Base SAS® 9.4 プロシジャガイド

第 4 版

SAS® ドキュメント

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2014. *Base SAS® 9.4 プロシジャガイド 第 4 版*. Cary, NC: SAS Institute Inc.

Base SAS® 9.4 プロシジャガイド 第 4 版

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

August 2014

目次

本書について

SAS 言語の構文規則

SAS 言語の構文規則の概要

SAS では、SAS 言語要素の構文ドキュメントに共通の規則を使用しています。これらの規則により、SAS 構文の構成要素を簡単に識別できます。規則は、次の項目に分類されます。

- 構文の構成要素
- スタイル規則
- 特殊文字
- SAS ライブラリと外部ファイルの参照

構文の構成要素

言語要素の多くでは、その構文の構成要素はキーワードと引数から構成されます。キーワードのみ必要な言語要素もあります。また、キーワードに等号(=)が続く言語要素もあります。複数の引数を含む構文で区切り記号を使用する場合と使用しない場合を説明するために、引数の構文の形式が複数示されています。

キーワード
プログラムの作成ときに使用する SAS 言語要素名です。キーワードはリテラルであり、通常、構文の先頭の単語です。CALL ルーチンでは、最初の 2 つの単語がキーワードです。

これらの例の SAS 構文では、キーワードには太字が使用されています。

CHAR (*string, position*)

CALL RANBIN (*seed, n, p, x*);

ALTER (*alter-password*)

BEST *w.*

REMOVE <*data-set-name*>

この例では、CALL ルーチンの最初の 2 つの単語がキーワードです。

CALL RANBIN(*seed*, *n*, *p*, *x*)

引数なしで 1 つのキーワードから構成される SAS ステートメント構文もあります。

DO;
... SAS code ...

END;

2 つのキーワード値のいずれか 1 つの指定が必要なシステムオプションもあります。

DUPLEX | **NODUPLEX**

プロシジャステートメントによっては、ステートメント構文中に複数のキーワードが含まれます。

CREATE <UNIQUE> **INDEX** *index-name* **ON** *table-name* (*column-1* <, *column-2*, …>)

引数
数値定数、文字定数、変数、式のいずれかです。引数は、キーワードに続くか、キーワードの後ろの等号に続きます。SAS では、引数を使用して、言語要素を処理します。引数が必須の場合もオプションの場合もあります。構文では、オプションの引数は山かっこ(< >)で囲まれます。

この例では、*string* と *position* がキーワード CHAR に続きます。これらの引数は、CHAR 関数の必須引数です。

CHAR (*string, position*)

引数ごとに値が指定されます。この例の SAS コードでは、引数 *string* の値は 'summer'、引数 *position* の値は 4 です。

```
x=char('summer', 4);
```

この例では、*string* および *substring* は必須引数ですが、*modifiers* と *startpos* はオプションです。

FIND(*string, substring* <,*modifiers*> <,*startpos*>

argument(s)
引数は必ず 1 つ必要であり、複数の引数が許可されます。引数の間はスペースで区切ります。カンマ(,)などの区切り記号は、引数間に必要ありません。

たとえば、MISSING ステートメントは、この形式で複数の引数を含みます。

MISSING *character(s)*;

<LITERAL_ARGUMENT> *argument-1* <<LITERAL_ARGUMENT> *argument-2 ...* >
引数は必ず 1 つ必要であり、リテラル引数がこの引数に関連付けられます。リテラルと引数のペアは複数指定できます。リテラルと引数の間に区切り記号は必要ありません。省略記号(...)は、追加のリテラルと引数が許可されることを示します。

たとえば、BY ステートメントはこの引数を含みます。

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …>;

argument-1 <option(s)> <argument-2 <option(s)> ...>
引数は必ず 1 つ必要であり、1 つ以上のオプションがこの引数に関連付けられます。複数の引数と関連するオプションを指定できます。引数とオプションの間に区切り記号は必要ありません。省略記号(...)は、追加の引数と関連するオプションが許可されることを示します。

たとえば、FORMAT プロシジャの PICTURE ステートメントは、この形式で複数の引数を含みます。

PICTURE name <(*format-option(s)*)>
<*value-range-set-1* <(*picture-1-option(s)*)>
<*value-range-set-2* <(*picture-2-option(s)*)> …>>;

argument-1=value-1 <argument-2=value-2 ...>
引数には値を割り当てる必要があり、複数の引数を指定できます。省略記号(...)は、追加の引数が許可されることを示します。引数間に区切り記号は必要ありません。

たとえば、LABEL ステートメントは、この形式で複数の引数を含みます。

LABEL *variable-1=label-1 <variable-2=label-2 …>*;

argument-1 <, argument-2, ...>
引数は必ず 1 つ必要であり、カンマまたは別の区切り記号で区切って複数の引数を指定できます。省略記号(...)は、カンマで区切られた引数が続くことを示します。SAS ドキュメントでは両方の形式が使用されます。

次に、この形式で指定された複数の引数の例を示します。

AUTHPROVIDERDOMAIN (*provider-1:domain-1 <, provider-2:domain-2, …>*

INTO :*macro-variable-specification-1 <, :macro-variable-specification-2, …>*

注: 通常、SAS ドキュメントのサンプルコードは、小文字の固定幅フォントを使用して表記されます。コードの作成には、大文字も、小文字も、大文字と小文字の両方も使用できます。

スタイル規則

SAS 構文の説明に使用されるスタイル規則には、大文字太字、大文字、斜体の規則も含まれます。

大文字太字
関数名やステートメント名などの SAS キーワードを示します。この例では、キーワード ERROR の表記には大文字太字が使用されています。

ERROR <*message*>;

大文字
リテラルの引数を示します。

この CMPMODEL=システムオプションの例では、BOTH、CATALOG、XML がリテラルです。

CMPMODEL=BOTH | CATALOG | XML |

斜体
ユーザー指定の引数または値を示します。斜体表記の項目は、ユーザー指定値であり、次のいずれかを表します。

- 非リテラル引数。この LINK ステートメントの例では、引数 *label* はユーザー指定値のため、斜体で表示されます。

  LINK *label*;

- 引数に割り当てられる非リテラル値。

  この FORMAT ステートメントの例では、引数 DEFAULT に変数の *default-format* が割り当てられます。

  FORMAT *variable(s) <format* > <DEFAULT = *default-format*>;

特殊文字

SAS 言語要素の構文には、次の特殊文字も使用されます。

=
等号は、一部の言語要素(システムオプションなど)のリテラル値を示します。

この MAPS システムオプションの例では、等号により MAPS の値が設定されます。

MAPS = *location-of-maps*

< >
山かっこはオプションの引数を示します。必須引数は山かっこで囲みません。

この CAT 関数の例では、少なくとも項目が 1 つ必要です。

CAT (*item-1* <, *item-2*, …>)

|
縦棒は、値グループから 1 つの値を選択できることを示します。縦棒で区切られている値は、相互排他です。

この CMPMODEL=システムオプションの例では、引数を 1 つのみ選択できます。

CMPMODEL=BOTH | CATALOG | XML

...
省略記号は、引数の繰り返しが可能なことを示します。引数と省略記号が山かっこで囲まれている場合、その引数はオプションです。繰り返される引数には、その引数の前や後ろに、区切り記号を入れる必要があります。

この CAT 関数の例では、複数の *item* 引数が許可され、カンマで区切る必要があります。

CAT (*item-1* <, *item-2*, …>)

'*value*'または"*value*"
一重引用符や二重引用符付きの引数は、その値にも一重引用符または二重引用符を付ける必要があることを示します。

この FOOTNOTE ステートメントの例では、引数 *text* に引用符が付けられています。

FOOTNOTE <*n*> <*ods-format-options* '*text*' | "*text*">;

;
セミコロンは、ステートメントまたは CALL ルーチンの終わりを示します。

この例では、各ステートメントがセミコロンで終了しています。

```
data namegame;
length color name $8;
color = 'black';
name = 'jack';
game = trim(color) || name;
run;
```

SAS ライブラリと外部ファイルの参照

多くの SAS ステートメントなどの言語要素では、SAS ライブラリと外部ファイルを参照します。論理名(ライブラリ参照名またはファイル参照名)から参照を作成するのか、引用符付きの物理ファイル名を使用するかを選択できます。論理名を使用する場合、通常、参照の作成に SAS ステートメント(LIBNAME または FILENAME)を使用するのか、動作環境のコントロール言語を使用するのかを選択します。複数の方法を使用して、SAS ライブラリと外部ファイルを参照できます。動作環境によっては使用できない方法があります。

SAS ドキュメントでは、外部ファイルを使用する例には斜体のフレーズ *file-specification* を使用します。また、SAS ライブラリを使用する例には斜体フレーズ *SAS-library* を引用符で囲んで使用します。

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```

Base SAS 9.4 プロシジャの新機能

概要

次のプロシジャが追加されました。

- PROC AUTHLIB
- PROC DELETE
- PROC DS2
- PROC FEDSQL
- PROC HDMD
- PROC JSON
- PROC LUA
- PROC PRESENV
- PROC PRODUCT_STATUS
- PROC SQOOP
- PROC STREAM

次の Base SAS プロシジャが拡張されました。

PROC APPEND	PROC HTTP
PROC CIMPORT	PROC IMPORT
PROC CONTENTS	PROC MIGRATE
PROC COPY	PROC OPTIONS
PROC CPORT	PROC PRINT
PROC DATASETS	PROC PRINTTO
PROC EXPORT	PROC PWENCODE
PROC FCMP	PROC QDEVICE
PROC FONTREG	PROC REPORT
PROC FORMAT	PROC SOAP
PROC HADOOP	PROC SORT
PROC HDMD	PROC XSL

新しい Base SAS プロシジャ

AUTHLIB プロシジャ

AUTHLIB プロシジャは、メタデータバインドライブラリの管理を可能にするユーティリティプロシジャです。詳細については、7 章, “AUTHLIB プロシジャ,” (102 ページ)を参照してください。

Base SAS 9.4 の 最初のメンテナンスリリースでは、 新しいオプションが AUTHLIB プロシジャに追加されました。REQUIRE_ENCRYPTION=YES オプションを CREATE または MODIFY ステートメントで使用することで、管理者はメタデータバインド型ライブラリ内のすべてのデータセットが作成時に自動的に暗号化されるように要求できます。詳細については、“メタデータバインド型データセットの暗号化を必須化する方法” (108 ページ)を参照してください。

Base SAS 9.4 の メンテナンスリリース 3 では、 新しいステートメントおよび MODIFY ステートメントの新しいオプションが AUTHLIB プロシジャに追加されました。

- PURGE ステートメントは、指定された置き換え日よりも古いメタデータ連結ライブラリ認証情報を削除します。詳細については、“PURGE ステートメント” (121 ページ)および “メタデータ連結ライブラリ認証情報の保持および削除” (107 ページ)を参照してください。
- MODIFY ステートメントには PURGE=オプションがあり、ライブラリ内のすべてのテーブルが新しい認証情報に正常に修正された場合に、保持されているすべてのメタデータ連結ライブラリ認証情報が自動的に削除されます。詳細については、“PURGE=YES | NO” (119 ページ)および“メタデータ連結ライブラリ認証情報の保持および削除” (107 ページ)を参照してください。

DELETE プロシジャ

DELETE プロシジャを使用すると、SAS ファイルをその格納先のディスクまたはテープから削除できます。詳細については、18 章, “DELETE プロシジャ,” (639 ページ)を参照してください。

DS2 プロシジャ

DS2 プロシジャを使用すると、Base SAS セッションから DS2 言語のステートメントをサブミットできます。DS2 は、高度なデータ操作とデータモデリングのアプリケーションに適した SAS プログラミング言語です。詳細については、20 章, “DS2 プロシジャ,” (655 ページ)を参照してください。

SAS 9.4 の 最初のメンテナンスリリースでは、 PROC DS2 INDB=オプションの名前が DS2ACCEL=に変更されました。INDB=は、現在でも別名としてサポートされています。ただし、このオプションのデフォルト値は YES から NO に変更されたため、DS2 コードはデータベースから実行されなくなります。

SAS 9.4 の メンテナンスリリース 2 では、 新しいオプション XCODE=によって、NLS トランスコーディングエラーが発生した場合の SAS セッションの動作が制御されます。また、SYSCC マクロ変数に、動作環境に返される現在の SAS 条件コードが含まれるようになりました。

SAS 9.4 の メンテナンスリリース 3 では、 PROC DS2 ステートメントの NOLIBS CONN=オプションを使用して、デフォルトのデータソース接続を指定したデータソース接続文字列で無効にすることができます。SAS 9.4 では、HAWQ および Impala データソースのサポートも追加されています。

FEDSQL プロシジャ

FEDSQL プロシジャを使用すると、Base SAS セッションから FedSQL 言語のステートメントをサブミットできます。FedSQL 言語は、ANSI SQL:1999 コア標準の SAS 実装です。詳細については、25 章, “FEDSQL プロシジャ,” (797 ページ)を参照してください。

SAS 9.4 の メンテナンスリリース 2 では、 新しいオプション XCODE=によって、NLS トランスコーディングエラーが発生した場合の SAS セッションの動作が制御されます。

SAS 9.4 の メンテナンスリリース 3 では、 HAWQ および Impala データソースのサポートが追加されています。

QUIT ステートメントの動作については、SAS 9.4 の メンテナンスリリース 3 の ドキュメントに記載されています。

HDMD プロシジャ

HDMD プロシジャによって、テーブルまたはファイルに対して SAS で定義されるメタデータが生成されます。このプロシジャは、Hive または HiveServer2 に依存しません。

JSON プロシジャ

JSON プロシジャは SAS データセットからデータを読み込み、そのデータを JSON 表現で外部ファイルに書き込みます。詳細については、35 章, “JSON プロシジャ,” (1027 ページ)を参照してください。

LUA プロシジャ

SAS 9.4 の メンテナンスリリース 3 では、 LUA プロシジャのサポートが追加されました。このプロシジャで、SAS セッション内の Lua コードを実行したり、Lua コマンド内で SAS 関数を呼び出したりすることができます。詳細については、36 章, “LUA プロシジャ,” (1058 ページ)を参照してください。

PRESENV プロシジャ

PRESENV プロシジャは、ある SAS セッションから別の SAS セッションにかけて、SAS コードのすべてのグローバルステートメントとマクロ変数を保存します。詳細については、44 章, “PRESENV プロシジャ,” (1333 ページ)を参照してください。

PRODUCT_STATUS プロシジャ

PRODUCT_STATUS は、システムにインストールされている SAS Foundation 製品の一覧を、それらの製品のバージョン番号とともに返します。詳細については、47 章, “PRODUCT_STATUS プロシジャ,” (1447 ページ)を参照してください。

SQOOP プロシジャ

SQOOP プロシジャで、SAS セッション内から Apache Sqoop にアクセスし、データベースと HDFS との間でデータを転送できます。SQOOP プロシジャのサポートが SAS 9.4

の メンテナンスリリース 3 に 追加されました。詳細については、60 章, “SQOOP プロシジャ,” (1827 ページ)を参照してください。

STREAM プロシジャ

STREAM プロシジャを使用すると、SAS マクロ指定を含む可能性のある任意のテキストで構成される入力ストリームを処理できます。マクロコードを拡張してファイルに保存できます。詳細については、62 章, “STREAM プロシジャ,” (1849 ページ)を参照してください。

Base SAS プロシジャの拡張

APPEND プロシジャ

APPEND プロシジャでは、次の拡張が行われました。

- ENCRYPTKEY=オプションでは、AES 暗号化データセットのキー値を指定します。詳細については、ENCRYPTKEY= (484 ページ)を参照してください。
- 拡張属性とは、SAS ファイル用にカスタマイズされたメタデータのことです。ユーザー定義の特性で、SAS データセットまたは変数と関連付けられます。詳細については、“拡張属性” (445 ページ)を参照してください。

CIMPORT プロシジャ

CIMPORT プロシジャでは次の拡張が行われました。

- PROC CIMPORT では、SAS 9.3 以前のバージョンの SAS から生成されたデータマイナーデータベースカタログエントリはインポートされません。警告メッセージが書き込まれます。
- データセットの拡張属性は、SAS 9.4 でサポートされます。データセットに拡張属性が含まれる場合に PROC CIMPORT を使用してファイルを転送するには、SAS 9.4 以上が必要です。拡張属性の詳細については、16 章, “DATASETS プロシジャ,” (436 ページ) および“Error and Warning Messages for Transport Files ” (*Moving and Accessing SAS Files*)を参照してください。
- SAS 9.4 の 最初のメンテナンスリリースでは、 ENCODINGINFO=オプションが追加され、トランスポートファイルでデータセットのエンコードを決定できるようになりました。エンコードの情報は、SAS ログに出力されます。詳細については、ENCODINGINFO= (324 ページ)を参照してください。
- SAS 9.4 の 最初のメンテナンスリリースでは、 タイムゾーンオフセットがあるデータセットを PROC CPORT (DATECOPY オプションを指定)および PROC CIMPORT を使用して移送できるようになりました。詳細については、“DATECOPY” (421 ページ)を参照してください。
- SAS 9.4 の メンテナンスリリース 2 では、 SORT オプションが PROC CIMPORT に追加されました。このオプションを設定すると、インポート対象のデータセットが、宛先オペレーティングシステムの照合シーケンスに従って再度ソートされます。詳細については、“SORT” (327 ページ)を参照してください。

詳細については、CIMPORT プロシジャ (321 ページ)を参照してください。

CONTENTS プロシジャ

CONTENTS プロシジャでは次の拡張が行われました。

- ENCRYPTKEY=オプションでは、AES 暗号化データセットのキー値を指定します。詳細については、ENCRYPTKEY= (484 ページ)を参照してください。
- 拡張属性とは、SAS ファイル用にカスタマイズされたメタデータのことです。ユーザー定義の特性で、SAS データセットまたは変数と関連付けられます。詳細については、“拡張属性” (445 ページ)を参照してください。
- TRANSCODE オプションは、変数がトランスコードされるかどうかを決定します。詳細については、“OUT=データセット” (534 ページ)を参照してください。

COPY プロシジャ

COPY プロシジャでは次の拡張が行われました。

- ENCRYPTKEY=オプションでは、AES 暗号化データセットのキー値を指定します。詳細については、ENCRYPTKEY= (484 ページ)を参照してください。
- 拡張属性とは、SAS ファイル用にカスタマイズされたメタデータのことです。ユーザー定義の特性で、SAS データセットまたは変数と関連付けられます。詳細については、“拡張属性” (445 ページ)を参照してください。

CPORT プロシジャ

CPORT プロシジャでは次の拡張が行われました。

- PROC CPORT では、SAS 9.3 以前のバージョンの SAS から生成されたデータマイナーデータベースカタログエントリはエックスポートされません。警告メッセージが書き込まれます。
- データセットの拡張属性は、SAS 9.4 でサポートされます。データセットに拡張属性が含まれる場合に PROC CPORT を使用してファイルを転送するには、SAS 9.4 以上を実行している必要があります。拡張属性の詳細については、16 章, “DATASETS プロシジャ,” (436 ページ) および“Error and Warning Messages for Transport Files ” (*Moving and Accessing SAS Files*)を参照してください。
- SAS 9.4 の 最初のメンテナンスリリースでは、 タイムゾーンオフセットがあるデータセットを PROC CPORT (DATECOPY オプションを指定)および PROC CIMPORT を使用して移送できるようになりました。詳細については、“DATECOPY” (421 ページ)を参照してください。

詳細については、CPORT プロシジャ (417 ページ)を参照してください。

DATASETS プロシジャ

DATASETS プロシジャでは次の拡張が行われました。

- DATASETS プロシジャは、拡張属性をサポートします。詳細については、“拡張属性” (445 ページ)を参照してください。DATASETS プロシジャに追加された次のステートメントを使用して拡張属性を管理できます。
  - XATTR ADD (523 ページ) ステートメントでは、拡張属性を変数またはデータセットに追加します。

  - XATTR DELETE (523 ページ) ステートメントでは、変数またはデータセットから拡張属性を削除します。
  - XATTR OPTIONS (524 ページ) ステートメントでは、拡張属性のオプションを指定します。
  - XATTR REMOVE (525 ページ) ステートメントでは、変数またはデータセットに対する拡張属性を削除します。
  - XATTR SET (525 ページ) ステートメントでは、変数またはデータセットに対する拡張属性を更新または追加します。
  - XATTR UPDATE (526 ページ) ステートメントでは、変数またはデータセットの属性を更新します。
- APPEND、COPY、CONTENTS ステートメントは AES 暗号化をサポートします。詳細については、"AES 暗号化データセットの追加" (463 ページ)、"AES 暗号化データファイルのコピー" (490 ページ)、および"ライブラリコンテンツと AES 暗号化" (477 ページ)を参照してください。
- CONTENTS ステートメントは、International Components for Unicode (ICU)リビジョン番号を印刷します。詳細については、"ICU 改訂番号の表示" (477 ページ)を参照してください。

EXPORT プロシジャ

EXPORT プロシジャでは次の拡張が行われました。

- SAS 9.4 の 最初のメンテナンスリリースでは、 CSV ファイルをエクスポートするときに、VALIDMEMNAME=EXTEND システムオプションが指定されている場合、SAS データセット名に一重引用符を使用できるようになりました。VALIDMEMNAME=を使用すると、SAS データセット名のルールが拡張されます。詳細については、"Using the External File Interface (EFI) " (*SAS/ACCESS Interface to PC Files: Reference*)を参照してください。
- JMP ファイルについては、次も当てはまります。
  - SAS 9.4 は JMP 7 以降の形式で保存された JMP ファイルからデータをインポートし、JMP 7 以降の形式でファイルにそのデータをエクスポートします。JMP 3 から 6 のファイル形式はサポートされなくなりました。これらのより新しい形式がサポートされるようになったため、JMP ファイルにアクセスして、JMP グラフビルダー iPad アプリなどのさまざまな方法で表示できます。
  - JMP ファイルの META データ型はサポートされなくなりました。代わりに、拡張属性が自動的に使用されます。META はプログラムに残すことができますが、その場合ログに NOTE が生成され、このステートメントが無視されます。
  - PROC EXPORT の META ステートメントは JMP ファイルでサポートされなくなり、無視されます。代わりに、拡張属性が自動的に使用されます。
  - JMP 変数名には、最大 255 文字を使用できます。
  - ROWSTATE データ型は JMP によって生成され、いくつかの低レベル文字を保存するために使用されます。PROC EXPORT で `_rowstate_`という名前の列が表示される場合、出力 JMP ファイルで変換され、列状態情報に返されます (JMP ファイルに列状態情報が含まれている場合、PROC IMPORT はこの情報を`_rowstate_`という名前を使用して新しい変数として保存します。)
  - 詳細については、"JMP Files" (*SAS/ACCESS Interface to PC Files: Reference*) を参照してください。

詳細については、21 章, "EXPORT プロシジャ" (675 ページ)を参照してください。

FCMP プロシジャ

FCMP プロシジャでは次の拡張が行われました。

- FCMP プロシジャに、RUN_MACRO 関数と DOSUBL 関数を比較するサンプルが含まれるようになりました。RUN_MACRO 起動時には、すべての変数が、設定できるように引数として渡されます。DOSUBL の呼び出しでは、すべてのマクロ変数は実行されるコードにインポートされてから、完了時にエクスポートされます。詳細については、“RUN_MACRO 関数” (769 ページ)を参照してください。
- PROC FCMP で作成する関数およびサブルーチンに使用できるプロシジャがリストされているテーブルに、PROC OPTMODEL が追加されました。詳細については、概要: FCMP プロシジャ (694 ページ)を参照してください。

SAS 9.4 の メンテナンスリリース 3 では、 STATIC ステートメントが追加されました。詳細については、“STRUCT ステートメント” (711 ページ) を参照してください。

FONTREG プロシジャ

SAS 9.4 の メンテナンスリリース 2 では、 次の拡張が FONTREG プロシジャに行われました。

- OPENTYPE ステートメントが追加されました。このステートメントは、有効な OpenType フォントファイルを検索する 1 つ以上のディレクトリを指定します。
- ファイル参照を使用する機能が追加されました。この機能により、FILENAME ステートメントとその機能を使用できるようになります。

詳細については、26 章, “FONTREG プロシジャ” (817 ページ)を参照してください。

FORMAT プロシジャ

FORMAT プロシジャでは次の拡張が行われました。

- %*n*B ディレクティブで使用する文字数を指定して、月を短縮形式でフォーマットできます。
- 入力形式、ピクチャ、または出力形式 1-32767 のデフォルトの長さを指定する範囲。

詳細については、FORMAT プロシジャ (833 ページ)を参照してください。

HADOOP プロシジャ

HADOOP プロシジャでは次の拡張が行われました。

- HADOOP プロシジャの PROPERTIES ステートメントを使用して、構成プロパティを Hadoop サーバーにサブミットできるようになりました。
- HDFS ステートメントで NOWARN オプションを指定して、存在しないファイルを削除しようとしたときに表示される警告メッセージを非表示にすることができるようになりました。
- SAS 9.4 の メンテナンスリリース 3 では、 HDFS ステートメントで、ファイルの内容を表示する CAT=オプション、ファイルのアクセス許可を変更する CHMOD=オプション、および HDFS ファイルを一覧表示する LS=オプションがサポートされます。
- SAS 9.4 の メンテナンスリリース 3 では、 いくつかの HDFS ステートメントオプションで、HDFS ファイルの指定時にワイルドカード文字を使用したり、指定されたディ

レクトリおよびサブディレクトリで操作を実行する再帰アクションを要求したりすることができるようになりました。

- SAS 9.4 の メンテナンスリリース 3 では、 Hadoop クラスタに接続するための新しいメソッドがあります。Hadoop クラスタ構成ファイルを、SAS クライアントマシンにアクセスできる物理的な場所にコピーしてから、SAS 環境変数を構成ファイルの場所に設定できます。
- SAS 9.4 の メンテナンスリリース 3 では、 Apache Oozie RESTful API を通じて MapReduce プログラムと Pig 言語コードを Hadoop クラスタにサブミットできます。

詳細については、30 章, “HADOOP プロシジャ,” (947 ページ)を参照してください。

HTTP プロシジャ

HTTP プロシジャでは次の拡張が行われました。

- SSLCALISTLOC システムオプションで、HTTPS プロトコルを使用する接続に信頼ソースを設定できます。SAS ソフトウェアの前のバージョンでこの信頼ソースの設定に使用されていた Java システムオプションは現在サポートされていません。
- SAS 9.4 の 最初のメンテナンスリリースでは、 HTTP_TOKENAUTH オプションが追加されて、SAS コンテンツサーバーにアクセスするために使用できるワンタイムパスワードをメタデータサーバーから生成できるようになりました。
- SAS 9.4 の 最初のメンテナンスリリースでは、 PROC HTTP でユーザー ID 認証がサポートされます。接続先のサーバーが NTLM (Windows のみ)または Kerberos 認証プロトコルをサポートしている場合、ユーザー名とパスワードを指定する必要はありません。現在のユーザー ID に権限がある場合、認証が確立されます。
- SAS 9.4 の メンテナンスリリース 3 では、 PROC HTTP でメソッドのサポートが拡張され、HTTP/1.1 標準をサポートし、ターゲットサーバーでサポートされるすべてのメソッドが含まれるようになりました。また、HEADER ステートメント、認証タイプ仕様、HTTP/1.1 機能(永続接続、cookie のキャッシング、EXPECT_100_CONTINUE のサポートなど)が追加されました。入力データは引用符付きの文字列で指定するか、ファイル参照からサブミットできます。カスタム要求ヘッダーは HEADERS ステートメントで名前と値のペアとして指定するか、ファイル参照名から完全にフォーマットされた入力ファイルをサブミットして指定できます。これをサポートするウェブサーバーについては、プロシジャはデフォルトで接続キャッシングと Cookie キャッシングを使用します。プロシジャ内で、この 2 つの種類のキャッシングの動作を切り替えて、キャッシュをクリアするには、プロシジャの引数を指定します。または、マクロ変数を使用して、Cookie キャッシングをオフにします。リダイレクトを必要とする要求の応答ヘッダーを管理するための HEADEROUT_OVERWRITE 引数も提供されます。

詳細については、32 章, “HTTP プロシジャ,” (983 ページ)を参照してください。

IMPORT プロシジャ

IMPORT プロシジャでは次の拡張が行われました。

- SAS 9.4 の 最初のメンテナンスリリースでは、 CSV ファイルをインポートするときに、VALIDMEMNAME=EXTEND システムオプションが指定されている場合、SAS データセット名に一重引用符を使用できるようになりました。VALIDMEMNAME=を使用すると、SAS データセット名のルールが拡張されます。詳細については、“Using the External File Interface (EFI) ” (*SAS/ACCESS Interface to PC Files: Reference*)を参照してください。
- JMP ファイルについては、次も当てはまります。

- SAS 9.4 は JMP 7 以降の形式で保存された JMP ファイルからデータをインポートし、JMP 7 以降の形式でファイルにそのデータをエクスポートします。JMP 3 から 6 のファイル形式はサポートされなくなりました。これらのより新しい形式がサポートされるようになったため、JMP ファイルにアクセスして、JMP グラフビルダー iPad アプリなどのさまざまな方法で表示できます。
- JMP ファイルの META データ型はサポートされなくなりました。代わりに、拡張属性が自動的に使用されます。META はプログラムに残すことができますが、その場合ログに NOTE が生成され、このステートメントが無視されます。
- PROC IMPORT の META ステートメントは JMP ファイルでサポートされなくなり、無視されます。代わりに、拡張属性が自動的に使用されます。拡張属性による JMP ファイルのインポート時には、拡張属性は新規 SAS データセットに自動的に割り当てられます。
- JMP 変数名には、最大 255 文字を使用できます。
- ROWSTATE データ型は JMP によって生成され、いくつかの低レベル文字を保存するために使用されます。JMP ファイルに列状態情報が含まれている場合、PROC IMPORT はこの情報を`_rowstate_`という名前を使用して新しい変数として保存します(PROC EXPORT で`_rowstate_`という名前の列が表示される場合、JMP 出力ファイルで変換され、列状態情報に返されます)。

詳細については、“JMP Files” (*SAS/ACCESS Interface to PC Files: Reference*)および33 章, “IMPORT プロシジャ” (1005 ページ)を参照してください。

MIGRATE プロシジャ

MIGRATE プロシジャでは次の拡張が行われました。

- データセットで拡張属性がサポートされます。詳細については、MIGRATE プロシジャ (1176 ページ)を参照してください。
- SAS 9.4 の メンテナンスリリース 3 では、 BUFSIZE のデフォルト値が変更されています。新しいデフォルト値は、現在のセッションのバッファページのサイズです。前の動作をそのまま使用して、ソースライブラリからメンバのページサイズをコピーする場合は、BUFSIZE=KEEPSIZE を指定します。

OPTIONS プロシジャ

OPTIONS プロシジャでは次の拡張が行われました。

- LISTOPTSAVE オプションでは、OPTSAVE プロシジャまたは DMOPTSAVE コマンドを使用して保存できるシステムオプションが一覧表示されます。詳細については、39 章, “OPTIONS プロシジャ” (1197 ページ)を参照してください。
- SAS 9.4 の メンテナンスリリース 2 では、 OPTIONS プロシジャは、SAS ログでパスワードを表示する際、実際の長さにかかわらず 8 個の X で表示します。

PRINT プロシジャ

PRINT プロシジャでは、次の拡張が行われました。

- PROC PRINT SUMLABEL=オプションと GRANDTOTAL_LABEL=オプションを使用して、BY グループ合計および総計値のラベルを指定できます。
- HEADER 場所に対する PROC PRINT ステートメントの STYLE=オプションスタイル属性は、Obs 列ヘッダーに影響しなくなりました。Obs 列ヘッダーのスタイル属性は、OBSHEADER 場所を使用して指定します。

詳細については、PRINT プロシジャ (1337 ページ)を参照してください。

PRINTTO プロシジャ

PROC PRINTTO PRINT=ステートメントで、LISTING 出力先が開きます。PRINTTO プロシジャを使用する前に ODS LISTING ステートメントを指定する必要がなくなりました。詳細については、46 章, “PRINTTO プロシジャ” (1427 ページ)を参照してください。

SAS 9.4 の メンテナンスリリース 3 では、 SAS ログと LISTING 出力ファイルが以前保存されていた場所を復元できます。SAS では、自動マクロ変数に SAS ログと LISTING 出力ファイルのパスが保存されます。詳細については、“前の SAS ログまたは LISTING 出力ファイルの場所の復元” (1434 ページ)を参照してください。

PWENCODE プロシジャ

SAS/SECURE を使用する場合は、保存されたパスワード用の新しい暗号化タイプ SAS004 (64 ビット salt を使用する AES 暗号化)を使用できます。salt のサイズが 64 ビットに拡大されて、PKCS #5 v2.0:Password-Based Cryptography Standard http://www.rsa.com/rsalabs/node.asp?id=2127 の最小推奨 salt サイズに準拠するようになりました。

詳細については、51 章, “PWENCODE プロシジャ” (1495 ページ)を参照してください。

QDEVICE プロシジャ

QDEVICE プロシジャでは次の拡張が行われました。

- PROC QDEVICE DEVLOC=オプションを使用すると、SAS/GRAPH Gdevice*n* ライブラリと Sashelp ライブラリ以外のデバイスライブラリを指定できます。デバイスの最初の発生についてレポートすることも、Gdevice*n* および Sashelp ライブラリ内のデバイスのすべての発生についてレポートすることもできます。
- PROC QDEVICE CATALOG=オプションを使用すると、検索対象として、DEVICES カタログ以外のカタログを指定できます。
- DEVICE ステートメントと PRINTER ステートメントでは、ワイルドカード*と?を デバイス名とプリンタ名に使用できます。
- Windows プリンタデバイスタイプの値は、TYPE 変数でレポートされるように、Printer Interface Device です。Printer Interface Device は、System Printer という値に置き換わるものです。
- すべてのレポートで、NAMETYPE 変数名が TYPE に変更されました。
- 一般レポートには、新たに次の情報が含まれています。
  - ALIAS 変数は、フォントの別名をレポートします。
  - ANIMATION 変数は、プリンタアニメーションのステータスについてレポートします。
  - FVERSION 変数は、フォントのバージョンをレポートします。
  - COMPRESSION 変数は、圧縮が使用される条件を示すようになりました。
  - COMPMETHOD 変数は、圧縮方法を示します。
  - MODULE 変数はデバイスドライバモジュールの名前をレポートします。
  - レポート出力データセットの文字変数の長さは固定長になりました。

- フォントレポートには、フォント別名とフォントバージョンに関する情報が含まれるようになりました。新しい変数はそれぞれ ALIAS と FVERSION です。
- 一般レポートを除くすべてのレポートでは、プロトタイプに関する情報が削除されています。PROTOTYPE 変数情報は一般レポートでのみレポートされるようになりました。
- レポート出力データセットの文字変数の長さは、128 文字の固定長です。レポートを結合または連結するときに、LENGTH ステートメントは必要なくなりました。

詳細については、QDEVICE プロシジャ (1504 ページ)を参照してください。

REPORT プロシジャ

REPORT プロシジャでは次の拡張が行われました。

- PROC REPORT の In-Database 処理では、Aster、Greenplum、および HADOOP データベース管理システムをサポートしています。
- SAS 9.4 の メンテナンスリリース 3 では、 PROC REPORT のデータベース内処理で、Impala、HAWK、および SAP HANA データベース管理システムをサポートします。
- NOWINDOWS (NOWD)は、PROC REPORT のデフォルトのウィンドウ環境になりました。
- SAS 9.4 の メンテナンスリリース 2 では、 PROC REPORT での ODS スタイルの使用法を説明する新しいセクションが追加されました。詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1655 ページ)を参照してください。
- SAS 9.4 の メンテナンスリリース 2 では、 PROC REPORT が統計キーワード P20、P30、P40、P60、P70、および P80 をサポートするようになりました。詳細については、“PROC REPORT で使用できる統計量” (1654 ページ)を参照してください。

詳細については、55 章, “REPORT プロシジャ,” (1580 ページ)を参照してください。

SOAP プロシジャ

SAS 9.4 の メンテナンスリリース 2 では、 SAS がロックダウン状態のときは、SOAP プロシジャは使用できません。詳細については、58 章, “SOAP プロシジャ,” (1771 ページ)を参照してください。

SORT プロシジャ

SORT プロシジャでは、次の拡張が行われました。

- PROC SORT は拡張属性をサポートします。PROC SORT では、データセットに定義された属性が出力データセットにコピーされます。詳細については、“拡張属性” (445 ページ)を参照してください。
- PROC SORT で使用されるユーティリティファイルのページサイズは、新しい STRIPESIZE=システムオプションの影響を受けます。詳細については、“STRIPESIZE= System Option” (*SAS System Options: Reference*)を参照してください。
- PROC SORT で使用される International Components for Unicode (ICU)ライブラリがバージョン 4.8.1 にアップグレードされました。この新しい ICU バージョンは、Unicode Common Locale Data Repository (CLDR)のバージョン 2.0 のローカルデータを使用し、言語のサポートを強化し、ソフトウェアの修正を提供します。詳細に

ついては、http://site.icu-project.org/download/48 および http://cldr.unicode.org/index/downloads/cldr-2-0 を参照してください。

SAS で使用される ICU バージョンの変更は、前のバージョンの SAS で並べ替えられた一部のデータセットの解釈に影響する可能性があります。これらの影響については、59 章, “SORT プロシジャ,” (1785 ページ)、14 章, “COPY プロシジャ,” (409 ページ)、、38 章, “MIGRATE プロシジャ,” (1176 ページ)を参照してください。

- CONTENTS プロシジャまたは CONTENTS ステートメントの出力に、言語ソートされたデータセットの ICU バージョン番号が示されるようになりました。
- SAS 9.4 の メンテナンスリリース 3 では、 PROC SORT のデータベース内処理で、Impala、HAWK、および SAP HANA データベース管理システムをサポートします。

詳細については、59 章, “SORT プロシジャ” (1785 ページ)を参照してください。

XSL プロシジャ

XSL プロシジャは、パラメータ値を XSL スタイルシートに渡すことができる PARAMETER ステートメントを提供するようになりました。詳細については、67 章, “XSL プロシジャ,” (2065 ページ)を参照してください。

ソフトウェアの拡張

SAS では、高度暗号化標準(AES)で作成されたファイルへのアクセスがサポートされています。詳細については、“AES 暗号化” (24 ページ)を参照してください。

International Components for Unicode (ICU)ライブラリは、バージョン 4.2 からバージョン 4.8 にアップグレードされました。ICU は文字データの言語照合のために SAS で使用されます。ICU バージョン 4.8 の詳細については、ICU の Web サイト(http://site.icu-project.org/download/48)を参照してください。

SAS の 1 つのリリースで言語ソートされたデータセットは、別のリリースで並べ替え済みとして認識されない場合があります。ICU バージョンの変更の影響については、次を参照してください。

- “ データセットの言語ソートと ICU” (1790 ページ)
- 13 章, “CONTENTS プロシジャ,” (401 ページ)
- 14 章, “COPY プロシジャ,” (409 ページ)
- 38 章, “MIGRATE プロシジャ,” (1176 ページ)

ドキュメントの拡充

*Base SAS プロシジャガイド*に、次の変更が行われました。

- “Base SAS プロシジャのスレッド処理” (25 ページ) に、スレッド処理をサポートする SAS プロシジャに関する情報が含まれます。
- “PROC FCMP と DATA ステップの構成要素オブジェクト” (730 ページ) に、ハッシングに関する情報が含まれます。

- SAS 9.4 の 最初のメンテナンスリリースでは、 リンクとそれを説明するテキストが、PROC FCMP で使用可能な Microsoft Excel 関数に追加されました。
- 31 章, “HDMD プロシジャ,”に関する情報は、*SAS/ACCESS for Relational Databases: Reference* からこのドキュメントに移動しました。
- 38 章, “MIGRATE プロシジャ,” (1176 ページ) に、BUFSIZE=オプションを使用して、移行されたデータセットのパフォーマンス向上に関する情報が含まれます。

1 部

概念

1 章

適切なプロシジャの選択

Base SAS プロシジャの関数カテゴリ

レポート作成

これらのプロシジャを使用して、データのリスト表示（詳細レポート）、要約レポート、カレンダ、レター、ラベル、マルチパネルレポート、およびグラフ付きレポートなどの有益な情報を表示できます。

CALENDAR	PLOT	SQL*
CHART*	PRINT	SUMMARY*
FREQ*	QDEVICE*	TABULATE *
MEANS*	REPORT*	TIMEPLOT

* これらのプロシジャによって、レポートが作成され、統計が計算されます。

統計

これらのプロシジャによって、モーメント、等量分類、信頼区間、度数カウント、クロス集計、相関分析および分布テストに基づいた記述統計など、基本的な統計測定値が計算されます。データのランク付けや標準化も行われます。

CHART	RANK	SUMMARY
CORR	REPORT	TABULATE
FREQ	SQL	UNIVARIATE
MEANS	STANDARD	

ユーティリティ

これらのプロシジャによって、次の基本ユーティリティ操作が実行されます。

- データセットの作成、編集、ソートおよび移送
- 移送データセットの作成および復元
- ユーザー定義の出力形式の作成
- データセットのコピー、追加および比較などの基本的なファイル管理
- ディスクまたはテープからの永続または一時データファイルの削除

APPEND	FEDSQL	PRESENV
AUTHLIB	FONTREG	PRINTTO
CATALOG	FSLIST	PRTDEF
CIMPORT	GROOVY	PRTEXP
COMPARE	HADOOP	PWENCODE
CONTENTS	IMPORT††	REGISTRY
CONVERT*	INFOMAPS †††	RELEASE*
COPY	JAVAINFO	SCAPROC
CPORT	JSON	SORT
CV2VIEW***	METADATA‡	ソース
DATASETS	METALIB‡	SQL*
DBCSTAB†	METAOPERATE‡	STREAM

DELETE	MIGRATE	TAPECOPY
表示	OPTIONS	TAPELABEL *
DOCUMENT **	OPTLOAD	TEMPLATE
DS2	OPTSAVE	TRANSPOSE*
EXPORT††	PDS	TRANSTAB **
FCMP	PDSCOPY*	XSL
	PMENU*	

* このプロシジャの説明については、ご利用の動作環境に応じた SAS ドキュメントを参照してください。
** このプロシジャの説明については、*SAS Output Delivery System: ユーザーガイド*を参照してください。
*** このプロシジャの説明については、*SAS/ACCESS for Relational Databases: Reference* を参照してください。
† このプロシジャの説明については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。
†† このプロシジャの説明については、*SAS/ACCESS Interface to PC Files: Reference* を参照してください。
††† このプロシジャの説明については、*Base SAS ガイド(Information Map)*を参照してください。
‡ このプロシジャの説明については、*SAS Language Interfaces to Metadata* を参照してください。

レポート作成プロシジャ

次の表には、レポートのタイプに応じたレポート作成プロシジャが一覧表示されています。

表 1.1 *タスク別レポート作成プロシジャ*

レポートの種類	プロシジャ	説明
詳細レポート	PRINT	データのリストをすばやく作成できます。タイトル、フットノートおよび列の合計も設定できます。
	REPORT	PROC PRINT よりも詳細に制御およびカスタマイズを行えます。列と行の両方の合計も計算でき、DATA ステップ計算機能も使用できます。
	SQL	SQL(Structured Query Language)と、形式設定などの SAS 機能が組み合わされます。データを操作し、レポートの作成と同じステップで SAS データセットを作成できます。列と行の統計を作成できます。PROC PRINT や PROC REPORT ほどの出力制御機能は提供されません。
要約レポート	MEANS または SUMMARY	数値変数の記述統計を計算します。印刷されたレポートを生成し、出力データセットを作成できます。

レポートの種類	プロシジャ	説明
	PRINT	要約レポートを 1 つのみ作成します。BY 変数を合計できます。
	QDEVICE	グラフィックデバイスとユニバーサルプリンタに関するレポートを作成します。
	REPORT	PRINT、MEANS および TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートを作成できる単一のレポート作成ツールに組み込みます。出力データセットも作成できます。
	SQL	1 つ以上の SAS データセットまたは DBMS テーブルに対して記述統計を計算します。印刷されたレポートを作成したり、SAS データセットを作成したりすることができます。
	TABULATE	記述統計を表形式で生成します。スタブアンドバナーレポート(記述統計が含まれた多次元表)を作成できます。出力データセットも作成できます。
詳細に形式設定された多種多様なレポート		
カレンダ	CALENDAR	スケジュールカレンダと要約カレンダを生成します。休日期間および休暇、週次の勤務スケジュール、日次の勤務シフトのタスクをスケジュール設定できます。
マルチパネルレポート (電話帳リスト)	REPORT	マルチパネルレポートを生成します。
低解像度のグラフィカルレポート *		
	CHART	度数とその他の統計を表示する棒グラフ、ヒストグラム、BLOCK チャート、円グラフ、スターチャートを作成します。
	PLOT	1 つの変数を別の変数にプロットする散布図を作成します。
	TIMEPLOT	一定期間の 1 つ以上の変数のプロットを生成します。

* これらのレポートでは、データの簡単なグラフィカル画像がすばやく作成されます。高解像度のグラフィカルレポートを作成するには、SAS/GRAPH ソフトウェアを使用してください。

統計プロシジャ

利用可能な統計プロシジャ

次の表には、タスクに応じた統計プロシジャが一覧表示されています。表 A1.1 (2079 ページ)一般的によく使用されている統計値と、それらを計算するプロシジャを表示します。

表 1.2 タスク別基本統計量プロシジャ

レポートの種類	プロシジャ	説明
記述統計	CORR	単純な記述統計を計算します。
	MEANS または SUMMARY	記述統計を計算します。印刷された出力および出力データセットを生成できますデフォルトで、PROC MEANS は印刷された出力を生成し、PROC SUMMARY は出力データセットを作成します。
	REPORT	PROC TABULATE と大部分が同じである統計値を計算します。出力形式をカスタマイズできます。
	SQL	1 つ以上の DBMS テーブルのデータに関する記述統計値を計算します。印刷されたレポートを生成したり、SAS データセットを作成したりすることができます。
	TABULATE	記述統計に関する表レポートを生成します。出力データセットを作成できます。
	UNIVARIATE	最も広範囲にわたって一連の記述統計値を計算します。出力データセットを作成できます。
度数およびクロス集計表	FREQ	1 元から *n* 元の表を作成します。度数カウントをレポートします。カイ二乗検定を計算します。2 元から *n* 元のクロス集計表に対する関連性および一致の検定および測定値を計算します。正確検定および漸近検定を計算できます。出力データセットを作成できます。
	TABULATE	1 元および 2 元クロス集計表を作成します。出力データセットを作成できます。
	UNIVARIATE	1 元度数表を生成します。
相関分析	CORR	ピアソン、スピアマン、ケンドルの相関および偏相関を計算します。また、ヘフディングの従属(D)のメジャーおよびクローンバックのアルファ係数も計算します。
分布分析	UNIVARIATE	位置の検定と正規性の検定を計算します。

レポートの種類	プロシジャ	説明
	FREQ	1 元表の二項比率の検定を計算します。1 元表の適合度検定を計算します。2 元表の等分布のカイ 2 乗検定を計算します。
ロバスト推定	UNIVARIATE	スケール、トリム平均、ウィンザー化平均のロバスト推定を計算します。
データ変換		
ランクの計算	RANK	SAS データセットのオブザベーションに対して 1 つ以上の変数のランクを計算し、出力データセットを作成します。正規スコアまたはその他のランクスコアを生成できます。
データの標準化	STANDARD	所定の平均および標準偏差に標準化された変数を含む出力データセットを作成します。
低解像度グラフィック*		
	CHART	グラフ変数に対する統計である度数カウント、パーセント、累積度数、累積パーセント、合計、平均のいずれかを表示できるグラフィカルレポートを作成します。
	UNIVARIATE	幹葉図、箱ひげ図、正規確率プロットなどの記述プロットを作成します。

* 高解像度のグラフィカルレポートを作成するには、SAS/GRAPH ソフトウェアを使用してください。

効率に関する問題

分位点

大規模なサンプルサイズ n の場合、中央値などの分位点の計算には $n\log(n)$と比例する計算時間を要します。そのため、分位数を自動的に計算する UNIVARIATE などのプロシジャには、その他のデータ要約プロシジャよりも多くの時間が必要になることがあります。さらに、データはメモリに保持されるため、プロシジャには計算を実行するための記憶領域もさらに必要になります。デフォルトでは、レポートプロシジャ PROC MEANS、PROC SUMMARY、PROC TABULATE は分位数を自動的に計算しないため、必要なメモリは少なくなります。これらのプロシジャでは、新しい固定メモリ、通常はそれほどメモリ集中型ではない分位数推定方法を使用するオプションも使用できます。詳細については、“分位点” (1131 ページ)PROC MEANS のドキュメントを参照してください。

オブザベーショングループの統計の計算

オブザベーションの複数のグループの統計を計算するために、前のプロシジャを BY ステートメントで使用して、BY グループ変数を指定できます。ただし、BY グループ処理には、前もってデータセットを並べ替えたり、インデックス付けを行ったりする必要があります。非常に大規模なデータセットの場合は、大量のコンピュータリソースが必要となることがあります。並べ替えを行わずにグループ内の統計をさらに効率的に計算するには、CLASS ステートメントを次のいずれかのプロシジャとともに使用します:MEANS、SUMMARY または TABULATE。

統計プロシジャに関する追加情報

付録 1, “基本的な SAS 統計プロシジャ” (2077 ページ) には、Base SAS プロシジャが頻繁に計算する統計に関する標準キーワード、統計的表記および式が一覧表示されています。個々の統計プロシジャのセクションには、プロシジャ出力の解釈に役立つ統計値の概念が説明されています。

ユーティリティプロシジャ

次の表には、タスクに応じたユーティリティプロシジャがグループ化されています。

表 1.3 タスク別ユーティリティプロシジャ

タスク	プロシジャ	説明
情報を提供する	COMPARE	2 つの SAS データセットのコンテンツを比較します。
	CONTENTS	SAS ライブラリまたは特有ライブラリメンバのコンテンツを説明します。
	JAVAINFO	SAS が使用している Java 環境に関する診断情報を通知します。
	OPTIONS	すべての SAS システムオプションの現在の値をリストします。
	SCAPROC	SAS コードアナライザを実装します。これは、実行中に SAS ジョブからの入力、出力、マクロシンボルの使用に関する情報を取得します。
	SQL	個々の SAS データセット、および現在の SAS セッションでアクティブなすべての SAS ファイルのディクショナリテーブルによって情報を提供します。ディクショナリテーブルでは、マクロ、タイトル、インデックス、外部ファイルまたは SAS システムオプションに関する情報も提供できます。
SAS システムオプションを管理する	OPTIONS	すべての SAS システムオプションの現在の値をリストします。
	OPTLOAD	SAS レジストリや SAS データセットに保存されている SAS システムオプション設定を読み取ります。
	OPTSAVE	SAS システムオプション設定を SAS レジストリまたは SAS データセットに保存します。
印刷および Output Delivery System 出力に影響を及ぼす	DOCUMENT**	ODS 文書に保存されるプロシジャ出力を操作します。
	FONTREG	SAS レジストリにシステムフォントを追加します。

タスク	プロシジャ	説明
	FORMAT	データを表示および印刷するユーザー定義の出力形式を作成します。
	PRINTTO	プロシジャ出力をファイル、SAS カタログエントリ、プリンタに送ります。SAS ログをファイルにリダイレクトすることもできます。
	PRTDEF	プリンタ定義を作成します。
	PRTEXP	プリンタ定義属性を SAS データセットにエクスポートします。
	TEMPLATE**	ODS 出力をカスタマイズします。
データの作成、参照、編集	FCMP	SAS 関数およびサブルーチンを作成、検定および保存してから、他の SAS プロシジャで使用できます。
	FSLIST	SAS ソース行または SAS プロシジャ出力を含むファイルなどの外部ファイルを参照します。
	INFOMAPS†††	SAS Information Map を作成または更新します。
	PRESENV	ある SAS セッションから別の SAS セッションにかけて、SAS コードのすべてのグローバルステートメントとマクロ変数を保存します。このプロシジャが SAS セッションの最後に呼び出されると、グローバルステートメントとマクロ変数はすべて、1 つのファイルに書き込まれます。
	SQL	Structured Query Language と SAS 機能を使用して SAS データセットを作成します。
データの変換	DBCSTAB†	SAS がサポートするダブルバイト文字セットの変換テーブルを作成します。
	FORMAT	データを読み込むユーザー定義の入力形式と、データを表示するユーザー定義の出力形式を作成します。
	SORT	SAS データセットを 1 つ以上の変数によって並べ替えます。
	SQL	SAS データセットを 1 つ以上の変数によって並べ替えます。
	STREAM	SAS マクロ仕様を含むことができる任意のテキストで構成される入力ストリームを処理できます。マクロコードを拡張してファイルに保存できます。
	TRANSPOSE	オブザベーションが変数になり、変数がオブザベーションになるように SAS データセットを変換します。
	TRANTAB†	カスタマイズされた変換テーブルを作成、編集、表示します。

タスク	プロシジャ	説明
	XSL	XML ドキュメントを HTML、テキスト、または他の種類の XML ドキュメントなどの別の出力形式に変換します。
SAS ファイルを管理する	APPEND	1つの SAS データセットを別のデータセットの最後に追加します。
	AUTHLIB	metadata-bound ライブラリを管理します。
	CATALOG	SAS カタログエントリを管理します。
	CIMPORT	PROC CPORT が(通常は別の動作環境で)作成する移送順次ファイルを、元の形式に SAS カタログ、SAS データセット、SAS ライブラリとしてリストアします。
	CONVERT*	BMDP システムファイル、OSIRIS システムファイル、SPSS ポータブルファイルを SAS データセットに変換します。
	COPY	SAS ライブラリまたはライブラリの特定メンバをコピーします。
	CPORT	SAS カタログ、SAS データセットまたは SAS ライブラリを PROC CIMPORT がその元の形式にリストア可能な移送順次ファイルに(通常は別の動作環境で)変換します。
	CV2VIEW***	SAS/ACCESS ビューディスクリプタを PROC SQL ビューに変換します。
	DATASETS	SAS ファイルを管理します。
	DELETE	永続または一時 SAS ファイルを削除します。
	EXPORT††	データを SAS データセットから読み込み、外部データソースに書き込みます。
	IMPORT††	データを外部データソースから読み込み、SAS データセットに書き込みます。
	JSON	データを SAS データセットから読み込み、JSON 表現で外部データソースに書き込みます。
	MIGRATE	SAS ライブラリのメンバを SAS の最新リリースに移行します。
	PDS*	分割されたデータセットのメンバをリスト、削除、名前変更します。
	PDSCOPY*	分割されたデータセットをディスクからテープ、ディスクからディスク、テープからテープ、またはテープからディスクにコピーします。

タスク	プロシジャ	説明
	PROTO	C または C++プログラミング言語で記述される外部関数のバッチモードでの登録を行えるようにします。
	REGISTRY	レジストリ情報を SAS レジストリの USER 部分にインポートします。
	RELEASE*	z/OS 環境のディスクデータセットの最後の未使用スペースを解放します。
	ソース*	ソースライブラリデータセットを簡単にバックアップして処理できます。
	SQL	SAS データセットを連結します。
	TAPECOPY*	テープボリューム全体またはファイル全体を 1 つ以上のテープボリュームから 1 つの出力テープボリュームにコピーします。
	TAPELABEL*	z/OS 環境の IBM 標準ラベル付テープボリュームのラベル情報をリストします。
コントロールウィンドウ	PMENU	SAS アプリケーション用にカスタマイズされたメニューを作成します。
その他	表示	SAS/AF アプリケーションを実行します。
	DS2	Base SAS セッションから DS2 言語ステートメントをサブミットします。
	FEDSQL	Base SAS セッションから FedSQL 言語ステートメントをサブミットします。
	GROOVY	Java 仮想マシン(JVM)で、SAS コードによって Groovy コードを実行できるようにします。
	HADOOP	SAS が Hadoop データに対して Apache Hadoop コードを実行できるようにします。
	PWENCODE	SAS プログラムで使用するパスワードをエンコードします。
SAS Metadata Repository のメタデータを管理します。	METADATA‡	XML 文字列形式のメソッド呼び出しを SAS Metadata Server に送信します。
	METALIB‡	ライブラリ内のテーブルと一致するようにメタデータを更新します。

タスク	プロシジャ	説明
	METAOPERATE‡	メタデータサーバーで管理タスクを実行します。

* これらプロシジャの説明については、ご利用の動作環境に対応する SAS ドキュメントを参照してください。
** このプロシジャの説明については、*SAS Output Delivery System: ユーザーガイド*を参照してください。
*** このプロシジャの説明については、*SAS/ACCESS for Relational Databases: Reference* を参照してください。
† このプロシジャの説明については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。
†† このプロシジャの説明については、*SAS/ACCESS Interface to PC Files: Reference* を参照してください。
††† このプロシジャの説明については、*Base SAS ガイド(Information Map)*を参照してください。
‡ このプロシジャの説明については、*SAS Language Interfaces to Metadata* を参照してください。

Base SAS プロシジャの簡単な説明

APPEND プロシジャ (p. 93)
1 つの SAS データセットから別の SAS データセットの最後にオブザベーションを追加します。

AUTHLIB プロシジャ (p. 102)
これは、対応するメタデータ保護テーブルオブジェクトに関連付けられた物理ライブラリであるメタデータバインド型ライブラリを管理します。メタデータバインドライブラリ内の各物理テーブルには、特定のメタデータオブジェクトを指す情報がヘッダーに含まれています。

CALENDAR プロシジャ (p. 168)
SAS データセットのデータを月次カレンダ形式で表示します。SAS データセットからのデータを月次カレンダ形式で表示します。PROC CALENDAR は、当月の休日の表示、タスクのスケジュール設定、およびさまざまな勤務スケジュールが含まれた複数のカレンダのデータの処理を行えます。

CATALOG プロシジャ (p. 249)
SAS カタログでエントリを管理します。PROC CATALOG は対話型の非ウィンドウプロシジャであり、カタログのコンテンツの表示、カタログ全体またはカタログの特定のエントリのコピー、カタログ内のエントリの名前変更、交換、削除を行えます。

CHART プロシジャ (p. 275)
縦棒グラフおよび横棒グラフ、ブロックチャート、円グラフ、スターチャートを作成します。縦棒グラフおよび横棒グラフ、ブロックチャート、円グラフ、スターチャートを作成します。これらのチャートには、単一の変数または複数の変数の値がわかりやすくビジュアル表示されます。PROC CHART を使用して値に関連する統計を表示することもできます。PROC CHART を使用して値に関連する統計を表示することもできます。

CIMPORT プロシジャ (p. 321)
CPORT プロシジャによって作成された移送ファイルを、動作環境に適した形式でその元の形式(SAS ライブラリ、カタログまたはデータセット)にリストアします。CPORT プロシジャに加えて、PROC CIMPORT でも SAS ライブラリ、カタログ、データセットを動作環境間で移動できます。SAS ライブラリ、データセット、カタログを、移送ファイルと呼ばれる特殊形式で書き込みます。CIMPORT プロシジャに加えて、PROC CPORT でも SAS ライブラリ、データセット、カタログを動作環境間で移動できます。

COMPARE プロシジャ (p. 340)
2 つの SAS データセットのコンテンツを比較します。PROC COMPARE を使用して、単一のデータセット内の異なる変数の値を比較することもできます。PROC COMPARE は、実行する比較に関するさまざまなレポートを作成します。

CONTENTS プロシジャ (p. 401)
SAS ライブラリの 1 つ以上のファイルのコンテンツの説明を印刷します。

CONVERT プロシジャ
BMDP システムファイル、OSIRIS システムファイル、SPSS ポータブルファイルを SAS データセットに変換します。詳細については、動作環境に対応する SAS のドキュメントを参照してください。詳細については、動作環境に対応する SAS のドキュメントを参照してください。

COPY プロシジャ (p. 409)
SAS ライブラリ全体またはライブラリの特定のメンバをコピーします。特定の種類のライブラリメンバに処理を限定できます。

CORR プロシジャ
これらの変数に対する変数と記述統計の間の Pearson 積率相関係数と重み付けされた積率相関係数を比較します。また、PROC CORR は 3 つのノンパラメトリックな連関の指標(Spearman の順位相関係数、Kendall の tau-b、Hoeffding の従属性の指標である D 統計)、偏相関係数(Pearson の偏相関係数、Spearman の偏順位相関係数、Kendall の偏 tau-b)、および Cronbach のアルファ係数を計算します。詳細については、*Base SAS Procedures Guide:*の*統計プロシジャ*に関する情報を参照してください。

CPORT プロシジャ (p. 417)
SAS ライブラリ、データセット、カタログを、移送ファイルと呼ばれる特殊形式で書き込みます。CIMPORT プロシジャに加えて、PROC CPORT でも SAS ライブラリ、データセット、カタログを動作環境間で移動できます。

CV2VIEW プロシジャ
SAS/ACCESS ビューディスクリプタを PROC SQL ビューに変換します。SAS システム 9 以降、PROC SQL ビューはプラットフォーム非依存であり、LIBNAME ステートメントを使用できるようになったため、SAS/ACCESS ビューディスクリプタを PROC SQL ビューに変換することをお勧めします。詳細については、*SAS/ACCESS for Relational Databases: Reference* を参照してください。

DATASETS プロシジャ (p. 436)
SAS ファイルと SAS 世代グループをリスト、コピー、名前変更、削除し、インデックスを管理して、SAS データセットを SAS ライブラリに追加します。このプロシジャによって、APPEND、CONTENTS、COPY プロシジャのすべての機能が提供されます。データセット内の変数を修正することもできます。ラベルやパスワードなどのデータセット属性を管理したり、整合性制約を作成および削除したりすることもできます。

DBCSTAB プロシジャ
SAS がサポートするダブルバイト文字セットの変換テーブルを作成します。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

DELETE プロシジャ (p. 639)
保存先のディスクまたはテープから SAS ファイルを削除します。

DISPLAY プロシジャ (p. 651)
SAS/AF アプリケーションを実行します。SAS/AF アプリケーションの作成の詳細については、*Guide to SAS/AF Applications Development* を参照してください。

DOCUMENT プロシジャ
ODS 文書に保存されるプロシジャ出力を操作します。PROC DOCUMENT を使用して、出力オブジェクトと階層を参照および編集したり、サポートされている ODS

出力形式でそれらを再生したりすることができます。詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

DS2 プロシジャ (p. 655)
Base SAS セッションの DS2 言語ステートメントをサブミットできます。

EXPORT プロシジャ (p. 675)
SAS データセットからデータを読み込み、外部データソースに書き込みます。

FCMP プロシジャ (p. 694)
SAS 関数およびサブルーチンを作成、検定および保存してから、他の SAS プロシジャで使用できます。PROC FCMP では、DATA ステップステートメントのわずかな変更は受け入れられます。PROC FCMP で処理される関数およびサブルーチンで、SAS プログラミング言語のほとんどの機能を使用できます。

FEDSQL プロシジャ (p. 797)
Base SAS セッションから FedSQL 言語ステートメントをサブミットできます。

FONTREG プロシジャ (p. 817)
SAS レジストリにシステムフォントを追加します。

FORMAT プロシジャ (p. 834)
文字変数または数値変数の、ユーザ定義の入力形式と出力形式を作成します。PROC FORMAT はフォーマットライブラリのコンテンツを印刷し、他の入力形式または出力形式に書き込む制御データセットを作成して、入力形式または出力形式を作成するための制御データセットを読み込みます。

FREQ プロシジャ
1 元から *n* 元の度数表を作成し、度数カウントをレポートします。PROC FREQ は、1 元から *n* 元表に対するカイ二乗検定と、2 元から *n* 元のクロス集計表に対する関連性および一致の検定および統計値を計算できます。また、リスクと 2×2 表のリスクの差異、傾向検定、Cochran-Mantel-Haenszel 統計を計算できます。出力データセットを作成することもできます。詳細については、*Base SAS Procedures Guide: の統計プロシジャに関する情報を参照してください。統計プロシジャ。*

FSLIST procedure (p. 925)
外部ファイルのコンテンツを表示するか、テキストを外部ファイルから SAS テキストエディタにコピーします。

GROOVY プロシジャ (p. 937)
Java 仮想マシン(JVM)で、SAS コードによって Groovy コードを実行できるようにします。

HADOOP プロシジャ (p. 947)
SAS が Hadoop データに対して Apache Hadoop コードを実行できるようにします。

HTTP プロシジャ (p. 983)
ハイパーテキスト転送プロトコル(HTTP)要求を発行します。

IMPORT プロシジャ
データを外部データソースから読み込み、SAS データセットに書き込みます。

INFOMAPS プロシジャ (p. 1005)
SAS Information Map を作成または更新します。詳細については、*Base SAS ガイド (Information Map)*を参照してください。

JAVAINFO プロシジャ (p. 1025)
SAS が使用している Java 環境に関する診断情報を通知します。診断情報を使用して、SAS Java 環境が正しく設定されていることを確認できます。また、SAS テクニカルサポートに問題を報告するときにも診断情報が役立ちます。

JSON プロシジャ (p. 1027)
データを SAS データセットから読み込み、JSON 表現で外部データソースに書き込みます。

MEANS プロシジャ (p. 1092)
すべてのオブザベーションにわたる変数とオブザベーショングループ内の変数に対し記述統計量を計算します。すべてのオブザベーションに対して、数値変数とオブザベーショングループ内の数値変数に対する記述統計を計算します。特定の統計を含み、オブザベーショングループに対する最小値と最大値を特定する出力データセットを作成することもできます。

METADATA プロシジャ
XML 文字列形式のメソッド呼び出しを SAS Metadata Server に送信します。詳細については、*SAS Language Interfaces to Metadata* を参照してください。

METALIB プロシジャ
ライブラリ内の表と一致するように SAS メタデータレポジトリのメタデータを更新します。詳細については、*SAS Language Interfaces to Metadata* を参照してください。

METAOPERATE プロシジャ
メタデータサーバーで管理タスクを実行します。詳細については、*SAS Language Interfaces to Metadata* を参照してください。

MIGRATE プロシジャ (p. 1176)
SAS ライブラリのメンバを SAS の最新リリースに移行します。SAS ライブラリのメンバを SAS の最新リリースに移行します。移行は、同じエンジンファミリー内で行う必要があります。たとえば、V6、V7、または V8 は V9 に移行できますが、V6TAPE は V9TAPE に移行する必要があります。

OPTIONS プロシジャ (p. 1197)
すべての SAS システムオプションの現在の値をリストします。

OPTLOAD プロシジャ (p. 1213)
SAS システムオプション設定を SAS レジストリまたは SAS データセットから読み取って、実行します。

OPTSAVE プロシジャ (p. 1217)
SAS システムオプション設定を SAS レジストリまたは SAS データセットに保存します。

PDS プロシジャ
分割されたデータセットのメンバをリスト、削除、名前変更します。詳細については、*z/OS 版 SAS* を参照してください。

PDSCOPY プロシジャ
分割されたデータセットをディスクからテープ、ディスクからディスク、テープからテープ、またはテープからディスクにコピーします。詳細については、*z/OS 版 SAS* を参照してください。

PLOT プロシジャ (p. 1224)
1 つの変数を別の変数にプロットする散布図を作成します。プロットの各ポイントの座標は、入力データセットの 1 つ以上のオブザベーションの 2 つの変数値に対応します。

PMENU プロシジャ (p. 1291)
DATA ステップウィンドウ、マクロウィンドウ、SAS/AF ウィンドウ、またはカスタマイズされたメニューを指定できる SAS アプリケーションで使用するメニューを定義します。

PRESENV プロシジャ (p. 1333)
ある SAS セッションから別の SAS セッションにかけて、SAS コードのすべてのグローバルステートメントとマクロ変数を保存します。このプロシジャが SAS セッション

の最後に呼び出されると、グローバルステートメントとマクロ変数はすべて、1 つのファイルに書き込まれます。

PRINT プロシジャ (p. 1338)
すべての変数または一部の変数を使用して、SAS データセットのオブザベーションを印刷します。PROC PRINT は、数値変数の合計と小計を印刷することもできます。

PRINTTO プロシジャ (p. 1427)
SAS プロシジャ出力および SAS ログに対する出力先を定義します。

PROTO プロシジャ (p. 1449)
C または C++プログラミング言語で記述される外部関数をバッチモードで登録できます。これらの関数は、SAS のほかに、C-language 構造と型で使用できます。これらの関数は PROC PROTO に登録された後、FCMP プロシジャで宣言される SAS 関数またはサブルーチンから呼び出すことができます。登録後は、COMPILE プロシジャで宣言される SAS 関数またはサブルーチン、あるいはメソッドブロックから呼び出すこともできます。

PRTDEF プロシジャ (p. 1473)
個々の SAS ユーザーまたはすべての SAS ユーザーに対するプリンタ定義を作成します。

PRTEXP プロシジャ (p. 1489)
プリンタ定義属性を簡単に複製、修正できるように SAS データセットにエクスポートします。

PWENCODE プロシジャ (p. 1495)
SAS プログラムで使用するパスワードをエンコードします。

QDEVICE プロシジャ (p. 1504)
グラフィックデバイスとユニバーサルプリンタに関するレポートを作成します。

RANK プロシジャ (p. 1543)
SAS データセットのオブザベーション全体に対して、1 つ以上の数値変数の順位を計算します。ランクは新しい SAS データセットに書き込まれます。または、PROC RANK は正規スコアまたはその他のランクスコアを作成します。ランクは新しい SAS データセットに書き込まれます。または、PROC RANK は正規スコアまたはその他のランクスコアを作成します。

REGISTRY プロシジャ (p. 1563)
レジストリ情報を SAS レジストリの USER 部分にインポートします。

RELEASE プロシジャ
z/OS 環境のディスクデータセットの最後の未使用スペースを解放します。詳細については、*z/OS 版 SAS* を参照してください。

REPORT プロシジャ (p. 1580)
PRINT プロシジャ、MEANS プロシジャ、TABULATE プロシジャの機能と DATA ステップの機能を、詳細レポートと要約レポートの両方を作成できる 1 つのレポート作成ツールに組み合わせます。

SCAPROC プロシジャ (p. 1761)
SAS コードアナライザを実装します。これは、実行中に SAS ジョブからの入力、出力、マクロシンボルの使用に関する情報を取得します。

SOAP プロシジャ (p. 1771)
ファイル参照を持つファイルから XML 入力を読み取り、ファイル参照を持つ別のファイルに XML 出力を書き込みます。

SORT プロシジャ (p. 1785)
SAS データセットを 1 つ以上の変数によって並べ替えます。PROC SORT は、結果として得られた並べ替え済みのオブザベーションを新しい SAS データセットに保存するか、元のデータセットを置き換えます。

SOURCE プロシジャ
ソースライブラリデータセットを簡単にバックアップして処理できます。詳細については、動作環境に対応する SAS のドキュメントを参照してください。

SQL プロシジャ
SAS で使用する SQL のサブセットを実装します。SQL は標準化され、広く使用されている言語であり、SAS データセット、SQL ビュー、DBMS テーブルに加え、これらのテーブルに基づくビューのデータを取得して更新します。PROC SQL は、テーブル、ビュー、要約、統計、レポートを作成し、並べ替えや連結などのユーティリティ機能を実行することもできます。詳細については、*SAS SQL プロシジャユーザーガイド*を参照してください。

STANDARD プロシジャ (p. 1833)
SAS データセットの一部またはすべての変数を所定の平均および標準偏差に標準化し、標準化された値を含む新しい SAS データセットを生成します。

STREAM (p. 1849)
SAS マクロ仕様を含めることができる任意のテキストで構成された入力ストリームを処理できます。マクロコードを拡張してファイルに保存できます。マクロコードを拡張してファイルに保存できます。

SUMMARY プロシジャ (p. 1861)
SAS データセットのすべてのオブザベーションにわたる変数とオブザベーショングループ内の変数に対して記述統計量を計算し、結果を新しい SAS データセットに出力します。

TABULATE プロシジャ (p. 1866)
記述統計量を表形式で表示します。各表のセルの値が、表のページ、行、列を定義する変数および統計値から計算されます。各セルに関連付けられている統計値がそのカテゴリのすべてのオブザベーションで得られた値で計算されます。結果を SAS データセットに書き込むことができます。

TAPECOPY プロシジャ
テープボリューム全体またはファイル全体を 1 つ以上のテープボリュームから 1 つの出力テープボリュームにコピーします。詳細については、*z/OS 版 SAS* を参照してください。

TAPELABEL プロシジャ
z/OS 環境の IBM 標準ラベル付きテープボリュームのラベル情報をリストします。詳細について は、*z/OS 版 SAS* を参照してください。

TEMPLATE プロシジャ
SAS ジョブ全体に対する ODS 出力、または単一の ODS 出力オブジェクトに対する ODS 出力をカスタマイズします。詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

TIMEPLOT プロシジャ (p. 2013)
一定期間の 1 つ以上の変数のプロットを生成します。

TRANSPOSE プロシジャ (p. 2039)
オブザベーションを変更するデータセットを変数に、またはその逆に置き換えます。

TRANTAB プロシジャ
カスタマイズされた変換テーブルを作成、編集、表示します。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

UNIVARIATE プロシジャ
記述統計(分位値を含む)、信頼区間、および数値変数のロバスト推定を計算します。数値変数の分布に関する詳細を提供します。これには、正規性の検定、分布を示すプロット、度数テーブルおよび位置の検定が含まれます。詳細については、*Base SAS Procedures Guide:の統計プロシジャに関する情報を参照してください。統計プロシジャ*。

XSL プロシジャ (p. 2065)
XML ドキュメントを HTML、テキスト、または他の種類の XML ドキュメントなどの別の出力形式に変換します。

2 章

Base SAS プロシジャの使用に必要な基本概念

言語の概念

一時 SAS データセットと永久 SAS データセット

SAS データセット名の指定

SAS データセットには、1 レベルの名前か、2 レベルの名前を指定できます。通常、一時 SAS データセットの名前は 1 レベルのみで、WORK ライブラリに保存されます。WORK ライブラリは SAS セッションの開始時に自動的に定義され、SAS セッションの終了時に自動的に削除されます。プロシジャでは、1 レベルの名前で指定される SAS データセットの読み取りと書き込みが WORK ライブラリに対して実行されるとみなされます。その他の場合を示すには、USER ライブラリを指定します。詳細については、“USER ライブラリ” (22 ページ)を参照してください。たとえば、次の PROC PRINT ステップは同じです。2 番目の PROC PRINT ステップでは、DEBATE データセットが WORK ライブラリ内にあるものとみなされます。

```
proc print data=work.debate;
```

```
run;

proc print data=debate;
run;
```

SAS システムオプション、WORK=、WORKINIT および WORKTERM は一時ライブラリと永久ライブラリの操作方法に影響します。詳細については、*SAS システムオプション: リファレンス*を参照してください。

通常、2 レベルの名前は、永久 SAS データセットを表します。2 レベルの名前の形式は、*libref.SAS-data-set* となります。*libref* は、一時的に SAS ライブラリと関連付けられている名前です。SAS ライブラリは、動作環境に SAS データセットを保存する外部保存場所です。LIBNAME ステートメントは、ライブラリ参照名と SAS ライブラリを関連付けます。次の PROC PRINT ステップでは、PROCLIB はライブラリ参照名で、EMP はライブラリ内の SAS データセットです。

```
libname proclib 'SAS-library';
proc print data=proclib.emp;
run;
```

USER ライブラリ

USER ライブラリを指定して、1 レベルの名前を永久 SAS データセットに使用できます。USER ライブラリを LIBNAME ステートメントまたは SAS システムオプション USER=を使用して割り当てることができます。USER ライブラリを指定した後、プロシジャでは 1 レベルの名前のデータセットが WORK ライブラリではなく USER ライブラリに存在するとみなされます。たとえば、次の PROC PRINT ステップでは、DEBATE が USER ライブラリに存在するとみなされます。

```
options user='SAS-library';
proc print data=debate;
run;
```

注: USER ライブラリを定義した場合、WORK.*SAS-data-set* を指定することで WORK ライブラリを続けて使用できます。

SAS システムオプション

一部の SAS システムオプション設定は、プロシジャ設定に影響します。次の SAS システムオプションは、SAS プロシジャと使用する可能性が最も高いオプションです。

- BYLINE | NOBYLINE
- DATE | NODATE
- DETAILS | NODETAILS
- FMTERR | NOFMTERR
- FORMCHAR=
- FORMDLIM=
- LABEL | NOLABEL
- LINESIZE=
- NUMBER | NONUMBER
- PAGENO=
- PAGESIZE=
- REPLACE | NOREPLACE

- SOURCE | NOSOURCE

SAS システムオプションの詳細説明については、*SAS システムオプション: リファレンス* を参照してください。

データセットオプション

データセットを読み込む、または出力データセットを作成するプロシジャのほとんどは、データセットオプションを受け入れます。SAS データセットオプションは、データセット指定の後にかっこ内に表示されます。次に例を示します。

```
proc print data=stocks(obs=25 pw=green);
```

個別のプロシジャの章には、適切な場合にデータセットオプションを使用できるアラームが含まれています。

SAS データセットオプションは、次のとおりです。

ALTER=	OBS=
BUFNO=	OBSBUF=
BUFSIZE=	OUTREP=
CNTLLEV=	POINTOBS=
COMPRESS=	PW=
DLDMGACTION=	PWREQ=
DROP=	READ=
ENCODING=	RENAME=
ENCRYPT=	REPEMPTY=
FILECLOSE=	REPLACE=
FIRSTOBS=	REUSE=
GENMAX=	SORTEDBY=
GENNUM=	SPILL=
IDXNAME=	TOBSNO=
IDXWHERE=	TYPE=
IN=	WHERE=
INDEX=	WHEREUP=
KEEP=	WRITE=

LABEL=

SAS データセットオプションの詳細説明については、*SAS データセットオプション: リファレンス*を参照してください。

グローバルステートメント

これらのグローバルステートメントは、DATALINES、CARDS、PARMCARDS ステートメントの後を除く、SAS プログラムで使用できます。

comment	ODS
DM	OPTIONS
ENDSAS	PAGE
FILENAME	RUN
FOOTNOTE	%RUN
%INCLUDE	SASFILE
LIBNAME	SKIP
%LIST	TITLE
LOCK	X

前述のステートメントで ODS ステートメント以外のものについては、*SAS ステートメント: リファレンス*を参照してください。ODS ステートメントについては、“Output Delivery System (ODS)” (66 ページ)および *SAS Output Delivery System: ユーザーガイド*を参照してください。

AES 暗号化

SAS では、高度暗号化標準(AES)で作成されたファイルへのアクセスがサポートされています。AES 暗号化は Base SAS の一部です。これはどの SAS プロシジャにも特化していません。

AES 暗号化で作成されたデータセットにアクセスするには、暗号化キー値に ENCRYPTKEY=オプションを付与してください。AES 保護データセットへのアクセス時に ENCRYPTKEY=キー値を省略すると、ダイアログボックスが表示され、ENCRYPTKEY=キー値の追加を求められます。詳細については、“AES Encryption” (*SAS Language Reference: Concepts*)および “ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)を参照してください。

プロシジャの概念

入力データセット

Base SAS プロシジャの多くには、入力 SAS データセットが必要です。この例のように、プロシジャステートメントで DATA=オプションを使用して、入力 SAS データセットを指定します。

```
proc print data=emp;
```

DATA=オプションを省略すると、プロシジャでは SAS システムオプション _LAST_=の値が使用されます。_LAST_=のデフォルトは、現在の SAS ジョブまたはセッションで作成された最新の SAS データセットです。_LAST_=の詳細説明については、*SAS データセットオプション: リファレンス*を参照してください。

Base SAS プロシジャのスレッド処理

スレッド処理で、複数の実行可能なコードを同時に実行できます。いくつかの SAS/STAT プロシジャや高性能解析(HPA)プロシジャなど、多くの SAS プロシジャがスレッド処理をサポートしています。ただし、すべての SAS プロシジャがスレッド処理をサポートしているわけではありません。スレッド処理をサポートしているかどうかを調べるには、使用中のプロシジャに関するドキュメントを参照してください。

次の Base SAS プロシジャはスレッド処理をサポートしています。

- 37 章, “MEANS プロシジャ,” (1092 ページ)
- 55 章, “REPORT プロシジャ,” (1580 ページ)
- 59 章, “SORT プロシジャ” (1785 ページ)
- 63 章, “SUMMARY プロシジャ,” (1861 ページ)
- “SQL” (*SAS SQL Procedure User's Guide*)
- 64 章, “TABULATE プロシジャ,” (1866 ページ)

関連項目:

システムオプション

- “CPUCOUNT= System Option” (*SAS System Options: Reference*)
- “THREADS System Option” (*SAS System Options: Reference*)

その他のドキュメント

- “Support for Parallel Processing” (*SAS Language Reference: Concepts*)
- *SAS Scalable Performance Data Server:User's Guide*

データ値の順序の制御

データ値の並べ替え

プロシジャでは、データ値の並べ替えスキームが適用され、特定の条件がプロシジャでのデータの並べ替えに影響します。

- 動作環境固有の照合順序
- BY ステートメント
- 単一の分類変数
- 複数の分類変数
- 出力形式
- ORDER=オプション
- その他の並べ替えオプション

データの並べ替えの例では、Sasuser.Houses データセットを使用します。

```
data sasuser.houses;
input style $ 1-9 sqfeet 10-13 bedrooms 15 baths 17-19 street $ 21-36 price 38-44;
datalines;
RANCH    1250 2 1.0 Sheppart Avenue   64000
SPLIT    1190 1 1.0 Rand Street       65850
CONDO    1400 2 1.5 Market Street     80050
TWOSTORY 1810 4 3.0 Garris Street    107250
RANCH    1500 3 3.0 Kemble Avenue     86650
SPLIT    1615 4 3.0 West Drive        94450
SPLIT    1305 3 1.5 Graham Avenue     73650
CONDO    1390 3 2.5 Hampshire Avenue  79650
TWOSTORY 1040 2 1.0 Sanders Road      55850
CONDO    2105 4 2.5 Jeans Avenue     127150
RANCH    1535 3 3.0 State Highway     89100
TWOSTORY 1240 2 1.0 Fairbanks Circle  69250
RANCH     720 1 1.0 Nicholson Drive   34550
TWOSTORY 1745 4 2.5 Highland Road    102950
CONDO    1860 2 2.0 Arcata Avenue    110700
run;

proc datasets lib=sasuser memtype=data;
   modify houses;
   attrib price format=dollar8.;
run;

proc print data=sasuser.houses;
run;
```

図 2.1 Sasuser.Houses データセット

The SAS System

Obs	style	sqfeet	bedrooms	baths	street	price
1	RANCH	1250	2	1.0	Sheppard Avenue	$64,000
2	SPLIT	1190	1	1.0	Rand Street	$65,850
3	CONDO	1400	2	1.5	Market Street	$80,050
4	TWOSTORY	1810	4	3.0	Garris Street	$107,250
5	RANCH	1500	3	3.0	Kemble Avenue	$86,650
6	SPLIT	1615	4	3.0	West Drive	$94,450
7	SPLIT	1305	3	1.5	Graham Avenue	$73,650
8	CONDO	1390	3	2.5	Hampshire Avenue	$79,350
9	TWOSTORY	1040	2	1.0	Sanders Road	$55,850
10	CONDO	2105	4	2.5	Jeans Avenue	$127,150
11	RANCH	1535	3	3.0	State Highway	$89,100
12	TWOSTORY	1240	2	1.0	Fairbanks Circle	$69,250
13	RANCH	720	1	1.0	Nicholson Drive	$34,550
14	TWOSTORY	1745	4	2.5	Highland Road	$102,950
15	CONDO	1860	2	2.0	Arcata Avenue	$110,700

動作環境によるデータの並べ替え

動作環境では、ASCII または EBCIDIC のどちらかの照合順序を使用してデータを並べ替えます。

- Windows および UNIX では、ASCII 照合順序を使用します。
- z/OS では、EBCIDIC 照合順序を使用します。

次のサンプルコードを Windows および z/OS 上で、システムごとにタイトルを変えて実行しました。

```
data order;
   input x $1.;
   datalines;
1
a
A
z
Z
\
;

proc print;
run;
```

次の表に、PRINT プロシジャ出力における ASCII と EBCIDIC の違いを示します。

ASCII 照合順序	EBCDIC 照合順序
1	a
A	z
Z	A
\	\
a	Z
z	1

詳細については、“Collating Sequence” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

BY ステートメントを使用したデータの並べ替え

1 つ以上の変数でデータを並べ替えるには、まず SORT プロシジャを使用する必要があります。データを並べ替えた後、プロシジャの BY ステートメントで同じ変数を使用し、データの並べ替え方を示します。プロシジャでは、BY 変数に基づいて、データを BY グループにサブセット化します。

```
proc sort data=sasuser.houses out=houses;
   by style;
run;
proc freq data=houses;
  by style;
  tables bedrooms /nopercent;
run;
```

ここでは、家のスタイルを表す Style 変数で並べ替えた最初の 2 つの BY グループを示します。

図 2.2 *2 つの BY グループのうち昇順で最初のグループ*

The FREQ Procedure

style=CONDO

bedrooms	Frequency	Cumulative Frequency
2	2	2
3	1	3
4	1	4

図 2.3 2 つの BY グループのうち昇順で 2 番目のグループ

The FREQ Procedure

style=RANCH

bedrooms	Frequency	Cumulative Frequency
1	1	1
2	1	2
3	2	4

デフォルトでは、SORT プロシジャは昇順で BY グループを並べ替えます。逆順に並べ替えるには、データセットを処理する PROC SORT およびそれに続くプロシジャの BY ステートメントで DESCENDING オプションを使用します。

```
proc sort data=sasuser.houses out=houses;
   by descending style;
run;
proc freq data=houses;
  by descending style;
  tables bedrooms /nopercent;
  run;
```

図 2.4 2 つの BY グループのうち降順で最初のグループ

The FREQ Procedure

style=TWOSTORY

bedrooms	Frequency	Cumulative Frequency
2	2	2
4	2	4

図 2.5 2 つの BY グループのうち降順で 2 番目のグループ

The FREQ Procedure

style=SPLIT

bedrooms	Frequency	Cumulative Frequency
1	1	1
3	1	2
4	1	3

BY ステートメントには、NOTSORTED という別のオプションもあります。NOTSORTED オプションは、値が同じグループになった BY ステートメントの変数をリストする場合で、昇順にも降順にも並べ替えない場合に役立ちます。

単一の分類変数を使用したデータの並べ替え

分類変数は、分析にとって意味のあるグループにデータを編成します。データセットまたは BY グループ内のすべてのデータ値がプロシジャによって読み込まれた後、値がグループ化され、並べ替えられます。

次の表に、分類変数を定義するプロシジャおよびそのステートメントの一部を示します。

プロシジャ	ステートメント
FREQ	TABLES
MEANS	CLASS
REPORT	DEFINE GROUP、ORDER または ACROSS オプションを使用
SUMMARY	CLASS
TABULATE	CLASS

ほとんどのプロシジャでは、出力形式または並べ替えオプションがない場合、単一の分類変数に対するデフォルトの並べ替えスキームは昇順です。

```
proc means data=sasuser.houses nway mean;
   class style;
   var sqfeet;
run;
```

図 2.6 単一の分類変数に対するデフォルトの昇順並べ替え

The MEANS Procedure

Analysis Variable : sqfeet Square footage		
Style of homes	N Obs	Mean
CONDO	4	1688.75
RANCH	4	1251.25
SPLIT	3	1370.00
TWOSTORY	4	1458.75

PROC REPORT のデフォルトのデータ並べ替え動作については、“ORDER=オプションを使用したデータの並べ替え” (41 ページ)を参照してください。

複数の分類変数を使用したデータの並べ替え

複数の分類変数を使用してデータのサブグループを作成する場合、上位のグループ化変数とみなされるのは 1 つの変数だけです。他の変数はすべて、直前の上位変数のサブグループの作成に使用されます。

次の表に、上位変数を指定するステートメントをプロシジャごとに示します。プロシジャステートメント内の最初の変数が上位変数です。表の例ではすべて、A が上位変数です。

プロシジャ	上位変数を指定するステートメント	例

FREQ	TABLES	`proc freq;`
		`  tables A * B * C * D;`
		`run;`

MEANS	CLASS	`proc means;`
		`  class A B C D;`
		`run;`

REPORT	COLUMN	`proc report;`
		`  column A B C D;`
		`  define C / group;`
		`  define A / group;`
		`  define B / group;`
		`  define D / group;`
		
		`run;`

SUMMARY	CLASS	`proc summary;`
		`  class A B C D;`
		`run;`

プロシジャ	上位変数を指定するステートメント	例

TABULATE	TABLE	`proc tabulate;`
		`  class D C B A;`
		`  table A, B, C * D;`
		`run;`

データセット全体または BY グループに対する各分類変数のマスタ順序を作成することにより、まず並べ替えスキームが決定されます。次に、マスタ順序が階層の各サブグループに適用されます。順序は、どの分類サブグループでも同じです。

次の例を考えます。

```
proc tabulate data=sasuser.houses format=3. noseps;
   class style bedrooms;
   table style*bedrooms, n / rts=23;
run;
```

Style および Bedrooms のマスタ順序は別々に決定されます。Style は上位分類変数です。Bedrooms は、Style の各値のサブグループを形成します。Bedrooms の順序は、Style の値ごとに再決定されるものではなく、サブグループを生成する前に決定されたマスタ順序が適用されます。

CLASS ステートメントで ORDER=オプションを指定しない場合、フォーマットされていない値を使用してデータが並べ替えられ、SORT プロシジャと同一の順序になります。Style のマスタ順序はアルファベット順の昇順で、Bedrooms のマスタ順序は数値の昇順です。

図 2.7 ORDER=オプションを使用しない場合のマスタ順序

The SAS System

		N
style	bedrooms	
CONDO	2	2
	3	1
	4	1
RANCH	1	1
	2	1
	3	2
SPLIT	1	1
	3	1
	4	1
TWOSTORY	2	2
	4	2

次の例では、順序で度数カウントも考慮する場合を考えます。

```
proc tabulate data=sasuser.houses format=3. noseps order=freq;
   class style bedrooms;
   table style*bedrooms, n / rts=23;
run;
```

PROC TABULATE では、複数の変数で度数カウントが同一の場合、プロシジャがデータを読み込む順序をマスタ順序として使用します。PROC TABULATE がデータセットを読み込むとき、Ranch の度数カウントは 4、Split が 3、Condo が 4、TwoStory が 4 です。このため、上位変数 Style のマスタ順序は、Ranch、Condo、TwoStory、Split の順です。

次に、bedrooms のマスタ順序が決定されます。bedrooms=2 の度数は 5、bedrooms=4 の度数は 4、bedrooms=3 の度数は 4、bedrooms=1 の度数は 2 です。bedrooms のマスタ順序は 2、4、3、1 です。出力は次のとおりです。

図 2.8 *マスタ順序を使用した場合の値の順序*

The SAS System

		N
Style of homes	Number of bedrooms	
RANCH	2	1
	3	2
	1	1
CONDO	2	2
	4	1
	3	1
TWOSTORY	2	2
	4	2
SPLIT	4	1
	3	1
	1	1

TABLE ステートメントに PRINTMISS オプションを追加して度数カウントを表示すると、順序がわかりやすくなります。

図 2.9 度数カウントによるマスタ順序

The SAS System

		N
Style of homes	Number of bedrooms	
RANCH	2	1
	4	.
	3	2
	1	1
CONDO	2	2
	4	1
	3	1
	1	.
TWOSTORY	2	2
	4	2
	3	.
	1	.
SPLIT	2	.
	4	1
	3	1
	1	1

出力形式を使用したデータの並べ替え:プロシジャのデフォルトの動作

FREQ、MEANS、SUMMARY、TABULATE の各プロシジャは、デフォルトで分類変数の値を昇順に並べ替えますが、その際に使用されるのは、フォーマットされた値ではなく、SAS データセット内の実際の値です。

REPORT プロシジャのデフォルトの順序は、順序変数、グループ変数または列変数のフォーマットされた値に基づく昇順です。

次の表では、分類変数に出力形式が適用される場合の、プロシジャのデフォルトの並べ替えスキームをまとめています。

プロシジャ	変数の種類	出力形式の指定	並べ替えの基準
FREQ	数値または文字	有りまたは無し	実際の値
MEANS	数値または文字	有りまたは無し	実際の値
REPORT	数値	有りまたは無し	フォーマットされた値 *

プロシジャ	変数の種類	出力形式の指定	並べ替えの基準
	文字	有り 無し	フォーマットされた値 実際の値
SUMMARY	数値または文字	有りまたは無し	実際の値
TABULATE	数値または文字	有りまたは無し	実際の値

* BEST*w.d* は、出力形式が割り当てられていない場合のデフォルトの出力形式です。

出力形式を使用したデータの並べ替え:実際の値の最小値を使用

分類変数の複数の値をフォーマットし、ORDER=INTERNAL を使用する場合、データの並べ替えには、フォーマットされた値ではなく、データセットの実際の値が使用されます。特定の分類水準に適用する値は、出力形式範囲に対してデータセット内で発生した実際の値のうち、最小値です。これは、印刷出力を生成している場合か、出力データセットを生成している場合かにかかわりません。

次の例では、GROUP B で発生する実際の値の最小値が 1 であるので、GROUP B が GROUP A の前に表示されます。GROUP A では実際の値の最小値が 3 です。GROUP A で発生する可能性のある実際の値の最小値は 0 ですが、0 はデータ内に存在しません。

```
proc format;
   value numf 0,3,4='GROUP A'
              1,2='GROUP B';

proc report data=sasuser.houses;
   column bedrooms;
   define bedrooms / group format=numf. order=internal;
run;
```

図 2.10 *出力形式範囲の最小値による並べ替え*

The SAS System

bedrooms
GROUP B
GROUP A

実際の値の最小値を使用する別の状況としては、互いに独立した複数のグループまたは範囲が出力形式に含まれる場合があります。

次の例では、DEPT=PET に対して、値 OTHER は最後に表示されますが、DEPT=PLANT に対しては最初に表示されます。これは、ID に対するマスタ並べ替え順序が、サブグループの作成前に決定されるからです。ID=199 と ID=299 には、同じ出力形式 OTHER が設定されています。フォーマットされた値 OTHER だけが、ID のマスタ順序の中に作成されたため、OTHER は DEPT=PLANT に対して最初に並べられます。

```
data sample;

   length dept $ 5;
```

```
   input dept id;
   datalines;
PET   100
PET   110
PET   120
PET   199
PLANT 200
PLANT 210
PLANT 220
PLANT 299
;

proc format;
   value idfmt
         100='CAT'
         110='DOG'
         120='FISH'
         199='OTHER'
         200='CACTUS'
         210='IVY'
         220='FERN'
         299='OTHER';

proc tabulate data=sample noseps;
   class dept id;
   table dept*id, n;
   format id idfmt.;
run;
```

図 2.11 実際の値の最小値による並べ替え

The SAS System

		N
dept	id	
PET	CAT	1
	DOG	1
	FISH	1
	OTHER	1
PLANT	OTHER	1
	CACTUS	1
	IVY	1
	FERN	1

出力形式を使用したデータの並べ替え:欠損値

データに欠損値が存在し、出力形式範囲に入っている場合、指定するプロシジャおよびオプションによって、出力への影響には次の 2 つの可能性があります。

1. 出力形式グループ全体が無効として処理されます。
2. 出力形式範囲全体が内部値の最小値となり、その結果、その出力形式範囲が並べ替えスキームで最初に表示される可能性があります。

分類変数を使用するプロシジャのほとんどに MISSING オプションがあります。このオプションを使用すると、欠損値を有効な分類水準とみなすかどうかを指定できます。PROC FREQ には、オプション MISSPRINT があります。このオプションでは、欠損分類水準を表示しますが、統計量の計算には使用しません。

MEANS、REPORT、SUMMARY、TABULATE の各プロシジャでは、出力形式を適用する前に、欠損値を有効または無効と分類します。MISSING オプションを指定しない場合、非欠損分類水準を欠損分類水準と同じグループにする出力形式範囲を使用すると、非欠損分類水準は有効として処理されますが、欠損分類水準は無効になります。

PROC FREQ では、欠損値を有効または無効に分類する前に、出力形式が適用されます。MISSING または MISSPRINT を使用しない場合、非欠損分類水準と欠損分類水準を出力形式範囲で同じグループにすると、非欠損分類水準も欠損分類水準も無効とみなされます。次の例では、PROC FREQ と PROC REPORT の影響の違いを示します。

```
proc format;
   value bedfmt 1='ONE' 2='TWO' other='OTHER';

data houses;
   set sasuser.houses end=last;
   output;
   if last then do;
      bedrooms=.;
      output;
   end;
   format bedrooms bedfmt.;
run;

proc print data=houses;
   title "PROC PRINT";
   title2 "WORK.HOUSES";
   var bedrooms;
   format bedrooms;
run;

proc freq data=houses;
   title1 "PROC FREQ";
   title2 "Without MISSING Specified";
   tables bedrooms / nocum nopercent;
run;

proc report data=houses;
   title1 "PROC REPORT";
   title2 "Without MISSING Specified";
   column beddrooms n;
   define bedrooms /group width=8;
run;
```

アウトプット 2.1 *PROC FREQ とPROC REPORT のデータ順序の比較*

PROC PRINT
WORK.HOUSES

Obs	bedrooms
1	2
2	1
3	2
4	4
5	3
6	4
7	3
8	3
9	2
10	4
11	3
12	2
13	1
14	4
15	2
16	.

PROC FREQ
Without MISSING Specified

The FREQ Procedure

Number of bedrooms	
bedrooms	Frequency
ONE	2
TWO	5
Frequency Missing = 9	

PROC REPORT
Without MISSING Specified

Number of bedrooms	n
ONE	2
OTHER	8
TWO	5

PROC FREQ では、フォーマットされた分類水準 OTHER が含まれないのに対し、PROC REPORT では含まれます。これは、PROC FREQ の場合、欠損値を無効と分類する前に BEDFMT.出力形式が Bedrooms 変数に適用されるからです。PROC REPORT では、出力形式を適用する前に欠損値が無効と分類されます。これにより、通常は欠損分類水準と同じグループにされる非欠損分類水準が、有効として処理されます。

度数カウントを観測することにより、これを確認できます。Work.Houses データセットには合計 16 個のオブザベーションがあります。PROC FREQ では、9 個の無効な分類水準がレポートされます。PROC REPORT では、1 つの分類水準だけが無効として処理されます。出力形式グループ内の最小の内部値が欠損値である場合、グループ全体が欠損として処理されます。欠損値は、数値変数でも文字変数でも、取り得る内部値の最小値とみなされるので、欠損値によって出力形式グループ全体が最小の内部値を持つことになります。内部値で並べ替える場合、欠損値は先頭に位置付けられます。次のコードでは、前の例に MISSING オプションを追加して、これを示します。

```
proc freq data=houses;
   title1 "PROC FREQ";
   title2 "With MISSING Specified";
   tables bedrooms / nocum nopercent missing ;
run;

proc report data=houses missing;
   title1 "PROC REPORT";
   title2 "With MISSING Specified";
   column bedrooms n;
   define bedrooms / group width=8;
run;
```

アウトプット 2.2 *MISSING を指定した場合の PROC FREQ および PROC REPORT*

PROC FREQ
With MISSING Specified

The FREQ Procedure

bedrooms	Frequency
OTHER	9
ONE	2
TWO	5

PROC REPORT
With MISSING Specified

bedrooms	n
ONE	2
OTHER	9
TWO	5

PROC FREQ の結果では、グループ OTHER が先頭に並べられています。PROC REPORT の結果では、OTHER が 2 番目に並べられています。デフォルトでは、PROC FREQ は内部値で並べ替え、PROC REPORT はフォーマットされた値で並べ替えます。度数カウントはどちらのプロシジャでも同じです。PROC FREQ では、OTHER が Work.Houses データセット内の欠損度数に対応します。PROC REPORT の場合、Work.Houses に含まれていなかった単一の欠損値ごとに度数が 1 ずつ増加します。

出力形式を使用したデータの並べ替え:BY 変数

BY ステートメント内の変数に出力形式が適用される場合、内部値が出力形式範囲で連続していない場合を除き、BY 変数の値はグループ化されます。

```
proc sort data=sasuser.houses out=houses;
   by bedrooms;
run;

proc format;
   value numf 3='GROUP A' 1,2,4='GROUP B';
run;

proc report data=houses;
   by bedrooms;
   format bedrooms numf.;
   column price;
run;
```

アウトプット 2.3 フォーマットされた BY 変数を使用した並べ替え

ORDER=オプションを使用したデータの並べ替え

ORDER=オプションを使用すると、プロシジャで使用する並べ替えスキームを選択できます。ORDER=オプションには、値 INTERNAL または UNFORMATTED、FORMATTED、DATA、FREQ を設定できます。

注: MEANS、REPORT、SUMMARY、TABULATE の各プロシジャでは、フォーマットされていないデータの並べ替えに INTERNAL と UNFORMATTED の両方を ORDER=オプションの値として受け入れます。

次のプロシジャでオプションを使用できます。

プロシジャ	SAS 製品	デフォルト値
CATMOD	SAS/STAT	INTERNAL
FREQ	Base SAS	INTERNAL
GLM	SAS/STAT	FORMATTED
LIFEREG	SAS/STAT	FORMATTED
LOGISTIC	SAS/STAT	FORMATTED
MEANS	Base SAS	UNFORMATTED
PROBIT	SAS/STAT	FORMATTED

プロシジャ	SAS 製品	デフォルト値
REPORT	Base SAS	FORMATTED
SUMMARY	Base SAS	UNFORMATTED
TABULATE	Base SAS	UNFORMATTED

次のトピックで ORDER=の各値について説明します。

ORDER=INTERNAL オプションを使用したデータの並べ替え

ORDER=INTERNAL を指定し、他の並べ替えオプションを指定しない場合、分類変数値は実際のフォーマットされていない値で昇順に並べられます。

```
proc format;
   value numf 1='ONE' 2='TWO' 3='THREE' 4='FOUR';
run;
proc report data=sasuser.houses;
   column bedrooms;
   define bedrooms /group format=numf8. order=internal;
run;
```

アウトプット 2.4 *ORDER=INTERNAL の場合の出力*

The SAS System

Number of bedrooms
ONE
TWO
THREE
FOUR

bedrooms の値は ONE、TWO、THREE、FOUR で、数値 1、2、3、4 に対応します。ORDER=INTERNAL を指定しないと、値はフォーマットされた値でアルファベット順に、FOUR、ONE、THREE、TWO と並べられます。

ORDER=FORMATTED オプションを使用したデータの並べ替え

ORDER=FORMATTED を指定し、他の並べ替えオプションを指定しない場合、分類変数値はフォーマットされた値でアルファベット順に並べられます。

```
proc format;
   value numf 1='ONE' 2='TWO' 3='THREE' 4='FOUR';
run;
proc freq data=sasuser.houses order=formatted;
   tables bedrooms / nopercent;
   format bedrooms numf8. ;
run;
```

アウトプット 2.5 *ORDER=FORMATTED の場合の出力*

The SAS System

The FREQ Procedure

Number of bedrooms		
bedrooms	Frequency	Cumulative Frequency
FOUR	4	4
ONE	2	6
THREE	4	10
TWO	5	15

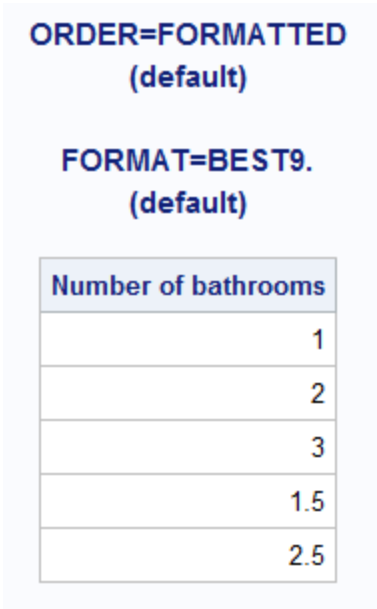
ORDER=FORMATTED を指定する場合、分類変数に関連付けられた出力形式がないと、その出力は内部値、すなわちフォーマットされていない値で並べられます。

特殊なケースでは、PROC REPORT のデフォルトの動作が ORDER=FORMATTED であるため、この設定によって問題が発生する場合があります。PROC REPORT では数値のデフォルト出力形式として BEST*w*.が使用されるためです。次に例を示します。

```
proc report data=sasuser.houses;
   title1 "ORDER=FORMATTED";
   title2 "(default)";
   title4 "FORMAT=BEST9.";
   title5 "(default)";
   column baths;
   define baths / group;
run;
```

出力形式が指定されていないので、ここでは BEST9.がデフォルト出力形式です。PROC REPORT のデフォルト設定は ORDER=FORMATTED であるため、値はフォーマットされた値を使用して並べ替えられます。これは文字比較によって複雑になり、誤解を招く結果になる場合があります。比較される値は、" 1"、" 2"、" 3"、"1.5"、"2.5"です。1 桁の値には先頭にブランクがあります。ブランク文字は数字やピリオド(.)の前に並べられるので、値"1"、"2"および"3"は 1.5 や 2.5 の前に並べられます。

アウトプット 2.6 値の先頭にブランクがある場合の PROC REPORT のデフォルトのフォーマット



ORDER=FORMATTED
(default)

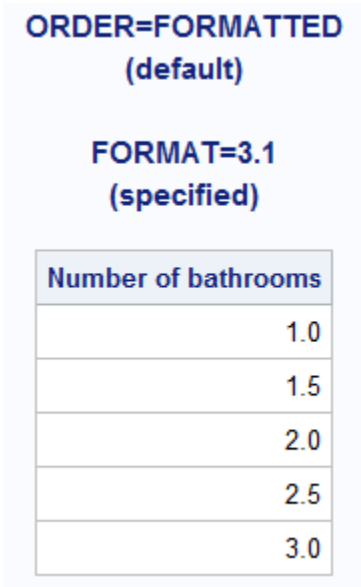
FORMAT=BEST9.
(default)

Number of bathrooms
1
2
3
1.5
2.5

この問題は次の出力で、3.1 出力形式を使用して先頭のブランクが比較に出現しないようにすることによって修正できます。

```
proc report data=sasuser.houses;
   title1 "ORDER=FORMATTED";
   title2 "(default)";
   title4 "FORMAT=3.1";
   title5 "(specified)";
   column baths;
   define baths / group format=3.1;
run;
```

アウトプット 2.7 値の先頭にブランクがある場合の PROC REPORT のフォーマット



ORDER=FORMATTED
(default)

FORMAT=3.1
(specified)

Number of bathrooms
1.0
1.5
2.0
2.5
3.0

また、この問題は次の出力でも修正できます。フォーマットされた値ではなく、内部値を使用して順序が決定されるからです。

```
proc report data=sasuser.houses nowd;
   title1 "ORDER =INTERNAL";
   title2 "(specified)";
   title4 "FORMAT=BEST9.";
   title5 "(default)";
   column baths;
   define baths / group order=internal;
run;
```

アウトプット 2.8 *内部値に基づいた PROC FORMAT の出力*

ORDER =INTERNAL
(specified)

FORMAT=BEST9.
(default)

Number of bathrooms
1
1.5
2
2.5
3

ORDER=DATA オプションを使用したデータの並べ替え

ORDER=DATA を指定すると、プロシジャによるデータの初期読み込み方法に従って順序が設定されます。ORDER=DATA は、BY ステートメントや複数の分類変数の使用により複雑になる場合があります。次に単純なケースを示します。

```
proc tabulate data=sasuser.houses order=data format=3. noseps;
   class style;
   table style, n;
run;
```

アウトプット 2.9 *ORDER=DATA を使用した PROC TABULATE*

	N
Style of homes	
RANCH	4
SPLIT	3
CONDO	4
TWOSTORY	4

Style の出力順序は、昇順でも降順でもありません。RANCH は、入力データセットで発生する最初の値なので、最初に表示されます。SPLIT は、2 番目に発生するので、2 番目に表示されます。以下同様です。BY ステートメントを使用すると、新しい BY グループごとに、新しいデータセットの処理のように先頭で順序がリセットされます。

複数の分類変数を使用する場合、データセット全体または BY グループで、分類変数ごとに独立に順序が決定されます。並べ替えスキームは、まず最上位の順序変数のマスタ順序を使用し、次に 2 番目、3 番目を使用して作成されます。次に例を示します。

```
proc tabulate data=sasuser.houses order=data format=3. noseps;
   class style bedrooms;
   table style*bedrooms, n;
run;
```

アウトプット 2.10 *2 つの分類変数を使用した PROC TABULATE*

		N
Style of homes	Number of bedrooms	
RANCH	2	1
	1	1
	3	2
SPLIT	1	1
	4	1
	3	1
CONDO	2	2
	4	1
	3	1
TWOSTORY	2	2
	4	2

比較すると、Style の順序は RANCH、SPLIT、CONDO、TWOSTORY です。Bedrooms の順序は 2、1、4、3 です。Bedrooms の 4 つの値をすべて持つ STYLE の値がないため、これは出力からすぐにはわかりません。値の順序は、変数ごとに独立に作成され、サブグループには依存しません。このことは、TABLE ステートメントに PRINTMISS オプションを追加するか、Sasuser.Houses データセットの Style と Bedrooms の順序を比較することによって確認できます。“複数の分類変数を使用したデータの並べ替え” (31 ページ)を参照してください。

ORDER=FREQ オプションを使用したデータの並べ替え

ORDER=FREQ は、分類変数をその値ごとの度数によって並べ替えることを指定します。PROC FREQ を除き、欠損分類水準は、非欠損分類水準と同じように度数カウントによって並べ替えられます。PROC FREQ では、欠損分類水準を、その度数カウントにかかわらず、常に先頭に表示します。PROC REPORT を除き、すべての Base SAS プロシジャは度数を降順で表示します。PROC REPORT は、デフォルトで、度数を昇順で表示します。PROC REPORT のオプション DESCENDING を ORDER=FREQ とともに使用すると、分類水準を度数カウントの降順で並べ替えます。

2 つの分類水準の度数が同一の場合、第 2 の並べ替えアルゴリズムが使用されます。PROC FREQ を除き、すべての Base SAS プロシジャでは、ORDER=DATA を第 2 の並べ替えメソッドとして使用します。

PROC FREQ で度数カウントの重複が発生し、ORDER=FREQ が指定されていた場合、PROC FREQ では第 2 の並べ替えメソッドとして ORDER=FORMATTED を使用します。PROC FREQ で出力形式が適用されていなかった場合、ORDER=INTERNAL メソッドを使用して同数の順序が決定されます。PROC REPORT を ORDER=FREQ および DESCENDING オプションとともに使用する場合、同数が発生すると、ORDER=DATA メソッドが使用されますが、データの発生とは逆順に分類水準が表示されます。

次の表に、ORDER=オプションを使用する Base SAS プロシジャの動作をまとめます。

プロシジャ	並べ替え方向	出力形式の指定	同数の場合の値の順序
FREQ	降順	いいえ	ORDER=INTERNAL
		はい	ORDER=FORMATTED
MEANS	降順	有りまたは無し	ORDER=DATA
REPORT	昇順	有りまたは無し	ORDER=DATA
	降順	有りまたは無し	ORDER=DATA *
SUMMARY	降順	有りまたは無し	ORDER=DATA
TABULATE	降順	有りまたは無し	ORDER=DATA

* DESCENDING オプションを指定すると、第 1 メソッド ORDER=FREQ と第 2 メソッド ORDER=DATA の両方で水準が降順に表示されます。

次に、ORDER=FREQ を TABULATE、FREQ および REPORT プロシジャで指定する場合の例を示します。まず、出力形式を指定しない場合の PROC TABULATE および PROC FREQ の出力を示します。

```
proc format;
   value bedfmt 1='ONE' 2='TWO' 3='THREE' 4='FOUR';

proc tabulate data=sasuser.houses order=freq noseps format=3.;
   title1 "PROC TABULATE";
   title2 "Without Format";
   class bedrooms;
   table bedrooms, n;
run;

proc freq data=sasuser.houses order=freq;
   title1 "PROC FREQ";
   title2 "Without Format";
   tables bedrooms / nocum nopercent;
run;
```

どちらの出力も、データ値を度数カウントの降順で表示します。

アウトプット 2.11 ORDER=FREQ の例

PROC TABULATE
Without Format

	N
Number of bedrooms	
2	5
4	4
3	4
1	2

PROC FREQ
Without Format

The FREQ Procedure

Number of bedrooms	
bedrooms	Frequency
2	5
3	4
4	4
1	2

PROC REPORT では、出力形式を指定しない場合、出力を昇順で表示します。

```
proc report data=sasuser.houses;
   title1 "PROC REPORT";
   title2 "Without Format";
   title3 "Without DESCENDING";
   column bedrooms n;
   define bedrooms / group order=freq;
run;
```

アウトプット 2.12　出力形式を指定しない場合の PROC REPORT

PROC REPORT
Without Format
Without DESCENDING

Number of bedrooms	n
1	2
4	4
3	4
2	5

PROC REPORT で出力を降順に表示するには、DEFINE ステートメントで DESCENDING を指定する必要があります。

```
proc report data=sasuser.houses;
   title1 "PROC REPORT";
   title2 "Without Format";
   title3 "With DESCENDING";
   column bedrooms n;
   define bedrooms / group order=freq descending;
run;
```

PROC REPORT
Without Format
With DESCENDING

Number of bedrooms	n
2	5
4	4
3	4
1	2

PROC TABULATE および PROC REPORT では、ORDER=DATA メソッドを使用して、同一の値を処理します。

```
proc tabulate data=sasuser.houses order=freq noseps format=3.;
   title1 "PROC TABULATE";
   title2 "With Format";
   class bedrooms;
   table bedrooms, n;
   format bedrooms bedfmt.;
run;
proc report data=sasuser.houses;
   title1 "PROC REPORT";
   title2 "With Format";
   title3 "Without DESCENDING";
```

```
   column bedrooms n;
   define bedrooms / group order=freq format=bedfmt8.;
run;
```

アウトプット 2.13 *降順の出力例*

PROC TABULATE
With Format

	N
Number of bedrooms	
TWO	5
FOUR	4
THREE	4
ONE	2

PROC REPORT
With Format
Without DESCENDING

Number of bedrooms	n
ONE	2
FOUR	4
THREE	4
TWO	5

PROC FREQ では、出力形式が指定されている場合、値が同一のときには ORDER=FORMATTED を使用します。

```
proc freq data=sasuser.houses order=freq;
   title1 "PROC FREQ";
   title2 "With Format";
   tables bedrooms / nocum nopercent;
   format bedrooms bedfmt.;
run;
```

アウトプット 2.14 *出力形式を指定して PROC FREQ を使用する場合の降順の出力*

PROC FREQ
With Format

The FREQ Procedure

Number of bedrooms	
bedrooms	Frequency
TWO	5
FOUR	4
THREE	4
ONE	2

出力形式を指定しない場合、PROC FREQ では ORDER=INTERNAL が使用されます。

PROC REPORT では、DESCENDING が適用される場合、データ値が度数カウントの降順で表示され、度数が同一のデータ値は発生順の逆順で表示されます。つまり、同一の値の処理には ORDER=DATA が使用され、データ値は逆順で表示されます。

アウトプット 2.15 *出力形式を指定して DESCENDING オプション付きで PROC REPORT を使用する場合の出力*

PROC REPORT
With Format
With DESCENDING

Number of bedrooms	n
TWO	5
FOUR	4
THREE	4
ONE	2

ORDER=DATA を使用するので、BY ステートメントを追加する場合、各 BY グループが別のデータセットであるかのように分類変数の順序が作成されます。
ORDER=FREQ で複数の分類変数を使用する場合、分類変数ごとに独立に順序が決定されます。その後、まず最上位の順序変数の順序を使用し、次に 2 番目、3 番目の順序変数を使用して、全体の並べ替えスキームが適用されます。

次の例では、Baths の度数を検討します。

```
proc tabulate data=sasuser.houses order=freq format=3. noseps;
   class baths;
   table baths, n;
run;
```

アウトプット 2.16 PROC TABULATE の使用による Baths の度数

	N
Number of bathrooms	
1	5
3	4
2.5	3
1.5	2
2	1

Bedrooms を最上位の順序変数にし、Baths を次に上位の順序変数にすると、まず Bedrooms の値で度数の降順に行が並べ替えられ、次に Baths で並べ替えられると予測できます。

```
proc tabulate data=sasuser.houses order=freq format=3. noseps;
   class baths bedrooms;
   table bedrooms*baths, n;
run;
```

N 統計量では、組み合わせの度数は降順に表示されません。まず BEDROOMS の度数の降順で順序が設定され、次に BATHS で設定されます。このことは、各変数の順序を、アウトプット 2.11 (48 ページ)で使用した順序と比較することにより確認できます。BEDROOMS のマスタ並べ替え順序は 2、4、3、1 であり、BATHS のマスタ並べ替え順序は 1、3、2.5、1.5、2 です

アウトプット 2.17 Bedrooms を上位順序変数にした PROC TABULATE

		N
Number of bedrooms	Number of bathrooms	
2	1	3
	1.5	1
	2	1
4	3	2
	2.5	2
3	3	2
	2.5	1
	1.5	1
1	1	2

RUN グループ処理

RUN グループ処理を使用して、プロシジャを終了せずに PROC ステップを RUN ステートメントでサブミットできます。別の PROC ステートメントを発行せずにプロシジャを続

けて使用できます。プロシジャを終了するには、RUN CANCEL ステートメントか、QUIT ステートメントを使用します。RUN グループ処理は、複数の Base SAS プロシジャでサポートされています。

CATALOG	DS2	PLOT	TRANTAB
DATASETS	FEDSQL	PMENU	

詳細については、個別のプロシジャのセクションを参照してください。

注: PROC SQL は、各クエリを自動的に実行します。RUN ステートメントも RUN CANCEL ステートメントも影響しません。

BY グループの情報を含むタイトルの作成

BY グループ処理

BY グループ処理では、BY ステートメントを使用して、変数の値別に並べ替え、グループ化、インデックス付けが行われるオブザベーションを処理します。デフォルトで、プロシジャステップで BY グループ処理を使用すると、BY 行が各グループを識別します。このセクションでは、カスタマイズされた BY 行として機能するタイトルを作成する方法を説明します。

デフォルトの BY 行を表示しない

BY グループ処理情報をタイトルに挿入する場合は、通常、デフォルトの BY 行を非表示にします。非表示にするには、SAS システムオプション NOBYLINE を使用します。

注: 次のプロシジャに対し BY グループ情報をタイトルに挿入する場合、NOBYLINE オプションを使用する必要があります。

- MEANS
- PRINT
- STANDARD
- SUMMARY

BY ステートメントを NOBYLINE オプションと使用する場合、これらのプロシジャは常に BY グループごとに新しいページを開始します。この動作により、複数の BY グループが単一のページに表示されなくなり、タイトルの情報がページのレポートと一致するようになります。

BY グループ情報をタイトルに挿入する

BY グループ情報をタイトルに挿入するための一般形式は、次のとおりです。

#BY-specification<.suffix>

BY-specification
次の指定のうちのいずれかになります。

BYVAL*n* | BYVAL(*BY-variable*)
指定された BY 変数の値をタイトルに置きます。BY 変数を次のオプションのうちいずれかで指定します。

n は、BY ステートメントの *n* 番目の BY 変数です。

BY-variable は、値をタイトルに挿入する BY 変数の名前です。

BYVAR*n* | BYVAR(*BY-variable*)
指定された BY 変数のラベルまたは名前(ラベルが存在しない場合)をタイトルに置きます。BY 変数を次のオプションのうちいずれかで指定します。

n は、BY ステートメントの *n* 番目の BY 変数です。

BY-variable は、名前をタイトルに挿入する BY 変数の名前です。

BYLINE
完全なデフォルトの BY 行をタイトルに挿入します。

suffix
タイトルに挿入する BY グループ情報の直後に置くテキストを提供します。BY グループ情報と接尾辞の間にスペースは表示されません。

例:タイトルに各 BY 変数の値を挿入する

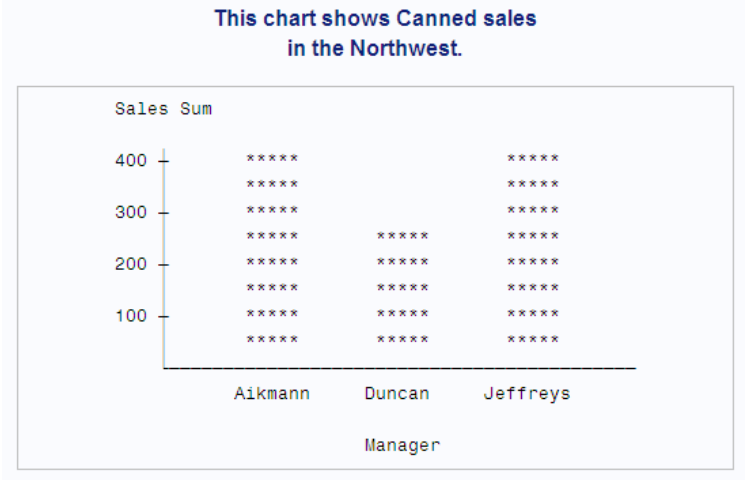
次の例では、次のアクションを説明します。

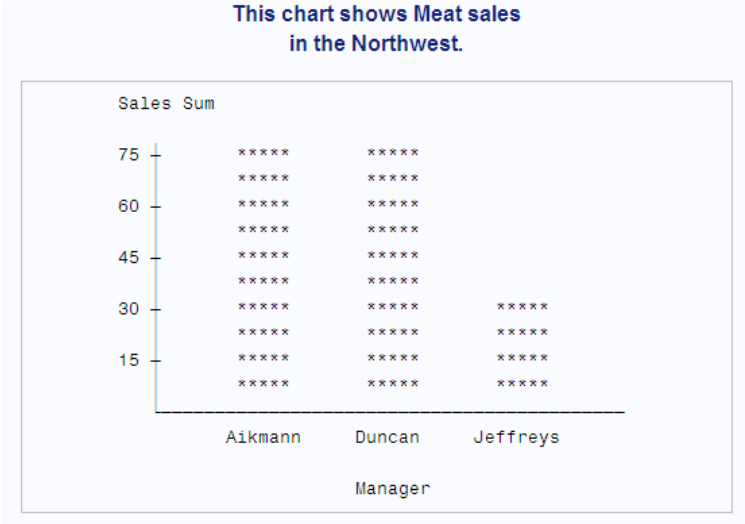
1. 4 つの地域からの店舗に関するデータを含むデータセット GROC を作成します。店舗にはそれぞれ 4 つの部門があります。データセットを作成する DATA ステップについては、"GROC" (2155 ページ)を参照してください。
2. Region および Department 別にデータを並べ替えます。
3. SAS システムオプション NOBYLINE を使用して、BY グループ処理で生成される出力に通常表示される BY 行を非表示にします。
4. PROC CHART を使用して、売上のチャートを Region および Department 別に作成します。最初の TITLE ステートメントで、#BYVAL2 は 2 番目の BY 変数、Department の値をタイトルに挿入します。2 番目の TITLE ステートメントで、#BYVAL(Region)は Region の値をタイトルに挿入します。Region の後の最初のピリオドは、接尾辞が続くことを示します。2 番目のピリオドは接尾辞です。
5. SAS システムオプション BYLINE を使用して、BY グループ処理によるデフォルトの BY 行の作成に返します。

```
data groc; 1
   input Region $9. Manager $ Department $ Sales;
   datalines;
Southeast    Hayes       Paper       250
Southeast    Hayes       Produce     100
Southeast    Hayes       Canned      120
Southeast    Hayes       Meat         80
...more lines of data...
Northeast    Fuller      Paper       200
Northeast    Fuller      Produce     300
Northeast    Fuller      Canned      420
Northeast    Fuller      Meat        125
;

proc sort data=groc;  2
   by region department;
run;
options nobyline nodate pageno=1
        linesize=64 pagesize=20;  3
proc chart data=groc;  4
   by region department;
   vbar manager / type=sum sumvar=sales;
   title1 'This chart shows #byval2 sales';
```

```
   title2 'in the #byval(region)..';
run;
options byline;  5
```

この部分出力には、カスタマイズされた BY 行を含む 2 つの BY グループが表示されます。





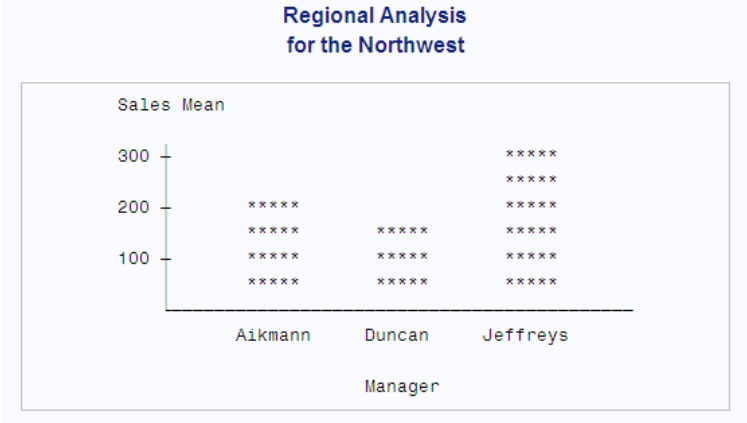
例:タイトルに BY 変数の名前を挿入する

この例では、タイトルに BY 変数の名前と値を挿入します。プログラムにより次のアクションが実行されます。

1. SAS システムオプション NOBYLINE を使用して、BY グループ処理で生成される出力に通常表示される BY 行を非表示にします。
2. PROC CHART を使用して売上のチャートを Region 別に作成します。最初の TITLE ステートメントで、#BYVAR(Region)は変数 Region の名前をタイトルに挿入します(Region にラベルがある場合、#BYVAR は名前の代わりにラベルを使用します)。接尾辞 `al` がラベルに追加されます。2 番目の TITLE ステートメントで、#BYVAL1 は最初の BY 変数、Region の値をタイトルに挿入します。
3. SAS システムオプション BYLINE を使用して、BY グループ処理によるデフォルトの BY 行の作成に返します。

```
options nobyline nodate pageno=1
        linesize=64 pagesize=20;  1
proc chart data=groc;  2
   by region;
   vbar manager / type=mean sumvar=sales;
   title1 '#byvar(region).al Analysis';
   title2 'for the #byval1';
run;
options byline;  3
```

この部分出力には、カスタマイズされた BY 行を含む 1 つの BY グループが表示されます。



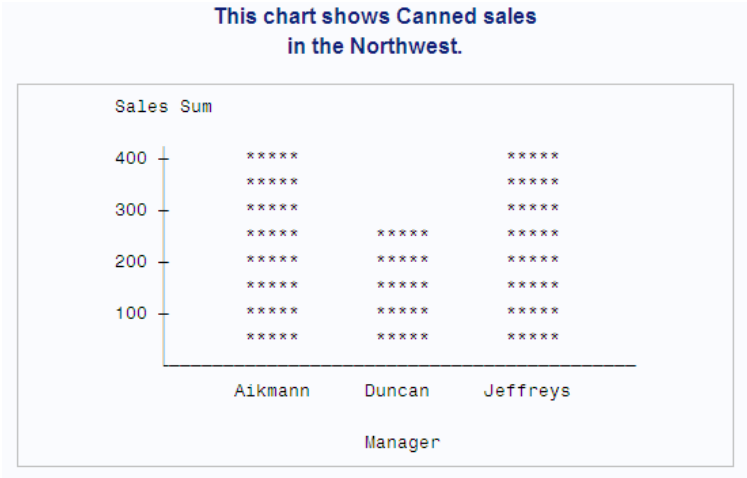
例:タイトルに完全な BY 行を挿入する

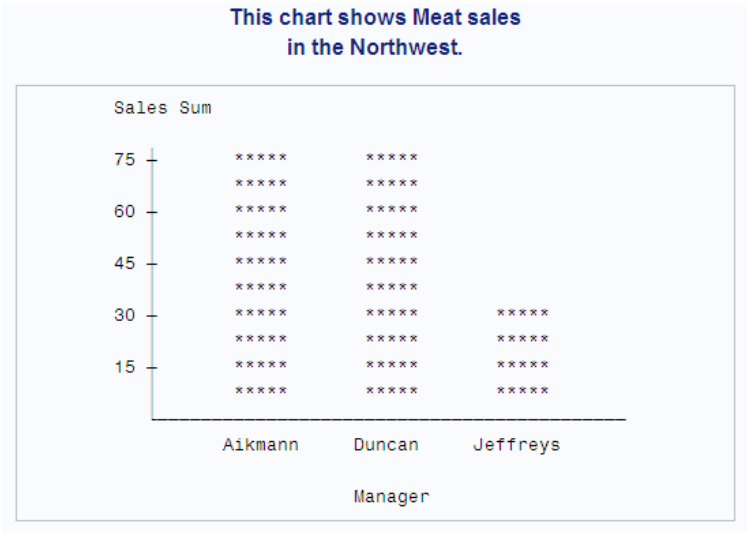
この例では、完全な BY 行をタイトルに挿入します。プログラムにより次のアクションが実行されます。

1. SAS システムオプション NOBYLINE を使用して、BY グループ処理で生成される出力に通常表示される BY 行を非表示にします。
2. PROC CHART を使用して、売上のチャートを Region および Department 別に作成します。TITLE ステートメントで、#BYLINE は完全な BY 行をタイトルに挿入します。
3. SAS システムオプション BYLINE を使用して、BY グループ処理によるデフォルトの BY 行の作成に返します。

```
options nobyline nodate pageno=1
        linesize=64 pagesize=20;  1
proc chart data=groc;  2
   by region department;
   vbar manager / type=sum sumvar=sales;
   title 'Information for #byline';
run;
options byline;  3
```

この部分出力には、カスタマイズされた BY 行を含む 2 つの BY グループが表示されます。





BY グループの指定のエラー処理

SAS では不正な#BYVAL、#BYVAR または#BYLINE 指定に対し、エラーメッセージまたは警告メッセージを発行しません。代わりに、その項目のテキストがタイトルの一部になります。

変数名リストを示すショートカット

プロシジャの複数のステートメントでは、複数の変数名が使用できます。各変数名を指定する代わりに、これらのショートカット表記を使用できます。

表記	意味
x1-x*n*	変数 X1 から X*n* を指定します。番号は連続している必要があります。
x:	文字 X で始まるすべての変数を指定します。
x--a	X と A の間のすべての変数を指定します。この表記では、データセットの変数の位置を使用します。

表記	意味
x-numeric-a	X と A の間のすべての数値変数を指定します。この表記では、データセットの変数の位置を使用します。
x-character-a	X と A の間のすべての文字変数を指定します。この表記では、データセットの変数の位置を使用します。
numeric	すべての数値変数を指定します。
character	すべての文字変数を指定します。
all	すべての変数を指定します。

注: ショートカットを使用して、PROC DATASETS の INDEX CREATE ステートメントで変数名をリストすることはできません。

詳細については、*SAS 言語リファレンス: 解説編*を参照してください。

フォーマットされた値

フォーマットされた値の使用

通常、変数値を印刷またはグループ化する場合、Base SAS プロシジャはフォーマットされた値を使用します。このセクションには、Base SAS プロシジャによるフォーマットされた値の使用方法例が含まれています。

例: データセットのフォーマットされた値を印刷する

次の例では、データセット PROCLIB.PAYROLL のフォーマットされた値を印刷します。(このデータセットを作成する DATA ステップの詳細については、"PROCLIB.PAYROLL" (2163 ページ)を参照してください)。PROCLIB.PAYROLL で、変数 Jobcode は従業員のジョブとレベルを示します。たとえば、**TA1** は、従業員がチケット取扱店の開始レベルであることを示します。

```
options nodate pageno=1
        linesize=64 pagesize=40;
proc print data=proclib.payroll(obs=10)
           noobs;
   title  'PROCLIB.PAYROLL';
   title2 'First 10 Observations Only';
run;
```

次の例は、PROCLIB.PAYROLL の印刷の一部です。

PROCLIB.PAYROLL
First 10 Observations Only

IdNumber	Gender	Jobcode	Salary	Birth	Hired
1919	M	TA2	34376	12SEP60	04JUN87
1653	F	ME2	35108	15OCT64	09AUG90
1400	M	ME1	29769	05NOV67	16OCT90
1350	F	FA3	32886	31AUG65	29JUL90
1401	M	TA3	38822	13DEC50	17NOV85
1499	M	ME3	43025	26APR54	07JUN80
1101	M	SCP	18723	06JUN62	01OCT90
1333	M	PT2	88606	30MAR61	10FEB81
1402	M	TA2	32615	17JAN63	02DEC90
1479	F	TA3	38785	22DEC68	05OCT89

次の PROC FORMAT ステップでは出力形式$JOBFMT.を作成します。これによりジョブごとに詳細な名前が割り当てられます。

```
proc format;
    value $jobfmt
          'FA1'='Flight Attendant Trainee'
          'FA2'='Junior Flight Attendant'
          'FA3'='Senior Flight Attendant'
          'ME1'='Mechanic Trainee'
          'ME2'='Junior Mechanic'
          'ME3'='Senior Mechanic'
          'PT1'='Pilot Trainee'
          'PT2'='Junior Pilot'
          'PT3'='Senior Pilot'
          'TA1'='Ticket Agent Trainee'
          'TA2'='Junior Ticket Agent'
          'TA3'='Senior Ticket Agent'
          'NA1'='Junior Navigator'
          'NA2'='Senior Navigator'
          'BCK'='Baggage Checker'
          'SCP'='Skycap';
run;
```

この PROC MEANS ステップの FORMAT ステートメントは、一時的に$JOBFMT.出力形式と変数 Jobcode を関連付けます。

```
options nodate pageno=1
        linesize=64 pagesize=60;
proc means data=proclib.payroll mean max;
   class jobcode;
   var salary;
   format jobcode $jobfmt.;
   title 'Summary Statistics for';
   title2 'Each Job Code';
run;
```

PROC MEANS は次の出力を生成します。$JOBFMT.出力形式が使用されます。

Summary Statistics for Each Job Code

The MEANS Procedure

Analysis Variable : Salary			
Jobcode	N Obs	Mean	Maximum
Baggage Checker	9	25794.22	26896.00
Flight Attendant Trainee	11	23039.36	23979.00
Junior Flight Attendant	16	27986.88	28978.00
Senior Flight Attendant	7	32933.86	33419.00
Mechanic Trainee	8	28500.25	29769.00
Junior Mechanic	14	35576.86	36925.00
Senior Mechanic	7	42410.71	43900.00
Junior Navigator	5	42032.20	43433.00
Senior Navigator	3	52383.00	53798.00
Pilot Trainee	8	67908.00	71349.00
Junior Pilot	10	87925.20	91908.00
Senior Pilot	2	10504.50	11379.00
Skycap	7	18308.86	18833.00
Ticket Agent Trainee	9	27721.33	28880.00
Junior Ticket Agent	20	33574.95	34803.00
Senior Ticket Agent	12	39679.58	40899.00

注: 出力形式が文字列であるため、数値変数の値が数値計算に必要な場合は数値変数用出力形式は無視されます。

例:フォーマットされたデータをグループ化/分類する

フォーマットされた変数を使用してデータのグループ化または分類を行う場合、プロシジャはフォーマットされた値を使用します。次の例では、出力形式$CODEFMT.を作成し、割り当てます。これは、ジョブコードごとにレベルを 1 つのカテゴリにグループ化します。PROC MEANS は、統計量を$CODEFMT.出力形式のグループに基づいて計算します。

```
proc format;
    value $codefmt
          'FA1','FA2','FA3'='Flight Attendant'
          'ME1','ME2','ME3'='Mechanic'
          'PT1','PT2','PT3'='Pilot'
          'TA1','TA2','TA3'='Ticket Agent'
                'NA1','NA2'='Navigator'
                      'BCK'='Baggage Checker'
                      'SCP'='Skycap';
run;
```

```
options nodate pageno=1
        linesize=64 pagesize=40;
proc means data=proclib.payroll mean max;
   class jobcode;
   var salary;
   format jobcode $codefmt.;
   title 'Summary Statistics for Job Codes';
   title2 '(Using a Format that Groups the Job Codes)';
run;
```

PROC MEANS は次の出力を生成します。

Summary Statistics for Job Codes
(Using a Format that Groups the Job Codes)

The MEANS Procedure

Analysis Variable : Salary			
Jobcode	N Obs	Mean	Maximum
Baggage Checker	9	25794.22	26896.00
Flight Attendant	34	27404.71	33419.00
Mechanic	29	35274.24	43900.00
Navigator	8	45913.75	53798.00
Pilot	20	72176.25	91908.00
Skycap	7	18308.86	18833.00
Ticket Agent	41	34076.73	40899.00

例:出力形式と変数を一時的に関連付ける

出力形式と変数を一時的に関連付ける場合、FORMAT ステートメントを使用できます。たとえば、次の PROC PRINT ステップは DOLLAR8.出力形式と、この PROC PRINT ステップのみの期間に対する変数 Salary とを関連付けます。

```
options nodate pageno=1
        linesize=64 pagesize=40;
proc print data=proclib.payroll(obs=10)
           noobs;
   format salary dollar8.;
   title 'Temporarily Associating a Format';
   title2 'with the Variable Salary';
run;
```

PROC PRINT は次の出力を生成します。

Temporarily Associating a Format with the Variable Salary

IdNumber	Gender	Jobcode	Salary	Birth	Hired
1919	M	TA2	$34,376	12SEP60	04JUN87
1653	F	ME2	$35,108	15OCT64	09AUG90
1400	M	ME1	$29,769	05NOV67	16OCT90
1350	F	FA3	$32,886	31AUG65	29JUL90
1401	M	TA3	$38,822	13DEC50	17NOV85
1499	M	ME3	$43,025	26APR54	07JUN80
1101	M	SCP	$18,723	06JUN62	01OCT90
1333	M	PT2	$88,606	30MAR61	10FEB81
1402	M	TA2	$32,615	17JAN63	02DEC90
1479	F	TA3	$38,785	22DEC68	05OCT89

例: 一時的に出力形式と変数の関連付けを解除する

変数の出力形式がプロシジャで使用しない永久出力形式である場合、FORMAT ステートメントを使用して出力形式と変数との関連付けを一時的に解除します。

この例では、DATA ステップの FORMAT ステートメントが$YRFMT.変数と変数 Year を永久に関連付けます。そのため、変数を PROC ステップで使用すると、プロシジャはフォーマットされた値を使用します。ただし、PROC MEANS ステップには、この PROC MEANS ステップのみに対する Year から$YRFMT.出力形式との関連付けを解除する FORMAT ステートメントが含まれています。PROC MEANS は出力で Year の保存されている値を使用します。

```
proc format;
   value $yrfmt  '1'='Freshman'
                 '2'='Sophomore'
                 '3'='Junior'
                 '4'='Senior';
run;
data debate;
    input Name $ Gender $  Year $  GPA  @@;
    format year $yrfmt.;
    datalines;
Capiccio m 1 3.598 Tucker   m 1 3.901
Bagwell  f 2 3.722 Berry    m 2 3.198
Metcalf  m 2 3.342 Gold     f 3 3.609
Gray     f 3 3.177 Syme     f 3 3.883
Baglione f 4 4.000 Carr     m 4 3.750
Hall     m 4 3.574 Lewis    m 4 3.421
;

options nodate pageno=1
        linesize=64 pagesize=40;
proc means data=debate mean maxdec=2;
   class year;
   format year;
   title 'Average GPA';
run;
```

PROC MEANS は次の出力を生成します。YRFMT.出力形式は使用されません。

Average GPA

The MEANS Procedure

Analysis Variable : GPA		
Year	N Obs	Mean
1	2	3.75
2	3	3.42
3	3	3.56
4	4	3.69

出力形式と BY グループ処理

プロシジャはデータセットの処理時に、出力形式が BY 変数に割り当てられているかどうかを決定するためにチェックします。割り当てられている場合、プロシジャはフォーマットされた値が変わるまでオブザベーションを現在の BY グループに追加します。BY 変数の *nonconsecutive* 内部値に同じフォーマットされた値が含まれている場合、値は異なる BY グループにグループ化されます。そのため、2 つの BY グループは、同じフォーマットされた値で作成されます。また、BY 変数の異なる *consecutive* 内部値に同じフォーマットされた値が含まれている場合は、同じ BY グループに含まれます。

出力形式とエラーチェック

出力形式が見つからない場合、処理は停止され SAS ログにエラーメッセージが印刷されます。システムオプション NOFMTERR を使用して、この動作を行わないようにすることができます。NOFMTERR を使用すると、出力形式が見つからない場合、デフォルトの出力形式が使用され、処理が続行されます。SAS では通常、デフォルトで、BEST *w*.フォーマットが数値変数に、$*w*.フォーマットが文字変数に使用されます。

注: ユーザー作成の出力形式を検索できるようにするには、SAS システムオプション FMTSEARCH=を使用します。出力形式の保存方法については、“入力形式と出力形式の保存 ” (836 ページ)を参照してください。

ライブラリのすべてのデータセットを処理する

SAS マクロ機能を使用して、ライブラリのすべてのデータセットで同じプロシジャを実行できます。マクロ機能は、Base SAS ソフトウェアの一部です。

“例 9: SAS ライブラリのすべてのデータセットを印刷する” (1421 ページ) に、ライブラリのすべてのデータセットの印刷方法を示します。同じマクロ定義を使用して、ライブラリのすべてのデータセットでプロシジャを実行できます。プログラムの PROC PRINT の部分を適切なプロシジャコードと置き換えるだけです。

動作環境固有のプロシジャ

複数の Base SAS プロシジャは、1 つの動作環境または 1 つのリリースに固有です。付録 2, “動作環境固有のプロシジャ” (2115 ページ)には、追加情報を含むテーブルが

含まれます。これらのプロシジャの詳しい説明については、ご使用の動作環境の SAS ドキュメントを参照してください。

統計量の説明

次のテーブルでは、複数の Base SAS プロシジャで利用可能な共通の記述統計量を識別します。利用可能な統計量に関する詳細情報と理論情報については、“キーワードと式” (2078 ページ)を参照してください。

表 2.1 Base SAS プロシジャが計算する共通記述統計量

統計量	説明	プロシジャ
信頼区間		FREQ、MEANS または SUMMARY、TABULATE、UNIVARIATE
CSS	修正済み平方和	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
CV	変動係数	MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
適合度検定		FREQ、UNIVARIATE
KURTOSIS	尖度	MEANS または SUMMARY、TABULATE、UNIVARIATE
MAX	最大(最大)値	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
MEAN	平均値	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
MEDIAN	中央値(50番目の分位数)	CORR(ノンパラメトリックな相関指標の場合)、MEANS または SUMMARY、TABULATE、UNIVARIATE
MIN	最小(最小)値	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
MODE	最も頻度の高い値(一意でない場合、最小モードが使用されます)	UNIVARIATE
N	計算の基準となるオブザベーション数	CORR、FREQ、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
NMISS	欠損値の数	FREQ、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
NOBS	オブザベーション数	MEANS または SUMMARY、UNIVARIATE

統計量	説明	プロシジャ
PCTN	合計度数に対するセルまたは行の度数パーセント	REPORT、TABULATE
PCTSUM	総合計に対するセルまたは行合計比	REPORT、TABULATE
Pearson 相関		CORR
パーセント点		FREQ、MEANS または SUMMARY、REPORT、TABULATE、UNIVARIATE
RANGE	範囲	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
ロバスト統計量	トリム平均、ウィンザー化平均	UNIVARIATE
SKEWNESS	歪度	MEANS または SUMMARY、TABULATE、UNIVARIATE
Spearman 相関		CORR
STD	標準偏差	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
STDERR	平均の標準誤差	MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
SUM	合計	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
SUMWGT	重みの合計	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
位置の検定		UNIVARIATE
USS	未修正平方和	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
VAR	分散	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE

統計量の計算の必要条件

次に、表 2.1 (64 ページ)にリストされている統計量の計算要件を示します。推奨サンプルサイズは説明されていません。

- N と NMISS には非欠損オブザベーションは不要です。

- SUM、MEAN、MAX、MIN、RANGE、USS、CSS には、非欠損オブザベーションが少なくとも 1 つ必要です。
- VAR、STD、STDERR、CV には、オブザベーションが少なくとも 2 つ必要です。
- CV では、MEAN がゼロに等しくない必要があります。

統計量は、計算できない場合は欠損値としてレポートされます。

Output Delivery System (ODS)

Output Delivery System (ODS)により、SAS プロシジャと DATA ステップ出力の生成、保存、再作成を広範のフォーマットオプションによってより柔軟に行うことができます。ODS では、個別のプロシジャ、または DATA ステップのみからは利用できないフォーマット機能が提供されます。ODS はこうした制限を解決し、より簡単に出力をフォーマットできます。

バージョン 7 より前では、ほとんどの SAS プロシジャは従来のラインプリンタに指定された出力を生成しました。この種類の出力には制限があり、結果からほとんどの値を取得できません。

- 従来の SAS 出力は、Monospace フォントに制限されます。今日のデスクトップドキュメントエディタ、パブリッシングシステムの場合、印刷される出力により多くの多様性が必要です。
- 共通して使用されるプロシジャの一部は、出力データセットを生成しません。ODS より前は、これらのプロシジャのいずれかからの出力を別のプロシジャへの入力として使用する場合、PROC PRINTTO と DATA ステップを使用して結果を取得しました。

Output Delivery System の詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

3 章

複数のプロシジャで同じ機能を提供するステートメント

複数のプロシジャで同じ機能を提供するステートメント

概要

多数の Base SAS プロシジャで、複数の Base ステートメントに同じ機能があります。一部のステートメントについては *SAS ステートメント: リファレンス*にすべて記載され、その他はこのセクションに記載されています。各ステートメントの詳細についての参照先を次に示します。

ATTRIB
プロシジャ出力と出力データセットに影響します。ATTRIB ステートメントでは、入力データセットの変数が永久に変更されることはありません。LENGTH=オプションは影響しません。完全なドキュメントについては、*SAS ステートメント: リファレンス*を参照してください。

BY
出力を BY グループ順に並べ替えます。“BY” (68 ページ)を参照してください。

FORMAT
プロシジャ出力と出力データセットに影響します。FORMAT ステートメントでは、入力データセットの変数が永久に変更されることはありません。DEFAULT=オプションは無効です。完全なドキュメントについては、*SAS ステートメント: リファレンス*を参照してください。

FREQ
オブザベーションを、入力データセットに複数回表示されたように処理します。“FREQ” (72 ページ)を参照してください。

LABEL
プロシジャ出力と出力データセットに影響します。LABEL ステートメントでは、PROC DATASETS で MODIFY ステートメントと一緒に使用するとき以外は、入力データセットの変数が永久に変更されることはありません。完全なドキュメントについては、*SAS ステートメント: リファレンス*を参照してください。

QUIT
実行されなかったステートメントを実行し、プロシジャを終了します。“QUIT” (74 ページ)を参照してください。

WEIGHT
統計量計算の分析変数に対し重みを指定します。“WEIGHT” (74 ページ)を参照してください。

WHERE
各オブザベーションを処理可能にするために満たす必要のある特定条件を指定して、入力データセットをサブセット化します。“WHERE” (80 ページ)を参照してください。

ステートメント

BY

BY ステートメントの概要

出力を BY グループ順に並べ替えます。

詳細については、“BY グループの情報を含むタイトルの作成 ” (53 ページ)を参照してください。

BY <DESCENDING> *variable-1*
<… <DESCENDING> *variable-n*>
<NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは指定するすべての変数別に並べ替えるか、適切にインデックス付けする必要があります。BY ステートメントの変数は *BY 変数*といいます。

任意引数

DESCENDING
オブザベーションが BY ステートメントの DESCENDING の直後に続く変数別に降順で並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。オブザベーションは別の方法(時系列など)でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実

際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

PROC SORT ステップでは NOTSORTED オプションは使用できません。

注: GROUPFORMAT オプション(DATA ステップの BY ステートメントで使用可能)は、PROC ステップの BY ステートメントでは使用できません。

BY グループ処理

プロシジャでは、BY グループごとに出力が作成されます。たとえば、基本的な統計プロシジャとスコアリングプロシジャでは、BY グループごとに別々の分析が実行されます。レポートプロシジャでは、BY グループごとにレポートが作成されます。

注: PROC PRINT 以外のすべての Base SAS プロシジャでは、BY グループは独立して処理されます。PROC PRINT では、各 BY グループのオブザベーション数に加えて、全 BY グループのオブザベーション数のレポートも作成できます。同様に、PROC PRINT では、各 BY グループおよび全 BY グループの数値変数を合計できます。

各 PROC ステップでは 1 つの BY ステートメントしか使用できません。BY ステートメントを使用すると、プロシジャでは、BY 順に並べ替えたか、適切なインデックスが付いた入力データセットが求められます。入力データセットがこれらの基準を満たしていなければ、エラーが発生します。SORT プロシジャで並べ替えるか、BY 変数の適切なインデックスを作成します。

データの順序によっては、PROC ステップの BY ステートメントで NOTSORTED または DESCENDING オプションの使用が必要になる場合もあります。

- BY ステートメントの詳細については、*SAS ステートメント: リファレンス*を参照してください。
- PROC SORT の詳細については、59 章, “SORT プロシジャ,” (1785 ページ)を参照してください。
- インデックス作成の詳細については、“INDEX CREATE ステートメント” (503 ページ)を参照してください。

BY 変数値のフォーマッティング

BY ステートメントを指定してプロシジャをサブミットすると、BY グループの処理に関して次のアクションが実行されます。

1. プロシジャで、値を BY 変数の内部(未フォーマット)値順に並べ替えるどうかが決定されます。
2. プロシジャで、BY 変数に出力形式を適用されたかどうかが決定されます。BY 変数が数値で、ユーザー適用の出力形式がない場合は、BY グループ処理のために BEST12.出力形式が適用されます。
3. プロシジャは、BY 変数または変数の内部値と未フォーマット値が両方とも変更されるまで、現在の BY グループにオブザベーションを追加し続けます。

このプロセスでは、非連続の内部 BY 値が同一のフォーマットされた値を共有する場合などに、予想外の結果が出る可能性があります。この場合、フォーマットされた値は、異なる BY グループに表示されます。また、異なる連続内部 BY 値が同一のフォーマットされた値を共有している場合、そのオブザベーションは同一 BY グループにグループ化されます。

2 つのデータセットで長さが異なる BY 変数

プロシジャに BY ステートメントと 2 つの入力データセットが含まれている場合、一方の入力データセットはプライマリデータセット、他方はセカンダリデータセットと呼ばれます。プライマリデータセットは通常(常にではありません)DATA=データセットです。BY ステートメントは常にプライマリデータセットに適用されます。BY ステートメントの変数は、プライマリデータセットに出現している必要があります。

各プロシジャで、BY ステートメントをセカンダリデータセットに適用するかどうかが決定され、次のアクションのいずれかが実行されます。

- プロシジャで、BY ステートメントが常にセカンダリデータセットに適用される場合があります。この場合、BY ステートメントの変数が 1 つ以上、セカンダリデータセットに出現している必要があります。
- プロシジャで、BY ステートメントがセカンダリデータセットに一度も適用されない場合もあります。この場合、セカンダリデータセットには、BY ステートメントの変数は不要です。
- プロシジャで、セカンダリデータセットに BY 変数があるかどうかチェックされる場合があります。セカンダリデータセットに BY 変数が 1 つもない場合、セカンダリデータセットに対する BY 処理は発生しません。セカンダリデータセットに BY 変数が 1 つ以上あり、それがプライマリデータセットの BY 変数と一致する場合、セカンダリデータセットに対して BY 処理が行われます。セカンダリデータセットに全部ではなく一部の BY 変数がある場合、プロシジャでエラーメッセージが出力され、終了する場合があります。または、その特定プロシジャのドキュメントで説明しているその他のアクションが実行される場合もあります。

BY ステートメントがセカンダリデータセットに適用される場合、両方のデータセットに存在する BY 変数はそれぞれ、どちらのデータセットでも同じ種類(文字か数値)にする必要があります。BY 変数には、同一のフォーマットされた値か、同一のフォーマットされていない値のどちらかを含める必要があります。フォーマットされた値は、フォーマットされた長さとフォーマットされた値が同一の場合のみ一致します。フォーマットされていない値は、一致させるために長さを同一にする必要はありません。フォーマットされていない文字値は、末尾のブランクを削除後にフォーマットされていない値が同一になる場合に一致します。フォーマットされていない倍精度値は、値が同一の場合に一致します。

セカンダリデータセットには、プライマリデータセットにある BY 変数をすべて含める必要はありません。プロシジャでは、セカンダリデータセットに対して BY 変数のサブセットを定義できます。たとえば、プライマリデータセットに BY 変数 A、B、C、D がある場合、プロシジャでは、セカンダリデータセットに次の BY 変数を定義できます。

- A
- A、B
- A、B、C
- A、B、C、D

プライマリデータセットとセカンダリデータセットの両方に同数の BY 変数が含まれ、すべての BY 変数のバイト長とフォーマット長が同じ場合、(すべての BY 変数の)BY バッファにあるフォーマットされていない値かフォーマットされた値のどちらかが一致する必要があります。一致しない場合、各変数が比較されます。各変数のフォーマットされた値が先に比較されます。フォーマットされた長さが一致し、さらに、フォーマットされた値が一致する必要があります。フォーマットされた長さと値が一致しない場合は、バイト長が異なっても、フォーマットされていない値が比較されます。

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブランクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデ

ータセットの‘ABCD’とセカンダリデータセットの‘ABCD ’は一致します。セカンダリデータセットに‘ABCDEF’が含まれている場合は、一致しません。

BY ステートメントをサポートする Base SAS プロシジャ

CALENDAR	REPORT (非ウィンドウ環境のみ)
CHART	SORT (必須)
COMPARE	STANDARD
CORR	SUMMARY
FREQ	TABULATE
MEANS	TIMEPLOT
PLOT	TRANSPOSE
PRINT	UNIVARIATE
RANK	

注: SORT プロシジャでは、BY ステートメントでデータの並べ替え方法を指定します。その他のプロシジャでは、BY ステートメントでデータの現在の並べ替え方法を指定します。

例

この例では、PROC PRINT ステップで BY ステートメントを使用します。BY 変数 Year の値ごとに出力があります。DEBATE データセットは“例:一時的に出力形式と変数の関連付けを解除する” (62 ページ)で作成されます。

```
options nodate pageno=1 linesize=64
        pagesize=40;
proc print data=debate noobs;
   by year;
   title 'Printing of Team Members';
   title2 'by Year';
run;
```

Printing of Team Members by Year

Year=Freshman

Name	Gender	GPA
Capiccio	m	3.598
Tucker	m	3.901

Year=Sophomore

Name	Gender	GPA
Bagwell	f	3.722
Berry	m	3.198
Metcalf	m	3.342

Year=Junior

Name	Gender	GPA
Gold	f	3.609
Gray	f	3.177
Syme	f	3.883

Year=Senior

Name	Gender	GPA
Baglione	f	4.000
Carr	m	3.750
Hall	m	3.574
Lewis	m	3.421

FREQ

FREQ ステートメントの概要

オブザベーションを、入力データセットに複数回表示されたように処理します。

WEIGHT ステートメントと FREQ ステートメントは、両方のステートメントをサポートする任意のプロシジャの同一ステップ内で使用できます。

FREQ *variable*;

必須引数

variable

値がオブザベーションの度数を表す数値変数を指定します。FREQ ステートメントを使用する場合、プロシジャは各オブザベーションが *n* オブザベーションを表すとみなします。*n* は *variable* の値です。*variable* は整数でない場合、切り捨てられます。*variable* が 1 未満または欠損値の場合、プロシジャはそのオブザベーションを統計量の計算に使用しません。FREQ ステートメントを使用しない場合、各オブザベーションのデフォルト度数は 1 になります。

度数変数の合計は、オブザベーションの合計数を表します。

FREQ ステートメントをサポートするプロシジャ

- CORR
- MEANS または SUMMARY
- REPORT
- STANDARD
- TABULATE
- UNIVARIATE

例

この例のデータは、毎時記録された船の針路と速度(海里/時間)を表します。度数変数 Hours は船が同じ針路と速度を維持した時間数を表します。次の各 PROC MEANS ステップでは、平均進路と平均速度が計算されます。さまざまな結果によって、Hours を度数変数として使用する効果を示します。

次の PROC MEANS ステップでは、度数変数が使用されていません。

```
options nodate pageno=1 linesize=64 pagesize=40;

data track;
   input Course Speed Hours @@;
   datalines;
30  4  8 50 7 20
75 10 30 30 8 10
80  9 22 20 8 25
83 11  6 20 6 20
;

proc means data=track maxdec=2 n mean;
   var course speed;
   title 'Average Course and Speed';
run;
```

度数変数がなければ、各オブザベーションの度数は 1 で、オブザベーションの合計数は 8 です。

Average Course and Speed

The MEANS Procedure

Variable	N	Mean
Course	8	48.50
Speed	8	7.88

2 つ目の PROC MEANS ステップでは、度数変数として Hours が使用されます。

```
proc means data=track maxdec=2 n mean;
   var course speed;
   freq hours;
   title 'Average Course and Speed';
run;
```

Hours を度数変数として使用する場合、各オブザベーションの度数は Hours の値です。オブザベーションの合計数は 141(度数変数の値の合計)です。

Average Course and Speed

The MEANS Procedure

Variable	N	Mean
Course	141	49.28
Speed	141	8.06

QUIT

QUIT ステートメントの概要

実行されなかったステートメントを実行し、プロシジャを終了します。

QUIT;

QUIT ステートメントをサポートするプロシジャ

- CATALOG
- DATASETS
- PLOT
- PMENU
- SQL

WEIGHT

WEIGHT ステートメントの概要

統計量計算の分析変数に対し重みを指定します。

WEIGHT ステートメントと FREQ ステートメントは、両方のステートメントをサポートする任意のプロシジャの同一ステップ内で使用できます。

WEIGHT *variable*;

必須引数

variable

分析変数の値に重みを付ける数値変数を指定します。変数の値は、整数である必要はありません。

非正の値のさまざまな動作については、個々のプロシジャの WEIGHT ステートメント構文で説明します。

バージョン 7 より前の SAS の Base SAS プロシジャは、重みのないオブザベーションを分析から除外しませんでした。PROC GLM などのほとんどの SAS/STAT プロシジャは、欠損している重みだけでなく、負およびゼロの重みも常に分析から除外しています。WEIGHT ステートメントをサポートする Base SAS プロシジャでこれと同じ動作を実現するには、PROC ステートメントで EXCLNPWGT オプションを使用します。

重み値	プロシジャ
0	オブザベーションをオブザベーションの合計数にカウントします。
0 未満	重み値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントする
欠損値	分析からオブザベーションを除外する

プロシジャでは、w_iを WEIGHT 変数の値に置き換えます。この置き換え前の表記については、“キーワードと式” (2078 ページ)に記載されています。

WEIGHT ステートメントをサポートするプロシジャ

- CORR
- FREQ
- MEANS または SUMMARY
- REPORT
- STANDARD
- TABULATE
- UNIVARIATE

注: PROC FREQ では、WEIGHT ステートメントの変数値が各オブザベーションの発生度数を表します。詳細については、*Base SAS(R) 9.3 Procedures Guide:Statistical Procedures* の PROC FREQ に関する情報を参照してください。*Statistical Procedures.*

重み付き統計量の計算

WEIGHT ステートメントがサポートされるプロシジャでは、VARDEF=オプションもサポートされます。このオプションを使用すると、分散と標準偏差の計算に使用する分母を指定できます。

WEIGHT ステートメントを使用してモーメントを計算すると、*i* 番目のオブザベーションにσ^2/w_iと等しい分散があるとみなされます 。VARDEF=DF (デフォルト)を指定すると、計算された分散は、σ^2の重み付き最小二乗推定値になります。同様に、計算された標準偏差は σ の推定値になります。計算された分散は、*i* 番目のオブザベーションの分散の推定値ではないことに注意してください。これは、その分散がオブザベーションの重み(オブザベーションごとに異なる)にかかわるためです。

変数の値が、各オブザベーションの発生回数を表すカウントの場合は、WEIGHT ステートメントのかわりに FREQ ステートメントでこの変数を使用します。この場合、値はカウントなので整数にしてください。(FREQ ステートメントでは、非整数値はすべて切り捨てられます。FREQ 変数を使用して計算される分散は、オブザベーションの共通分散σ^2の推定値です。

注: データの発生元が層別サンプルの場合(このとき重みw_iは層別重みを表す)、WEIGHT ステートメントでも FREQ ステートメントでも、平均、分散、平均の分散の適切な層別推定値は提供されません。適切な分析を実行するには、PROC SURVEYMEANS の使用をお勧めします。これは *Base SAS(R) 9.3 Procedures Guide:*Statistical ProceduresStatistical Procedures に記載されている SAS/STAT プロシジャです。*Statistical Procedures.*

重み付き統計量の例

WEIGHT ステートメントの例として、30cm 幅の対象物のサイズを推定するよう 20 人に求めたとします。対象物からの距離が異なる場所にそれぞれを配置します。対象物からの距離が増加するにつれて、推定値の精度は低下します。

SAS データセット SIZE には、メートル単位の各距離(Distance)におけるセンチメートル単位の推定値(ObjectSize)、および各推定値の精度(Precision)が含まれます。最大偏差(20cm の過大見積もり)が最長距離(対象物から 7.5 メートル)で発生していることに注意してください。精度(1/Distance)として、対象物に近い位置での推定値ほど重みを増やし、遠い位置での推定値ほど重みを減らします。

次のステートメントで、データセット SIZE が作成されます。

```
options nodate pageno=1 linesize=64 pagesize=60;

data size;
   input Distance ObjectSize @@;
   Precision=1/distance;
   datalines;
1.5 30 1.5 20 1.5 30 1.5 25
3   43 3   33 3   25 3   30
4.5 25 4.5 36 4.5 48 4.5 33
6   43 6   36 6   23 6   48
7.5 30 7.5 25 7.5 50 7.5 38
;
```

次の PROC MEANS ステップでは、重みを無視して、物体サイズの平均推定値が計算されます。WEIGHT 変数がなければ、PROC MEANS では、すべてのオブザベーションに対してデフォルトの重み 1 が使用されます。したがって、すべての距離の物体サイズ推定値に、等しい重みが与えられます。物体サイズの平均推定値は、実際のサイズを 3.55cm 上回ります。

```
proc means data=size maxdec=3 n mean var stddev;
   var objectsize;
   title1 'Unweighted Analysis of the SIZE Data Set';
run;
```

Unweighted Analysis of the SIZE Data Set

The MEANS Procedure

Analysis Variable : ObjectSize			
N	Mean	Variance	Std Dev
20	33.550	80.892	8.994

次の 2 つの PROC MEANS ステップでは、WEIGHT ステートメントで精度(Precision)を使用し、値の異なる VARDEF=オプションの使用効果を示します。最初の PROC ステップでは、分散と標準偏差を含む出力データセットが作成されます。遠い位置での推定値の重みを減らすと、物体サイズの重み付き平均推定値が実際のサイズに近づきます。

```
proc means data=size maxdec=3 n mean var stddev;
   weight precision;
   var objectsize;
   output out=wtstats var=Est_SigmaSq std=Est_Sigma;
   title1 'Weighted Analysis Using Default VARDEF=DF';
run;

proc means data=size maxdec=3 n mean var std
                     vardef=weight;
   weight precision;
   var objectsize;
   title1 'Weighted Analysis Using VARDEF=WEIGHT';
run;
```

i 番目のオブザベーションの分散が$var(x_i) = \sigma^2/w_i$で、w_iは i 番目のオブザベーションの重みです。最初の PROC MEANS ステップでは、計算された分散はσ^2の推定値です。2 番目の PROC MEANS ステップでは、計算された分散は$(n - 1/n)\,\sigma^2/\overline{w}$の推定値です。このとき、$\overline{w}$は平均重みです。n が大きい場合、この値は平均重み付きのオブザベーションの分散の概算推定値です。

Weighted Analysis Using Default VARDEF=DF

The MEANS Procedure

Analysis Variable : ObjectSize			
N	Mean	Variance	Std Dev
20	31.088	20.678	4.547

Weighted Analysis Using VARDEF=WEIGHT

The MEANS Procedure

Analysis Variable : ObjectSize			
N	Mean	Variance	Std Dev
20	31.088	64.525	8.033

次のステートメントでは、各オブザベーションの重み付き分散と重み付き標準偏差を含むデータセットが作成、印刷されます。DATA ステップでは、重み付き分析からの分散と標準偏差を含む出力データセットが、元のデータセットと結合されます。各オブザベーションの分散を計算するには、Est_SigmaSq (VARDEF=DF の場合の重み付き分析によるσ^2の推定値)を各オブザベーションの重み(Precision)で除算します。各オブザベーションの標準偏差を計算するには、Est_Sigma (VARDEF=DF の場合の重み付き分析によるσの推定値)を各オブザベーションの重み(Precision)の平方根で除算します。

```
data wtsize(drop=_freq_ _type_);
   set size;
   if _n_=1 then set wtstats;
   Est_VarObs=est_sigmasq/precision;
   Est_StdObs=est_sigma/sqrt(precision);

proc print data=wtsize noobs;
   title 'Weighted Statistics';
   by distance;
   format est_varobs est_stdobs
          est_sigmasq est_sigma precision 6.3;
run;
```

Weighted Statistics

Distance=1.5

ObjectSize	Precision	Est_SigmaSq	Est_Sigma	Est_VarObs	Est_StdObs
30	0.667	20.678	4.547	31.017	5.569
20	0.667	20.678	4.547	31.017	5.569
30	0.667	20.678	4.547	31.017	5.569
25	0.667	20.678	4.547	31.017	5.569

Distance=3

ObjectSize	Precision	Est_SigmaSq	Est_Sigma	Est_VarObs	Est_StdObs
43	0.333	20.678	4.547	62.035	7.876
33	0.333	20.678	4.547	62.035	7.876
25	0.333	20.678	4.547	62.035	7.876
30	0.333	20.678	4.547	62.035	7.876

Distance=4.5

ObjectSize	Precision	Est_SigmaSq	Est_Sigma	Est_VarObs	Est_StdObs
25	0.222	20.678	4.547	93.052	9.646
36	0.222	20.678	4.547	93.052	9.646
48	0.222	20.678	4.547	93.052	9.646
33	0.222	20.678	4.547	93.052	9.646

Distance=6

ObjectSize	Precision	Est_SigmaSq	Est_Sigma	Est_VarObs	Est_StdObs
43	0.167	20.678	4.547	124.07	11.139
36	0.167	20.678	4.547	124.07	11.139
23	0.167	20.678	4.547	124.07	11.139
48	0.167	20.678	4.547	124.07	11.139

Distance=7.5

ObjectSize	Precision	Est_SigmaSq	Est_Sigma	Est_VarObs	Est_StdObs
30	0.133	20.678	4.547	155.09	12.453
25	0.133	20.678	4.547	155.09	12.453
50	0.133	20.678	4.547	155.09	12.453
38	0.133	20.678	4.547	155.09	12.453

WHERE

WHERE ステートメントの概要

各オブザベーションを処理可能にするために満たす必要のある特定条件を指定して、入力データセットをサブセット化します。

WHERE *where-expression*;

必須引数

where-expression

通常は一連のオペランドと演算子から成る有効な演算式または論理式です。WHERE 処理の詳細については、*SAS ステートメント: リファレンス*を参照してください。

WHERE ステートメントをサポートするプロシジャ

WHERE ステートメントと一緒に、SAS データセットを読み取る次の Base SAS プロシジャをどれでも使用できます。

CALENDAR	RANK
CHART	REPORT
COMPARE	SORT
CORR	SQL

DATASETS (APPEND ステートメント)	STANDARD
FREQ	TABULATE
MEANS または SUMMARY	TIMEPLOT
PLOT	TRANSPOSE
PRINT	UNIVARIATE

詳細

- CALENDAR および COMPARE プロシジャと、PROC DATASETS の APPEND ステートメントでは、複数の入力データセットが受け入れられます。詳細については、特定のプロシジャのドキュメントを参照してください。
- 出力データセットをサブセット化するには、WHERE=データセットオプションを使用します。

```
proc report data=debate nowd
            out=onlyfr(where=(year='1'));
run;
```

WHERE=の詳細については、*SAS ステートメント: リファレンス*を参照してください。

例

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。DEBATE データセットは“例:一時的に出力形式と変数の関連付けを解除する” (62 ページ)で作成されます。

```
options nodate pageno=1 linesize=64
        pagesize=40;

proc print data=debate noobs;
   where gpa>3.5;
   title 'Team Members with a GPA';
   title2 'Greater than 3.5';
run;
```

Team Members with a GPA Greater than 3.5

Name	Gender	Year	GPA
Capiccio	m	Freshman	3.598
Tucker	m	Freshman	3.901
Bagwell	f	Sophomore	3.722
Gold	f	Junior	3.609
Syme	f	Junior	3.883
Baglione	f	Senior	4.000
Carr	m	Senior	3.750
Hall	m	Senior	3.574

4 章

Base プロシジャのデータベース内処理

データベース内処理向けに拡張される Base プロシジャ

データベース内処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。機密データをデータソースから抽出する必要がないため、セキュリティを強化できます。より迅速な処理が可能になる理由は次のとおりです。

- データは、比較的低速なネットワーク接続を介して転送されるのではなく、高速の二次記憶装置を使用して、データソース上でローカル操作されます。
- データソースには、より多くの処理リソースが用意されている可能性があります。
- データソースには、実行するクエリを高度にパラレルでスケーラブルな方法で最適化できる機能が備えられている可能性があります。

SAS 9.2 のメンテナンスリリース 3 では、Base SAS プロシジャが拡張され、Teradata Enterprise Data Warehouse (EDW)、DB2、Oracle データソース内でデータを処理できるようになりました。SAS 9.3 では、プロシジャが拡張され、Netezza データソースでデータを処理できるようになりました。SAS 9.4 では、プロシジャが拡張され、Aster、Greenplum、HADOOP、HAWQ、Impala、SAP HANA データソースでデータを処理できるようになりました。データベース内プロシジャは、データソース内で集計や分析を実行できるようにするための、より洗練されたクエリを生成するために使用されます。

これらのデータベース内プロシジャはすべて SQL クエリを生成します。SAS/ACCESS または SQL をデータソースへのインターフェイスとして使用します。

次の Base SAS プロシジャでは、データベース内処理がサポートされています。

表 4.1 *データベース内 Base プロシジャ*

プロシジャ	説明
*Base SAS(R) 9.3 Procedures Guide:*Statistical Procedures の PROC FREQ *統計プロシジャ*	1 次元から *n* 次元の表を作成します。度数カウントをレポートします。2 次元から *n* 次元のクロス集計テーブルに対する関連性および一致の検定および統計量を計算します。正確検定および漸近検定を計算できます。出力データセットを作成できます。

プロシジャ	説明
PROC MEANS (1097 ページ)	記述統計量を計算します。印刷済出力および出力データセットを生成できます。デフォルトでは、PROC MEANS は印刷済出力を生成します。
PROC RANK (1546 ページ)*	SAS データセットのオブザベーションにわたる数値変数の順位を計算します。一部の順位スコアを生成できます。
PROC REPORT (1669 ページ)	PRINT プロシジャ、MEANS プロシジャ、TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートの作成が可能な 1 つのレポート作成ツールに組み合わせます。
PROC SORT (1810 ページ)*	SAS データセットオブザベーションを 1 つ以上の文字変数または数値変数の値を基準にして並べ替えます。
PROC SUMMARY (1861 ページ)	記述統計量を計算します。印刷済レポートを生成し、出力データセットを作成できます。デフォルトでは、PROC SUMMARY は出力データセットを作成します。
PROC TABULATE (1932 ページ)	データセットの一部またはすべての変数を使用して、記述統計量を表形式で表示します。

* Hadoop ではサポートされていません。

詳細については、*SAS/ACCESS for Relational Databases: Reference* の“In-Database Procedures in Teradata”を参照してください。

5 章

他のドキュメントで説明されている Base SAS プロシジャ

他のドキュメントで説明されている Base SAS プロシジャ

Base SAS プロシジャには、他の SAS ドキュメントで説明されているものもあります。次の表に、該当するプロシジャと、そのプロシジャを記載しているドキュメントを示します。

プロシジャ	説明	ドキュメント
CALLRFC	SAP システムのリモートファンクションコール(RFC)や RFC 互換機能を実行します。	*SAS/ACCESS Interface to R/3:ユーザーガイド*
CORR	ピアソン相関係数、アソシエーションの 3 つのノンパラメトリック測定、これらの統計量に関連する確率を計算します。	*Base SAS Procedures Guide:統計プロシジャ*
CV2VIEW	SAS/ACCESS ビューディスクリプタを SQL ビューに変換します。	*SAS/ACCESS for Relational Databases: Reference*
DBCSTAB	SAS がサポートするダブルバイト文字セットの変換テーブルを作成します。	*SAS 各国語サポート(NLS): リファレンスガイド*
DOCUMENT	ODS ドキュメント内のプロシジャやデータベースクエリの結果出力の再配置、複製または削除を可能にします。	*SAS Output Delivery System: ユーザーガイド*
EXPLODE	サイズ制限を超えているテキストで印刷を作成するために、各文字を文字行列に展開します。	EXPLODE プロシジャ

プロシジャ	説明	ドキュメント
FedSQL	後続の入力が FedSQL ステートメントであることを示します。	*SAS FedSQL Language Reference*
FORMS	封筒のラベル、宛名ラベル、外部テープラベル、ファイルカードなど、規則的なパターンがあるプリンタ用紙を作成します。	FORMS プロシジャ
FREQ	1 次元度数から N 次元の度数および分割(クロス集計)表を作成します。	*Base SAS Procedures Guide:統計プロシジャ*
GCHART	次の 6 種類のグラフ、BLOCK チャート、横棒グラフと縦棒グラフ、円グラフとドーナツグラフ、スターチャートを作成します。	*SAS/GRAPH: Reference*
GPLOT	ひと組の座標軸(X および Y)の 2 つ以上の変数の値をプロットします。	*SAS/GRAPH: Reference*
INFOMAPS	プログラムでの Information Map の作成を可能にします。	*Base SAS ガイド(Information Map)*
METADATA	XML 文字列を SAS Metadata Server に送信します。	*SAS Language Interfaces to Metadata*
METALIB	メタデータサーバーのメタデータを更新して、ライブラリのテーブルと一致させます。	*SAS Language Interfaces to Metadata*
METAOPERATE	SAS Metadata Server に関連する管理タスクのバッチモードでの実行を可能にします。	*SAS Language Interfaces to Metadata*
ODSLIST	リポートの個別コンポーネントを保存し、レポートの変更および再生を可能にします。	*SAS Output Delivery System: ユーザーガイド*
ODSTABLE	表形式出力テンプレートを作成します。	*SAS Output Delivery System: ユーザーガイド*
ODSTEXT	無限にカスタマイズやネストを実施できるパラグラフとリストを作成します。	*SAS Output Delivery System: ユーザーガイド*

プロシジャ	説明	ドキュメント
SGDESIGN	1 つ以上の入力 SAS データセット、およびユーザー定義の ODS Graphics Designer (SGD)ファイルからグラフを作成します。	*SAS ODS Graphics: プロシジャガイド*
SGPANEL	1 つ以上の分類変数の値に対するグラフセルのパネルを作成します。	*SAS ODS Graphics: プロシジャガイド*
SGPLOT	1 つ以上のプロットを作成し、単一の軸の組み合わせに重ね合わせます。	*SAS ODS Graphics: プロシジャガイド*
SGRENDER	Graph Template Language (GTL)で作成されたテンプレートからグラフ出力を作成します。	*SAS ODS Graphics: プロシジャガイド*
SGSCATTER	使用するプロットステートメントに応じて、複数の変数の組み合わせに対する散布図のパネルグラフを作成します。	*SAS ODS Graphics: プロシジャガイド*
SQL	SAS に SQL(Structured Query Language)を実装します。	*SAS SQL プロシジャユーザーガイド*
TEMPLATE	SAS 出力の表示のユーザー設定を可能にします。	*SAS Output Delivery System: ユーザーガイド*
TRANTAB	ユーザー設定された変換テーブルを作成、編集、表示して、SAS で提供される変換テーブルの参照と変更を可能にします。	*SAS 各国語サポート(NLS): リファレンスガイド*
UNIVARIATE	数値変数の統計分布を要約、ビジュアル化、分析およびモデル化するためのさまざまな記述測定、グラフィカル表示および統計方法を提供します。	*Base SAS Procedures Guide:統計プロシジャ*

すべての SAS プロシジャについては、*SAS Procedures by Name and Product* を参照してください。*SAS Procedures by Name and Product* の情報は、プロシジャ名およびプロダクト名のアルファベット順に並べられています。

2 部

プロシジャ

6 章

APPEND プロシジャ

概要: APPEND プロシジャ

APPEND プロシジャは、ある SAS データセットの末尾に、別の SAS データセットのオブザベーションを追加します。

大まかに言えば、APPEND プロシジャは DATASETS プロシジャの APPEND ステートメントと同様に機能します。APPEND プロシジャと PROC DATASETS の APPEND ステートメントの違いは、BASE=オプションと DATA=オプションの *libref* のデフォルト値のみです。PROC APPEND のデフォルトは WORK か USER です。APPEND ステートメントのデフォルトは、プロシジャの入力ライブラリのライブラリ参照名です。

構文: APPEND プロシジャ

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

APPEND ステートメントに関しての詳細なドキュメントは、DATASETS プロシジャの APPEND ステートメント (458 ページ)にあります。

ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

APPEND プロシジャの使用時には、BASE=オプションと DATA=オプションとともにデータセットオプションを使用できます。

PROC APPEND BASE=<*libref.*>*SAS-data-set*
<APPENDVER=V6>
<DATA=<*libref.*>*SAS-data-set*>
<ENCRYPTKEY=*key-value*>
<FORCE>
<GETSORT>
<NOWARN>;

ステートメント	タスク	例
“APPEND ステートメント”	ある SAS データセットのオブザベーションを別の SAS データセットの末尾に追加します	Ex. 1, Ex. 2

APPEND プロシジャの使用

データでなく、データセットのテーブルメタデータと構造のみコピーする場合は、次の例を参照してください。ここでは、Dataset1 は存在していません。

```
proc append base=dataset1 data=dataset2(obs=0);
run;

proc contents data=dataset1;
run;
quit;
```

例: APPEND プロシジャ

例 1: 2 つの SAS データセットを連結する

要素: PROC APPEND ステートメントオプション
BASE=
DATA=
FORCE

他の要素: OPTIONS ステートメント
PRINT プロシジャ

データセット: EXP ライブラリ

詳細

この例では、次のタスクについて説明します。

- ライブラリの印刷の抑制
- 2 つのデータセットの追加

- 追加後の新しいデータセットの出力

Exp.Results および Exp.Sur の各データセットを作成し、この例を使用して連結する前にそれらのデータセットを印刷するには、“EXP ライブラリ” (2153 ページ)を参照してください。

プログラム

```
options pagesize=40 linesize=64 nodate pageno=1;

LIBNAME exp 'SAS-library';

proc append base=exp.results data=exp.sur force;
run;

proc print data=exp.results noobs;
   title 'The Concatenated RESULTS Data Set';
run;
quit;
```

プログラムの説明

この例では、あるデータセットの末尾に別のデータセットを追加します。

The data set Exp.Sur contains the variable Wt6Mos, but the Exp.Results data set does not.

システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=オプションは開始ページ番号を指定します。LINESIZE=オプションで出力行長さを指定し、PAGESIZE=オプションで出力ページの行数を指定します。

```
options pagesize=40 linesize=64 nodate pageno=1;
```

LIBNAME ステートメントでライブラリを割り当てます。

```
LIBNAME exp 'SAS-library';
```

データセット Exp.Sur をデータセット Exp.Results に追加します。 PROC APPEND はデータセット Exp.Sur をデータセット Exp.Results に追加します。FORCE は Exp.Sur にある変数が Exp.Results にない場合でも PROC APPEND の追加操作を実行します。PROC APPEND は変数 Wt6Mos を Exp.Results に追加しません。

```
proc append base=exp.results data=exp.sur force;
run;
```

データセットを出力します。 アウトプット 6.3 (97 ページ)を参照してください。

```
proc print data=exp.results noobs;
   title 'The Concatenated RESULTS Data Set';
run;
quit;
```

出力:2 つの SAS データセットの連結

アウトプット 6.1 *Results データセット*

The RESULTS Data Set

id	treat	initwt	wt3mos	age
1	Other	166.28	146.98	35
2	Other	214.42	210.22	54
3	Other	172.46	159.42	33
5	Other	175.41	160.66	37
6	Other	173.13	169.40	20
7	Other	181.25	170.94	30
10	Other	239.83	214.48	48
11	Other	175.32	162.66	51
12	Other	227.01	211.06	29
13	Other	274.82	251.82	31

アウトプット 6.2 *Sur データセット*

The EXP.SUR Data Set

ID	treat	initwt	wt3mos	wt6mos	age
14	surgery	203.60	169.78	143.88	38
17	surgery	171.52	150.33	123.18	42
18	surgery	207.46	155.22	.	41

アウトプット 6.3 *Results データセットと Sur データセットの連結*

The Concatenated RESULTS Data Set

ID	TREAT	INITWT	WT3MOS	AGE
1	Other	166.28	146.98	35
2	Other	214.42	210.22	54
3	Other	172.46	159.42	33
5	Other	175.41	160.66	37
6	Other	173.13	169.40	20
7	Other	181.25	170.94	30
10	Other	239.83	214.48	48
11	Other	175.32	162.66	51
12	Other	227.01	211.06	29
13	Other	274.82	251.82	31
14	surgery	203.60	169.78	38
17	surgery	171.52	150.33	42
18	surgery	207.46	155.22	41

例 2: ソートインジケータの情報の取得

要素: PROC APPEND ステートメントオプション
GETSORT

他の要素: BY ステートメント
CONTENTS プロシジャ
ODS ステートメント
SORT プロシジャ

詳細

この例では、次のタスクについて説明します。

- 2 つのデータセットの作成:オブザベーションありとなしの 2 つのデータセットの作成
- データセットの降順ソート
- SORTEDBY データセットオプションを使用したソートインジケータの作成
- SORT プロシジャを使用したソートインジケータの作成

プログラム

```
data mtea;
```

```
    length var1 8.;
    stop;
run;

data phull;
    length var1 8.;
    do var1=1 to 100000;
    output;
end;
run;

proc sort data=phull;
    by DESCENDING var1;
run;

proc append base=mtea data=phull getsort;
run;

ods select sortedby;

proc contents data=mtea;
run;

data mysort(sortedby=var1);
    length var1 8.;
    do var1=1 to 10;
    output;
end;
run;

ods select sortedby;

proc contents data=mysort;
run;
quit;

data mysort;
    length var1 8.;
    do var1=1 to 10;
    output;
end;
run;

proc sort data=mysort;
    by var1;
run;

ods select sortedby;

proc contents data=mysort;
run;
quit;
```

プログラムの説明

次の例では、APPEND プロシジャメと GETSORT オプションを使って、ソートインジケータが継承できることを示します。

オブザベーションを含まない"シェル"データセットを作成します。

```
data mtea;
    length var1 8.;
    stop;
run;
```

同じ構造で多数のオブザベーションを持つ、もう1つのデータセットを作成します。 データセットを並べ替えます。

```
data phull;
    length var1 8.;
    do var1=1 to 100000;
    output;
end;
run;

proc sort data=phull;
    by DESCENDING var1;
run;

proc append base=mtea data=phull getsort;
run;

ods select sortedby;

proc contents data=mtea;
run;
```

ソートインジケータが、SORTEDBY データセットオプションを使用して作成されます。

```
data mysort(sortedby=var1);
    length var1 8.;
    do var1=1 to 10;
    output;
end;
run;

ods select sortedby;

proc contents data=mysort;
run;
quit;
```

ソートインジケータが PROC SORT で作成されます。

```
data mysort;
    length var1 8.;
    do var1=1 to 10;
    output;
end;
run;

proc sort data=mysort;
    by var1;
run;
```

```
ods select sortedby;

proc contents data=mysort;
run;
quit;
```

出力例

アウトプット 6.4 *降順ソート情報*

The CONTENTS Procedure

Sort Information	
Sortedby	DESCENDING var1
Validated	YES
Character Set	ANSI

アウトプット 6.5 *SORTEDBY= データセットオプションを使用したソートインジケータ情報*

The CONTENTS Procedure

Sort Information	
Sortedby	var1
Validated	NO
Character Set	ANSI

アウトプット 6.6 *SORT プロシジャを使用したソートインジケータ情報*

The CONTENTS Procedure

Sort Information	
Sortedby	var1
Validated	YES
Character Set	ANSI

7 章

AUTHLIB プロシジャ

概要: AUTHLIB プロシジャ

AUTHLIB プロシジャは、メタデータ連結ライブラリを管理するユーティリティプロシジャです。PROC AUTHLIB を使用すると、次のことを実行できます。

- SAS メタデータリポジトリ内のメタデータに物理ライブラリをバインドすることで、メタデータ連結ライブラリを作成する
- メタデータ連結ライブラリのパスワード値を変更する
- 置き換えられたパスワードと暗号化キー値(メタデータ連結ライブラリ認証情報とも呼ばれる)を消去する
- セキュリティ情報、保護ライブラリオブジェクト、保護テーブルオブジェクトを回復することで、メタデータ連結ライブラリを修復する
- 物理セキュリティ情報と、メタデータ連結ライブラリを保護するメタデータオブジェクトを削除する
- 物理ライブラリコンテンツと指定したメタデータ連結ライブラリ内の対応するメタデータオブジェクト間の不整合を報告する

SAS 9.3 のセカンドメンテナンスリリース以前の SAS リリースでは、ユーザーはメタデータバインド型データセットにアクセスできません。

注: メタデータにバインドされた z/OS 直接アクセスバウンドライブラリの場合、制約は若干幅広く、SAS の旧バージョンではライブラリにもそのメンバにもアクセスできません。

概念: AUTHLIB プロシジャ

メタデータバインド型ライブラリ

メタデータバインド型ライブラリは、対応するメタデータの保護付きテーブルオブジェクトに関連付けられる物理ライブラリです。メタデータ連結ライブラリ内の各物理テーブルには、特定のメタデータオブジェクトをポイントするヘッダー内の情報が格納されま

す。ポインタによって、物理テーブルとメタデータオブジェクト間のセキュリティバインドが作成されます。このバインドによって、ユーザーが SAS からアクセスを要求する方法に関係なく、物理テーブルのメタデータレイヤーアクセス要件が SAS によって普遍的に強制されます。詳細については、*SAS Guide to Metadata-Bound Libraries* を参照してください。

メタデータバインド型ライブラリパスワードの使用

メタデータバインド型ライブラリには、保護付きライブラリオブジェクトに保存される 1 セットのパスワードがあります。このパスワードセットは、メタデータバインド型ライブラリ内で作成される全データセットに追加されます。これらのパスワードは、データへのユーザーアクセス認証には使用されず、物理データの保護付きライブラリやテーブルメタデータオブジェクトへのバインドを修正するための管理者アクセスの認証に使用されます。また、データセットへのユーザーのアクセスを認証するプロセスでも検証されますが、ユーザーに与えられたアクセス権限を判定することはありません。

メタデータバインド型ライブラリパスワードは、メタデータバインド型ライブラリの管理者のみが知っておくものです。データライブラリ内のデータセットで以前に記録したメタデータオブジェクトや権限が失われた場合、SAS メタデータサーバーでこのパスワード情報を回復するか、または保護付きライブラリと保護付きテーブルオブジェクトを再作成する必要があります。

また、メタデータバインド型ライブラリパスワードによって、ユーザーは保護付きライブラリと保護付きテーブルオブジェクトを SAS メタデータサーバーからエクスポートし、認証を受けていないユーザーが作成した SAS メタデータサーバーやコントロールにインポートすることができなくなります。これによって、ユーザーが権限を変更した当該オブジェクトを不正なユーザーが使用できなくなります。

メタデータバインド型ライブラリパスワードは、常に暗号化された形式で保存および転送されます。暗号化されたパスワードは、転送から取り込まれて SAS に SAS 言語でパスワード値として表示される場合、データへのアクセスに使用されません。管理者は、PROC AUTHLIB ステートメントで使用されるパスワードの暗号化に、PWENCODE プロシジャを使用することも選択できます。暗号化パスワードを使用すると、管理者が入力する PROC AUTHLIB ステートメントのクリアテキストパスワードが偶然見られてしまうことがなくなります。

メタデータバインド型ライブラリセットのパスワードは 3 種類あり、それぞれ SAS データセットの READ=、WRITE=、および ALTER=パスワードに対応します。メタデータバインド型ライブラリの管理をより簡略化するには、PROC AUTHLIB ステートメントで PW=オプションを使用して 1 つのパスワード値を指定することをお勧めします。メタデータバインド型ライブラリのコンテキストでは READ=、WRITE=、および ALTER=の各オプションによってアクセス定義は作成されません。8 文字のパスワード 1 つでは会社のセキュリティ要件を満たさないことが心配な場合は、(READ=、WRITE=、および ALTER=)を使用して 3 種類のパスワード値を設定することもできます。これら 3 種類のオプションに別々の値を設定すると、24 文字のパスワードを作成できます。ただし、メタデータバインド型ライブラリに割り当てたすべてのパスワード値は記録しておく必要があります。次のような場合はパスワード値を指定する必要があります。

- ライブラリのバインドを解除する
- パスワードを変更する
- 物理ファイルと実際のメタデータオブジェクトに記録されたコンテンツ間の情報のバインドにおける不一致を修復する

詳細については、“メタデータバインド型ライブラリのパスワードの設定と変更” (104 ページ)を参照してください。

ヒント すべてのパスワード値は、最大 8 文字の有効な SAS 名にする必要があります。

注意:

メタデータ連結ライブラリのパスワードをなくすと、ライブラリのバインド解除やそのパスワードの変更ができなくなります。CREATE および MODIFY ステートメントで割り当てるパスワードは必ず記録しておいてください。

メタデータバインド型ライブラリのパスワードの設定と変更

メタデータバインド型ライブラリのパスワードは、CREATE ステートメントで設定し、MODIFY ステートメントで変更できます。オペレーティングシステムのライブラリ内のデータセットに保存されているパスワードは、MODIFY ステートメントとそれに従属する TABLES ステートメントで変更できます。また、そのライブラリがメタデータからバインド解除されていれば、REMOVE ステートメントでも変更できます。

CREATE、MODIFY、TABLES、REMOVE の各ステートメントのパスワードオプションでは、2 つの値をスラッシュ(/)で区切って指定できます(例:**`PW=password-value/new-password-value`**)。CREATE と MODIFY の各ステートメントの場合、スラッシュ(/)の後に新しいパスワードが指定されていなければ、メタデータまたはデータセットに設定されるパスワード値はスラッシュ(/)の前のパスワード値から取得されます。REMOVE ステートメントの場合も同様に、スラッシュ(/)を付記して新しいパスワード値を指定しないと、バインド解除プロセスの実行時にデータセットからパスワードを削除されます。ただし、CREATE、MODIFY、REMOVE の各ステートメントで TABLESONLY=YES を指定すると、各ステートメントの新しいパスワードは無視されます。

通常、CREATE または MODIFY のステートメントに後続する TABLES ステートメントには新しいパスワード値を指定しないでください。新しいパスワード値は、データセットがバインドされ(てい)るメタデータから取得されます。データセットごとに一意のパスワードを設定する場合は、REMOVE ステートメントに後続する TABLES ステートメントで新しいパスワード値を指定できます。その手順は次のとおりです。

1. REMOVE ステートメントで TABLESONLY=YES を指定し、個別の TABLES ステートメントで一意のパスワードを設定して、データセットのパスワードを変更します。
2. TABLESONLY=YES を含まない REMOVE ステートメントを使用しているメタデータ連結ライブラリを削除します。

関連項目:

- “例 1: 保護されないデータセットを格納する物理ライブラリのバインド” (137 ページ)
- “例 2: パスワードで保護されたデータセットを格納する物理ライブラリのバインド” (138 ページ)
- “例 3: 既存データセットが同じパスワードで保護されている場合のライブラリのバインド” (140 ページ)
- “例 4: 既存のデータセットが別のパスワードで保護されている場合のライブラリのバインド” (141 ページ)
- “例 5: データセットでのパスワードの変更” (143 ページ)
- “例 6: メタデータバインド型ライブラリパスワードの変更” (144 ページ)

暗号化データセットの検討事項

メタデータバインド型ライブラリの一部のデータは、SAS Proprietary Encryption と AES (Advanced Encryption Standard)のどちらかのアルゴリズムによって暗号化されます。

SAS Proprietary Encryption では、データセット作成時に ENCRYPT=YES が指定されます。AES では、データセットの作成時に ENCRYPT=AES と ENCRYPTKEY=キー値が指定されます。これらの暗号化データセットには、AUTHLIB プロシジャの処理時に特別な考慮が適用されます。

注意:

AES 暗号化は SAS 9.4 以降でのみサポートされます。 SAS 9.3 のセカンドメンテナンスリリースで必要とされるデータセットについては、AES 暗号化を使用しないでください。

SAS Proprietary 暗号化データセット

SAS Proprietary Encryption では、データセットの READ パスワードを暗号化キーの一部として使用します。ライブラリ内のメタデータバインド型データセットはすべて同じパスワードセットを共有するため、ファイルへのアクセス時に READ パスワードを指定する必要はありません。ただし、CREATE、MODIFY、または REMOVE ステートメント内のデータセットで READ パスワードが変更されると、新しいパスワード値を使用してデータを再暗号化する必要があります。この処理は、9.4 リリースではコピー置換処理によって自動的に実行されます。コピー置換処理の詳細については、“コピー置換処理” (136 ページ)を参照してください。

AES で暗号化されたデータセット

AES で暗号化されたデータセットにアクセスするには 2 つの方法があります。

- データセットを開くには、ユーザーが ENCRYPTKEY=キー値を入力する必要があります。
- 管理者は CREATE または MODIFY ステートメントの ENCRYPTKEY=オプションでメタデータバインド型ライブラリのオプションまたは必須の暗号化キーを記録しておく必要があります。

注: ENCRYPTKEY=値はパスフレーズであり、最大で 64 文字を使用できます。この値に基づいて実際の AES 暗号化キーは後で作成されますが、大部分の SAS ドキュメントでは暗号化キーと呼ばれています。

メタデータバインド型ライブラリのオプションまたは必須の ENCRYPTKEY=キー値を記録することで、メタデータは暗号化キー値のキーストアになります。パスワード値と同様に、キー値の保存と転送は常に暗号化された形式で行われます。暗号化されたキー値は、転送から取り込まれて SAS に SAS 言語で暗号化キー値として表示される場合、データへのアクセスには使用されません。詳細については、“メタデータバインド型ライブラリの暗号化オプションの設定と変更” (106 ページ)を参照してください。ライブラリの暗号化キーが記録されていない場合やデータセットが別のキーで暗号化されている場合は、TABLES ステートメントの ENCRYPTKEY=オプションで暗号化キー値を指定する必要があります。“TABLES ステートメント” (132 ページ)を参照してください。

注: AUTHLIB プロシジャでメタデータに記録されている暗号化キーは、SAS 9.4 のファーストメンテナンスリリースが適用されているかどうかにかかわらず、SAS データセットの作成時や置換時に SAS 9.4 リリースによって有効化されます。REQUIRE_ENCRYPTION=YES 属性が設定されている場合、AUTHLIB プロシジャの SAS 9.4 リリースはメタデータバインド型ライブラリの管理には使用できません。

注意:

ライブラリのメタデータに暗号化キーを記録する場合でも、ENCRYPT=AES を使用する場合はキーを他の場所にも記録しておく必要があります。 メタデータを損失したり、ENCRYPTKEY=キー値を忘れたりすると、データを失うことになります。SAS では、ENCRYPTKEY=キー値の復元についてはサポートできません。次の注意がログに記録されます。

```
NOTE:If you lose or forget the ENCRYPTKEY= value, there will be no way  to open the f
```

詳細については、“メタデータバインド型ライブラリの暗号化オプションの設定と変更” (106 ページ)を参照してください。

注意:

AES 暗号化を使用しているデータセットに参照一貫性制約が含まれる場合、更新アクセスを行うためにデータセットが開かれるときに、すべてのデータセットの暗号化キーが利用可能でなければなりません。 通常、SAS ではすべてのデータセットで同じ暗号化キーを共有する必要があります。メタデータに記録されているオプションまたは必須の暗号化キーを使用して、関連のデータセットに異なるキーを設定できます。ただし、別のライブラリのデータセットに関連付けられているデータセットを含む 1 つのライブラリで暗号化キーを変更すると、問題が発生する可能性があります。

ヒント メタデータバインド型ライブラリに AES 暗号化データセットが含まれている場合は、暗号化キーを記録し、AES で暗号化されたライブラリ内のどのメタデータバインド型データセットにもそのキーを使用することを推奨します。すべてのデータセットにその暗号化キーが使用されるようにする最善の方法は、暗号化を必須化することです。詳細については、“メタデータバインド型データセットの暗号化を必須化する方法” (108 ページ)を参照してください。

関連項目:

- “例 10: 既存データセットが SAS Proprietary で暗号化されている場合のライブラリのバインド” (149 ページ)
- “例 11: 既存データセットが AES で暗号化される場合のライブラリのバインド” (150 ページ)
- “例 12: 既存の AES 暗号化データセットの暗号化キーが異なる場合、オプションの記録された暗号化キーでライブラリをバインドする方法 ” (152 ページ)
- “例 13: 既存のデータセットが同じ暗号化キーで暗号化されている場合、AES 暗号化が必須のライブラリをバインドする方法” (154 ページ)
- “例 14: AES 暗号化を必要とするメタデータ連結ライブラリで暗号化キーを変更する方法” (156 ページ)

メタデータバインド型ライブラリの暗号化オプションの設定と変更

メタデータバインド型ライブラリの暗号化に影響を及ぼすオプションは次の 3 つです。

- REQUIRE_ENCRYPTION=
- ENCRYPT=
- ENCRYPTKEY=

これらのメタデータバインド型ライブラリの暗号化オプションは、CREATE ステートメントで設定し、MODIFY ステートメントで変更できます。オペレーティングシステムのライブラリにあるデータセットの暗号化は、CREATE および MODIFY ステートメントとそれに従属する TABLES ステートメントで変更できます。また、そのライブラリがメタデータからバインド解除されていれば、REMOVE ステートメントでも変更できます。ただし、CREATE、MODIFY、REMOVE の各ステートメントで TABLESONLY=YES を指定すると、各ステートメントの新しい暗号化オプションは無視されます。また、データセットの暗号化オプションを変更すると、コピー置換処理が自動的に実行され、出たが新しいオプションによって再暗号化されます。コピー置換処理の詳細については、“コピー置換処理” (136 ページ)を参照してください。

REQUIRE_ENCRYPTION=オプションを CREATE ステートメントで使用する場合、そのデフォルト値は NO です。REQUIRE_ENCRYPTION=オプションは、MODIFY ステートメントで YES または NO に変更できます。

ENCRYPT=オプションでは、使用する暗号化タイプを指定します。タイプは、AES、YES、NO のいずれかです。暗号化が必須でない場合、ENCRYPT=NO は無効です。メタデータバインド型ライブラリの暗号化キーを記録または変更するには、ENCRYPT=AES を指定する必要があります。記録された AES 暗号化キーによる必須の暗号化を SAS Proprietary アルゴリズムによる必須の暗号化に変更するには、MODIFY ステートメントで ENCRYPT=YES を指定します。このプロセスによって、記録された暗号化キーは削除されます。暗号化が必須でない場合、記録された暗号化キーを削除するには、MODIFY ステートメントで ENCRYPT=NO を指定します。REMOVE ステートメントでバインド解除時にデータセットの暗号化を変更するには、次のいずれかのタスクを実行します。

- TABLESONLY=YES と別の TABLES ステートメントの暗号化オプションでバインド解除されたデータセットに別の暗号化オプションを指定します
- TABLESONLY が YES でない場合、ENCRYPT=オプションによってバインド解除されたすべてのデータセットの暗号化を共通のものに変更します

パスワードオプションと同様に、ステートメントの ENCRYPTKEY=オプションでは 2 つの値をスラッシュ(/)で区切って指定できます。次に例を示します。

```
ENCRYPTKEY=key-value/new-key-value
```

CREATE および MODIFY ステートメントでは、メタデータまたはデータセットに記録する暗号化キー値は、

- ENCRYPT=AES
- でスラッシュ(/)の後に新しいキー値を指定しなければ、スラッシュ(/)の前の暗号化キー値から取得されます。

ENCRYPT=AES を指定しないと、暗号化キー値はデータセットを開くために使用されますが、メタデータには記録されません。パスワードオプションとは異なり、暗号化キー値の後にスラッシュ(/)を付記して暗号化キー値をブランクにする方法で暗号化キー値を削除しないでください。前のパラグラフで説明したとおり、ENCRYPT=YES または ENCRYPT=NO を使用してください。

暗号化が必須の場合、CREATE または MODIFY ステートメントに後続する TABLES ステートメントで新しいキー値を指定しないでください。新しいパスワード値は、データセットがバインドされ(てい)るメタデータから取得されます。暗号化が必須でない場合や TABLESONLY=YES を指定した REMOVE ステートメントに従っている場合、TABLES ステートメントで ENCRYPT=AES と新しいキー値を指定すれば、そのキー値によってデータセットを再暗号化できます。

関連項目:

- “例 15: 異なる暗号化キーによって AES で暗号化された既存のデータセットをもつライブラリのバインド” (158 ページ)
- “例 16: 既存のデータセットが異なる暗号化キーで暗号化されている場合、メタデータバインド型ライブラリを変更して AES 暗号化を必須化する方法” (160 ページ)

メタデータ連結ライブラリ認証情報の保持および削除

メタデータ連結ライブラリのパスワードと暗号化キーは、*メタデータ連結ライブラリ認証情報*としてまとめて参照されます。SAS 9.4 のメンテナンスリリース 3 よりも前のリリースでは、これらの認証情報が変更されると、置き換えられた値はメタデータから即時に削除されていました。別のユーザーがテーブルにアクセスしているため、テーブルが処理されないことがありました。

SAS 9.4 のメンテナンスリリース 3 からは、認証情報はメタデータに保持され、変更されていないデータセットを開いて使用できます。この保持機能により、ユーザーはテーブルの処理を続けることが可能となり、また管理者は認証情報の変更を完了できるようになります。保持された認証情報は、ライブラリ内のすべてのテーブルを処理する MODIFY ステートメントによって、すべてのテーブルがその認証情報によって正しく変更されたことが判別されると削除されます。

管理者は、既存のすべてのテーブルが正しく処理された後でも、認証情報を保持できます。認証情報を保持する理由を以下に挙げます。

- ビュー定義で古いパスワードを使用して基本テーブルに行および列レベルのセキュリティを実装したビューファイルの処理を行えます。SAS は、どのビューファイルにパスワードが含まれているかはわかりません。また、ビューファイルに含まれるパスワードを変更することもできません。管理者は新パスワードでビューを再定義する必要があります。
- これにより、変更前にバックアップからリストアしたデータセットを処理できるようになります。

管理者が古い認証情報を保持し、それらを削除しないように決定した場合、PURGE=NO オプションを MODIFY ステートメントに指定します。

注: 管理者は、置き換えられた認証情報を削除する準備が整うまで、すべてのテーブルを処理する各 MODIFY ステートメントに PURGE=NO オプションを指定する必要があります。

ライブラリにベストプラクティスに従っていないテーブルが含まれている場合、すべてのテーブルに MODIFY ステートメントを発行しても、古い認証情報の自動削除は行われない可能性があります。たとえば、オプションの暗号化が設定されたライブラリに対して保存された暗号化キーを変更する MODIFY ステートメントで、保存されたキーと一致しないキーを持つデータセットのキーは変更されない可能性があります。一部のデータセットが変更されなかったため、古い暗号化キーは削除されません。このような場合、PURGE ステートメントを使用して古い認証情報を削除する必要があります。

注: メタデータ連結テーブルへのアクセスが行われ、置き換えられた認証情報を使用してデータセットが正常に開かれると常に、Note が SAS ログに書き込まれます。この Note により、それらの認証情報が置き換えられた日時が特定されます。

詳細については、“PURGE ステートメント” (121 ページ)を参照してください。

メタデータバインド型データセットの暗号化を必須化する方法

管理者は SAS 9.4 のファーストメンテナンスリリースの開始時に、メタデータバインド型ライブラリ内のすべてのデータセットが作成時に自動的に暗号化されることを必須化できます。これを指定するには、CREATE または MODIFY ステートメントで REQUIRE_ENCRYPTION=YES オプションを使用します。必要とされる暗号化のタイプは、AES 暗号化キーが記録されているかどうかによります。暗号化キーが記録されている場合、保護付きライブラリオブジェクトにバインドされたデータセットはすべて、記録された暗号化キーによって自動的に AES で暗号化されます。AES 暗号化キーが記録されていない場合、すべてのデータセットは SAS Proprietary アルゴリズムで自動的に暗号化されます。

データセットを自動的に暗号化するため、コピー置換処理が使用されます。コピー置換処理の詳細については、“コピー置換処理” (136 ページ)を参照してください。データセットが別のキー値によって暗号化されている場合、そのキー値は記録された現在の暗号化キー値か、TABLES ステートメントの ENCRYPTKEY=オプションで指定した値でなければなりません。

注: AUTHLIB プロシジャでメタデータ連結ライブラリの REQUIRE_ENCRYPTION=YES 属性がメタデータに設定されている場合、SAS

9.4 のファーストメンテナンスリリースが適用されているかどうかにかかわらず、その属性は SAS データセットの作成時や置換時に SAS 9.4 によって有効化されます。REQUIRE_ENCRYPTION=YES 属性が設定されている場合、AUTHLIB プロシジャのプリメンテナンスバージョンはメタデータバインド型ライブラリの管理には使用できません。SAS 9.3 のセカンドメンテナンスリリースは REQUIRE_ENCRYPTION=YES 属性を有効化しません。REQUIRE_ENCRYPTION=YES 属性が設定されている場合は、その AUTHLIB プロシジャをライブラリの管理に使用しないでください。

関連項目:

- “例 15: 異なる暗号化キーによって AES で暗号化された既存のデータセットをもつライブラリのバインド” (158 ページ)
- “例 16: 既存のデータセットが異なる暗号化キーで暗号化されている場合、メタデータバインド型ライブラリを変更して AES 暗号化を必須化する方法” (160 ページ)

保護付きテーブルオブジェクトにバインドされないメタデータ連結ライブラリ内のデータセット

メタデータバインド型ライブラリ内の一部のデータセットには、メタデータバインド型ライブラリパスワードは格納されない可能性があります。これらのデータセットは、認証プロセスでバウンドライブラリの一部とはみなされません。これは次のいずれかの場合に発生します。

- バインドされる前にライブラリ内にデータセットが存在し、それらのパスワードがメタデータライブラリのパスワードと異なっていた場合
- データセットが AES で暗号化され、CREATE または MODIFY ステートメントでデータセットを開くために暗号化キーが使用できない場合

次の例を参照してください。

- “例 2: パスワードで保護されたデータセットを格納する物理ライブラリのバインド” (138 ページ)
- “例 12: 既存の AES 暗号化データセットの暗号化キーが異なる場合、オプションの記録された暗号化キーでライブラリをバインドする方法 ” (152 ページ)

また、オペレーティングシステムのコピーユーティリティによってデータセットがライブラリにコピーされる場合にも発生します。

データセットがコピーされる前にバインドされていた場合、元の保護付きライブラリでバインドされる保護付きテーブルオブジェクトでユーザーが持つ権限によって、そのデータセットは引き続き保護されます。

コピーされる前にバインドされていなかったデータセットは、新しいライブラリでもバインドされず、メタデータ権限によって保護されることもありません。データセットにパスワードが含まれている場合、データにアクセスするにはパスワードの入力が必要です。

MODIFY ステートメントを使用すれば、必要に応じてパスワードを変更し、ライブラリのバインド先の保護付きライブラリオブジェクト内の保護付きテーブルオブジェクトにデータセットをバインドできます。詳細については、“例 5: データセットでのパスワードの変更” (143 ページ)を参照してください。

構文: AUTHLIB プロシジャ

制限事項: SAS 9.3 のセカンドメンテナンスリリース以前の SAS リリースでは、ユーザーはメタデータバインド型データセットにアクセスできません。

AUTHLIB プロシジャは、SAS 管理者を使用対象者としています。メタデータレイヤーまたはホストレイヤーのいずれかで十分な権限を持たないユーザーは、このステートメントを使用できません。

AUTHLIB プロシジャは、SAS/SHARE サーバーを介したアクセス用に割り当てられるライブラリでは実行できません。

指定した物理ライブラリは、連結ライブラリにも、一時ライブラリにもならず、SAS/SHARE サーバー経由でアクセスできません。必ずメタデータ連結ライブラリに対応しているエンジンで処理してください。

要件 AUTHLIB プロシジャには、対象メタデータサーバーへの接続が必要です。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

参照項目: SAS Guide to Metadata-Bound Libraries

```
PROC AUTHLIB <option(s)>;
   CREATE
      SECUREDLIBRARY='secured-library-name'
        <SECUREDFOLDER='secured-folder-path'>
        <LIBRARY=libref>
        PW=all-password-value </ new-all-password-value> |
        ALTER=alter-password-value </ new-alter-password-value>
        READ=read-password-value </ new-read-password-value>
        WRITE=write-password-value </ new-write-password-value>
        <REQUIRE_ENCRYPTION=YES | NO>
        <ENCRYPT=YES | NO | AES>
        <ENCRYPTKEY=key-value </ new-key-value>>;
   MODIFY <LIBRARY=libref>
      PW=all-password </ new-all-password> |
        ALTER=alter-password </ new-alter-password>
        READ=read-password </ new-read-password>
        WRITE=write-password </ new-write-password>
        <TABLESONLY=YES | NO>
        <REQUIRE_ENCRYPTION=YES | NO>
        <ENCRYPT=YES | NO | AES>
        <ENCRYPTKEY=key-value </ new-key-value>>
        <PURGE=YES | NO>;
   PURGE CREDENTIALS | CREDS <LIBRARY=libref>
      PW=all-password |
        ALTER=alter-password
        READ=read-password
        WRITE=write-password
        BEFORE=datetime;
   REMOVE<LIBRARY=libref>
      PW=all-password </ <new-all-password>> |
        ALTER=alter-password </ <new-alter-password>>
        READ=read-password </ <new-read-password>>
        WRITE=write-password </ <new-write-password>>
        <TABLESONLY=YES | NO>
        <ENCRYPT=YES | NO | AES>
        <ENCRYPTKEY=key-value </ new-key-value>>;
   REPAIR ADD | UPDATE | DELETE
      LOCATION | METADATA
        SECUREDLIBRARY='secured-library-name'
        SECUREDFOLDER='secured-folder-path'
        <LIBRARY=libref>
        PW=all-password |
          ALTER=alter-password
            READ=read-password
            WRITE=write-password
        <TABLESONLY=YES | NO>
        <ENCRYPTKEY=key-value>;
   REPORT <LIBRARY=libref>
      <ENCRYPTKEY=key-value>;
   TABLES SAS-dataset(s) | _ALL_ | _NONE_
      </>
        <PW=all-password > </ <new-all-password>> |
        <ALTER=alter-password> </ <new-alter-password>>
        <READ=read-password> </ <new-read-password>>
        <WRITE=write-password> </ <new-write-password>>;
        <MEMTYPE= DATA | VIEW>
        <ENCRYPT=YES | NO | AES>
```

ステートメント	タスク	例
“PROC AUTHLIB ステートメント”	メタデータバインド型ライブラリを作成して管理します	
“CREATE ステートメント”	SAS メタデータサーバーで保護付きライブラリオブジェクトを作成し、ディレクトリまたはバウンドファイルに物理セキュリティ情報を記録します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 11, Ex. 10, Ex. 12, Ex. 13, Ex. 15
“MODIFY ステートメント”	メタデータ連結ライブラリのパスワード値および暗号化キー値の変更	Ex. 5, Ex. 6, Ex. 16
“PURGE ステートメント”	保持されているメタデータ連結ライブラリ認証情報で、指定した置き換え日よりも古いものをすべて削除します。	
“REMOVE ステートメント”	物理セキュリティ情報と、メタデータバインド型ライブラリを保護するメタデータオブジェクトを削除します	Ex. 7
“REPAIR ステートメント”	(物理データ内の)セキュリティ情報または(メタデータ内の)保護付きライブラリおよびテーブルオブジェク 回復します	
“REPORT ステートメント”	指定したメタデータ連結ライブラリについて、(不一致がないか識別するために)物理ライブラリコンテンツと対応するメタデータオブジェクトを比較します	Ex. 8
“TABLES ステートメント”	指定したメタデータバインド型ライブラリ内のどのテーブルが所定の AUTHLIB ステートメントによって影響を受けるかを指定します	Ex. 4, Ex. 9, Ex. 11, Ex. 10, Ex. 12, Ex. 15, Ex. 16

PROC AUTHLIB ステートメント

metadata-bound ライブラリを管理します。

構文

PROC AUTHLIB <*option(s)*>;

オプション引数の要約

LIBRARY=*libref*
: 保護付きライブラリオブジェクトが作成され、セキュリティ情報が保存される対象の物理ライブラリの名前です。

NOWARN
: エラー処理を行いません。

PWREQ=YES | NO
: ダイアログボックスのポップアップを制御します。

オプション引数

LIBRARY=*libref*
保護付きライブラリオブジェクトが作成され、セキュリティ情報が保存される対象の物理ライブラリの名前です。

LIBRARY=オプションが指定されていない場合、CREATE、MODIFY、REMOVE、REPORT、REPAIR の各ステートメントから LIBRARY=*libref* (物理ライブラリ)は使用されません。

別名 LIB=、DDNAME=、DD=

制限事項 物理ライブラリには、連結ライブラリや一時ライブラリを指定することはできず、SAS/SHARE サーバー経由でアクセスすることもできません。必ずメタデータバインドライブラリに対応しているエンジンで処理してください。

NOWARN
TABLES ステートメント内にデータセットが存在しないとき、`file not found` エラーメッセージは表示されません。

PWREQ=YES | NO
対話モードでデータセットパスワードのダイアログボックスのポップアップを制御します。

YES
入力を求められたときにパスワードを入力しない、または無効なパスワードを入力した場合、ダイアログボックスが表示されるように指定します。

NO
ダイアログボックスが表示されないようにします。パスワードを入力しない、または無効なパスワードを入力した場合、データセットは開かず、エラーメッセージが SAS ログに書き込まれます。

デフォルト PWREQ=NO

CREATE ステートメント

SAS メタデータリポジトリの対応するメタデータオブジェクトを生成し、物理ディレクトリ内とデータセット内のメタデータオブジェクトの記録を作成することにより、物理ライブラリをライブラリ内のデータセットと連結させます。

要件 AUTHLIB CREATE ステートメントには、ターゲットメタデータサーバ ーへの接続が必要です。詳しい要件については、“AUTHLIB ステートメントの使用要件” (135 ページ)を参照してください。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

構文

CREATE

SECUREDLIBRARY='*secured-library-name*'
<SECUREDFOLDER='*secured-folder-path*'>
<LIBRARY=*libref*>
PW=*all-password-value* </ *new-all-password-value*> |
ALTER=*alter-password-value* </ *new-alter-password-value*>
READ=*read-password-value* </ *new-read-password-value*>
WRITE=*write-password-value* </ *new-write-password-value*>
<REQUIRE_ENCRYPTION=YES | NO>
<ENCRYPT=YES | NO | AES>
<ENCRYPTKEY=*key-value* </ *new-key-value*>>;

必須引数

SECUREDLIBRARY='*secured-library-name*'
SAS Metadata Server の保護ライブラリオブジェクトに名前を付けます。

別名 SECLIB=

制限事項 保護フォルダの完全装飾パスを含む保護ライブラリオブジェクトパス名の合計の長さは、256 字未満です。

PW=*all-password-value* </ *new-all-password-value*>
メタデータ連結ライブラリの単一パスワードを設定します。

ALTER=*alter-password-value* </ *new-alter-password-value*>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを設定します。

READ=*read-password-value* </ *new-read-password-value*>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを設定します。

WRITE=*write-password-value* </ *new-write-password-value*>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを設定します。

ヒント すべてのパスワード値は、最大 8 文字の有効な SAS 名にする必要があります。

オプション引数

SECUREDFOLDER='*secured-folder-path*'
は`/System/Secured Libraries`フォルダツリー内にあるメタデータフォルダの名前です。このフォルダに保護ライブラリオブジェクトが作成されます。

SECUREDFOLDER=オプションが指定されていない場合、メタデータ連結ライブラリはファンデーションリポジトリの`/System/Secured Libraries`フォルダ内に直接作成されます。SECUREDFOLDER=オプションがスラッシュ(/)から始まっていない場合、これは相対パスであり、対象フォルダを探し出すために、その値は`/System/Secured Libraries/`に追加されます。SECUREDFOLDER=オプションがスラッシュ(/)から始まっている場合、これは絶対パスであり、その値は`/System/Secured Libraries`または`/<repository_name>/System/Secured Libraries`から始まる必要があります。

別名 SECFLDR=

制限事項 保護フォルダの完全装飾パスを含む保護ライブラリオブジェクトパス名の合計の長さは、256 字未満です。

ENCRYPT=YES | NO | AES
暗号化タイプを指定します。

YES
SAS Proprietary アルゴリズムを指定します。

NO
暗号化を指定しません。

AES
高度暗号化標準(AES)暗号化と、メタデータにキーを記録することを指定します。

要件 ENCRYPTKEY=オプションは、データファイルに AES 暗号化がある場合は必須です。

参照項目 “暗号化データセットの検討事項” (104 ページ)

ENCRYPTKEY=*key-value* </ *key-value*>
AES 暗号化のキー値を指定します。

要件 ENCRYPTKEY=オプションは、ライブラリまたはデータファイルに AES 暗号化がある場合は必須です。

注 ライブラリ内のすべてのデータセットの暗号化キー値はメタデータ連結ライブラリに保存できるため、承認されたユーザーはデータセットを開くたびに暗号化キー値を指定する必要はありません。詳細については、*SAS Guide to Metadata-Bound Libraries* の"Considerations for Data File Encryption"を参照してください。

ヒント ENCRYPTKEY=値はパスフレーズであり、最大で 64 文字を使用できます。この値に基づいて実際の AES 暗号化キーは後で作成されますが、大部分の SAS ドキュメントでは暗号化キーと呼ばれています。

参照項目 “暗号化データセットの検討事項” (104 ページ)

“ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)

LIBRARY=*libref*
保護ライブラリオブジェクトが作成され、セキュリティ情報が保存される物理ライブラリの名前です。

LIBRARY=オプションが指定されていない場合、AUTHLIB プロシジャからの物理ライブラリが使用されます。

別名 LIB=、DDNAME=、DD=

制限事項 物理ライブラリには、連結ライブラリや一時ライブラリを指定することはできず、SAS/SHARE サーバー経由でアクセスすることもできません。必ずメタデータバインドライブラリに対応しているエンジンで処理してください。

REQUIRE_ENCRYPTION=YES | NO

YES
メタデータ連結ライブラリのすべてのデータセットを自動的に暗号化することを指定します。

NO
メタデータ連結ライブラリのすべてのデータセットを自動的に暗号化しないことを指定します。

参照項目 “メタデータバインド型データセットの暗号化を必須化する方法” (108 ページ)

詳細

パスワードの指定

物理ライブラリにパスワードで保護されたデータセットが含まれていない場合は、CREATE ステートメントで PW=オプションまたは READ=オプション、WRITE=オプションおよび ALTER=オプションのいずれかを使用して、新しいメタデータ連結ライブラリパスワードを指定する必要があります。これが最も一般的なケースです。例については、“例 1: 保護されないデータセットを格納する物理ライブラリのバインド” (137 ページ)を参照してください。

物理ライブラリに含まれているパスワードで保護されたいくつかのデータセットが、現在の同一のパスワードセットを共有している場合は、CREATE ステートメントのパスワードオプションのスラッシュ(/)の前にデータセットの最も制限の厳しいパスワードを指定し、このスラッシュ(/)の後に新規パスワードを指定できます。例については、“例 3: 既存データセットが同じパスワードで保護されている場合のライブラリのバインド” (140 ページ)を参照してください。

物理ライブラリに含まれているパスワードで保護されたデータセットにさまざまなパスワードセットがある場合は、個別の TABLES ステートメントで各パスワードセットを使用するデータセットを指定するか(“例 4: 既存のデータセットが別のパスワードで保護されている場合のライブラリのバインド” (141 ページ)を参照)、続いて MODIFY ステートメントと TABLES ステートメントを使用すれば、ライブラリと CREATE ステートメントの連結後にパスワードを変更することができます(“例 5: データセットでのパスワードの変更” (143 ページ)を参照)。

暗号化キーの指定

AES 暗号化キーを使用して保護されるメタデータ連結ライブラリを作成したり、このライブラリにアクセスするには、暗号化キー値が必要です。ENCRYPT=AES および ENCRYPTKEY=*key-value* データセットオプションを使用する必要があります。

物理ライブラリにすべてが同じ AES 暗号化キーを共有する複数の AES 暗号化データセットが含まれている場合、CREATE ステートメントで ENCRYPTKEY=の後にキー値を指定できます。キーをメタデータに記録する場合は、ENCRYPT=AES を指定します。例については、を参照してください。たとえば、“例 13: 既存のデータセットが同じ暗号化キーで暗号化されている場合、AES 暗号化が必須のライブラリをバインドする方法” (154 ページ)を参照してください。

物理ライブラリに、異なる暗号化キーを使用する AES で暗号化されたデータセットが含まれている場合は、個別の TABLES ステートメントで各暗号化キーを使用するデータセットを指定できます。例については、“例 15: 異なる暗号化キーによって AES で暗号化された既存のデータセットをもつライブラリのバインド” (158 ページ)を参照してください。

ヒント 詳細については、*SAS Guide to Metadata-Bound Libraries* の"Considerations for Data File Encryption"を参照してください。

詳細については、“ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)および “ENCRYPT= Data Set Option” (*SAS Data Set Options: Reference*)を参照してください。

注意:
AES 暗号化を使用しているデータセットに参照一貫性制約が含まれる場合、更新アクセスを行うためにデータセットが開かれるときに、すべてのデータセットの暗号化キーが利用可能でなければなりません。 通常、SAS ではすべてのデータセットで同じ暗号化キーを共有する必要があります。メタデータに記録されているオプションまたは必須の暗号化キーを使用して、関連のデータセットに異なるキーを設定できます。ただし、別のライブラリのデータセットに関連付けられているデータセットを含む 1 つのライブラリで暗号化キーを変更すると、問題が発生する可能性があります。

注意:
相互に参照して関連付けられている、AES で暗号化されたデータセットの場合は、次のベストプラクティスに従って、データにアクセスできないようにしてください。 暗号化キーをライブラリのメタデータに保存します。保存されたキーは変更できますが、メタデータから削除したり、ライブラリとの連結を解除しないでください。

注意:
ライブラリのメタデータに暗号化キーを記録する場合でも、ENCRYPT=AES を使用する場合はキーを他の場所にも記録しておく必要があります。 メタデータを紛失したり、ENCRYPTKEY=キー値を忘れると、データを失うことになります。SAS では、ENCRYPTKEY=キー値の復元についてはサポートできません。次の注意がログに記録されます。

```
NOTE: If you lose or forget the ENCRYPTKEY= value,
there will be no way  to open the file or
recover the data.
```

MODIFY ステートメント

メタデータ連結ライブラリのパスワードと暗号化キーの値を変更します。

要件 AUTHLIB MODIFY ステートメントには、ターゲットメタデータサーバ ーへの接続が必要です。詳しい要件については、“AUTHLIB ステートメントの使用要件” (135 ページ)を参照してください。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

構文

MODIFY

<LIBRARY=*libref*>
PW=*all-password* </ *new-all-password*> |
ALTER=*alter-password* </ *new-alter-password*>
READ=*read-password* </ *new-read-password*>
WRITE=*write-password* </ *new-write-password*>
<TABLESONLY=YES | NO>
<REQUIRE_ENCRYPTION=YES | NO>
<ENCRYPT=YES | NO | AES>
<ENCRYPTKEY=*key-value* </ *new-key-value*>>
<PURGE=YES | NO>;

必須引数

PW=*all-password* </ *new-all-password*>
メタデータ連結ライブラリの単一パスワードを変更します。

ALTER=*alter-password* </ *new-alter-password*>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを変更します。

READ=*read-password* </ *new-read-password*>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを変更します。

WRITE=*write-password* </ *new-write-password*>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを変更します。

ヒント すべてのパスワード値は、最大 8 文字の有効な SAS 名にする必要があります。

オプション引数

ENCRYPT=YES | NO | AES
暗号化タイプを指定します。

YES
SAS Proprietary アルゴリズムを指定します。

NO
暗号化を指定しません。

AES
高度暗号化標準(AES)暗号化と、メタデータにキーを記録することを指定します。

要件 ENCRYPTKEY=オプションは、データファイルに AES 暗号化がある場合は必須です。

参照項目 “暗号化データセットの検討事項” (104 ページ)

ENCRYPTKEY=*key-value* </ *key-value*>
AES 暗号化のキー値を指定します。

要件 ENCRYPTKEY=オプションは、ライブラリまたはデータファイルに AES 暗号化がある場合は必須です。

注 ライブラリ内のすべてのデータセットの暗号化キー値はメタデータ連結ライブラリに保存できるため、承認されたユーザーはデータセットを開くたびに暗号化キー値を指定する必要はありません。詳細については、*SAS Guide to Metadata-Bound Libraries* の"Considerations for Data File Encryption"を参照してください。

ヒント ENCRYPTKEY=値はパスフレーズであり、最大で 64 文字を使用できます。この値に基づいて実際の AES 暗号化キーは後で作成されますが、大部分の SAS ドキュメントでは暗号化キーと呼ばれています。

参照項目 “暗号化データセットの検討事項” (104 ページ)

“ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)

LIBRARY=*libref*
メタデータ連結の物理ライブラリの名前です。

LIBRARY=オプションが指定されていない場合は、AUTHLIB プロシジャの物理ライブラリが使用されます。

別名 LIB=、DDNAME=、DD=

制限事項 物理ライブラリには、連結ライブラリや一時ライブラリを指定することはできず、SAS/SHARE サーバー経由でアクセスすることもできません。必ずメタデータバインドライブラリに対応しているエンジンで処理してください。

PURGE=YES | NO

YES
ライブラリ内のすべてのテーブルが新しい認証情報に正しく変更されると、保持されているすべてのメタデータ連結ライブラリの認証情報が削除されます。

デフォルト YES

NO
ライブラリ内のすべてのテーブルが正しく変更されても、置き換えられたメタデータ連結ライブラリの認証情報は削除されません。

参照項目 “メタデータ連結ライブラリ認証情報の保持および削除” (107 ページ)

REQUIRE_ENCRYPTION=YES | NO

YES
メタデータ連結ライブラリのすべてのデータセットを自動的に暗号化することを指定します。

NO
メタデータ連結ライブラリのすべてのデータセットを自動的に暗号化しないことを指定します。

参照項目 “メタデータバインド型データセットの暗号化を必須化する方法” (108 ページ)

TABLESONLY=YES | NO
MODIFY ステートメントアクションをライブラリレベルで適用するか、単にテーブルに適用するかを指定します。TABLESONLY=NO の場合、このアクションはライブラリとデータセットに適用されます。TABLESONLY=YES の場合、このアクションはデータセットにのみ適用されます。

デフォルト NO

ヒント TABLESONLY=YES を指定し、CREATE、MODIFY または REMOVE ステートメントで新しいパスワードまたは暗号化キーを指定すると、新しいパスワード値または暗号化キー値は無視されます。ライブラリがメタデータ連結であれば、現行のパスワードまたは暗号化キー値も必要になります。

詳細

MODIFY ステートメントの使用
MODIFY ステートメントでは、必要なメタデータ連結ライブラリのパスワードおよび暗号化オプションの値を変更できます。また、必要なメタデータ連結ライブラリのパスワ

ード値がないデータセット(テーブル)でパスワードを変更することもできます。MODIFY ステートメントの後に TABLES ステートメントを配置して、データセットの現在のパスワードと暗号化キーを指定します。

1 セットのパスワードがあるメタデータライブラリに物理ライブラリが連結している状態で、メタデータ連結ライブラリのパスワードを別のセットに変更する場合は、MODIFY ステートメントでメタデータ連結ライブラリのパスワードの現在の値と新しい値を/で区切って指定します。たとえば、“例 6: メタデータバインド型ライブラリパスワードの変更” (144 ページ)を参照してください。

物理ライブラリに、メタデータ連結ライブラリのパスワードとは異なるパスワードセットを使用するパスワードで保護されたデータセットがある場合は、MODIFY ステートメントと TABLE ステートメントを使用してメタデータ連結ライブラリの必須パスワードと合致するように、データセットのパスワードを変更できます。メタデータ連結ライブラリのパスワードは MODIFY ステートメントで指定します。パスワードの各セットを使用するデータセットは別の TABLES ステートメントで指定します。詳細については、“例 5: データセットでのパスワードの変更” (143 ページ)を参照してください。

ライブラリの暗号化オプションを変更する場合は、MODIFY ステートメントで新しいオプションを指定します。物理ライブラリに AES で暗号化されたデータセットが含まれている場合は、MODIFY または TABLE ステートメント内の ENCRYPTKEY=キー値を指定するか、ライブラリに記録された暗号化キーを使用して暗号化されたデータセットに変更を加える必要があります。たとえば、“例 16: 既存のデータセットが異なる暗号化キーで暗号化されている場合、メタデータバインド型ライブラリを変更して AES 暗号化を必須化する方法” (160 ページ)を参照してください。

詳細については、“TABLES ステートメント” (132 ページ)を参照してください。

注意:

相互に参照して関連付けられている、AES で暗号化されたデータセットの場合は、次のベストプラクティスに従って、データにアクセスできないようにしてください。暗号化キーをライブラリのメタデータに保存します。保存されたキーは変更できますが、メタデータから削除したり、ライブラリとの連結を解除しないでください。

注意:

ライブラリのメタデータに暗号化キーを記録する場合でも、ENCRYPT=AES を使用する場合はキーを他の場所にも記録しておく必要があります。メタデータを損失したり、ENCRYPTKEY=キー値を忘れたりすると、データを失うことになります。SAS では、ENCRYPTKEY=キー値の復元についてはサポートできません。

次のいずれかの理由により、バックアップパッケージから SecuredLibrary オブジェクトをインポートする必要が生じる場合があります。

- SecuredLibrary オブジェクトが不注意により削除された。
- メタデータ連結ライブラリを新しいメタデータサーバーにプロモートする。

パスワード値と暗号化キー値は、SecuredLibrary オブジェクトと一緒にはエクスポートされません。これにより、間違ったメタデータサーバーにインポートされることがなくなります。この場合、パスワードと記録された暗号化キー値を、インポートされた SecuredLibrary オブジェクトでリセットする必要があります。これを行わない場合は、インポートされた SecuredLibrary オブジェクトを参照する libname の割り当ては失敗し、次のメッセージが表示されます。

```
ERROR: The secured library object information for library library-name
could not be obtained from the metadata server or has invalid data.
ERROR: Association not found.
ERROR: Error in the LIBNAME statement.
```

たとえば、“例 18: インポートされた SecuredLibrary オブジェクトでの認証情報のリセット” (164 ページ)を参照してください。

LIBRARY=オプションの使用

AUTHLIB プロシジャのデフォルトライブラリを優先する場合は、LIBRARY=を使用します。

```
MODIFY <LIBRARY=library-name>
```

物理ライブラリへの連結を解除した保護ライブラリオブジェクトのパスワードまたは暗号化オプションを変更する場合は、SECUREDLIBRARY=オプションと SECUREDFOLDER=オプションで LIBRARY=_NONE_を指定して、保護ライブラリオブジェクトを配置します。

```
MODIFY <LIBRARY=_NONE_ SECUREDLIBRARY=secured-library-name>
                <SECUREDFOLDER=secured-folder-name>
```

注意:

保護ライブラリオブジェクトが物理ライブラリに連結している場合は LIB=_none_を使用しないでください。 LIB=_none_を使用すると、アクションが保護ライブラリオブジェクトでしか動作しなくなり、LIB=_none_は物理データに対して無効になります。

PURGE オプションの使用

メタデータ連結ライブラリのパスワードと暗号化キーは、*メタデータ連結ライブラリ認証情報*としてまとめて参照されます。認証情報の保持と削除の詳細については、“メタデータ連結ライブラリ認証情報の保持および削除” (107 ページ)を参照してください。

PURGE ステートメント

保持されているメタデータ連結ライブラリ認証情報で、指定した置き換え日よりも古いものをすべて削除します。

要件 AUTHLIB PURGE ステートメントには、ターゲットメタデータサーバーへの接続が必要です。詳しい要件については、“AUTHLIB ステートメントの使用要件” (135 ページ)を参照してください。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

構文

PURGE CREDENTIALS | CREDS <LIBRARY=*libref*>

PW=*all-password* |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*
BEFORE=*datetime*;

必須引数

PW=*all-password*
1 つのメタデータ連結ライブラリに対してパスワードを 1 つ指定します。

ALTER=*alter-password*
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを指定します。

READ=*read-password*
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを指定します。

WRITE=*write-password*
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを指定します。

ヒント すべてのパスワード値は、最大 8 文字の有効な SAS 名にする必要があります。

BEFORE=*datetime*
置き換えられたが保持されているすべての認証情報が削除される条件となる日時定数を指定します。

オプション引数

LIBRARY=*libref*
メタデータ連結ライブラリが作成され、セキュリティ情報が保存される物理ライブラリの名前です。

LIBRARY=オプションが指定されていない場合、AUTHLIB プロシジャからの物理ライブラリが使用されます。

別名	LIB=、DDNAME=、DD=
制限事項	物理ライブラリには、連結ライブラリや一時ライブラリを指定することはできず、SAS/SHARE サーバー経由でアクセスすることもできません。必ずメタデータバインドライブラリに対応しているエンジンで処理してください。

詳細

PURGE ステートメントの使用

*メタデータ連結ライブラリ認証情報*としてまとめて参照されます。メタデータ連結ライブラリ認証情報の削除についての詳細は、“メタデータ連結ライブラリ認証情報の保持および削除” (107 ページ)を参照してください。

LIBRARY=オプションの使用

AUTHLIB プロシジャのデフォルトライブラリを無効にする場合は、LIBRARY=オプションを使用します。

```
PURGE CREDENTIALS <LIBRARY=library-name>
```

物理ライブラリへの連結を解除した保護ライブラリオブジェクトの認証情報を削除する場合は、SECUREDLIBRARY=オプションと SECUREDFOLDER=オプションで LIBRARY=_NONE_を指定して、保護ライブラリオブジェクトを配置します。

```
PURGE CREDENTIALS <LIBRARY=_NONE_ SECUREDLIBRARY=secured-library-name>
<SECUREDFOLDER=secured-folder-name>
```

REMOVE ステートメント

メタデータ連結ライブラリを保護する物理セキュリティ情報とメタデータオブジェクトを削除します。

要件 これによりメタデータ連結ライブラリがなくなります。詳しい要件については、“AUTHLIB ステートメントの使用要件” (135 ページ)を参照してください。

注: SASProprietary 暗号化を使用するデータセットがある場合は、ENCRYPT=NO も指定して暗号化を削除しない限り、パスワードは削除できません。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

保護されていないデータセットを変更しない場合、REMOVE ステートメントを実行する前に保護されていないすべてのデータセットを物理ライブラリから移動します。

REMOVE ステートメントを使用する前に、REPORT ステートメントの実行を考慮してください。REPORT ステートメントからの出力により、メタデータ内に対応する保護テーブルオブジェクトがない物理テーブルが特定されます。このような物理テーブルが存在している特殊な環境においては、LIBNAME ステートメントで AUTHADMIN=YES を指定しない限り、テーブルのセキュリティ位置情報は REMOVE ステートメントからの影響を受けません。この状況では、LIBNAME ステートメントで AUTHADMIN=YES オプションを使用する必要があります。

例: “例 7: REMOVE ステートメントの使い方” (146 ページ)

“例 17: AES 暗号化が必要なメタデータバインド型ライブラリで REMOVE ステートメントを使用する方法” (162 ページ)

構文

REMOVE<LIBRARY=*libref*>

PW=*all-password* </ <*new-all-password*>> |
ALTER=*alter-password* </ <*new-alter-password*>>
READ=*read-password* </ <*new-read-password*>>
WRITE=*write-password* </ <*new-write-password*>>
<TABLESONLY=YES | NO>
<ENCRYPT=YES | NO | AES>
<ENCRYPTKEY=*key-value* </ *new-key-value*>>;

必須引数

PW=*all-password* </ <*new-all-password*>>
1 つのメタデータ連結ライブラリに対してパスワードを 1 つ指定します。

ALTER=*alter-password* </ <*new-alter-password*>>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを指定します。

READ=*read-password* </ <*new-read-password*>>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを指定します。

WRITE=*write-password* </ <*new-write-password*>>
メタデータ連結ライブラリに対し、最大で 3 つのパスワード値の中から 1 つを指定します。

オプション引数

ENCRYPT=YES | NO | AES
暗号化タイプを指定します。

YES
SAS Proprietary アルゴリズムを指定します。

NO
暗号化を指定しません。

AES
高度暗号化標準(AES)を指定し、新しいキー値でデータセットを暗号化することを指定する場合は必須です。

参照項目 “暗号化データセットの検討事項” (104 ページ)

ENCRYPTKEY=*key-value* </ *key-value*>
AES 暗号化のキー値を指定します。

ヒント ENCRYPTKEY=値はパスフレーズであり、最大で 64 文字を使用できます。この値に基づいて実際の AES 暗号化キーは後で作成されますが、大部分の SAS ドキュメントでは暗号化キーと呼ばれています。

参照項目 “ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)

LIBRARY=*libref*
メタデータ連結の物理ライブラリの名前です。

LIBRARY=オプションが指定されていない場合は、PROC AUTHLIB ステートメントの物理ライブラリが使用されます。

別名 LIB=、DDNAME=、DD=

制限事項 物理ライブラリには、連結ライブラリや一時ライブラリを指定することはできず、SAS/SHARE サーバー経由でアクセスすることもできません。必ずメタデータバインドライブラリに対応しているエンジンで処理してください。

TABLESONLY=YES | NO
REMOVE ステートメントアクションをライブラリレベルで適用するか、単にテーブルに適用するかを指定します。TABLESONLY=NO の場合、このアクションはライブラリとデータセットに適用されます。TABLESONLY=YES の場合、このアクションはリストされている個別のデータセットにのみ適用されます。TABLESONLY=NO の場合、このアクションはライブラリとデータセットに適用されます。TABLESONLY=YES の場合、このアクションはリストされている個別のデータセットにのみ適用されます。

デフォルト NO

ヒント TABLESONLY=YES と新しいパスワードまたは暗号化オプションを指定すると、新しいパスワードまたは暗号化オプションは無視されます。ライブラリがメタデータ連結であれば、現在のパスワード値も必要になります。

詳細

REMOVE ステートメントは、SAS ライブラリとその中のデータセットからメタデータ連結ライブラリを切り離す場合に使用します。このステートメントでは、SAS Metadata Server から保護ライブラリと保護テーブルオブジェクトも削除します。管理者が REMOVE ステートメントでパスワードの変更を指定しない限り、メタデータ連結ライブラリのパスワードで保護されている物理ライブラリ内にデータセットが残ります。メタデータ連結ライブラリ機能が削除されて、データセットのパスワードとメタデータ連結ライブラリのパスワードの適合という要件がなくなるため、新規パスワード値を指定するのではなく現行パスワードの後にスラッシュ(/)を使用して、データセットパスワードを削除できます。この操作を実行すると、データセットの SAS 保護がなくなったという警告が SAS ログに

示されます。ENCRYPTKEY でスラッシュ(/)の後に新しいキーを指定し、ENCRYPT=AES を指定して、データセットの暗号化キーを変更することもできます。ENCRYPT=YES を指定して SASProprietary 暗号化に変更できます。ENCRYPT=NO を指定してすべての暗号化を削除できます。

REMOVE ステートメントは、データセットに保存されているメタデータ連結ライブラリのパスワードと指定したパスワードが一致する場合、データセットから位置情報を削除します。また、データセットが AES で暗号化されたものである場合は、暗号化キーをメタデータに記録するか、REMOVE または TABLES ステートメントで指定する必要があります。ただし、参照した保護テーブルオブジェクトは、オペレーティングシステムライブラリが連結している保護ライブラリオブジェクトの下になければ、削除されません。SAS に書き込まれていないユーティリティにより、データセットが別のメタデータ連結ライブラリからこの連結ライブラリにコピーされている場合、このプロセスでは、REMOVE で他のメタデータ連結ライブラリに属する保護テーブルオブジェクトは削除されません。

注: 特定のメタデータ連結ライブラリによって保護されているすべての物理テーブルが、このライブラリ(ディレクトリ)内に残っていることを確認してください。このベストプラクティスは明確性を最大化し、REMOVE ステートメントを十分に活用するためには不可欠です。特別な環境(ホスト内の別のディレクトリにコピーされたテーブルなど)では、REMOVE ステートメントによる再配置されたデータセットの連結解除が回避される場合があります。

注意:

相互に参照して関連付けられている、AES で暗号化されたデータセットを含むライブラリの連結を解除する必要がある場合は、関連付けられているすべてのデータセットが AES で暗号化されないようにするか、関連付けられているすべてのデータセットが同じ暗号化キーを共有するようにします。 AES 暗号化を保持すると、キーを指定し、ホスト層アクセスがあるユーザーのみデータを使用できます。

REPAIR ステートメント

セキュリティ情報(物理データで)または保護ライブラリおよびテーブルオブジェクト(メタデータで)を回復します。

要件 AUTHLIB REPAIR ステートメントには、ターゲットメタデータサーバ ーへの接続が必要です。詳しい要件については、“AUTHLIB ステートメントの使用要件” (135 ページ)を参照してください。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

構文

REPAIR ADD | UPDATE | DELETE
LOCATION | METADATA
SECUREDLIBRARY='*secured-library-name*'
SECUREDFOLDER='*secured-folder-path*'
<LIBRARY=*libref*>
PW=*all-password* |
ALTER=*alter-password*
READ=*read-password*
WRITE=*write-password*
<TABLESONLY=YES | NO>
<ENCRYPTKEY=*key-value*>;

必須引数

ADD | UPDATE | DELETE
これらのいずれか 1 つを指定する必要があります。

LOCATION* | *METADATA
アクションが、ファイルシステムの物理的セキュリティ情報、SAS Metadata Server のメタデータオブジェクト、またはその両方のどれに適用されるか指定します。

PW=*all-password*
1 つのメタデータ連結ライブラリに対してパスワードを 1 つ指定します。

ALTER=*alter-password*
保護ライブラリオブジェクトおよび物理ライブラリのデータセットから、Alter パスワードを割り当て、変更または削除します。

READ=*read-password*
保護ライブラリオブジェクトおよび物理ライブラリのデータセットから、Read パスワードを割り当て、変更または削除します。

WRITE=*write-password*
保護ライブラリオブジェクトおよび物理ライブラリのデータセットから、Write パスワードを割り当て、変更または削除します。

オプション引数

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

要件 ENCRYPTKEY=データセットオプションは、ライブラリまたはデータファイルがライブラリメタデータに記録されていない場合に必要です。

注 ライブラリ内のすべてのデータセットの暗号化キー値はメタデータ連結ライブラリに保存できるため、承認されたユーザーはデータセットを開くたびに暗号化キー値を指定する必要はありません。詳細については、*SAS Guide to Metadata-Bound Libraries* の"Considerations for Data File Encryption"を参照してください。

ヒント ENCRYPTKEY=値はパスフレーズであり、最大で 64 文字を使用できます。この値に基づいて実際の AES 暗号化キーは後で作成されますが、大部分の SAS ドキュメントでは暗号化キーと呼ばれています。

参照項目 “ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)

LIBRARY=*libref*
セキュリティ情報が保存される物理ライブラリの名前です。

LIBRARY=オプションが指定されていない場合は、PROC AUTHLIB ステートメントの物理ライブラリが使用されます。

別名 LIB=、DDNAME=、DD=

制限事項 物理ライブラリには、連結ライブラリや一時ライブラリを指定することはできず、SAS/SHARE サーバー経由でアクセスすることもできません。必ずメタデータバインドライブラリに対応しているエンジンで処理してください。

SECUREDLIBRARY='*secured-library-name*'
SAS Metadata Server の保護ライブラリオブジェクトに名前を付けます。

別名 SECLIB=

制限事項 保護フォルダの完全装飾パスを含む保護ライブラリオブジェクトパス名の合計の長さは、256 字未満です。

SECUREDFOLDER='*secured-folder-path*'
保護ライブラリの修復または再作成を行う`/System/Secured Libraries` フォルダツリー内にあるメタデータフォルダの名前。

別名 SECFLDR=

制限事項 保護フォルダの完全装飾パスを含む保護ライブラリオブジェクトパス名の合計の長さは、256 字未満です。

TABLESONLY=YES | NO
REPAIR ステートメントアクションをライブラリレベルで適用するか、単にテーブルに適用するかを指定します。TABLESONLY=NO の場合、このアクションはライブラリとテーブルに適用されます。TABLESONLY=YES の場合は、テーブルにのみ適用されます。管理者は保護ライブラリとすべての保護テーブルを削除することなく、特定の保護テーブルオブジェクトを削除することができるため、これは特に REPAIR では重要です。

デフォルト NO

詳細

完全にテスト済みの REPAIR ステートメントの機能は REPAIR DELETE LOCATION です。メタデータオブジェクトを削除せずにメタデータ連結ライブラリ内またはそのライブラリ内のデータセット(あるいはその両方)のセキュリティ情報を削除する必要がある場合に、このオプションの組み合わせを使用します。

システム管理者は、存在していない保護テーブルオブジェクトを示している位置情報がデータセットに残っているという状況に陥る可能性があります。別の方法でデータセットにアクセスできるようにするには、REPAIR DELETE LOCATION を使用してこの位置情報を削除する必要があります。

REPAIR ステートメントを使用する場合は、ADD、UPDATE、DELETE のいずれかのアクションを必ず指定してください。LOCATION と METADATA のいずれかまたはその両方を使用して、指定したアクションの適用先として、ファイルシステムのメタデータのセキュリティ情報、SAS Metadata Server のメタデータオブジェクトまたはその両方のいずれかを指定できます。DELETE LOCATION 以外のこれらの他のアクションは完全にテストされておらず、プリプロダクション実装とみなされます。これらのアクションはこのドキュメントで説明されていますが、テクニカルサポートからアドバイスおよび指示があった場合のみ使用してください。

1 つ以上の TABLES ステートメントが REPAIR ステートメントに続いていれば、指定したデータセットで同じアクションを実行できます。REPAIR ステートメントの後に TABLES ステートメントが続いていない場合は、暗黙の TABLES _ALL_が使用されます。

オペレーティングシステムファイル内に保存されているメタデータセキュリティ情報と、修復が必要な SAS Metadata Server 内の保護ライブラリオブジェクトとの間に不一致があると、物理ライブラリに LIBNAME ステートメントを割り当てることができない可能性があります。物理ライブラリを所有し、メタデータ連結ライブラリのパスワードを把握している管理者は、AUTHADMIN=YES オプションを LIBNAME ステートメントに追加することで、ライブラリの割り当ておよびデータの修復を実行できます。REPAIR アクションの実行時には、AUTHADMIN=YES オプションを使用することをお勧めします。

注意:

メタデータ連結ライブラリの修復は高度なタスクです。このステートメントを使用する前に、必ず現在のバックアップ(メタデータと物理データの両方)を作成してください。

誤って削除されたメタデータ連結ライブラリのセキュリティ情報またはメタデータオブジェクトを復元するには、REPAIR ステートメントを使用します。管理者は REPAIR ステートメントを慎重に使用して、REPORT ステートメントにより報告された不整合を修復できます。REPORT リストに多数のグループがある場合は、次のことを実施することをお勧めします。

1. オペレーティングシステムディレクトリおよびメタデータ連結ライブラリを新規作成し、SAS Management Console を使用して、新規保護ライブラリオブジェクトに適切なデフォルトのライブラリ許可を設定します。
2. LIBNAME ステートメントの AUTHADMIN=YES、AUTHPW=または AUTHALTER=、AUTHWRITE=、AUTHREAD=オプションを使用して、現在のライブラリにアクセスします。
3. SAS COPY プロシジャを使用して SAS データセットを新規ライブラリにコピーします。いずれかのデータセットに参照整合性の制約がある場合は、CONSTRAINT=YES を使用します。保護ライブラリオブジェクトから継承された許可とは異なる許可を保護テーブルオブジェクトに設定するには、SAS Management Console を使用します。次に COPY プロシジャの使用例を示します。

メタデータ連結ライブラリ ABCDE には、Employees、EmpInfo、Product のデータセットもあります。REPORT ステートメントは、物理ライブラリのコンテンツと、対応するメタデータオブジェクトとの間の不整合を示しています。これは、これら相違点を解消する方法の一例を表しています。

```
libname klmno "SAS-library-2";

proc authlib lib=klmno;
 create securedfolder="Department XYZZY"
        securedlibrary="KLMNOEmps"
        pw=password;
run;
quit;

libname abcde "SAS-library"
   AUTHADMIN=yes
   AUTHPW=password;

proc copy in=abcde out=klmno ;run;
```

ログ 7.1 *PROC COPY を使用した相違点の解消*

```
88   proc copy in=abcde out=klmno ;run;

NOTE: Copying ABCDE.EMPINFO to KLMNO.EMPINFO (memtype=DATA).
NOTE: Data set ABCDE.EMPINFO.DATA has secured table object location information, but the
      secured library object location information that it contains:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 38C24AF4-9CF5-458B-8389-52092307007E
      is different from the registered location for the library ABCDE:
           SecuredFolder:
           SecuredLibrary:
           SecuredLibraryGUID:
      The data set might have been copied to this directory with a host copy utility.
NOTE: Permissions are obtained from the secured table and the secured library objects that are
      referenced in the header of the metadata-bound table.
NOTE: Metadata-bound library permissions are used for KLMNO.EMPINFO.DATA.
NOTE: Successfully added new secured table object "EMPINFO.DATA" to the secured library object
      at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
      KLMNO.EMPINFO.DATA.
NOTE: There were 5 observations read from the data set ABCDE.EMPINFO.
NOTE: The data set KLMNO.EMPINFO has 5 observations and 6 variables.
NOTE: Copying ABCDE.EMPLOYEES to KLMNO.EMPLOYEES (memtype=DATA).
NOTE: Data set ABCDE.EMPLOYEES.DATA has secured table object location information, but the
      secured library object location information that it contains:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 38C24AF4-9CF5-458B-8389-52092307007E
      is different from the registered location for the library ABCDE:
           SecuredFolder:
           SecuredLibrary:
           SecuredLibraryGUID:
      The data set might have been copied to this directory with a host copy utility.
NOTE: Permissions are obtained from the secured table and the secured library objects that are
      referenced in the header of the metadata-bound table.
NOTE: Metadata-bound library permissions are used for KLMNO.EMPLOYEES.DATA.
NOTE: Successfully added new secured table object "EMPLOYEES.DATA" to the secured library
      object at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
      KLMNO.EMPLOYEES.DATA.
```

```
NOTE: There were 5 observations read from the data set ABCDE.EMPLOYEES.
NOTE: The data set KLMNO.EMPLOYEES has 5 observations and 6 variables.
NOTE: Copying ABCDE.PRODUCT to KLMNO.PRODUCT (memtype=DATA).
NOTE: Data set ABCDE.PRODUCT.DATA has secured table object location information, but the
      secured library object location information that it contains:
           SecuredFolder:      /System/Secured Libraries/Department XYZZY
           SecuredLibrary:     ABCDEEmps
           SecuredLibraryGUID: 38C24AF4-9CF5-458B-8389-52092307007E
      is different from the registered location for the library ABCDE:
           SecuredFolder:
           SecuredLibrary:
           SecuredLibraryGUID:
      The data set might have been copied to this directory with a host copy utility.
NOTE: Permissions are obtained from the secured table and the secured library objects that are
      referenced in the header of the metadata-bound table.
NOTE: Metadata-bound library permissions are used for KLMNO.PRODUCT.DATA.
NOTE: Successfully added new secured table object "PRODUCT.DATA" to the secured library object
      at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
      KLMNO.PRODUCT.DATA.
NOTE: There were 5 observations read from the data set ABCDE.PRODUCT.
NOTE: The data set KLMNO.PRODUCT has 5 observations and 2 variables.
NOTE: PROCEDURE COPY used (Total process time):
      real time           0.14 seconds
      cpu time            0.04 seconds
```

次の REPAIR ステートメントのオプションの組み合わせは*プリプロダクション*であり、完全にはテストされていません。プリプロダクションとは、この機能が、開発およびテストを完全には完了していない ソフトウェアの準備リリースであることを意味しています。プリプロダクションソフトウェアは完全にはテストされていないため、使用する場合には注意が必要です。最終テストが完了した後、将来のリリース時には実稼働品質のコンポーネントまたは製品として提供される可能性が高くなります。

REPAIR ADD LOCATION
: メタデータ連結ライブラリ内またはメタデータ連結ライブラリ内のデータセットに、メタデータ連結ライブラリおよび保護テーブルのセキュリティ情報が見当たらないときに、このオプションの組み合わせを使用します。保護ライブラリおよび保護テーブルオブジェクトが SAS Metadata Server に存在している必要があります。

REPAIR UPDATE LOCATION
: メタデータ連結型ライブラリおよび保護テーブルのセキュリティ情報が、メタデータ連結ライブラリ内またはメタデータ連結ライブラリ内のデータセットに存在しているが、正しくないまたは存在しないメタデータオブジェクトを示している場合にこのオプションの組み合わせを使用します。位置情報を更新する保護ライブラリおよび保護テーブルオブジェクトが SAS Metadata Server 内に存在している必要があります。

REPAIR ADD METADATA LOCATION
: 保護ライブラリおよび保護テーブルオブジェクトが SAS Metadata Server から削除され、メタデータ連結ライブラリまたはメタデータ連結ライブラリ内のデータセットにセキュリティ情報が登録されていない場合に、このオプションの組み合わせを使用します。メタデータオブジェクトが SAS Metadata Server に作成され、これらオブジェクトのセキュリティ情報がメタデータ連結ライブラリおよびデータセットに登録されます。

REPAIR DELETE METADATA
: メタデータ連結ライブラリまたはそのライブラリ内のデータセットのセキュリティ情報を削除せずに、保護ライブラリ、保護テーブルメタデータオブジェクトまたはその両方を削除する必要がある場合にこのオプションの組み合わせを使用します。

REPAIR DELETE METADATA LOCATION
: 保護ライブラリ、保護テーブルメタデータオブジェクトまたはその両方と、メタデータ連結ライブラリまたはそのライブラリ内のデータセットのセキュリティ情報を削除する必要がある場合に、このオプションの組み合わせを使用します。

REPAIR UPDATE LOCATION
: メタデータ連結ライブラリ、データセット内、またはその両方のセキュリティ情報を更新して、さまざまな既存の保護ライブラリおよび保護テーブルメタデータオブジェクトを示すようにする必要がある場合に、このオプションの組み合わせを使用します。

  注: METADATA オプションは REPAIR UPDATE アクションでサポートされません。

REPORT ステートメント

指定したメタデータバインドライブラリについて、(不整合がないか識別するために)物理ライブラリコンテンツと対応するメタデータオブジェクトを比較します。

要件 AUTHLIB REPORT ステートメントには、ターゲットメタデータサーバ ーへの接続が必要です。詳しい要件については、“REPORT ステートメントの使用に対する要件” (131 ページ)を参照してください。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

例: “例 8: REPORT ステートメントの使い方” (147 ページ)

構文

REPORT

<LIBRARY=*libref*>
<ENCRYPTKEY=*key-value*>;

オプション引数

LIBRARY=*libref*
連結情報をレポートする物理ライブラリの名前です。

LIBRARY=オプションが指定されていない場合は、PROC AUTHLIB ステートメントの物理ライブラリが使用されます。

別名 LIB=、DDNAME=、DD=

制限事項 物理ライブラリには、連結ライブラリや一時ライブラリを指定することはできず、SAS/SHARE サーバー経由でアクセスすることもできません。必ずメタデータバインドライブラリに対応しているエンジンで処理してください。

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

ヒント ENCRYPTKEY=値はパスフレーズであり、最大で 64 文字を使用できます。この値に基づいて実際の AES 暗号化キーは後で作成されますが、大部分の SAS ドキュメントでは暗号化キーと呼ばれています。

参照項目 “ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)

詳細

REPORT ステートメントの使用に対する要件

管理者は、REPORT ステートメントを使用して、物理メタデータバインドライブラリとこれに対応するメタデータオブジェクトとの間の不整合を特定します。

REPORT ステートメントを使用するには、次の基準を満たす必要があります。

- 対象の物理ライブラリのホスト層の読み取りアクセスを持つアカウントで、SAS セッションを実行します。これは、ライブラリ参照を割り当てるために必要です。
- 対象の保護ライブラリオブジェクトと保護テーブルオブジェクトの ReadMetadata 権限を持つ ID で、メタデータサーバーに SAS セッションをメタデータサーバーに接続します。
- ライブラリに保護ライブラリオブジェクトの場所情報があるにもかかわらず、保護ライブラリオブジェクトを取得できない場合には、そのライブラリを割り当てるために LIBNAME=ステートメントで AUTHADMIN=YES オプションを使用します。

不整合のレポート

REPORT ステートメントを使用して、物理ライブラリコンテンツと、対応するメタデータオブジェクトとの間の不整合をレポートします。

標準以外の SAS 処理を使用してメタデータまたはオペレーティングシステムファイルを操作すると、物理ディレクトリ、データセット、保護ライブラリオブジェクト、および保護テーブルオブジェクトのメタデータセキュリティ情報間に不整合が生じることがあります。たとえば、オペレーティングシステムのコピーユーティリティを使用して、1 つのディレクトリからメタデータバインドライブラリディレクトリにオペレーティングシステムデータセットをコピーすると、このデータセットには、このメタデータバインドライブラリに対する適切なセキュリティ情報が含まれません。別の例として、管理者が SAS Management Console を使用して、誤って保護ライブラリオブジェクトや保護テーブルオブジェクトを削除してしまうことがあります。

REPORT ステートメントによって、そのライブラリのオペレーティングシステムディレクトリのデータセットごとに保護テーブルとメタデータバインドライブラリのセキュリティ情報がレポートされます。このデータセット情報は、すべてのデータセットが共有するメタデータバインドライブラリ属性によりグループ分けされます。物理ライブラリのデータセットが、このライブラリの保護ライブラリオブジェクトに正しく登録され、必要なパスワードがある場合は、これらデータセットと属性は、レポートの最初のグループとしてリストされます。それに続くグループは、メタデータバインドライブラリパスワードとは異なるパスワードを有するデータセットか、あるいはメタデータバインドライブラリのセキュリティ情報が、オペレーティングシステムディレクトリに登録されているメタデータバインドライブラリの場所情報に合致していないデータセットのグループです。

TABLES ステートメント

CREATE、MODIFY、REMOVE、REPAIR、REPORT ステートメントの後で使用してステートメントアクションを処理するテーブルを指定します。また、メタデータバインドライブラリのパスワードまたは記録されている暗号化キーと異なる場合は、データセットの現在のパスワードまたは暗号化キー値を TABLES ステートメントで指定できます。

デフォルト: TABLES ステートメントを一切指定しない場合は、TABLES _ALL_ステートメントがデフォルト動作になります。

要件 TABLES ステートメントは、必ず CREATE、MODIFY、REMOVE、REPAIR、REPORT ステートメントまたは別の TABLES の後に指定してください。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

例: “例 9: TABLES ステートメントの使用” (148 ページ)

構文

TABLES *SAS-dataset(s)* | _ALL_ | _NONE_
</>
<PW=*all-password* > </ <*new-all-password*>> |
<ALTER=*alter-password*> </ <*new-alter-password*>>
<READ=*read-password*> </ <*new-read-password*>>
<WRITE=*write-password*> </ <*new-write-password*>>;
<MEMTYPE= DATA | VIEW>
<ENCRYPT=YES | NO | AES>
<ENCRYPTKEY=*key-value*< / *new-key-value*>>;

必須引数

***SAS-dataset(s)* | _ALL_ | _NONE_**

SAS-dataset(s)
1 つ以上の SAS データセットの名前

ALL
すべてのデータセットに適用するパスワードオプションを指定します。

NONE
以前の CREATE または MODIFY または REPAIR ステートメントのアクションをライブラリレベルまでに制限し、これをテーブルに適用しないようにします。

オプション引数

/
パスワードや MEMTYPE=など、任意のオプションを含む場合に必要になります。一例を以下に示します。

```
tables table-name / pw=password;
```

ENCRYPT=YES | NO | AES
暗号化タイプを指定します。

YES
SAS Proprietary アルゴリズムを指定します。

NO
暗号化を指定しません。

AES
高度暗号化標準(AES)暗号化を指定します。アクションステートメントで新しいキー値および TABLESONLY=YES を使用して変更および暗号化する場合に必要になります。

参照項目 “暗号化データセットの検討事項” (104 ページ)

ENCRYPTKEY=*key-value* </ *key-value*>
AES 暗号化のキー値を指定します。

要件 ENCRYPTKEY=データセットオプションは、データファイルに AES 暗号化があり、キーがライブラリに記録されていない場合に必要です。

ヒント ENCRYPTKEY=値はパスフレーズであり、最大で 64 文字を使用できます。この値に基づいて実際の AES 暗号化キーは後で作成されますが、大部分の SAS ドキュメントでは暗号化キーと呼ばれています。

参照項目 “暗号化データセットの検討事項” (104 ページ)

“ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)

MEMTYPE= DATA | VIEW
処理を単一メンバタイプの DATA または VIEW に制限します。これを指定しない場合、デフォルトは両タイプとなります。

DATA
SAS データファイルのメンバタイプを指定します。

VIEW
SAS ビューメンバタイプを指定します。

別名 MTYPE=, MT=

デフォルト ALL

PW=*all-password* </ *<new-all-password>>*
データセットの現在のパスワードを指定します。

ALTER=*alter-password* </ *<new-alter-password>>*
データセットの現在の ALTER=パスワードを指定します。

READ=*read-password* </ *<new-read-password>>*
データセットの現在の READ=パスワードを指定します。

WRITE=*write-password* </ *<new-write-password>>*
データセットの現在の WRITE=パスワードを指定します。

ヒント すべてのパスワード値は、最大 8 文字の有効な SAS 名にする必要があります。

詳細

TABLES ステートメントの使用

TABLES ステートメントは主に、メタデータバインドライブラリの現在の必須パスワードおよび暗号化キーと異なる場合に、データセットの現在のパスワードおよび暗号化キーを指定するために使用します。TABLES ステートメントは、通常 CREATE ステートメントまたは MODIFY ステートメントの後に指定し、データセットのパスワードおよび暗号化キーをメタデータバインドライブラリパスワードおよび暗号化キーに変更します。例については、“例 4: 既存のデータセットが別のパスワードで保護されている場合のライブラリのバインド” (141 ページ)を参照してください。

TABLES _NONE_を使用すると、以前の CREATE、MODIFY または REPAIR ステートメントのアクションをライブラリレベルまでに制限して、アクションがテーブルに適用されないようにすることができます。TABLES ステートメントを指定しない場合は、TABLES _ALL_がデフォルト動作になります。パスワードまたは暗号化キー値を指定してすべてのデータセットを開く場合に使用されるようにする場合は、明示的に TABLES _ALL_を書き込むことをお勧めします。

TABLES ステートメントの CREATE ステートメントとの併用

CREATE ステートメントの後に 1 つ以上の TABLES ステートメントを指定することで、メタデータバインドライブラリのパスワードおよび暗号化キーと異なる場合に、現在のパスワードまたは暗号化キー値を指定できます。TABLES ステートメントを使用しない場合、バインドされるデータセットは次の 2 つのグループのみになります。

- パスワードまたは暗号化キーが設定されていないデータセット
- メタデータ連結ライブラリと一致するパスワードまたは暗号化キー値が設定されたデータセット

実際には、TABLES ステートメントを省略すると、1 つの TABLES _ALL_ステートメントを指定したことと同じことになります。詳細については、“CREATE ステートメント” (113 ページ)を参照してください。

TABLES ステートメントの MODIFY ステートメントとの併用

MODIFY ステートメントの後に 1 つ以上の TABLES ステートメントを指定することで、データセットのパスワードおよび暗号化キー値の変更を指定できます。MODIFY ステートメントに後続する TABLES ステートメントがない場合、暗黙的に TABLES _ALL_が使用されます。現在のパスワードまたは暗号化キーが異なるデータセット(テーブル)が複数ある場合には、別の TABLES ステートメントが必要です。詳細については、“MODIFY ステートメント” (117 ページ)を参照してください。

TABLES ステートメントの REPAIR ステートメントとの併用

REPAIR ステートメントを使用する場合は、ADD、UPDATE、DELETE のいずれかのアクションを必ず指定してください。LOCATION と METADATA のいずれかまたはその両方を使用して、指定したアクションの適用先として、ファイルシステムのメタデータのセキュリティ情報、SAS Metadata Server のメタデータオブジェクトまたはその両方のいずれかを指定できます。REPAIR ステートメントの後に 1 つ以上の TABLES ステートメントを指定すると、指定したデータセットで同一のアクションを実行できます。ただし、REPAIR ステートメントの後に指定する TABLES ステートメントで新パスワードまたは暗号化キー値は指定できません。詳細については、“REPAIR ステートメント” (125 ページ)を参照してください。

TABLES ステートメントの REMOVE ステートメントとの併用

TABLESONLY=YES を含む TABLES ステートメントを REMOVE ステートメントで使用すると、メタデータバインドライブラリの特定のテーブルの位置情報および保護テーブルオブジェクトのみを削除できます。TABLES ステートメントとともに TABLESONLY=YES を使用しない場合、その保護ライブラリオブジェクトとすべての保護テーブルオブジェクトが、REMOVE ステートメントにより削除されます。

REMOVE ステートメントの後に TABLES ステートメントを使用すると、ENCRYPT=NO オプションにより、前述のテーブルが削除されるときにデータセットの暗号化が削除されます。詳細については、“暗号化データセットの検討事項” (104 ページ)を参照してください。このプロセスは、管理者がデータセットからパスワードまたは暗号化を削除しようとする場合にのみ必要になります。

メタデータに対する物理ライブラリのバインドを削除する場合、あるいは物理ライブラリが保護ライブラリにバインドされていない場合は、データセットのパスワードまたは暗号化を他の値に変更することをお勧めします。メタデータバインドライブラリの共通のパスワードまたは暗号化への変更に限らず、REMOVE ステートメントで、現在のパスワードと新パスワードをスラッシュ(/)で区切って両方を指定することもできます。異なるデータセットに一意のパスワードまたは暗号化を設定する場合は、次の 2 つのステップを使用します。

1. TABLESONLY=YES を設定した REMOVE ステートメントと、一意のパスワードおよび暗号化ごとに個別の TABLE ステートメントを使用して、データセットの PW=オプションを変更します。
2. TABLESONLY=YES を**設定せずに** REMOVE ステートメントを使用してメタデータ連結ライブラリを削除します。

TABLES ステートメントの REPORT ステートメントとの併用

TABLES ステートメントは、構文上は REPORT ステートメントとともに使用できますが、あまり使用されません。TABLES を指定すると、掲載されているテーブル(使用されている場合)にレポートが限定されます。詳細については、“REPORT ステートメント” (130 ページ)を参照してください。

AUTHLIB プロシジャの使用

AUTHLIB ステートメントの使用要件

REPORT ステートメントを除き、AUTHLIB プロシジャ内のすべてのステートメントでは、次の条件を満たす必要があります。

- 対象物理ライブラリのホストレイヤーコントロールを持つアカウントに従い、SAS セッションが実行される。ホストコントロール権限を持つユーザーのみが物理ライブラリをメタデータにバインドできるようにするには、SAS セッションが次の通り特権付きのホストアカウントに従い実行される必要があります。
  - UNIX では、アカウントはディレクトリの所有者である必要があります。
  - Windows では、アカウントはディレクトリのフルコントロール権限を持つ必要があります。
  - z/OS では、UNIX ファイルシステムライブラリについて、アカウントはディレクトリの所有者である必要があります。
  - z/OS では、直接アクセスバウンドライブラリについて、アカウントはライブラリデータセットへの RACF ALTER アクセス認証を持つ必要があります。
- SAS セッションは、保護付き対象データフォルダへの ReadMetadata および WriteMemberMetadata 権限を持つ ID として SAS メタデータサーバーへ接続されます。
- CREATE、MODIFY、REPAIR、REMOVE の各ステートメントにパスワードを入力する必要があります。

REPORT ステートメント要件は制約が少なく、ステートメントに記録されます。

コピー置換処理

SAS 9.4 リリースでは、データセットの再暗号化にコピー置換処理が使用されます。

SAS 9.4 のセカンドメンテナンスリリース以前では、ホスト環境以外で AUTHLIB コードを実行すると、CREATE、MODIFY、REPAIR、REMOVE のアクションに対してメタデータバインド型データセットが表示されません。SAS 9.4 のセカンドメンテナンスリリースでは、ほとんどのメタデータバインド型データファイルのバインドやバインド変更、CEDA (クロス環境データアクセス)によってアクセスされるファイルの表示にコピー置換処理が使用されます。ただし、CEDA によってアクセスされたメタデータバインド型データセットは、インデックス、拡張属性、一貫性制約が含まれていると検出され、コピー置換処理が試行されず、試行しても失敗します。

コピー置換処理では次のステップが実行されます。

1. データセットの名前が_TEMP_ENCRYPT_FILE_NAME_に変更されます。
2. データセットが元のデータセット名でコピーされ、そのプロセスでデータが再暗号化されます。
3. __TEMP_ENCRYPT_FILE_NAME__ファイルが削除されます。

コピー置換処理について次の SAS ログの例を参照してください。

- “例 12: 既存の AES 暗号化データセットの暗号化キーが異なる場合、オプションの記録された暗号化キーでライブラリをバインドする方法 ” (152 ページ)
- “例 16: 既存のデータセットが異なる暗号化キーで暗号化されている場合、メタデータバインド型ライブラリを変更して AES 暗号化を必須化する方法” (160 ページ)

結果: AUTHLIB プロシジャ

REPORT ステートメントによって、次の出力が生成されます。

アウトプット 7.1 REPORT ステートメントの使い方

The OS library is properly registered to this SecuredLibrary. These data sets are properly registered to SecuredTables in it.
SecuredLibrary Path: /System/Secured Libraries/Department XYZZY/ABCDEEmps
SecuredLibrary Guid: F25E6004-EF15-4792-B0BD-9A8499435A07
Registered in OS Path: C:\lib1
Password Set: 0

MemberName	MemberType	SecuredTableName	SecuredTableGUID
PRODUCT	DATA	PRODUCT.DATA	5057208E-7EB0-4090-BD6D-57EA856DEA8B

The OS library is properly registered to this SecuredLibrary. These data sets have no registered SecuredTable location information.
SecuredLibrary Path: /System/Secured Libraries/Department XYZZY/ABCDEEmps
SecuredLibrary Guid: F25E6004-EF15-4792-B0BD-9A8499435A07
Registered in OS Path: C:\lib1
Password Set: 1

MemberName	MemberType	SecuredTableName	SecuredTableGUID
EMPINFO	DATA		
EMPLOYEES	DATA		

例: AUTHLIB プロシジャ

例 1: 保護されないデータセットを格納する物理ライブラリのバインド

要素: PROC AUTHLIB ステートメントオプション
CREATE ステートメントオプション:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=

詳細

この例では、パスワードや AES 暗号化が施されていないデータセットを含む物理ライブラリのバインドについて説明します。

プログラム

```
proc authlib lib=zyxwvut;

    create securedfolder="Department XYZZY"
```

```
        securedlibrary="ZYXWVUTEmps"
        pw=secretpw;
run;
quit;
```

プログラムの説明

ライブラリ ZYXWVUT には、パスワードを持たない 3 つのデータセット(Employees、EmpInfo、Product)が格納されています。

```
proc authlib lib=zyxwvut;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 PW=オプションでメタデータバインド型ライブラリのパスワードを指定します。

```
    create securedfolder="Department XYZZY"
        securedlibrary="ZYXWVUTEmps"
        pw=secretpw;
run;
quit;
```

結果:RANK プロシジャ ライブラリとデータセットは、パスワードによってバインドされます *secretpw*.PROC AUTHLIB はデータへのアクセスに制約がないため、バインドは簡単に行われます。

ログの例

ログ 7.2 *保護されないデータセット*

```
79   proc authlib lib=zyxwvut; 80 81    create securedfolder="Department XYZZY" 82
securedlibrary="ZYXWVUTEmps" 83           pw=XXXXXXXX; 84 85   run; NOTE:Successfully created a
secured library object for the physical library ZYXWVUT and recorded its location
as:SecuredFolder:      /System/Secured Libraries/Department XYZZY SecuredLibrary:     ZYXWVUTEmps
SecuredLibraryGUID:1A323C03-A3D8-4A83-9615-2BC2CB9FAAE2 NOTE:Successfully added new secured table
object "EMPINFO.DATA" to the secured library object at path "/System/Secured Libraries/Department
XYZZY/ZYXWVUTEmps" for data set ZYXWVUT.EMPINFO.DATA.NOTE:The passwords on ZYXWVUT.EMPINFO.DATA were
successfully modified.NOTE:Successfully added new secured table object "EMPLOYEES.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/ZYXWVUTEmps" for data set
ZYXWVUT.EMPLOYEES.DATA.NOTE:The passwords on ZYXWVUT.EMPLOYEES.DATA were successfully
modified.NOTE:Successfully added new secured table object "PRODUCT.DATA" to the secured library object
at path "/System/Secured Libraries/Department XYZZY/ZYXWVUTEmps" for data set
ZYXWVUT.PRODUCT.DATA.NOTE:The passwords on ZYXWVUT.PRODUCT.DATA were successfully modified.86   quit;
```

例 2: パスワードで保護されたデータセットを格納する物理ライブラリのバインド

要素: PROC AUTHLIB ステートメントオプション
CREATE ステートメントオプション:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=

詳細

この例では、READ=、WRITE=、ALTER=の同じパスワードをもつ 2 つのデータセットとパスワードをもたない 1 つのデータセットが物理ライブラリに含まれている場合、同様の CREATE ステートメントを使用するとどうなるか、例 1 として説明します。どのデータセットも AES で暗号化されていません。

プログラム

```
proc authlib lib=abcde;

    create securedfolder="Department XYZZY"
        securedlibrary="ABCDEEmps"
        pw=secretpw;
run;
quit;
```

プログラムの説明

ライブラリ ABCDE には、Employees、EmpInfo、Product という各データセットが格納されています。 ただし、ライブラリ ABCDE では、Employees および EmpInfo データセットは、ステートメントによってライブラリが保護される前に、READ=パスワード **abcd**、WRITE=パスワード **efgh**、ALTER=パスワード **ijkl** で保護されています。3 つ目のデータセット Product はパスワードで保護されていません。

```
proc authlib lib=abcde;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 PW=オプションでメタデータバインド型ライブラリのパスワードを指定します。

```
    create securedfolder="Department XYZZY"
        securedlibrary="ABCDEEmps"
        pw=secretpw;
run;
quit;
```

結果:RANK プロシジャ ABCDE ライブラリがバインドされ、保護されていない Product データセットがバインドされ、パスワードが設定されました。保護されたデータセットはバインドされず、現在のパスワードは指定されていないため変更されませんでした。

ログの例

ログ 7.3 パスワード保護されたデータセット

```
179  proc authlib lib=abcde; 180 181   create securedfolder="Department XYZZY" 182
securedlibrary="ABCDEEmps" 183          pw=XXXXXXXX; 184 185  run; NOTE:Successfully created a secured
library object for the physical library ABCDE and recorded its location as:SecuredFolder:      /System/
Secured Libraries/Department XYZZY SecuredLibrary:     ABCDEEmps SecuredLibraryGUID:4881263D-C346-41F7-
AC49-BF9181AF13D2 ERROR:The ALTER password is the most restrictive on ABCDE.EMPINFO.DATA.You must
supply its value in order to alter or add any passwords.ERROR:The ALTER password is the most
restrictive on ABCDE.EMPLOYEES.DATA.You must supply its value in order to alter or add any
passwords.NOTE:Successfully added new secured table object "PRODUCT.DATA" to the secured library
object at path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set
ABCDE.PRODUCT.DATA.NOTE:The passwords on ABCDE.PRODUCT.DATA were successfully modified.NOTE:Some
statement actions not processed because of errors noted above.186  quit; NOTE:The SAS System stopped
processing this step because of errors.
```

例 3: 既存データセットが同じパスワードで保護されている場合のライブラリのバインド

要素: PROC AUTHLIB ステートメントオプション
CREATE ステートメントオプション:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=

詳細

この例では、PROC AUTHLIB CREATE ステートメントで前述の例の Employees および EmpInfo データセットのパスワードを指定する方法について説明します。どのデータセットも AES で暗号化されていません。

プログラム

```
proc authlib lib=abcde;

    create securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=ijkl/secretpw;
run;
quit;
```

プログラムの説明

ライブラリ ABCDE には、Employees、EmpInfo、Product という各データセットも格納されています。 ただし、ライブラリ ABCDE では、Employees および EmpInfo データセットは、ステートメントによってライブラリが保護される前に、READ=パスワード **abcd**、WRITE=パスワード **efgh**、ALTER=パスワード **ijkl** で保護されています。3 つ目のデータセット Product はパスワードで保護されていません。

```
proc authlib lib=abcde;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 データセットの ALTER=パスワード `ijkl` は、PW=引数内で新しいパスワード *secretpw* の前にスラッシュ(/)で区切って指定します。

```
      create securedlibrary="ABCDEEmps"
          securedfolder="Department XYZZY"
          pw=ijkl/secretpw;
run;
quit;
```

結果:RANK プロシジャ ライブラリ ABCDE がバインドされます。3 つのデータセットはすべて同じパスワード *secretpw* によってバインドされます。

ログの例

ログ 7.4 *同じパスワードで保護されているデータセットが格納されたライブラリの保護*

```
39   proc authlib lib=abcde; 40   create securedlibrary="ABCDEEmps" 41   securedfolder="Department
XYZZY" 42   pw=XXXX/XXXXXXXX; 43   run; NOTE:Successfully created a secured library object for the
physical library ABCDE and recorded its location as:SecuredFolder:      /System/Secured Libraries/
Department XYZZY SecuredLibrary:     ABCDEEmps SecuredLibraryGUID:9F746F86-2336-4E2F-A67E-BFB77DEC27F0
NOTE:Successfully added new secured table object "DEPTNAME.DATA" to the secured library object at path
"/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.DEPTNAME.DATA.NOTE:The
passwords on ABCDE.DEPTNAME.DATA were successfully modified.NOTE:Successfully added new secured table
object "EMPINFO.DATA" to the secured library object at path "/System/Secured Libraries/Department
XYZZY/ABCDEEmps" for data set ABCDE.EMPINFO.DATA.NOTE:The passwords on ABCDE.EMPINFO.DATA were
successfully modified.NOTE:Successfully added new secured table object "EMPLOYEE.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set
ABCDE.EMPLOYEE.DATA.NOTE:The passwords on ABCDE.EMPLOYEE.DATA were successfully modified.44   quit;
```

例 4: 既存のデータセットが別のパスワードで保護されている場合のライブラリのバインド

要素: PROC AUTHLIB ステートメントオプション
- CREATE ステートメントオプション:
- ALTER=
- READ=
- SECUREDLIBRARY=
- SECUREDFOLDER=
- WRITE=
- TABLE ステートメントオプション:
- ALTER=
- PW=
- READ=
- WRITE=

詳細

この例では、パスワードの異なる 3 つのデータセットが格納されたライブラリ KLMNO をバインドする方法について説明します。どのデータセットも AES で暗号化されていま

せん。READ=、WRITE=、ALTER=の各パスワードオプションを指定して、メタデータバインド型ライブラリの長いパスワードを作成する方法についても説明します。

プログラム

```
proc authlib lib=klmno;

    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        read=abcdefgh
        write=ijklmno
        alter=pqrstuvw;

    tables employees /
        pw=lmno;
    tables empinfo /
        read=abcd
        write=efgh
        alter=ijkl;
    tables product;
run;
quit;
```

プログラムの説明

ライブラリ KLMNO には、Employees、EmpInfo、Product という各データセットが格納されます。 Employees データセットは PW=パスワード **lmno** で保護されます。EmpInfo データセットは、READ=パスワード **abcd**、WRITE=パスワード **efgh**、ALTER=パスワード **ijkl** によって保護されます。Product データセットは保護されません。

```
proc authlib lib=klmno;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 メタデータバインド型ライブラリの長いパスワードを作成するには、READ=パスワード **abcdefgh**、WRITE=パスワード **ijklmno**、ALTER=パスワード **pqrstuvw** の各値を指定します。

```
    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        read=abcdefgh
        write=ijklmno
        alter=pqrstuvw;
```

TABLES ステートメントを使用して、各データセットの現在のパスワードを指定します。 TABLES ステートメントを使用する場合、すべてのデータセットに TABLES ステートメントを指定する必要があります。

```
    tables employees /
        pw=lmno;
    tables empinfo /
        read=abcd
        write=efgh
        alter=ijkl;
    tables product;
run;
quit;
```

結果:RANK プロシジャ ライブラリ KLMNO がバインドされ、3 つのデータセットがすべて同じパスワードでバインドされます。パスワードは、READ=パスワード **abcdefgh**、WRITE=パスワード **ijklmno**、ALTER=パスワード **pqrstuvw** です。

ログの例

ログ 7.5 *別のパスワードで保護された既存データセットが格納されるライブラリの保護*

```
177  libname klmno "c:\lib2"; NOTE:Libref KLMNO was successfully assigned as follows:Engine:        V9
Physical Name: c:\lib2 178 179  proc authlib lib=klmno; 180  create securedlibrary="KLMNOEmps" 181
securedfolder="Department XYZZY" 182  read=XXXXXXXX 183  write=XXXXXXX 184  alter=XXXXXXXX; 185
tables employees / 186  pw=XXXX; 187  tables empinfo / 188  read=XXXX 189  write=XXXX 190  alter=XXXX;
191  tables product; 192  run; NOTE:Successfully created a secured library object for the physical
library KLMNO and recorded its location as:SecuredFolder:      /System/Secured Libraries/Department
XYZZY SecuredLibrary:     KLMNOEmps SecuredLibraryGUID:BC74E81F-E86B-402E-8C16-F9A94A078F81
NOTE:Successfully added new secured table object "EMPLOYEES.DATA" to the secured library object at
path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.EMPLOYEES.DATA.NOTE:The
passwords on KLMNO.EMPLOYEES.DATA were successfully modified.NOTE:Successfully added new secured table
object "EMPINFO.DATA" to the secured library object at path "/System/Secured Libraries/Department
XYZZY/KLMNOEmps" for data set KLMNO.EMPINFO.DATA.NOTE:The passwords on KLMNO.EMPINFO.DATA were
successfully modified.NOTE:Successfully added new secured table object "PRODUCT.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
KLMNO.PRODUCT.DATA.NOTE:The passwords on KLMNO.PRODUCT.DATA were successfully modified.193  quit;
```

例 5: データセットでのパスワードの変更

要素: PROC AUTHLIB ステートメントオプション
MODIFY ステートメントオプション:
PW=
TABLESONLY=
TABLES ステートメントオプション:
PW=

詳細

この例では、メタデータバインド型ライブラリのパスワードを一致させるため、既存データセットのパスワードを変更するための別のアプローチを示します。MODIFY ステートメントを使用します。ここで、MODIFY ステートメントは、メタデータバインド型ライブラリのパスワードに一致させるため、例 2 (138 ページ) の Employees および EmpInfo データセットのデータセットパスワードの変更に使用されます。どのデータセットも AES で暗号化されていません。

また、MODIFY ステートメントは、ライブラリのバインド完了後にオペレーティングシステムのコマンドによってメタデータ連結ライブラリにコピーされるデータセットのパスワードの変更にも使用されます。

プログラム

```
proc authlib lib=abcde;

    modify tablesonly=yes
        pw=secretpw;
```

```
    tables _all_ /
        pw=ijkl/secretpw;
run;
quit;
```

プログラムの説明

ライブラリ ABCDE には、Employees、EmpInfo、Product という各データセットが格納されています。ライブラリはメタデータバインド型ライブラリのパスワード *secretpw* によってバインドされます。ただし、ライブラリ ABCDE では、Employees および EmpInfo データセットはライブラリにバインドされず、ALTER=パスワード **ijkl** で保護されます。3 つ目のデータセット Product はすでにバインドされています。

```
proc authlib lib=abcde;
```

MODIFY ステートメントは、Employees および EmpInfo データセットのパスワードをメタデータバインド型ライブラリのパスワードに合わせて変更する場合に使用します。TABLESONLY= ステートメントは、テーブルパスワードのみを変更する場合に指定します。

```
    modify tablesonly=yes
        pw=secretpw;
```

TABLES ステートメントを指定する必要があります。既存データセットの ALTER パスワードは、メタデータバインド型パスワードの前の PW=引数内で指定され、TABLES ステートメント内でスラッシュ (/)によって区切られます。

```
    tables _all_ /
        pw=ijkl/secretpw;
run;
quit;
```

結果:RANK プロシジャ 3 つのデータセットがすべてパスワード *secretpw* によってバインドされます。

ログの例

ログ 7.6 *データセットパスワードの変更*

```
76   proc authlib lib=abcde; 77   modify tablesonly=yes 78   pw=XXXXXXXX; 79   tables _all_ / 80
pw=XXXX/XXXXXXXX; 81   run; NOTE:The passwords on ABCDE.DEPTNAME.DATA do not require
modification.NOTE:The passwords on ABCDE.EMPINFO.DATA do not require modification.NOTE:The passwords
on ABCDE.EMPLOYEE.DATA do not require modification.82   quit;
```

例 6: メタデータバインド型ライブラリパスワードの変更

要素: PROC AUTHLIB ステートメントオプション
MODIFY ステートメントオプション:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=

詳細

この例では、メタデータバインド型ライブラリのパスワードが漏洩したと考えられる場合、MODIFY ステートメントを使用してライブラリのパスワードを変更する方法について説明します。次のコードは、指定したパスワードを使用するライブラリ内のすべてのデータセット、あるいはパスワードのないライブラリ内のすべてのデータセットの、ライブラリパスワードとデータセットパスワードを変更するものです。この例では、データセットは AES で暗号化されません。ライブラリに AES で暗号化されたデータが含まれている場合は、後の例を参照してください。

プログラム

```
proc authlib lib=abcde;

    modify securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secretpw/new-password;
run;
quit;
```

プログラムの説明

ライブラリ ABCDE はパスワード変更が必要です。

```
proc authlib lib=abcde;
```

ライブラリのパスワードとデータセットのパスワードを変更するには、MODIFY ステートメントを使用します。 保護付きライブラリオブジェクトの名前とメタデータフォルダの名前はオプションですが、名前を指定すれば、ライブラリが保護付きライブラリオブジェクトにバインドされていることを変更前に確認できます。これは SAS Management Console が Modify アクションからコードをサブミットするとき、オペレーティングシステムの正しいライブラリパスが指定されたことを確認するために使用します。

```
    modify securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secretpw/new-password;
run;
quit;
```

結果:RANK プロシジャ ライブラリ ABCDE はバインドされたまま、そのライブラリパスワードが *new-password* に変更されます。3 つのデータセットがすべてバインドされたまま、それらのパスワードが *new-password* によって変更されます。*secretpw* 以外のパスワードをもつデータセットがあれば、SAS ログにエラーメッセージが表示されます。

ログの例

ログ 7.7 メタデータバインド型ライブラリパスワードの変更

```
217  proc authlib lib=abcde; 218    modify securedlibrary="ABCDEEmps" 219
securedfolder="Department XYZZY" 220        pw=XXXXXXXX/XXXXXXXX; 221 222  run; NOTE:The passwords for
the secured library object with path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" were
successfully modified."NOTE:The passwords on ABCDE.EMPINFO.DATA were successfully modified.NOTE:The
passwords on ABCDE.EMPLOYEES.DATA were successfully modified.NOTE:The passwords on ABCDE.PRODUCT.DATA
were successfully modified.223  quit;
```

例 7: REMOVE ステートメントの使い方

要素: PROC AUTHLIB ステートメントオプション
REMOVE ステートメントオプション:
PW=

詳細

この例では、メタデータ連結ライブラリのバインドを解除する方法について説明します。このコードでは次が実行されます。

- ライブラリとそのテーブルを記述するメタデータを SAS メタデータリポジトリから削除します
- 物理ライブラリとデータセットからセキュリティバインドを削除します
- 割り当てたパスワードをデータセットから削除し、保護しないまま残します

パスワードの後のスラッシュ(/)はオプションで、データセットからパスワードを削除または置換するために使用されます。例 4 (141 ページ),に示すように、ライブラリが READ=、WRITE=、ALTER=の各パスワードによってバインドされている場合、すべてのパスワードを指定する必要があり、各パスワードにはスラッシュ(/)が必要です。どのデータセットも AES で暗号化されていません。

プログラム

```
proc authlib lib=abcde;

    remove
        pw=currntpw/;
run;
quit;
```

プログラムの説明

メタデータバインド型ライブラリ ABCDE のバインド解除

```
proc authlib lib=abcde;
```

メタデータバインド型ライブラリのバインドを解除するには、REMOVE ステートメントを使用します。データセットからパスワードを削除するには、パスワードの後のスラッシュ(/)を使用します。

```
    remove
        pw=currntpw/;
run;
quit;
```

結果:RANK プロシジャ ライブラリ ABCDE とそれにバインドされたすべてのデータセットは、バインドが解除されます。バインドを解除されたデータセットからすべてのパスワードが削除され、保護が解除されます。

ログの例

ログ 7.8 *メタデータ連結ライブラリのバインド解除*

```
195  proc authlib lib=abcde; 196  remove 197  pw=XXXXXXXX/; 198  run; WARNING:Some or all the
passwords on ABCDE.DEPTNAME.DATA were removed along with the secured library object location, leaving
the data set unprotected.NOTE:The secured table object location for ABCDE.DEPTNAME.DATA was
successfully removed.WARNING:Some or all the passwords on ABCDE.EMPINFO.DATA were removed along with
the secured library object location, leaving the data set unprotected.NOTE:The secured table object
location for ABCDE.EMPINFO.DATA was successfully removed.WARNING:Some or all the passwords on
ABCDE.EMPLOYEE.DATA were removed along with the secured library object location, leaving the data set
unprotected.NOTE:The secured table object location for ABCDE.EMPLOYEE.DATA was successfully
removed.NOTE:Successfully deleted the secured library object that was located at:SecuredFolder:      /
System/Secured Libraries/Department XYZZY SecuredLibrary:     ABCDEEmps SecuredLibraryGUID:
9F746F86-2336-4E2F-A67E-BFB77DEC27F0 NOTE:Successfully deleted the recorded location of the secured
library object for the physical library ABCDE.199  quit;
```

例 8: REPORT ステートメントの使い方

要素: PROC AUTHLIB ステートメントオプション
REPORT ステートメント

詳細

この例では、ライブラリのバインドを確認する方法について説明します。

プログラム

```
proc authlib lib=abcde;

    report;
run;
quit;
```

プログラムの説明

メタデータバインド型ライブラリ ABCDE のバインドを確認します。

```
proc authlib lib=abcde;
```

REPORT ステートメントを使用します。

```
    report;
run;
quit;
```

結果:RANK プロシジャ REPORT ステートメントの結果については、“出力例” (148 ページ)を参照してください。

ログの例

ログ 7.9 *レポートの作成*

```
49   proc authlib lib=abcde; 50    report; 51   run; 52   quit;
```

出力例

アウトプット 7.2 *ABCDE ライブラリの REPORT ステートメント結果*

The OS library is properly registered to this SecuredLibrary. These data sets are properly registered to SecuredTables in it.

SecuredLibrary Path: /System/Secured Libraries/Department XYZZY/ABCDEEmps

SecuredLibrary Guid: F25E6004-EF15-4792-B0BD-9A8499435A07

Registered in OS Path: C:\lib1

Password Set: 0

MemberName	MemberType	SecuredTableName	SecuredTableGUID
EMPINFO	DATA	EMPINFO.DATA	C8EA8780-83E8-4F32-9912-14F109FA1D50
EMPLOYEES	DATA	EMPLOYEES.DATA	A0F173F6-9D0B-40A0-B38F-C56433CE22E1
PRODUCT	DATA	PRODUCT.DATA	5057208E-7EB0-4090-BD6D-57EA856DEA8B

例 9: TABLES ステートメントの使用

要素: PROC AUTHLIB ステートメントオプション
CREATE ステートメントオプション:
ALTER=
READ=
SECUREDLIBRARY=
SECUREDFOLDER=
WRITE=
TABLE ステートメントオプション:
ALTER=
PW=
READ=
WRITE=

詳細

例 4 (141 ページ) では、TABLES ステートメントの使い方を説明します。

例 10: 既存データセットが SAS Proprietary で暗号化されている場合のライブラリのバインド

要素: PROC AUTHLIB ステートメントオプション
CREATE ステートメントオプション:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=
TABLES ステートメントオプション:
PW=
READ=

詳細

次の例では、SAS Proprietary で暗号化されたデータセットのバインドとパスワード変更の方法を説明します。

プログラム

```
proc authlib lib=klmno;

    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        pw=pqrstuvw;

    tables employees /
        pw=lmno;
    tables empinfo /
        read=abcd;
    tables product;
run;
quit;
```

プログラムの説明

ライブラリ KLMNO には、Employees、EmpInfo、Product という 3 つのデータセットが格納されます。 このライブラリでは、Employees データセットは PW=パスワードで保護されます **lmno**.EmpInfo データセットは、READ=パスワード **abcd** で保護されます。Employees と EmpInfo のどちらのデータセットも、SAS Proprietary で暗号化されます。Product データセットは保護されません。

```
proc authlib lib=klmno;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 ライブラリのパスワードを **pqrstuvw** に設定します。

```
    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        pw=pqrstuvw;
```

これらのデータセットはパスワードが異なるため、それらのパスワードを変更するには、すべてのデータセットの TABLES ステートメントを指定する必要があります。

```
        tables employees /
            pw=lmno;
        tables empinfo /
            read=abcd;
        tables product;
    run;
    quit;
```

結果:RANK プロシジャ ライブラリ KLMNO がバインドされます。3 つのデータセットはすべてバインドされ、同じ PW=パスワード **pqrstuvw** を使用します。データセット Employees および EmpInfo はコピー置換され、パスワード **pqrstuvw** で暗号化されます。データセット Product はバインドされますが、暗号化はされません。

ログの例

ログ 7.10 *SAS Proprietary データセットを格納する KLMNO ライブラリの TABLES ステートメント*

```
265  proc authlib lib=klmno; 266  create securedlibrary="KLMNOEmps" 267  securedfolder="Department
XYZZY" 268  pw=XXXXXXXX; 269  tables employees / 270  pw=XXXX; 271  tables empinfo / 272  read=XXXX;
273  tables product; 274  run; NOTE:Successfully created a secured library object for the physical
library KLMNO and recorded its location as:SecuredFolder:      /System/Secured Libraries/Department
XYZZY SecuredLibrary:     KLMNOEmps SecuredLibraryGUID:E71881CD-8C54-4E21-A8B5-FD7D4FBDAA7D
NOTE:Copying data set KLMNO.EMPLOYEES in place to encrypt with the new secured library passwords or
encryption options.NOTE:Renaming the data set KLMNO.EMPLOYEES to
KLMNO.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__ to
KLMNO.EMPLOYEES.NOTE:Metadata-bound library permissions are used for
KLMNO.EMPLOYEES.DATA.NOTE:Successfully added new secured table object "EMPLOYEES.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
KLMNO.EMPLOYEES.DATA.NOTE:There were 5 observations read from the data set
KLMNO.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set KLMNO.EMPLOYEES has 5 observations and 6
variables.NOTE:Deleting the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
KLMNO.EMPLOYEES.DATA were successfully modified.NOTE:Copying data set KLMNO.EMPINFO in place to
encrypt with the new secured library passwords or encryption options.NOTE:Renaming the data set
KLMNO.EMPINFO to KLMNO.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
KLMNO.__TEMP_ENCRYPT_FILE_NAME__ to KLMNO.EMPINFO.NOTE:Metadata-bound library permissions are used for
KLMNO.EMPINFO.DATA.NOTE:Successfully added new secured table object "EMPINFO.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
KLMNO.EMPINFO.DATA.NOTE:There were 5 observations read from the data set
KLMNO.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set KLMNO.EMPINFO has 5 observations and 6
variables.NOTE:Deleting the data set KLMNO.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
KLMNO.EMPINFO.DATA were successfully modified.NOTE:The passwords on KLMNO.PRODUCT.DATA do not require
modification.NOTE:Successfully added new secured table object "PRODUCT.DATA" to the secured library
object at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.PRODUCT.DATA.
275  quit;
```

例 11: 既存データセットが AES で暗号化される場合のライブラリのバインド

要素: PROC AUTHLIB ステートメントオプション
CREATE ステートメントオプション:
PW=
SECUREDLIBRARY=
SECUREDFOLDER=

TABLES ステートメントオプション:
ENCRYPTKEY=

詳細

この例では、AES で暗号化されたデータセットをバインドする方法について説明します。どのデータセットにもパスワードはありません。

注意:

この例のように、メタデータバインド型ライブラリには暗号化キーの異なる AES 暗号化データセットを入れないことを強く推奨します。 デフォルトの暗号化キーをメタデータに記録し、すべての AES 暗号化データセットを変換して、そのキーを使用することをお勧めします。そうすれば、ユーザーもプログラムでも、データセットを開くときにキーを指定する必要がなくなります。この例に続いて、次の例ではこのプロセスの動作を説明します。

プログラム

```
proc authlib lib=klmno;

    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        pw=pqrstuvw;

    tables employees /
        encryptkey=lmno;
    tables empinfo /
        encryptkey=abcd;
    tables product;
run;
quit;
```

プログラムの説明

ライブラリ KLMNO には、Employees、EmpInfo、Product という 3 つのデータセットが格納されます。 このライブラリでは、Employees データセットは AES で暗号化され、その ENCRYPTKEY=値は **lmno** です。EmpInfo データセットは AES で暗号化され、その ENCRYPTKEY=値は **abcd** です。Product データセットは保護されません。

```
proc authlib lib=klmno;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 ライブラリのパスワードを **pqrstuvw** に設定します。

```
    create securedlibrary="KLMNOEmps"
        securedfolder="Department XYZZY"
        pw=pqrstuvw;
```

TABLES ステートメントを使用して、各データセットの暗号化キーを指定します。 TABLES ステートメントは、すべてのデータセットに指定する必要があります。

```
    tables employees /
        encryptkey=lmno;
```

```
    tables empinfo /
        encryptkey=abcd;
    tables product;
run;
quit;
```

結果:RANK プロシジャ ライブラリ KLMNO がバインドされます。3 つのデータセットがすべてバインドされます。Employees と EmpInfo のどちらのデータセットも、AES で暗号化されます。Product データセットは暗号化されません。Employees および Empinfo データセットの暗号化キー値は異なります。この例のように、メタデータバインド型ライブラリには暗号化キーの異なる AES 暗号化データセットを入れないことを強く推奨します。

ログの例

ログ 7.11 *AES で暗号化されたデータセットを格納する KLMNO ライブラリの TABLES ステートメント*

```
351  proc authlib lib=klmno; 352  create securedlibrary="KLMNOEmps" 353  securedfolder="Department
XYZZY" 354  pw=XXXXXXXX; 355  tables employees / 356  encryptkey=XXXX; 357  tables empinfo / 358
encryptkey=XXXX; 359  tables product; 360  run; NOTE:Successfully created a secured library object for
the physical library KLMNO and recorded its location as:SecuredFolder:      /System/Secured Libraries/
Department XYZZY SecuredLibrary:     KLMNOEmps SecuredLibraryGUID:48E2C4C7-ADE1-49D2-BBFE-14E5EAAB8961
NOTE:Successfully added new secured table object "EMPLOYEES.DATA" to the secured library object at
path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set KLMNO.EMPLOYEES.DATA.NOTE:The
passwords on KLMNO.EMPLOYEES.DATA were successfully modified.NOTE:Successfully added new secured table
object "EMPINFO.DATA" to the secured library object at path "/System/Secured Libraries/Department
XYZZY/KLMNOEmps" for data set KLMNO.EMPINFO.DATA.NOTE:The passwords on KLMNO.EMPINFO.DATA were
successfully modified.NOTE:Successfully added new secured table object "PRODUCT.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/KLMNOEmps" for data set
KLMNO.PRODUCT.DATA.NOTE:The passwords on KLMNO.PRODUCT.DATA were successfully modified.361  quit;
```

例 12: 既存の AES 暗号化データセットの暗号化キーが異なる場合、オプションの記録された暗号化キーでライブラリをバインドする方法

要素: PROC AUTHLIB ステートメントオプション
CREATE ステートメントオプション:
ENCRYPT=
ENCRYPTKEY=
PW=
SECUREDLIBRARY=
SECUREDFOLDER=
TABLES ステートメントオプション:
ENCRYPT=
ENCRYPTKEY=

詳細

この例では、オプションの記録された暗号化キーでライブラリをバインドする方法について説明します。どのデータセットにもパスワードはありません。

ENCRYPTKEY=DEF の EmpInfo データセットを作成して参照している SAS コードが存在し、記録されたライブラリキーは必要ないため、コードから ENCRYPTKEY=DEF

の指定を削除します。データを再作成するコードは、データセットの再作成時にオプションの記録されたキーが使用されるように、ENCRYPT=AES オプションを保持する必要があります。

プログラム

```
proc authlib lib=abcde;

    create securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        encrypt=aes
        encryptkey=optionalkey;


    tables employee;
    tables empinfo /
        encryptkey=def/optionalkey
        encrypt=aes;
    tables deptname;
run;
quit;
```

プログラムの説明

ライブラリ ABCDE には、Employees、EmpInfo、DeptName という各データセットが格納されます。 このライブラリでは、EmpInfo データセットは AES で暗号化され、その ENCRYPTKEY=値は **def** です。

```
proc authlib lib=abcde;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 オプションの暗号化キーは、メタデータバインド型ライブラリに指定します。

```
    create securedlibrary="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        encrypt=aes
        encryptkey=optionalkey;
```

各データセットに TABLES ステートメントが必要です。

```
    tables employee;
    tables empinfo /
        encryptkey=def/optionalkey
        encrypt=aes;
    tables deptname;
run;
quit;
```

結果:RANK プロシジャ ABCDE ライブラリがバインドされ、オプションの暗号化キーが保存されます。ステートメントが実行されると、3 つのデータセットは次のようになります。Employee データセットは、新しいメタデータバインド型ライブラリのパスワードによって更新されますが、暗号化はされません。DeptName データセットは、メタデータバイ

ンド型ライブラリのパスワードによって更新されますが、暗号化はされません。EmpInfo データセットはコピーされ、記録されたオプションのキーで再暗号化され、新しいメタデータバインド型ライブラリのパスワードを取得します。次のプログラムでは、EmpInfo の TABLES ステートメントに現在および新規のオプションのキーを両方入力する必要があります。新しいキーを指定しないと、データセットは **def** キーで暗号化されたままとなります。

ログの例

ログ 7.12 *暗号化キー値を記録された暗号化キーに変更する方法*

```
467  libname abcde "c:\lib1"; NOTE:Libref ABCDE was successfully assigned as follows:Engine:        V9
Physical Name: c:\lib1 468 469  proc authlib lib=abcde; 470  create securedlibrary="ABCDEEmps" 471
securedfolder="Department XYZZY" 472  pw=XXXXXX 473  encrypt=aes 474  encryptkey=XXXXXXXXXXX; 475
tables employee; 476  tables empinfo / 477  encryptkey=XXX/XXXXXXXXXXX 478  encrypt=aes; 479  tables
deptname; 480  run; NOTE:Successfully created a secured library object for the physical library ABCDE
and recorded its location as:SecuredFolder:      /System/Secured Libraries/Department XYZZY
SecuredLibrary:     ABCDEEmps SecuredLibraryGUID:8E683650-B306-4871-A92D-16D481EC6456
NOTE:Successfully added new secured table object "EMPLOYEE.DATA" to the secured library object at path
"/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set ABCDE.EMPLOYEE.DATA.NOTE:The
passwords on ABCDE.EMPLOYEE.DATA were successfully modified.NOTE:Copying data set ABCDE.EMPINFO in
place to encrypt with the new secured library passwords or encryption options.NOTE:Renaming the data
set ABCDE.EMPINFO to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.NOTE:Metadata-bound library permissions are used for
ABCDE.EMPINFO.DATA.NOTE:Successfully added new secured table object "EMPINFO.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set
ABCDE.EMPINFO.DATA.NOTE:There were 5 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.EMPINFO has 5 observations and 6
variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.EMPINFO.DATA were successfully modified.NOTE:Successfully added new secured table object
"DEPTNAME.DATA" to the secured library object at path "/System/Secured Libraries/Department XYZZY/
ABCDEEmps" for data set ABCDE.DEPTNAME.DATA.NOTE:The passwords on ABCDE.DEPTNAME.DATA were
successfully modified.480  quit;
```

例 13: 既存のデータセットが同じ暗号化キーで暗号化されている場合、AES 暗号化が必須のライブラリをバインドする方法

要素: PROC AUTHLIB ステートメントオプション
CREATE ステートメントオプション:
ENCRYPT=
ENCRYPTKEY=
PW=
REQUIRE_ENCRYPTION
SECUREDLIBRARY=
SECUREDFOLDER=

詳細

この例では、このメタデータバインド型ライブラリ内のすべてのデータセットが AES で暗号化されて同じ暗号化キーをもつ必要があるライブラリのバインド方法について説明します。

プログラム

```
proc authlib lib=abcde;

    create  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=abc ;
run;
quit;
```

プログラムの説明

ライブラリ ABCDE には、Employees、EmpInfo、DeptName という 3 つのデータセットが格納されます。 データセット EmpInfo の暗号化キー値は **abc** です。他の 2 つのデータセットは AES で暗号化されません。どのデータセットにもパスワードはありません。

```
proc authlib lib=abcde;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 REQUIRE_ENCRYPTION=YES は、メタデータバインド型ライブラリ内のすべてのデータセットが自動的に AES で暗号化されるように指定します。

```
    create  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=abc ;
run;
quit;
```

結果:RANK プロシジャ ライブラリ ABCDE がバインドされ、すべてのデータセットがバインドされ、同じ暗号化キーによって AES で暗号化されます。

ログの例

ログ 7.13 *データセットが同じ暗号化キーですでに暗号化されている場合、AES 暗号化を必要とするライブラリ ABCDE*

```
40   proc authlib lib=abcde; 41   create  seclib="ABCDEEmps" 42         securedfolder="Department
XYZZY" 43         pw=XXXXXX 44         require_encryption=yes 45         encrypt=aes 46
encryptkey=XXX ; 47   run; NOTE:Setting library to require encryption.NOTE:Required encryption will
use AES encryption with the recorded key.NOTE:Successfully created a secured library object for the
physical library ABCDE and recorded its location as:SecuredFolder:      /System/Secured Libraries/
Department XYZZY SecuredLibrary:     ABCDEEmps SecuredLibraryGUID:9FD6C5D9-EF00-4CDC-8D0A-348D08BB329E
NOTE:Copying data set ABCDE.DEPTNAME in place to do required encryption with the library's required
encryption key and passwords.NOTE:Renaming the data set ABCDE.DEPTNAME to
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to
ABCDE.DEPTNAME.NOTE:Metadata-bound library permissions are used for
ABCDE.DEPTNAME.DATA.NOTE:Successfully added new secured table object "DEPTNAME.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set
ABCDE.DEPTNAME.DATA.NOTE:There were 10 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.DEPTNAME has 10 observations and 2
variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.DEPTNAME.DATA were successfully modified.NOTE:Successfully added new secured table object
"EMPINFO.DATA" to the secured library object at path "/System/Secured Libraries/Department XYZZY/
ABCDEEmps" for data set ABCDE.EMPINFO.DATA.NOTE:The passwords on ABCDE.EMPINFO.DATA were successfully
modified.NOTE:Copying data set ABCDE.EMPLOYEE in place to do required encryption with the library's
required encryption key and passwords.NOTE:Renaming the data set ABCDE.EMPLOYEE to
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to
ABCDE.EMPLOYEE.NOTE:Metadata-bound library permissions are used for
ABCDE.EMPLOYEE.DATA.NOTE:Successfully added new secured table object "EMPLOYEE.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set
ABCDE.EMPLOYEE.DATA.NOTE:There were 22 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.EMPLOYEE has 22 observations and 11
variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.EMPLOYEE.DATA were successfully modified.48   quit;
```

例 14: AES 暗号化を必要とするメタデータ連結ライブラリで暗号化キーを変更する方法

要素: PROC AUTHLIB ステートメントオプション
MODIFY ステートメントオプション:
ENCRYPT=
ENCRYPTKEY=
PW=

詳細

この例では、メタデータバインド型ライブラリの暗号化キーが漏洩したと考えられる場合、MODIFY ステートメントを使用してライブラリの暗号化キーを変更する方法について説明します。

プログラム

```
proc authlib lib=abcde;

    modify
       pw=secret
       encrypt=aes
       encryptkey=/new;
```

```
run;
quit;
```

プログラムの説明

ライブラリ ABCDE には、Employees、EmpInfo、DeptName という 3 つのデータセットが格納されます。 メタデータバインド型ライブラリには AES 暗号化が必要なため、このライブラリでは、すべてのデータセットが暗号化キー値 **abc** によって AES で暗号化されます。

```
proc authlib lib=abcde;
```

ライブラリの暗号化キーとデータセットの暗号化キーを変更するには、MODIFY ステートメントを使用します。 ENCRYPT=AES を指定する必要があります。

```
      modify
         pw=secret
         encrypt=aes
         encryptkey=/new;

run;
quit;
```

結果:RANK プロシジャ ライブラリ ABCDE は、同じパスワードと新しい暗号化キーによってバインドされたままです。3 つのデータセットはすべて、同じパスワードと新しい暗号化キーによってバインドされたままです。データセットがコピー置換され、新しいキー値によって暗号化されます。

ログ 7.14 暗号化キー ABCDE ライブラリの変更

```
502  proc authlib lib=abcde; 503  modify 504  pw=XXXXXX 505  encrypt=aes 506
encryptkey=/XXX; 507  run; NOTE:Changing the required encryption key.NOTE:The
passwords on ABCDE.DEPTNAME.DATA do not require modification.NOTE:Copying data
set ABCDE.DEPTNAME in place to do required encryption with the library's
required encryption key and passwords.NOTE:Renaming the data set ABCDE.DEPTNAME
to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.NOTE:There were 4
observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The
data set ABCDE.DEPTNAME has 4 observations and 2 variables.NOTE:Deleting the
data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.EMPINFO.DATA do not require modification.NOTE:Copying data set
ABCDE.EMPINFO in place to do required encryption with the library's required
encryption key and passwords.NOTE:Renaming the data set ABCDE.EMPINFO to
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.NOTE:There were 5 observations
read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set
ABCDE.EMPINFO has 5 observations and 6 variables.NOTE:Deleting the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on ABCDE.EMPLOYEE.DATA do
not require modification.NOTE:Copying data set ABCDE.EMPLOYEE in place to do
required encryption with the library's required encryption key and
passwords.NOTE:Renaming the data set ABCDE.EMPLOYEE to
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.NOTE:There were 5
observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The
data set ABCDE.EMPLOYEE has 5 observations and 6 variables.NOTE:Deleting the
data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords and/or encryption
options for the secured library object with path "/System/Secured Libraries/
Department XYZZY/ABCDEEmps" were successfully modified."NOTE:All data sets in
library ABCDE are properly protected with the metadata-bound library passwords
and encryption options.Replaced Passwords and encryption keys were
purged.NOTE:Purged 1 versions of the replaced passwords and encryption keys
older than 2015-05-04T15:40:57-05:00.508  quit; NOTE:Renaming the data set
ABCDE.EMPLOYEE to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.NOTE:There were 22
observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The
data set ABCDE.EMPLOYEE has 22 observations and 11 variables.NOTE:Deleting the
data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords and/or encryption
options for the secured library object with path "/System/Secured Libraries/
Department XYZZY/ABCDEEmps" were successfully modified.
```

例 15: 異なる暗号化キーによって AES で暗号化された既存のデータセットをもつライブラリのバインド

要素: PROC AUTHLIB ステートメントオプション
- CREATE ステートメントオプション:
- ENCRYPT=
- ENCRYPTKEY=
- PW=
- REQUIRE_ENCRYPTION
- SECUREDLIBRARY=
- SECUREDFOLDER=
- TABLES ステートメントオプション:
- ENCRYPTKEY=

詳細

この例では、異なる暗号化キーを含むメタデータバインド型ライブラリ内のすべてのデータセットを、必須の AES 暗号化と同じ暗号化キーをもつように変更する方法について説明します。どのデータセットにもパスワードはありません。

プログラム

```
proc authlib lib=abcde;


    create  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=new ;

    tables employee /
        encryptkey=abc;
    tables empinfo /
        encryptkey=def;
    tables deptname ;
run;
quit;
```

プログラムの説明

ライブラリ ABCDE には、Employee、EmpInfo、DeptName という 3 つのデータセットが格納されます。 Employee および EmpInfo データセットは、異なるキーによってすでに AES で暗号化されています。DeptName データセットは暗号化されません。

```
proc authlib lib=abcde;
```

CREATE ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 REQUIRE_ENCRYPTION=YES は、メタデータバインド型ライブラリ内のすべてのデータセットが自動的に AES で暗号化されるように指定します。

```
    create  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=new ;
```

TABLES ステートメントを使用して、各データセットの暗号化キーを指定します。 各データセットに TABLES ステートメントが必要です。

```
    tables employee /
        encryptkey=abc;
    tables empinfo /
        encryptkey=def;
    tables deptname ;
run;
```

```
quit;
```

結果:RANK プロシジャ ライブラリ ABCDE がバインドされます。メタデータバインド型ライブラリ ABCDE 内のすべてのデータセットは、必須のキーで暗号化されるようにコピー置換されています。

ログの例

ログ 7.15 *各データセットの暗号化キー値が異なる場合、AES 暗号化を必要とするライブラリ ABCDE*

```
554  proc authlib lib=abcde; 555  create seclib="ABCDEEmps" 556  securedfolder="Department XYZZY" 557
pw=XXXXXX 558  require_encryption=yes 559  encrypt=aes 560  encryptkey=XXX ; 561  tables employee /
562  encryptkey=XXX; 563  tables empinfo / 564  encryptkey=XXX; 565  tables deptname ; 566  run;
NOTE:Setting library to require encryption.NOTE:Required encryption will use AES encryption with the
recorded key.NOTE:Successfully created a secured library object for the physical library ABCDE and
recorded its location as:SecuredFolder:      /System/Secured Libraries/Department XYZZY
SecuredLibrary:     ABCDEEmps SecuredLibraryGUID:097E9A84-D6E8-488E-B779-1E2AB0670036 NOTE:Copying
data set ABCDE.EMPLOYEE in place to do required encryption with the library's required encryption key
and passwords.NOTE:Renaming the data set ABCDE.EMPLOYEE to
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to
ABCDE.EMPLOYEE.NOTE:Metadata-bound library permissions are used for
ABCDE.EMPLOYEE.DATA.NOTE:Successfully added new secured table object "EMPLOYEE.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set
ABCDE.EMPLOYEE.DATA.NOTE:There were 5 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.EMPLOYEE has 5 observations and 6
variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.EMPLOYEE.DATA were successfully modified.NOTE:Copying data set ABCDE.EMPINFO in place to do
required encryption with the library's required encryption key and passwords.NOTE:Renaming the data
set ABCDE.EMPINFO to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.NOTE:Metadata-bound library permissions are used for
ABCDE.EMPINFO.DATA.NOTE:Successfully added new secured table object "EMPINFO.DATA" to the secured
library object at path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set
ABCDE.EMPINFO.DATA.NOTE:There were 5 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.EMPINFO has 5 observations and 6
variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.EMPINFO.DATA were successfully modified.NOTE:Copying data set ABCDE.DEPTNAME in place to do
required encryption with the library's required encryption key and passwords.NOTE:Renaming the data
set ABCDE.DEPTNAME to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.NOTE:Metadata-bound library permissions are used
for ABCDE.DEPTNAME.DATA.NOTE:Successfully added new secured table object "DEPTNAME.DATA" to the
secured library object at path "/System/Secured Libraries/Department XYZZY/ABCDEEmps" for data set
ABCDE.DEPTNAME.DATA.NOTE:There were 4 observations read from the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.DEPTNAME has 4 observations and 2
variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.DEPTNAME.DATA were successfully modified.567  quit;
```

例 16: 既存のデータセットが異なる暗号化キーで暗号化されている場合、メタデータバインド型ライブラリを変更して AES 暗号化を必須化する方法

要素: PROC AUTHLIB ステートメントオプション
MODIFY ステートメントオプション:
ENCRYPT=
ENCRYPTKEY=
PW=
REQUIRE_ENCRYPTION
SECUREDLIBRARY=

SECUREDFOLDER=
TABLES ステートメントオプション:
ENCRYPTKEY=

詳細

この例は前の例と似ています。違いはライブラリがメタデータにすでにバインドされていることです。したがって、MODIFY ステートメントは、バインドを変更して AES 暗号化を必須化する場合に使用します。

プログラム

```
proc authlib lib=abcde;

    modify  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=new;

    tables employee /
        encryptkey=abc;
    tables empinfo /
        encryptkey=def;
    tables deptname ;
run;
quit;
```

プログラムの説明

ライブラリ ABCDE には、Employees、EmpInfo、DeptName という 3 つのデータセットが格納されます。 このライブラリでは、Employees データセットの暗号化キー値は **abc** です。EmpInfo データセットの暗号化キー値は **def** です。DeptName データセットは AES で暗号化されません。

```
proc authlib lib=abcde;
```

MODIFY ステートメントを使用して、メタデータフォルダの名前を入力し、SAS メタデータサーバーで保護付きライブラリオブジェクトに名前を付けます。 REQUIRE_ENCRYPTION=YES オプションを使用して、メタデータバインド型ライブラリ内のすべてのデータセットが AES で暗号化されることを必須化します。保護付きライブラリオブジェクトの名前とメタデータフォルダの名前はオプションですが、名前を指定すれば、ライブラリが保護付きライブラリオブジェクトにバインドされていることを変更前に確認できます。

```
    modify  seclib="ABCDEEmps"
        securedfolder="Department XYZZY"
        pw=secret
        require_encryption=yes
        encrypt=aes
        encryptkey=new;
```

TABLES ステートメントを使用して、各データセットの暗号化キーを指定します。 各データセットに TABLES ステートメントが必要です。

```
    tables employee /
        encryptkey=abc;
    tables empinfo /
        encryptkey=def;
    tables deptname ;
run;
quit;
```

結果:RANK プロシジャ ライブラリ ABCDE はバインドされたままです。MODIFY ステートメントは、バインドを変更して AES 暗号化を必須化しました。3 つのデータセットはすべて、必須の暗号化キーで暗号化するため、コピー置換されます。

ログの例

ログ 7.16 *AES 暗号化が必要なライブラリ ABCDE と、各データセットの暗号化キー値を記録された暗号化キー値に変更する方法*

```
628  proc authlib lib=abcde; 629  modify seclib="ABCDEEmps" 630  securedfolder="Department XYZZY" 631
pw=XXXXXX 632  require_encryption=yes 633  encrypt=aes 634  encryptkey=XXX; 635  tables employee /
636  encryptkey=XXX; 637  tables empinfo / 638  encryptkey=XXX; 639  tables deptname ; 640  run;
NOTE:Changing library to require encryption.NOTE:Required encryption will use AES encryption with the
recorded key.NOTE:The passwords on ABCDE.EMPLOYEE.DATA do not require modification.NOTE:Copying data
set ABCDE.EMPLOYEE in place to do required encryption with the library's required encryption key and
passwords.NOTE:Renaming the data set ABCDE.EMPLOYEE to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying
the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.NOTE:There were 5 observations read
from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.EMPLOYEE has 5 observations
and 6 variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.EMPINFO.DATA do not require modification.NOTE:Copying data set ABCDE.EMPINFO in place to do
required encryption with the library's required encryption key and passwords.NOTE:Renaming the data
set ABCDE.EMPINFO to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.NOTE:There were 5 observations read from the data
set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.EMPINFO has 5 observations and 6
variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords on
ABCDE.DEPTNAME.DATA do not require modification.NOTE:Copying data set ABCDE.DEPTNAME in place to do
required encryption with the library's required encryption key and passwords.NOTE:Renaming the data
set ABCDE.DEPTNAME to ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.NOTE:There were 4 observations read from the data
set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set ABCDE.DEPTNAME has 4 observations and 2
variables.NOTE:Deleting the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The passwords and/or
encryption options for the secured library object with path "/System/Secured Libraries/Department
XYZZY/ABCDEEmps" were successfully modified."641  quit;
```

例 17: AES 暗号化が必要なメタデータバインド型ライブラリで REMOVE ステートメントを使用する方法

要素: PROC AUTHLIB ステートメントオプション
REMOVE ステートメントオプション:
PW=
ENCRYPT=

詳細

この例では、メタデータ連結ライブラリのバインドを解除する方法について説明します。このコードでは次が実行されます。

- ライブラリとそのテーブルを記述するメタデータを SAS メタデータリポジトリから削除します
- 物理ライブラリとデータセットからセキュリティバインドを削除します
- 割り当てたパスワードと暗号化をデータセットから削除し、保護しないまま残します

パスワードの後のスラッシュ(/)はオプションで、データセットからパスワードを削除または置換するために使用されます。例 4 (141 ページ)に示すように、ライブラリが READ=、WRITE=、ALTER=の各パスワードによってバインドされている場合、すべてのパスワードを指定する必要があり、各パスワードにはスラッシュ(/)が必要です。

プログラム

```
proc authlib lib=abcde;


    remove
        pw=currntpw/
        encrypt=no;
run;
quit;
```

プログラムの説明

メタデータバインド型ライブラリ ABCDE のバインド解除

```
proc authlib lib=abcde;
```

メタデータバインド型ライブラリのバインドを解除するには、REMOVE ステートメントを使用します。 データセットからパスワードを削除するには、パスワードの後のスラッシュ(/)を使用します。ENCRYPT=NO を指定すると、すべてのデータセットから暗号化が削除されます。

```
    remove
        pw=currntpw/
        encrypt=no;
run;
quit;
```

結果:RANK プロシジャ ライブラリ ABCDE とそれにバインドされたすべてのデータセットは、バインドが解除されます。バインドが解除されたデータセットからすべてのパスワードと暗号化が削除され、データセットが保護されない状態となります。

ログ 7.17 AES 暗号化が必要なメタデータバインド型ライブラリで REMOVE ステートメントを使用する方法

```
642  proc authlib lib=abcde; 643  remove 644  pw=XXXXXX/ 645  encrypt=no; 646
run; NOTE:Copying data set ABCDE.DEPTNAME in place to remove
encryption.NOTE:Renaming the data set ABCDE.DEPTNAME to
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.DEPTNAME.NOTE:There were 4
observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The
data set ABCDE.DEPTNAME has 4 observations and 2 variables.NOTE:Deleting the
data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.WARNING:Some or all the passwords on
ABCDE.DEPTNAME.DATA were removed along with the secured library object location,
leaving the data set unprotected.NOTE:The secured table object location for
ABCDE.DEPTNAME.DATA was successfully removed.NOTE:Copying data set ABCDE.EMPINFO
in place to remove encryption.NOTE:Renaming the data set ABCDE.EMPINFO to
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPINFO.NOTE:There were 5 observations
read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The data set
ABCDE.EMPINFO has 5 observations and 6 variables.NOTE:Deleting the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.WARNING:Some or all the passwords on
ABCDE.EMPINFO.DATA were removed along with the secured library object location,
leaving the data set unprotected.NOTE:The secured table object location for
ABCDE.EMPINFO.DATA was successfully removed.NOTE:Copying data set ABCDE.EMPLOYEE
in place to remove encryption.NOTE:Renaming the data set ABCDE.EMPLOYEE to
ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:Copying the data set
ABCDE.__TEMP_ENCRYPT_FILE_NAME__ to ABCDE.EMPLOYEE.NOTE:There were 5
observations read from the data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.NOTE:The
data set ABCDE.EMPLOYEE has 5 observations and 6 variables.NOTE:Deleting the
data set ABCDE.__TEMP_ENCRYPT_FILE_NAME__.WARNING:Some or all the passwords on
ABCDE.EMPLOYEE.DATA were removed along with the secured library object location,
leaving the data set unprotected.NOTE:The secured table object location for
ABCDE.EMPLOYEE.DATA was successfully removed.NOTE:Successfully deleted the
secured library object that was located at:SecuredFolder:      /System/Secured
Libraries/Department XYZZY SecuredLibrary:     ABCDEEmps SecuredLibraryGUID:
157F7ACD-5B71-4BC3-A490-DCED4BD275E8 NOTE:Successfully deleted the recorded
location of the secured library object for the physical library ABCDE.647  quit;
```

例 18: インポートされた SecuredLibrary オブジェクトでの認証情報のリセット

要素: PROC AUTHLIB ステートメントオプション
MODIFY ステートメントオプション:
LIBRARY=
PW=
ENCRYPT=
ENCRYPTKEY=

詳細

この例は、バックアップパッケージからインポートされる SecuredLibrary オブジェクトでパスワードと暗号化キーをリセットする方法を示しています。

- AUTHADMIN=YES オプションを指定せずに LIBNAME ステートメントを実行すると、インポートによってリストアされる、関連づけられたパスワード値がないため、操作は失敗します。
- AUTHADMIN=YES オプションを使用して AUTHLIB プロシジャを有効にし、物理ライブラリで連結情報を使用して実行します。

- MODIFY ステートメントを使用して、“例 13: 既存のデータセットが同じ暗号化キーで暗号化されている場合、AES 暗号化が必須のライブラリをバインドする方法” (154 ページ) からライブラリでメタデータ連結ライブラリのパスワードと暗号化キー値がリセットされますが、これは、これらの値を設定せずにバックアップパッケージから SecuredLibrary オブジェクトがインポートされたことを想定しています。

プログラム

```
libname abcde "sas-library" ;

libname abcde "sas-library" authadmin=yes;

proc authlib lib=abcde;
    modify
        pw=secret
        encrypt=aes
        encryptkey=value;
run;
quit;

libname abcde "sas-library";
```

プログラムの説明

ライブラリ ABCDE には、Employees、EmpInfo、DeptName という 3 つのデータセットが格納されます。 関連づけられたパスワード値がないため、この LIBNAME ステートメントは失敗します。

```
libname abcde "sas-library" ;
```

AUTHADMIN=YES オプションを使用します。 AUTHADMIN=YES オプションを使用すると、AUTHLIB プロシジャが有効になり、物理ライブラリ内の連結情報を使用して実行されます。

```
libname abcde "sas-library" authadmin=yes;
```

MODIFY ステートメントを使用して、メタデータ連結ライブラリのパスワードと暗号化キー値をリセットします。 PW=オプションを使用すると、パスワードがリセットされます。ENCRYPTKEY=オプションを使用すると、暗号化キー値がリセットされます。

```
proc authlib lib=abcde;
    modify
        pw=secret
        encrypt=aes
        encryptkey=value;
run;
quit;
```

AUTHADMIN=YES オプションなしの LIBNAME ステートメントの再発行 .管理での必要性がなくなったらすぐに AUTHADMIN=YES を設定せずにライブラリを再割り当てし、ライブラリへのその他すべてのアクセスが管理モードにならないようにすることをお勧めします。これを行う場合、認証情報も必ずリセットしてください。

```
libname abcde "sas-library";
```

ログ 7.18 認証情報のリセット

```
253 libname abcde "library-name" ; ERROR:The secured library object information for library ABCDE
could not be obtained from the metadata server or has invalid data.ERROR:Association not
found.ERROR:Error in the LIBNAME statement.254  libname abcde "library-name" authadmin=yes;
NOTE:Libref ABCDE was successfully assigned as follows:Engine:            V9 Physical Name:
library-name Secured Library:    /System/Secured Libraries/Department XYZZY/ABCDEEmps Authenticated
ID:   user-id@site as user-id Encryption Key:     YES Require Encryption:YES 255  proc authlib
lib=abcde; 256  modify 257        pw=XXXXXX 258        encrypt=aes 259        encryptkey=XXX ; 260
run; NOTE:Required encryption will use AES encryption with the recorded key.NOTE:The passwords on
ABCDE.DEPTNAME.DATA do not require modification.NOTE:The passwords on ABCDE.EMPINFO.DATA do not
require modification.NOTE:The passwords on ABCDE.EMPLOYEE.DATA do not require modification.261  quit;
```

8 章

CALENDAR プロシジャ

概要: CALENDAR プロシジャ

CALENDAR プロシジャの動作について

CALENDAR プロシジャは SAS データセットのデータを月次カレンダ形式で表示します。イベントを休日や業務外期間を避けてスケジュールする*スケジュールカレンダ*や、データを集計してイベントや休日などを日別に表示する*サマリーカレンダ*を作成することができます。PROC CALENDAR を使うと、次のタスクが実行できます。

- 作業を休日や業務外期間を避けてスケジュールします。
- 休日を表示します。
- *複数のカレンダ*のデータを単一ステップで処理して、それを個々のカレンダ、各カレンダを特定して組み合わせたカレンダ、または各カレンダを特定せずに組み合わせたカレンダに出力します。
- 単一の PROC ステップで、異なった休日や週ベースの勤務スケジュールや日次作業シフトを、複数のカレンダに適用できます。
- 月の日数やオブザベーション数をベースに、変数の平均や合計を計算します。

PROC CALENDAR は、SAS/OR(プロジェクト管理スケジュールツール)の PROC CPM との連携に特化した機能も持っています。

PROC CALENDAR が作成可能なカレンダの種類について

単純スケジュールカレンダ

次の出力は、最もシンプルな種類のスケジュールカレンダになります。このカレンダ出力は、銀行役員の予定しているアクティビティを示します。次のステートメントは、アウトプット 8.1 (169 ページ)を作成します。

```
proc calendar data=allacty;
   start date;
   dur long;
run;
```

このカレンダに表示されているアクティビティデータに関しては、“例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)を参照してください。

次のカレンダは 2 種類のデフォルトカレンダのうちの 1 種類(24 時間、1 週 7 日)を使います。

アウトプット 8.1 単純スケジュールカレンダ

The SAS System

July 2002

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	1 +Dist. Mtg./+	2 +Mgrs. Meeting/	3 +Interview/J+ District 6=+	4	5 +VIP Banquet+	6
7	8 +============Trade Show/Knox============+ +======================Sales Drive/District 6======================+	9	10	11 +Planning Co+ +Mgrs. Meeting/	12 +Seminar/Whi+ District 7=+	13
14	15 +======================Sales Drive/District 7======================+	16 +Dentist/JW=+	17 +Bank Meetin+	18 +NewsLetter + +Planning Co+	19 +Co. Picnic/+ +Seminar/Whi+	20
21	22 +=========Inventors Show/Melvin=========+	23	24 +Birthday/Ma+	25 +==Close Sale/ +Planning Co+	26 WYGIX Co.===+	27
28	29	30	31			

詳細スケジュールカレンダ

次の出力は、PROC CALENDAR で作成された詳細スケジュールカレンダになります。このカレンダを作成するステートメントは次のタスクを実行します。

- アクティビティを休日を避けてスケジュールします。

- 個々のカレンダを特定します
- 同じレポートに複数のカレンダを出力します。
- カレンダごとに異なる祝日を適用します。
- カレンダごとに異なる勤務パターンを適用します。

このカレンダを作成するプログラムの説明については、“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)を参照してください。

アウトプット 8.2 詳細スケジュールカレンダ

Well Drilling Work Schedule: Combined Calendars

	July 2002						
	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
CAL1		1 +========Drill Well/$1,000.00========>	2	3	4 Independence	5 +Assemble T> +Lay Power > <Drill Well+	6
CAL2				+================Excavate/$3,500.00================>			
CAL1	7	8 +============Build Pump House/$2,000.00============+ <=============Assemble Tank/$1,000.00=============+ <Lay Power Line/$2,000.0+	9	10	11 +Pour Foundation/$1,500.>	12	13
CAL2		<Excavate/$>	**Vacation**	<Excavate/$+			
CAL1	14	15 +======================Install Pump/$500.00======================+ <=====Pour Foundation/$1,500.00======+	16	17	18	19 +Install Pi>	20
CAL1	21	22 +====================Erect Tower/$2,500.00====================> <Install Pipe/$1,000.00=+	23	24	25	26	27
CAL1	28	29 <Erect Towe+	30	31			

単純サマリーカレンダ

次の出力は、病院の食堂で提供される毎日の食事数を表示する単純サマリーカレンダを示しています。

```
proc calendar data=meals;
   start date;
   sum brkfst lunch dinner;
   mean brkfst lunch dinner;
```

```
run;
```

サマリーカレンダでは、ある特定の日の情報は、変数のその日の値です。変数は数値または文字で、必要に応じてフォーマットできます。数値変数の場合は、SUM や MEAN オプションを使って合計値や平均値も計算できます。これらの統計量は、次の出力のように、カレンダの下のボックスに表示されます。このカレンダに表示されているデータセットは、“例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)で作成されます。

アウトプット 8.3 *単純サマリーカレンダ*

The SAS System

December 2008

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	1 123 234 238	2 188 188 198	3 123 183 176	4 200 267 243	5 176 165 177	6
7	8 178 198 187	9 165 176 187	10 187 176 231	11 176 187 222	12 187 187 123	13
14	15 176 165 177	16 156 . 167	17 198 143 167	18 178 198 187	19 165 176 187	20
21	22 187 187 123	23	24	25	26	27
28	29	30	31			

	Sum	Mean
Brkfst	2763	172.688
Lunch	2830	188.667
Dinner	2990	186.875

詳細スケジュールとプロジェクト管理のタスク

さらに複雑なスケジュールタスクに関しては、SAS/OR の CPM プロシジャの使用をお勧めします。PROC CALENDAR では、アクティビティには開始日の指定が必須になります。あるタスクの開始が他のタスクの完了に依存するケースで、その完了が遅れた場合、スケジュールの再計算には時間がかかります。手動で日付を再計算する代わり

に PROC CPM を使用すると、初期開始日やアクティビティの期間、そして他のタスクの *後続*となるタスクの情報をベースにして、プロジェクトアクティビティの日付を計算できます。例については、“例 6: 前のタスクの完了に基づき、スケジュールを計算する” (231 ページ)を参照してください。

概念: CALENDAR プロシジャ

カレンダの種類

PROC CALENDAR は、スケジュールとサマリーの 2 種類のカレンダを作成します。

表 8.1 *スケジュールカレンダおよびサマリーカレンダの要約*

カレンダの種類	タスク	制限
スケジュールカレンダ	作業を休日や業務外期間を避けてアクティビティをスケジュールします。	合計と平均を計算できません。
スケジュールカレンダ	一日以上継続するアクティビティをスケジュールします。	
サマリーカレンダ	合計と平均を計算します。	アクティビティの期間は 1 日です。

注: PROC ステップで DUR または FIN ステートメントを使用しない場合、PROC CALENDAR はサマリーカレンダを作成します。

スケジュールカレンダ

定義

アクティビティと休日の開始時と終了時を示すカレンダ形式のレポートです。

必要なステートメント

START ステートメント、および DUR または FIN ステートメントのいずれかを指定する必要があります。DUR または FIN ステートメントを使用しない場合、PROC CALENDAR はサマリーカレンダレポートを作成するとみなします。

表 8.2 *必要なステートメント*

ステートメント	変数値
“START ステートメント” (204 ページ)	アクティビティの開始日
“DUR ステートメント” (197 ページ)	アクティビティの期間

ステートメント	変数値
“FIN ステートメント” (198 ページ)	アクティビティの終了日

例

- “単純スケジュールカレンダ” (168 ページ)
- “詳細スケジュールカレンダ” (169 ページ)
- “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)
- “例 2: 複数のカレンダを含むスケジュールカレンダ” (212 ページ)
- “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
- “例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)
- “例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)
- “例 6: 前のタスクの完了に基づき、スケジュールを計算する” (231 ページ)

サマリーカレンダ

定義

継続期間が一日のアクティビティと休日を表示するカレンダ形式のレポートで、合計と平均の形式で要約情報を提供できます。

必要なステートメント

START ステートメントを指定する必要があります。このステートメントは、アクティビティの開始日を含むアクティビティのデータセットの変数を特定します。

1 日にイベントが複数ある場合

サマリーカレンダレポートでは、指定日に 1 つのアクティビティのみ表示されます。そのため、複数のアクティビティの START 値が同じ場合は、最後に読み込まれたオブザベーションのみ使用されます。このような状況では、PROC SUMMARY を使用して、開始日ごとに 1 アクティビティが含まれるようにデータセットをまとめる場合があります。

例

- “単純サマリーカレンダ” (171 ページ)
- “例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)
- “例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

デフォルトのカレンダ

説明

PROC CALENDAR は、単純な適用に対応した 2 つのデフォルトカレンダを提供しています。2 つの単純な勤務パターンのうち 1 つを使用して適用できる場合は、詳細な

勤務シフトや週次勤務パターンを指定しなくてもカレンダを作成できます。次の条件が当てはまる場合は、デフォルトカレンダの使用を検討してください。

- 1 日 8 時間の週 5 日勤務かまたは 1 日 24 時間の週 7 日の勤務(次の表を参照)を使用して適用する
- すべてのアクティビティを同じカレンダに出力する
- 個々のカレンダを特定する必要がない

表 8.3 *デフォルトカレンダの設定と例*

スケジュールされた勤務日	INTERVAL=	DAYLENGTH=のデフォルト値	業務期間	例
7 (月-日)	DAY	24	1 日 24 時間	2
5 (月-金)	WORKDAY	8	1 日 8 時間	1

デフォルトカレンダが予期せずに作成される場合

特別カレンダの生成に必要な情報を一部提供しなかった場合、PROC CALENDAR はデフォルトカレンダを作成しようとします。次のエラーが発生すると、PROC CALENDAR は、デフォルト機能を備えたカレンダを生成します。

- ACTIVITIES データセットに CALID 変数が含まれていない場合、PROC CALENDAR はデフォルトカレンダを生成します。
- HOLIDAYS データセットと CALENDAR データセットの*両方*に CALID 変数が含まれていない場合、PROC CALENDAR は、*ACTIVITIES データセットに CALID 変数が含まれていてもデフォルトカレンダを生成します。*
- ACTIVITIES データセットと CALENDAR データセットに CALID 変数が含まれており、HOLIDAYS データセットには含まれていない場合、デフォルトの休日が使用されます。

例

- の週 7 日のデフォルトカレンダを参照 アウトプット 8.1 (169 ページ)
- の週 5 日のデフォルトカレンダを参照 “例 1: 休日表示付きスケジュールカレンダ: 週 5 日” (208 ページ)

カレンダと複数のカレンダ

定義

カレンダ
: 週次勤務パターンを表す論理エンティティで、週次勤務スケジュールと日次シフトで構成されます。PROC CALENDAR には、1 日 8 時間の週 5 日勤務または 1 日 24 時間の週 7 日勤務の 2 つのデフォルト勤務パターンが含まれます。CALENDAR データセットと WORKDAYS データセットを使用して独自の勤務パターンを定義することもできます。

カレンダレポート
: アクティビティ、休日および業務外期間を表示するカレンダ形式のレポート。カレンダレポートには、複数のカレンダを 3 つのうちいずれかの形式で含めることができます。

個別のカレンダ(separate)
特定されたカレンダをそれぞれ別の出力ページに出力します。

各カレンダを特定して組み合わせたカレンダ(combined)
カレンダをすべて同じ出力ページに出力し、各カレンダを特定します。

各カレンダを特定せずに組み合わせたカレンダ(mixed)
カレンダをすべて同じ出力ページに出力しますが、個々の所属カレンダは特定されません。

複数のカレンダ
複数の週次勤務パターン表す論理エンティティです。

複数のカレンダを作成する利点

複数の勤務スケジュールや複数の週次勤務パターンに従うアクティビティのカレンダレポートを出力する場合に、複数のカレンダを作成します。たとえば、建設プロジェクトレポートの場合、プロジェクトの要素ごとに、作業員に対して異なる勤務スケジュールおよび週次勤務パターンの使用が必要になる場合があります。

複数のカレンダの別の用途は、アクティビティを特定し、同じカレンダレポートに出力できるようにすることです。たとえば、アクティビティが 1 つの部内の別々の部門に属していると特定した場合、すべての部門のアクティビティを同じカレンダに表示するカレンダレポートを印刷するよう選択できます。

最後に、複数のカレンダを使用すると、カレンダごとに別々のカレンダレポートを一度に作成できます。たとえば、アクティビティが部門によって識別される場合は、各課のアクティビティを別々のページに印刷したカレンダレポートを作成できます。

複数のカレンダの識別方法

PROC CALENDAR は単一 PROC ステップで各種類(ACTIVITIES、HOLIDAYS、CALENDAR、WORKDAYS)のデータセットを 1 つのみ処理可能なため、PROC CALENDAR では、アクティビティ、休日、または週次勤務パターンがどのカレンダに属するかを識別できる必要があります。CALID ステートメントを使用して変数を指定し、その値によって適切なカレンダを識別します。この変数には、数値または文字が可能です。

特殊変数名_CAL_を使用することも、別の変数名を使用することもできます。ACTIVITIES データセットで別の名前の変数が CALID 変数として使用されている場合でも、PROC CALENDAR は、HOLIDAYS データセットと CALENDAR データセットで自動的に_CAL_という名前の変数を検索します。そのため、HOLIDAYS データセットと CALENDAR データセットで名前_CAL_を使用すると、異なるカレンダアプリケーションでこれらのデータセットをさらに再利用しやすくなります。

HOLIDAYS データセットまたは CALENDAR データセットと複数カレンダの使用

複数のカレンダで HOLIDAYS データセットまたは CALENDAR データセットを使用する場合、PROC CALENDAR は変数値を次のように処理します。

- HOLIDAYS データセットと CALENDAR データセットのいずれかに存在する CALID 変数の各値によって、カレンダが定義されます。
- CALID 値が HOLIDATA=データセットには存在し、CALEDATA=データセットには存在しない場合、デフォルトカレンダの勤務スケジュールが使用されます。
- CALID 値が CALEDATA=データセットには存在し、HOLIDATA=データセットには存在しない場合、デフォルトカレンダの休日が使用されます。
- CALID 値が HOLIDATA=データセットにも CALEDATA=データセットにも存在しない場合、デフォルトカレンダの勤務スケジュールと休日が使用されます。

- CALID 変数が HOLIDAY データセットにも CALENDAR データセットにもない場合、PROC CALENDAR は代わりにデフォルト変数_CAL_を検索します。データセットに CALID 変数も_CAL_変数も見つからなかった場合、そのデータセットのオブザベーションはデフォルトカレンダに適用されます。

複数のカレンダを含むレポートの種類

異なるオブザベーションと異なるカレンダを関連付けられるため、異なる勤務スケジュールや異なる勤務シフトに従ったアクティビティ、または異なる休日を含むアクティビティを示すカレンダレポートを印刷できます。次のタスクを実行できます。

- 別々のカレンダを同じページに印刷し、それぞれを識別します。
- 別々のカレンダを識別せずに同じページに印刷します。
- 識別された各カレンダを別々のページに印刷します。

たとえば、部内のすべての部門のアクティビティを示すカレンダがあるとします。各部門に固有のカレンダ ID 値を設定し、必要に応じて、個別の週次勤務パターン、日次勤務シフトおよび休日も設定できます。

異なるカレンダと関連付けられているアクティビティを同じ ACTIVITIES データセットに置く場合は、PROC CALENDAR を使用して、次を印刷するカレンダレポートを作成します。

- 各部門のスケジュールとイベントを別々のページに印刷します(個別出力)。
- 部全体のスケジュールとイベントを印刷し、それぞれを部門によって識別します(結合出力)。
- 部全体のスケジュールとイベントを印刷しますが、部門による識別は*行われません*(混合出力)

複数カレンダ機能は、PROC CALENDAR が SAS/OR ソフトウェア(プロジェクト管理ツール)の PROC CPM の出力を処理できるようにするために特に追加されました。“例 6: 前のタスクの完了に基づき、スケジュールを計算する” (231 ページ)を参照してください。

CALID ステートメントと特殊変数_CAL_によるカレンダの識別方法

複数のカレンダを識別するには、CALID ステートメントを使用して変数を指定し、その値によってイベントに属するカレンダを識別します。この変数には、数値または文字が可能です。

特殊変数名_CAL_を使用することも、別の変数名を使用することもできます。ACTIVITIES データセットで別の名前の変数が CALID 変数として使用されている場合でも、PROC CALENDAR は、HOLIDAYS データセットと CALENDAR データセットで自動的に_CAL_という名前の変数を検索します。そのため、HOLIDAYS データセットと CALENDAR データセットで名前_CAL_を使用すると、異なるカレンダアプリケーションでこれらのデータセットをさらに再利用しやすくなります。

HOLIDAYS データセットまたは CALENDAR データセットを使用する場合

複数のカレンダで HOLIDAYS データセットまたは CALENDAR データセットを使用する場合、PROC CALENDAR は変数値を次のように処理します。

- HOLIDAYS データセットと CALENDAR データセットのいずれかに存在する CALID 変数の各値によって、カレンダが定義されます。
- CALID 値が HOLIDATA=データセットには存在し、CALEDATA=データセットには存在しない場合、デフォルトカレンダの勤務スケジュールが使用されます。
- CALID 値が CALEDATA=データセットには存在し、HOLIDATA=データセットには存在しない場合、デフォルトカレンダの休日が使用されます。

- CALID 値が HOLIDATA=データセットにも CALEDATA=データセットにも存在しない場合、デフォルトカレンダの勤務スケジュールと休日が使用されます。
- CALID 変数が HOLIDAYS データセットにも CALENDAR データセットにもない場合、PROC CALENDAR は代わりにデフォルト変数_CAL_を検索します。データセットに CALID 変数も_CAL_変数も見つからなかった場合、そのデータセットのオブザベーションはデフォルトカレンダに適用されます。

例

- “例 2: 複数のカレンダを含むスケジュールカレンダ” (212 ページ)
- “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
- “例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)
- “例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

入力データセット

アプリケーションの複雑さに応じて、複数のデータセットがカレンダの作成に必要になる場合があります。次の表にあるように、PROC CALENDAR は 4 つのデータセットをそれぞれ 1 つずつ処理できます。

表 8.4 PROC CALENDAR に使用可能な 4 つの入力データセット

データセット	説明	オプション
ACTIVITIES	*オブザベーション*にはそれぞれ、1 つのアクティビティに関する情報が含まれます。	“DATA=*SAS-data-set*” (188 ページ)
HOLIDAYS	*オブザベーション*にはそれぞれ、休日に関する情報が含まれます。	“HOLIDATA=*SAS-data-set*” (191 ページ)
CALENDAR	*オブザベーション*がそれぞれ、1 つの週次勤務スケジュールを定義します。	“CALEDATA=*SAS-data-set*” (187 ページ)
WORKDAYS	*変数*がそれぞれ、業務期間と業務外期間が交互に並ぶ 1 つの日次スケジュールを表します。	“WORKDATA=*SAS-data-set*” (194 ページ)

ACTIVITIES データセット

目的

DATA=オプションで指定する ACTIVITIES データセットには、PROC CALENDAR によってスケジュールされるアクティビティに関する情報が含まれます。各オブザベーションが 1 つのアクティビティを表します。

必要条件と制限

- ACTIVITIES データセットは必須です。(DATA=オプションで ACTIVITIES データセットを指定しない場合、PROC CALENDAR は_LAST_データセットを使用します。
- 1 つの ACTIVITIES データセットのみ使用できます。
- ACTIVITIES データセットは START 変数によって並べ替えるか、またはインデックス付けする必要があります。
- CALID (カレンダ識別子)変数を使用して、複数のカレンダを別々のページに表示する出力を作成する場合は、ACTIVITIES データセットを CALID 変数によって並べ替えるか、またはインデックス付けをしてから、START 変数で同様の処理を行います。
- BY ステートメントを使用する場合は、ACTIVITIES データセットを、BY 変数によって並べ替えるか、またはインデックス付けする必要があります。

構造

ACTIVITIES データセットの*オブザベーション*にはそれぞれ 1 つのアクティビティに関する情報が含まれます。1 つの変数に開始日が含まれている必要があります。スケジュールカレンダを作成する場合は、別の変数にアクティビティ期間か終了日のいずれかが含まれている必要があります。他の変数には、アクティビティに関する追加情報を含めることができます。

表 8.5 必要なステートメント

変数コンテンツ	ステートメント	カレンダの種類
開始日	“START ステートメント” (204 ページ)	スケジュール 要約
期間	“DUR ステートメント” (197 ページ)	スケジュール
終了日	“FIN ステートメント” (198 ページ)	スケジュール

サマリーカレンダの 1 日に複数のアクティビティ

サマリーカレンダに表示できるのは、指定日に 1 つのアクティビティのみです。そのため、複数のアクティビティの START 値が同じ場合は、読み込まれる最後のオブザベーションのみ使用されます。このような状況では、PROC SUMMARY は、データセットを折りたたんで開始日ごとに 1 アクティビティが含まれるようにするために使用されます。

例

例セクションの例はすべて、ACTIVITIES データセットを使用します。

HOLIDAYS データセット

目的

HOLIDATA=オプションで指定する HOLIDAYS データセットを使用して、次を識別できます。

- カレンダ出力の休日。

- 勤務のスケジューリングに使用できない日。(スケジュールカレンダでは、PROC CALENDAR はこれらの日にアクティビティをスケジュールしません。

構造

HOLIDAYS データセットの各オブザベーションには、少なくとも休日開始日が含まれている必要があります。期間か終了日を指定しない限り、休日は 1 日のみ続きます。必須ではありませんが、休日名を指定することをお勧めします。休日名を含む変数を指定しない場合、PROC CALENDAR は用語 **DATE** を使用して各休日を識別します。

表 8.6 必要なステートメント

変数コンテンツ	ステートメント
開始日	“HOLISTART ステートメント” (200 ページ)
名前	“HOLIVAR ステートメント” (201 ページ)
期間	“HOLIDUR ステートメント” (199 ページ)
終了日	“HOLIFIN ステートメント” (199 ページ)

並べ替え不要

HOLIDAYS データセットは、並べ替えやインデックス付けの必要はありません。

SAS 日付値と SAS 日時値の使用

PROC CALENDAR は、SAS 日時値を使用して時間を計算します。データが DATE. 出力形式の場合でも、プロシジャは自動的に時間を分単位および秒単位で計算します。そのため、日付値のみを指定すると、PROC CALENDAR は次のようなメッセージを SAS ログに印刷します。

```
NOTE: All holidays are assumed to start at the
      time/date specified for the holiday variable
      and last one DTWRKDAY.
WARNING: The units of calculation are SAS datetime
         values while all the holiday variables are
         not. All holidays are converted to SAS
         datetime values.
```

一般的な HOLIDAYS データセットの作成

PROC CALENDAR 出力を必要とするアプリケーションが多数ある場合は、標準の休日を含む一般的な HOLIDAYS データセットの作成を検討してください。一般的な休日から始めて、アプリケーション固有の休日や業務外イベントを含むオブザベーションを追加できます。

休日と業務外期間

業務外期間中は休日をスケジュールしません。HOLIDATA=データセットで定義される休日は、勤務スケジュールで定義される業務外期間中に発生しません。たとえば、週間勤務日が月曜日から金曜日と定義されている場合は、日曜日を休暇日としてスケ

ジュールできません。このような矛盾が発生した場合、休日のスケジュールは、業務外日の後に続く次の使用可能な業務期間に変更されます。

例

例セクションの例はすべて、HOLIDAYS データセットを使用します。

CALENDAR データセット

目的

CALEDATA=オプションで指定する CALENDAR データセットを使用すると、異なるカレンダの勤務スケジュールを指定できます。

構造

CALENDAR データセットのオブザベーションがそれぞれ 1 つの週次勤務スケジュールを定義します。次に示す DATA ステップで作成されるデータセットは、2 つのカレンダ、CALONE と CALTWO の週次勤務スケジュールを定義します。

```
data cale;
   input _sun_ $ _mon_ $ _tue_ $ _wed_ $ _thu_ $ /
         _fri_ $ _sat_ $ _cal_ $ d_length time6.;
   datalines;
holiday workday workday workday workday
workday holiday calone 8:00
holiday shift1 shift1 shift1 shift1
shift2 holiday caltwo 9:00
;
```

この CALENDAR データセットの変数は次のとおりです。

_SUN_から_SAT_まで
カレンダに表示される曜日名。これらの変数の値には、勤務シフト名が含まれます。勤務シフトの有効値は次のとおりです。

- **WORKDAY** (デフォルト勤務シフト)
- **HOLIDAY**(業務外期間)
- WORKDATA=データセットの変数名(この例では、**SHIFT1** と **SHIFT2**)

CAL
CALID (カレンダ識別子)変数。この変数の値によって、異なるカレンダが識別されます。この変数が存在しない場合は、このデータセットの最初のオブザベーションによって、ACTIVITIES データセットのすべてのカレンダに適用される勤務スケジュールが定義されます。

CALID 変数に欠損値が含まれている場合は、デフォルトカレンダの文字値または数値(**DEFAULT** または 0)が使用されます。詳細については、次を参照してください。"デフォルトのカレンダ" (174 ページ)

D_LENGTH
日の長さの識別子の変数。D_LENGTH の値は、カレンダ計算に使用される標準勤務時間の長さを示します。勤務時間の長さを設定するには、この変数を CALENDAR データセットに置くか、または DAYLENGTH=オプションを使用します。

この変数の欠損値が DAYLENGTH=オプションで指定した時間数のデフォルトとなります。DAYLENGTH=オプションを指定しない場合、日の長さは

INTERVAL=DAY の場合は 24 時間、INTERVAL=WORKDAY の場合は 8 時間がデフォルトとなります。

WORKDAYS データセットの代わりにデフォルト勤務シフトを使用

CALENDAR データセットは、WORKDAYS データセットがあってもなくても使用できます。WORKDAYS データセットがない場合、CALENDAR データセットの WORKDAY は、INTERVAL=オプションの設定に応じて、2 つの標準勤務時間のいずれかと等しくなります。

表 8.7 勤務時間の設定

INTERVAL=	勤務シフト開始時間	日の長さ
DAY	00:00	24 時間
WORKDAY	9:00	8 時間

標準勤務時間の長さは、DAYLENGTH=オプション、または CALENDAR データセットの D_LENGTH 変数でリセットできます。WORKDAYS データセットの他の勤務シフトを定義できます。

例

次の例では、CALENDAR データセットについて説明しています。

- “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
- “例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)
- “例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)

WORKDAYS データセット

目的

WORKDATA=オプションで指定した WORKDAYS データセットを使用して、CALEDATA=データセットで指定した日次勤務シフトを定義できます。

デフォルト勤務シフトを使用するか固有勤務シフトを作成する

アプリケーションで 2 つのデフォルト勤務シフトのうちいずれかが使用可能な場合、WORKDAYS データセットは不要です。

表 8.8 デフォルト勤務シフト

INTERVAL=	勤務シフト開始時間	日の長さ
DAY	00:00	24 時間
WORKDAY	9:00	8 時間

次を参照してください。“INTERVAL=DAY | WORKDAY ” (192 ページ)
“INTERVAL=DAY | WORKDAY ” (192 ページ)

構造

WORKDAYS データセットの各変数に、業務期間と業務外期間が交互に並ぶ日次スケジュールが 1 つずつ含まれます。たとえば、この DATA ステップでは、2 つの勤務シフトの指定を含むデータセットが作成されます。

```
data work;
   input shift1 time6. shift2 time6.;
   datalines;
7:00  7:00
12:00 11:00
13:00   .
17:00   .
;
```

変数 SHIFT1 は 1 つの業務外期間(昼食時間)を含む 1 日 10 時間勤務を、変数 SHIFT2 の指定は業務外期間を含まない 1 日 4 時間勤務をそれぞれ指定します。

欠損値の処理方法

欠損値は、最初のオブザベーションで 00:00、その他すべてのオブザベーションで 24 がデフォルトとなります。24:00 の値が 2 つ連続すると、ゼロの期間の長さが定義され、無視されます。

例

次を参照してください。“例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)

入力データセットの欠損値

次の表に、PROC CALENDAR で使用されるデータセットの変数の欠損値に対する処理を要約します。

表 8.9 *PROC CALENDAR での欠損値の処理*

データセット	変数	欠損値の処理
ACTIVITIES (DATA=)	“_CAL_” (181 ページ)	デフォルトカレンダ値が使用されます。
	“START ステートメント” (204 ページ)	オブザベーションは使用されません。
	“DUR ステートメント” (197 ページ)	1.0 が使用されます。
	“FIN ステートメント” (198 ページ)	START 値+日の長さが使用されます。
	“VAR ステートメント” (205 ページ)	サマリーカレンダまたは MISSING オプションが指定される場合、欠損値が使用されます。その他の場合、値は使用されません。

データセット	変数	欠損値の処理
	“SUM ステートメント” (204 ページ) , “MEAN ステートメント” (201 ページ)	0
CALENDAR (CALEDATA=)	“_CAL_” (181 ページ)	デフォルトカレンダ値が使用されます。
	“_SUN_から_SAT_まで” (181 ページ)	デフォルトカレンダの対応するシフトが使用されます。
	“D_LENGTH” (181 ページ)	可能な場合は DAYLENGTH=値が使用されます。または INTERVAL=DAY の場合は 24:00 が使用されます。その他の場合は、8:00 が使用されます。
	“SUM ステートメント” (204 ページ) , “MEAN ステートメント” (201 ページ)	0
HOLIDAY (HOLIDATA=)	“_CAL_” (181 ページ)	すべての休日がすべてのカレンダに適用されます。
	“HOLISTART ステートメント” (200 ページ)	オブザベーションは使用されません。
	“HOLIDUR ステートメント” (199 ページ)	可能な場合は、HOLIFIN 値が HOLIDUR 値の代わりに使用されます。その他の場合は 1.0 が使用されます。
	“HOLIFIN ステートメント” (199 ページ)	可能な場合は、HOLIDUR 値が HOLIFIN 値の代わりに使用されます。その他の場合は、HOLISTART 値+日の長さが使用されます。
	“HOLIVAR ステートメント” (201 ページ)	値は使用されません。
WORKDAYS (WORKDATA=)	任意	最初のオブザベーションについては、00:00 が使用されます。その他の場合は 24:00 が使用されます。

構文: CALENDAR プロシジャ

要件: START ステートメントを使用する必要があります。

スケジュールカレンダの場合は、DUR または FIN ステートメントも使用する必要があります。

ヒント: DUR または FIN ステートメントを使用する場合、PROC CALENDAR はスケジュールカレンダを作成します。

FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを PROC CALENDAR と併用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

グローバルステートメントを使用することもできます。リストは、“グローバルステートメント” (24 ページ)および“Global Statements” (*SAS Statements: Reference*)を参照してください。

PROC CALENDAR <*option(s)*>;
  START *variable*;
  BY <DESCENDING> *variable-1*
    <<DESCENDING> *variable-2* …>
    <NOTSORTED>;
  CALID *variable*
    </ OUTPUT=COMBINE | MIX | SEPARATE>;
  DUR *variable*;
  FIN *variable*;
  HOLISTART *variable*;
    HOLIDUR *variable*;
    HOLIFIN *variable*;
    HOLIVAR *variable*;
  MEAN *variable(s)* </ FORMAT=*format-name*>;
  OUTSTART *day-of-week*;
    OUTDUR *number-of-days*;
    OUTFIN *day-of-week*;
  SUM *variable(s)* </ FORMAT=*format-name*>;
  VAR *variable(s)*;

ステートメント	タスク	例
“PROC CALENDAR ステートメント”	SAS データセットのデータを月次カレンダ形式で表示します	Ex. 1, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 8
“BY ステートメント”	アクティビティを BY グループごとに別々に処理し、BY 変数の値ごとに個別のカレンダを作成します。	
“CALID ステートメント”	カレンダ識別子変数の値によって定義されたグループのアクティビティを処理します	Ex. 2, Ex. 3, Ex. 4, Ex. 6, Ex. 7, Ex. 8
“DUR ステートメント”	各アクティビティの期間を含む変数を指定します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
“FIN ステートメント”	各アクティビティの終了日を含む ACTIVITIES データセットの変数を指定します	Ex. 6
“HOLIDUR ステートメント”	スケジュールカレンダの各休日の期間を含む HOLIDAYS データセットの変数を指定します	Ex. 1, Ex. 5
“HOLIFIN ステートメント”	各休日の終了日を含む HOLIDAYS データセットの変数を指定します	
“HOLISTART ステートメント”	各休日の開始日を含む HOLIDAYS データセットの変数を指定します	Ex. 1, Ex. 5

ステートメント	タスク	例
“HOLIVAR ステートメント”	値が休日のラベル付けに使用される HOLIDAYS データセットの変数を指定します。	Ex. 1, Ex. 5
“MEAN ステートメント”	平均値が毎月計算される ACTIVITIES データセットの数値変数を指定します。	
“OUTDUR ステートメント”	表示される週の長さを日単位で指定します	
“OUTFIN ステートメント”	カレンダに表示する最後の曜日を指定します	Ex. 3, Ex. 4, Ex. 8
“OUTSTART ステートメント”	カレンダに表示する開始曜日を指定します	Ex. 3, Ex. 4, Ex. 8
“START ステートメント”	各アクティビティの開始日を含む ACTIVITIES データセットの変数を指定します。	Ex. 1
“SUM ステートメント”	各月の合計を出す ACTIVITIES データセットの数値変数を指定します	Ex. 8
“VAR ステートメント”	アクティビティごとに表示する変数を指定します	Ex. 6

PROC CALENDAR ステートメント

SAS データセットのデータを月次カレンダ形式で表示します。

例:
“例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)
“例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)
“例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)
“例 6: 前のタスクの完了に基づき、スケジュールを計算する” (231 ページ)
“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

構文

PROC CALENDAR <*option(s)*>;

オプション引数の要約

Control printing

FILL
: アクティビティが存在しない場合でも、すべての月を表示します。

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
: 外枠や分割線などに使用される文字を定義します。

HEADER=SMALL | MEDIUM | LARGE
: 月名の印刷で使用するヘッダーの種類を指定します。

LOCALE
月名と曜日名をローカル言語で表示します。

MISSING
欠損値の表示方法を指定します。

WEEKDAYS
出力の土曜日と日曜日の表示を非表示にします。

Control summary information

LEGEND
値がカレンダに表示される変数の名前を印刷します。

MEANTYPE=NOBS | NDAYS
計算する平均値の種類を各月に指定します。

Specify data sets containing

CALEDATA=*SAS-data-set*
週次勤務スケジュール

DATA=*SAS-data-set*
アクティビティ

HOLIDATA=*SAS-data-set*
休日

WORKDATA=*SAS-data-set*
一意のシフトパターン

Specify time or duration

DATETIME
START および FIN 変数に DATETIME.出力形式の値が含まれるように指定します。

DAYLENGTH=*hours*
標準勤務時間数を指定します。

INTERVAL=DAY | WORKDAY
DUR 変数と HOLIDUR 変数の単位を指定します。

オプション引数

CALEDATA=*SAS-data-set*
CALENDAR データセット(複数のカレンダの週次勤務スケジュールを含む SAS データセット)を指定します。

デフォルト
CALEDATA=オプションを省略すると、PROC CALENDAR はデフォルト勤務スケジュールを使用します。

ヒント
CALENDAR データセットは、複数のカレンダや非標準の勤務スケジュールを使用する場合に便利です。

参照項目
“デフォルトのカレンダ” (174 ページ)

“CALENDAR データセット ” (181 ページ)

例
“例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)

“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)

DATA=*SAS-data-set*

ACTIVITIES データセット(すべてのアクティビティの開始日、およびアクティビティごとに表示する変数を含む SAS データセット)を指定します。アクティビティは、開始日によって並べ替え、またはインデックス付けを行う必要があります。

デフォルト DATA=オプションを省略すると、最も新しく作成された SAS データセットが使用されます。

参照項目 “ACTIVITIES データセット ” (178 ページ)

例 “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)

DATETIME

START および FIN 変数に DATETIME.出力形式の値が含まれるように指定します。

デフォルト DATETIME オプションを省略すると、PROC CALENDAR は、START 値と FIN 値が DATE.出力形式であるとみなします。

例 “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)

“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)

“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

DAYLENGTH=*hours*

時間値は SAS TIME 値である必要があります。

デフォルト INTERVAL=DAY の場合は 24 (デフォルト)、INTERVAL=WORKDAY の場合は 8.

制限事項 DAYLENGTH=はスケジュールカレンダにのみ適用されます。

操作 DAYLENGTH=オプションを指定し、CALENDAR データセットに D_LENGTH 変数が含まれている場合、PROC CALENDAR は、D_LENGTH が欠損値のときに限り、DAYLENGTH=値を使用します。

INTERVAL=DAY で、CALEDATA=データセットがない場合は、DAYLENGTH=値の指定は無効になります。

ヒント DAYLENGTH=オプションは、DUR ステートメントを使用しており、勤務スケジュールにさまざまな長さの日(たとえば週 5 日の半日勤務)が含まれる場合に有用です。日の長さがさまざまな勤務週では、期間の計算に使用する標準的な日の長さを設定する必要があります。たとえば、勤務日の期間が 3.0 日のアクティビティは、DAYLENGTH=8:00 の場合は 24 時間、DAYLENGTH=10:00 の場合は 30 時間続きます。

DAYLENGTH=オプションを指定する代わりに、CALEDATA=データセットの D_LENGTH 変数を使用して、勤務時間の長さを指定できます。この方法を使用すると、異なるカレンダに対して異なる標準的な日の長さを指定できます。

参照項目 “CALENDAR データセット ” (181 ページ) 標準勤務時間の長さの設定の詳細については、を参照してください。

FILL

最初のアクティビティから最後のアクティビティまでのすべての月を、開始日から終了日まで、アクティビティが含まれない月も含めて表示します。

デフォルト FILL を指定しない場合、PROC CALENDAR は*アクティビティ*を含む月のみ印刷します。*休日*のみ含む月は印刷されません)。

例 “例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'

カレンダのセルの外枠と分割線の構成に使用される文字、および PROC CALENDAR 出力で休日やアクティビティの続行を示すために使用される識別マーカー(アスタリスクや矢印など)を定義します。

position(s)

SAS フォーマット文字列における 1 つ以上の文字の位置を識別します。スペースまたはカンマで位置を区切ります。

デフォルト *position(s)*)を省略する場合、20 の可能なすべてのシステムフォーマット文字を指定することと同じです。

範囲 PROC CALENDAR では、SAS で提供している 20 のフォーマット文字のうち 17 が使用されます。

参照項目 表 8.10 (190 ページ)に、PROC CALENDAR で使用されるフォーマット文字を示します。

図 8.1 (190 ページ)に、PROC CALENDAR 出力での使用法を示します。

formatting-character(s)

指定位置に使用する文字をリストします。PROC CALENDAR は、*formatting-character(s)*の文字を表示されている順序で *position(s)*に割り当てます。たとえば、次のオプションでは、アスタリスク(*)を 12 番目の位置に、シングルハイフン(-)を 13 番目に割り当てます。その他の文字は変更されません。

```
formchar(12 13)='*-'
```

これらの新しい設定によって、アクティビティ行が次の行から、

```
+=================ACTIVITY===============+
```

次の行に変更されます。

```
*-----------------ACTIVITY--------------*
```

図 8.1 *PROC CALENDAR 出力のフォーマット文字*

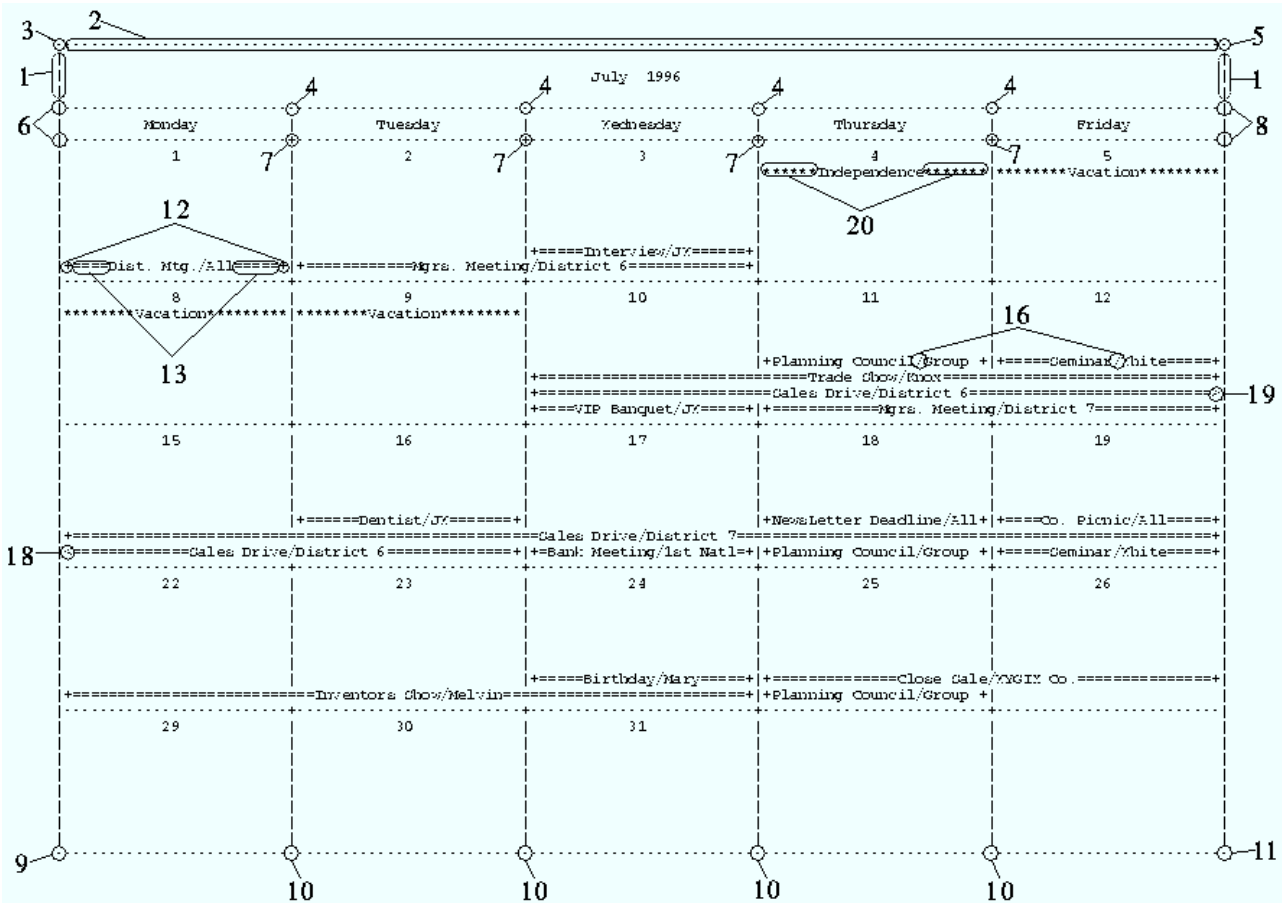


表 8.10 *PROC CALENDAR によって使用されるフォーマット文字*

Position	デフォルト	表示に使用
1	|	縦棒
2	-	横棒
3	-	セル:左上隅
4	-	セル:中央上交点
5	-	セル:右上隅
6	|	セル:中央左側
7	+	セル:中央交点
8	|	セル:中央右側
9	-	セル:左下隅
10	-	セル:中央下交点

Position	デフォルト	表示に使用
11	-	セル:右下隅
12	+	アクティビティの開始と終了
13	=	アクティビティ行
16	/	アクティビティの区切り
18	<	アクティビティ続行開始
19	>	アクティビティ続行終了
20	*	休日マーカー

操作 SAS システムオプション FORMCHAR=では、デフォルトのフォーマット文字を指定します。SAS システムオプションは、フォーマット文字の文字列列全体を定義します。プロシジャの FORMCHAR=オプションでは、選択した文字を再定義できます。

ヒント 16 進数文字を含む *formatting-characters* の文字を使用できます。16 進数文字を使用する場合、**x** を終了引用符の後に付ける必要があります。たとえば、次のオプションでは、16 進文字 2-D が 3 番目のフォーマット文字に、16 進文字 7C が 7 番目の文字にそれぞれ割り当てられます。その他の文字は変わりません。**formchar(3,7)='2D7C'x**

参照項目 どの 16 進コードをどの文字に使用するかについては、ハードウェアのドキュメントを参照してください。

HEADER=SMALL | MEDIUM | LARGE
月名の印刷で使用するヘッダーの種類を指定します。

SMALL
月と年が 1 行に印刷します。

MEDIUM
月と年が高さ 4 行分のボックス内に印刷されます。

LARGE
月がアスタリスク(*)を使用して高さ 7 行分で印刷されます。スペースがあれば年も含まれます。

デフォルト MEDIUM

HOLIDATA=*SAS-data-set*
HOLIDAYS データセット(出力に表示する休日を含む SAS データセット)を指定します。1 つの変数に休日名、もう 1 つの変数に各休日の開始日が含まれている必要があります。PROC CALENDAR は、スペースがあればカレンダ出力の休日をアスタリスク(*)でマーク付けます。

操作 カレンダに休日を表示するには、HOLIDAYS データセットと HOLISTART ステートメントが必要です。休日の指定には、HOLIVAR ステートメントをお勧めします。1 日よりも長く続く休日がある場合、HOLIDUR は必須です。

ヒント HOLIDAYS データセットは、並べ替える必要はありません。

参照項目 “HOLIDAYS データセット ” (179 ページ)

例 “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)

“例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)

INTERVAL=DAY | WORKDAY

DUR 変数と HOLIDUR 変数の単位を、2 つのデフォルトの日の長さのうちいずれかに指定します。

DAY

DUR 変数と HOLIDUR 変数の値を 1 日 24 時間単位で指定し、デフォルトの週 7 日のカレンダを指定します。たとえば、DUR 値 3.0 は 72 時間として処理されます。デフォルトカレンダ勤務スケジュールは勤務日 7 日で構成され、すべて長さ 24:00 で 00:00 に開始します。

WORKDAY

DUR 変数と HOLIDUR 変数の値を 1 日 8 時間単位で指定します。また、WORKDAY は、デフォルトカレンダに月曜日から金曜日までの週 5 日を含めるように指定します。これはすべて長さ 08:00 で 09:00 に開始されます。WORKDAY が指定されると、PROC CALENDAR は、DAYLENGTH=オプション、CALEDATA=データセットまたはデフォルトカレンダでの定義どおりに、DUR 変数と HOLIDUR 変数の値を勤務時間単位で処理します。たとえば、勤務時間が 8 時間の場合は、DUR 値 3.0 は 24 時間として処理されます。

例 “例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)

デフォルト DAY

操作 CALEDATA=データセットがない場合、PROC CALENDAR はデフォルトカレンダで定義された勤務スケジュールを使用します。

WEEKDAYS オプションは、INTERVAL=値を自動的に WORKDAY に設定します。

参照項目 “カレンダと複数のカレンダ” (175 ページ) および “CALENDAR データセット ” (181 ページ) (INTERVAL=オプションと勤務時間指定の詳細、“デフォルトのカレンダ” (174 ページ)

例 “例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)

LEGEND

値がカレンダに表示される変数の名前を印刷します。この識別テキスト(凡例ボックス)は、スペースがあれば各月のページ下部に表示されます。スペースがなければ、次のページに印刷されます。PROC CALENDAR は、各変数を名前またはラベル(存在する場合)で識別します。凡例の変数の順序は、カレンダでの順序と一致します。

制限事項 LEGEND はサマリーカレンダにのみ適用されます。

操作 SUM および MEAN ステートメントを使用する場合、凡例ボックスに SUM 値と MEAN 値も含まれます。

例 “例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

LOCALE
LOCALE= SAS システムオプションの値によって示される言語で月と曜日の名前を印刷します。PROC CALENDAR の LOCALE オプションは、週の開始日を変更しません。

デフォルト LOCALE を指定しない場合、月と曜日の名前は英語で印刷されます。

MEANTYPE=NOBS | NDAYS
計算する平均値の種類を各月に指定します。

NOBS
その月に表示される *observations* の数の平均が計算されます。

NDAYS
その月に表示される *days* の数の平均が計算されます。

デフォルト NOBS

制限事項 MEANTYPE=はサマリーカレンダにのみ適用されます。

操作 通常、PROC CALENDAR は各月のすべての日を表示します。ただし、OUTSTART ステートメントを OUTDUR または OUTFIN ステートメントとあわせて使用する場合は、一部の日が省略されることがあります。

例 “例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)

MISSING
カレンダの種類に基づいて、欠損値の処理方法を決定します。

サマリーカレンダ
アクティビティがスケジュールされていない日がある場合、PROC CALENDAR は、欠損値に対して指定された SAS 出力形式またはユーザー定義の出力形式を使用して、その日の変数値を印刷します。

デフォルト MISSING を省略すると、アクティビティのない日には値が含まれません。

スケジュールカレンダ
欠損値を含む変数は、欠損値に対して指定された出力形式を使用して、アクティビティのラベルに表示されます。

デフォルト MISSING を指定しない場合、PROC CALENDAR は、アクティビティのラベル付けで欠損値を無視します。

参照項目 “入力データセットの欠損値” (183 ページ) (欠損値の詳細)

WEEKDAYS
出力の土曜日と日曜日の表示を非表示にします。また、INTERVAL=オプションの値が WORKDAY になるように指定します。

```
  proc calendar weekdays;
    start date;
run;
proc calendar interval=workday;
    start date;
outstart monday;
outfin friday;
run;
```

デフォルト WEEKDAYS を省略すると、カレンダに 7 日がすべて表示されます。

ヒント ここで示されているように、WEEKDAYS オプションは INTERVAL=WORKDAY を OUTSTART および OUTFIN ステートメントと組み合わせて使用する代わりに指定できます。

例 “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)

WORKDATA=*SAS-data-set*

workdays data set (標準勤務日の勤務パターンを定義する SAS データセット)を指定します。WORKDAYS データセットの各数値は、1 勤務日の一意の勤務シフトパターンを示します。

ヒント WORKDAYS データセットは、CALENDAR データセットと組み合わせると便利です。

参照項目 “WORKDAYS データセット ” (182 ページ) および “CALENDAR データセット ” (181 ページ)

例 “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)

“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)

BY ステートメント

アクティビティを BY グループごとに別々に処理し、BY 変数の値ごとに個別のカレンダを作成します。

サポート: Summary and schedule calendars

参照項目: “CALID ステートメント” (195 ページ)

“BY” (68 ページ) (主な説明)

例: “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* …>
<NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できますが、データセットのオブザベーションは、指定する変数の順序に並べ替えるか、または適切なインデックスを付ける必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
オブザベーションが BY ステートメントの文字 DESCENDING の直後に続く変数で降順に並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。オブザベーションは別の方法(時系列など)でグループ化されます。

詳細

CALID ステートメントを使用して、CALID 変数の値によって示される異なるカレンダに適用されるアクティビティを処理できます。ただし、1 つの CALID 変数のみ指定可能なため、行えるのは 1 レベルのグループ化のみです。たとえば、企業内の複数の部門のアクティビティを表示するカレンダレポートを作成する場合は、CALID 変数の値を含む部門をそれぞれ識別し、すべての部門のカレンダを表示するカレンダ出力を作成できます。

ただし、BY ステートメントを使用すると、さらにアクティビティを関連グループに分割できます。たとえば、部門のカレンダを部ごとにグループ化したカレンダ出力を印刷できます。アクティビティのオブザベーションには、アクティビティが属する部門を識別する変数と、部門が存在する部を識別する変数が含まれている必要があります。部門を識別する変数は CALID ステートメントで指定します。部を識別する変数は BY ステートメントで指定します。

CALID ステートメント

カレンダ識別子変数の値によって定義されるグループのアクティビティを処理します。

サポート: サマリーカレンダとスケジュールカレンダ

ヒント: CALID は、複数のスケジュールカレンダの作成、および SAS/OR ソフトウェアでの使用に便利です。

参照項目: “CALENDAR データセット ” (181 ページ)

例:
“例 2: 複数のカレンダを含むスケジュールカレンダ” (212 ページ)
“例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)
“例 6: 前のタスクの完了に基づき、スケジュールを計算する” (231 ページ)
“例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)
“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

構文

CALID *variable*
</ OUTPUT=COMBINE | MIX | SEPARATE>;

必須引数

variable
オブザベーションにどのカレンダのデータが含まれているかを識別する文字変数または数値変数。

要件 CALID 変数を指定する場合は、その変数を ACTIVITIES データセットと HOLIDAYS データセットの両方に含める必要があります。どちらのデータセットにも CALID 変数が含まれていない場合は、デフォルトカレンダが使用されます。

操作 SAS/OR ソフトウェアではこの変数を使用して、オブザベーションにどのカレンダのデータが含まれているかを識別します。

ヒント この変数を作成するために CALID ステートメントを使用する必要はありません。デフォルト変数_CALID_を入力データセットに含めることができます。

参照項目 “CALENDAR データセット ” (181 ページ)

オプション引数

OUTPUT=COMBINE|MIX|SEPARATE
複数のカレンダ出力の表示に必要なスペース量を制御します。

COMBINE
アクティビティを含む月ごとに 1 ページ生成し、各日を CALID 値によって分割します。

制限事項 入力データは、START 変数によって並べ替えるか、インデックス付けする必要があります。

例 “例 2: 複数のカレンダを含むスケジュールカレンダ” (212 ページ)

“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)

MIX
アクティビティを含む月ごとに 1 ページ生成しますが、CALID 値によってアクティビティを識別しません。

制限事項 入力データは、START 変数によって並べ替えるか、インデックス付けする必要があります。

ヒント MIX を指定すると、出力に必要なスペースが最小になります。

例 “例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)

SEPARATE
CALID 変数の値ごとに個別のページを生成します。

制限事項 入力データは、CALID 変数、START 変数の順に並べ替えるか、適切な複合インデックスを含める必要があります。

例 “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)

“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

“例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)

デフォルト COMBINE

DUR ステートメント

各アクティビティの期間を含む変数を指定します。

別名: DURATION

操作: DUR ステートメントと FIN ステートメントの両方を使用すると、DUR は無視されます。

サポート: スケジュールカレンダ

ヒント: スケジュールカレンダを生成するには、DUR ステートメントか FIN ステートメントのいずれかを使用する必要があります。

例: “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)
“例 2: 複数のカレンダを含むスケジュールカレンダ” (212 ページ)
“例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)
“例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)

構文

DUR *variable*;

必須引数

variable
スケジュールカレンダの各アクティビティの期間が含まれます。

範囲 期間には実数値または整数値が可能です。

制限事項 この変数は ACTIVITIES データセットに含まれている必要があります。

参照項目 アクティビティ期間の詳細については、次を参照してください。 “ACTIVITIES データセット ” (178 ページ) と “CALENDAR データセット ” (181 ページ)

詳細

アクティビティ開始(START 変数で指定)からの全期間が測定されます。出力では、1 日の一部だけ続くアクティビティはすべて 1 日中続くものとして表示されます。

PROC CALENDAR ステートメントの INTERVAL=オプションは、次にのように固有の値に応じて自動的に期間変数の単位を設定します。

表 8.11 INTERVAL=設定

INTERVAL=	デフォルト期間単位長
DAY (デフォルト)	24 時間
WORKDAY	8 時間

次のうちいずれかを使用して、デフォルト期間単位長を無効にできます。

- DAYLENGTH=オプション
- CALEDATA=データセットの D_LENGTH 変数

FIN ステートメント

各アクティビティの終了日を含む ACTIVITIES データセットの変数を指定します。

別名: FINISH

操作: FIN ステートメントと DUR ステートメントの両方を使用すると、FIN が使用されます。

サポート: スケジュールカレンダ

ヒント: スケジュールカレンダを生成するには、FIN ステートメントか DUR ステートメントのいずれかを使用する必要があります。

例: “例 6: 前のタスクの完了に基づき、スケジュールを計算する” (231 ページ)

構文

FIN *variable*;

必須引数

variable
各アクティビティの終了日が含まれます。

制限事項

variable の値は、SAS 日付値か SAS 日時値のいずれかである必要があります。

FIN 変数に日時値が含まれる場合は、PROC CALENDAR ステートメントで DATETIME オプションを指定する必要があります。

START 変数と FIN 変数の両方の出力形式が一致している必要があります。たとえば、一方に日時値が含まれている場合は、他方も同様である必要があります。

HOLIDUR ステートメント

スケジュールカレンダの各休日の期間を含む HOLIDAYS データセットの変数を指定します。

別名: HOLIDURATION

デフォルト: HOLIDUR ステートメントも HOLIFIN ステートメントも使用しない場合、すべての休日の期間は 1 日になります。

制限事項: HOLIDUR ステートメントは HOLIFIN とは併用できません。

サポート: スケジュールカレンダ

例: “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)
“例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)

構文

HOLIDUR *variable*;

必須引数

variable
各休日の期間が含まれます。

範囲 期間には実数値または整数値が可能です。

制限事項 この変数は HOLIDAYS データセットに含まれている必要があります。

例 “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

詳細

- HOLIFIN ステートメントと HOLIDUR ステートメントの両方を使用すると、PROC CALENDAR は HOLIFIN 変数値を使用して各休日の期間を定義します。
- 休日期間変数の *unit* を、期間変数の単位と同じように設定します。INTERVAL=オプションと DAYLENGTH=オプション、または CALEDATA=データセットのいずれかを使用します。
- 休日開始(HOLISTART 変数で指定)からの全期間が測定されます。出力では、少なくとも半日続く休日はすべて 1 日中続くものとして表示されます。

HOLIFIN ステートメント

各休日の終了日を含む HOLIDAYS データセットの変数を指定します。

別名: HOLIFINISH

デフォルト: HOLIFIN ステートメントも HOLIDUR ステートメントも使用しない場合、すべての休日の期間は 1 日になります。

サポート: スケジュールカレンダ

構文

HOLIFIN *variable*;

必須引数

variable
各休日の終了日が含まれます。

制限事項

この変数は HOLIDAYS データセットに含まれている必要があります。

variable の値は、SAS 日付値または SAS 日時値のいずれかである必要があります。

HOLIFIN 変数に日時値が含まれる場合、PROC CALENDAR ステートメントで DATETIME オプションを指定する必要があります。

HOLIFIN ステートメントも HOLIDUR ステートメントも指定しない場合、すべての休日の期間は 1 日になります。

詳細

HOLIFIN ステートメントと HOLIDUR ステートメントの両方を使用すると、PROC CALENDAR は HOLIFIN 変数のみ使用します。

HOLISTART ステートメント

各休日の開始日を含む HOLIDAYS データセットの変数を指定します。

別名: HOLISTA、HOLIDAY

要件 HOLIDAYS データセットを使用する場合は、HOLISTART が必須になります。

サポート: サマリーカレンダとスケジュールカレンダ

例: “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)
“例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)

構文

HOLISTART *variable*;

必須引数

variable
各休日の開始日が含まれます。

制限事項 *variable* の値は、SAS 日付値または SAS 日時値のいずれかである必要があります。

HOLISTART 変数に日時値が含まれる場合は、PROC CALENDAR ステートメントで DATETIME オプションを指定します。

詳細

- HOLIDAYS データセットは、並べ替える必要はありません。
- HOLIFIN ステートメントまたは HOLIDUR ステートメントを使用しない限り、すべての休日は 1 日のみ続きます。
- 同日に 2 つ以上の休日が発生すると、PROC CALENDAR は最初のオブザベーションのみ使用します。

HOLIVAR ステートメント

値が休日のラベル付けに使用される HOLIDAYS データセットの変数を指定します。

別名: HOLIVARIABLE、HOLINAME

デフォルト: HOLIVAR ステートメントを使用しない場合、PROC CALENDAR は *DATE* を使用して休日を識別します。

サポート: サマリーカレンダとスケジュールカレンダ

例: “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)
“例 5: ブランク表示または休日表示付きのスケジュールカレンダ” (228 ページ)

構文

HOLIVAR *variable*;

必須引数

variable
値が休日のラベル付けに使用される変数。通常、この変数には休日名が含まれます。

範囲 文字または数値。

制限事項 この変数は HOLIDAYS データセットに含まれている必要があります。

HOLIVAR ステートメントを指定しない場合、PROC CALENDAR は **DATE** を使用して休日を識別します。

ヒント HOLIVAR 変数には自由にフォーマットできます。

MEAN ステートメント

各月の平均値が計算される ACTIVITIES データセットの数値変数を指定します。

サポート: サマリーカレンダ

ヒント: 複数の MEAN ステートメントを使用できます。

参照項目: “例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)

構文

MEAN *variable(s)* </ FORMAT=*format-name*>;

必須引数

variable(s)
各月の平均値が計算される数値変数。

制限事項 この変数は ACTIVITIES データセットに含まれている必要があります。

オプション引数

FORMAT=*format-name*
要求された平均の表示に使用される SAS 出力形式またはユーザー定義出力形式を指定します。

別名 F=

デフォルト BEST.出力形式

例 “例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)

詳細

- スペースがあれば、平均はサマリーカレンダページの下部に表示されます。スペースがない場合は次のページに表示されます。
- LEGEND オプションを指定した場合、平均は **LEGEND** ボックスに表示されます。
- PROC CALENDAR は、MEAN ステートメントで指定された変数が VAR ステートメントで指定されていなくても、自動的にカレンダ出力に表示します。

OUTDUR ステートメント

表示される週の長さを日単位で指定します。

別名: OUTDURATION

要件 OUTSTART ステートメントが必須です。

構文

OUTDUR *number-of-days*;

必須引数

number-of-days
表示される週の長さを日単位で表す整数。

詳細

表示する週の長さに関する情報をプロシジャに提供するには、OUTDUR ステートメントまたは OUTFIN ステートメントのいずれかを使用します。両方を使用すると、PROC CALENDAR は OUTDUR ステートメントを無視します。

OUTFIN ステートメント

カレンダに表示する最後の曜日を指定します。

別名: OUTFINISH

要件 OUTSTART ステートメントが必須です。

参照項目: “例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

例: “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)
“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

構文

OUTFIN *day-of-week*;

必須引数

day-of-week
表示する最後の曜日名。たとえば、

```
outfin friday;
```

詳細

表示する週の長さに関する情報をプロシジャに提供するには、OUTFIN ステートメントまたは OUTDUR ステートメントのいずれかを使用します。両方を使用すると、PROC CALENDAR は OUTFIN ステートメントのみ使用します。

OUTSTART ステートメント

カレンダに表示する開始曜日を指定します。

別名: OUTSTA

デフォルト: OUTSTART を使用しない場合、各カレンダ週は日曜日で始まります。

参照項目: “例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

例: “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ)
“例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ)
“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

構文

OUTSTART *day-of-week*;

必須引数

day-of-week
カレンダの各週の開始曜日名。たとえば、

```
outstart monday;
```

詳細

デフォルトでは、カレンダに 7 つの曜日がすべて表示されます。表示日数と開始曜日を制御するには、OUTDUR または OUTFIN を OUTSTART と組み合わせて使用します。

START ステートメント

各アクティビティの開始日を含む ACTIVITIES データセットの変数を指定します。

別名: STA、DATE、ID

要件 START はサマリーカレンダとスケジュールカレンダの両方に必須です。

例: “例 1: 休日表示付きスケジュールカレンダ:週 5 日” (208 ページ)

構文

START *variable*;

必須引数

variable
各アクティビティの開始日が含まれます。

制限事項

この変数は ACTIVITIES データセットに含まれている必要があります。

variable の値は、SAS 日付値または SAS 日時値のいずれかである必要があります。

日時値を使用する場合、PROC CALENDAR ステートメントで DATETIME オプションを指定します。

START 変数と FIN 変数の両方の出力形式が一致している必要があります。たとえば、一方に日時値が含まれている場合は、他方も同様である必要があります。

SUM ステートメント

各月の合計を出す ACTIVITIES データセットの数値変数を指定します。

サポート: サマリーカレンダ

ヒント: 合計中の変数に異なる出力形式を適用するには、複数の SUM ステートメントを使用します。

例: “例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)
“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

構文

SUM *variable(s)* </ FORMAT=*format-name*>;

必須引数

variable(s)
各月の合計を出す数値変数を 1 つ以上指定します。

制限事項 この変数は ACTIVITIES データセットに含まれている必要があります。

オプション引数

FORMAT=*format-name*
要求した合計の表示に使用する SAS 出力形式またはユーザー定義出力形式を指定します。

別名 F=

デフォルト BEST.出力形式

例 “例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ” (239 ページ)
“例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)” (244 ページ)

詳細

- スペースがあれば、合計はカレンダページの下部に表示されます。スペースがなければ、次のページに表示されます。
- LEGEND オプションを指定した場合、合計は **LEGEND** ボックスに表示されます。
- PROC CALENDAR は、SUM ステートメントで指定された変数が VAR ステートメントで指定されていなくても、自動的にカレンダ出力に表示します。

VAR ステートメント

アクティビティごとに表示する変数を指定します。

別名: VARIABLE

例: “例 6: 前のタスクの完了に基づき、スケジュールを計算する” (231 ページ)

構文

VAR *variable(s)*;

必須引数

variable(s)
カレンダに表示する変数を 1 つ以上指定します。

範囲	*variable* の値には文字または数値が可能です。
制限事項	これらの変数は ACTIVITIES データセットに含まれている必要があります。
ヒント	この変数に出力形式を適用できます。

詳細

VAR を使用しない場合

VAR ステートメントを使用しない場合、プロシジャは、ACTIVITIES データセット内の BY、CALID、START、DUR、FIN 変数を除くすべての変数をデータセットで発生する順序で表示します。ただし、LINESIZE=と PAGESIZE=の設定でカレンダに十分なスペースが許可されない場合、一部の変数は表示されません。

変数の表示

- PROC CALENDAR は、変数を VAR ステートメントに表示される順序で表示します。ただし、LINESIZE=および PAGESIZE=の設定でカレンダに十分なスペースが許可されない場合、一部の変数は表示されません。
- また、PROC CALENDAR は、カレンダ出力の各アクティビティに対して SUM ステートメントまたは MEAN ステートメントで指定した変数もすべて表示します。その変数は、VAR ステートメントで指定していなくても表示されます。

結果:CALENDAR プロシジャ

PROC CALENDAR 出力量に影響する要素

印刷カレンダ出力量は、次の変数によって異なります。

- ACTIVITIES データセットの日付範囲
- FILL オプションの指定の有無
- BY ステートメント
- CALID ステートメント

PROC CALENDAR は常に、アクティビティを含む月ごとに 1 つずつカレンダを印刷します。FILL オプションを指定すると、プロシジャは、アクティビティを含まない月も含めて、最初のアクティビティから最後のアクティビティまでのすべての月を印刷します。BY ステートメントを使用すると、BY 値ごとに 1 つずつ出力セットが印刷されます。CALID ステートメントと OUTPUT=SEPARATE を使用すると、CALID 変数値ごとに 1 つずつ出力セットが印刷されます。

サイズによる PROC CALENDAR 出力形式への影響

PROC CALENDAR は、SAS システムオプション PAGESIZE=および LINESIZE=の定義どおりに、常にカレンダを 1 ページ内におさめようとします。PAGESIZE=値と LINESIZE=値によって十分なスペースができなかった場合、PROC CALENDAR は凡例ボックスを別のページに印刷することがあります。PROC CALENDAR は、必要に応じて、出力をページに合わせるために値の切り捨てや省略を行い、その旨を示すメッセージを SAS ログに印刷します。

アクティビティ期間を表示するラインに影響する要素

スケジュールカレンダでは、アクティビティ期間は、アクティビティの各日に引かれた連続ラインによって示されます。各アクティビティの変数値は、同じライン上に、スラッシュ(/)で区切って印刷されます。各アクティビティの最初と最後はプラス記号(+)で示されます。アクティビティが、ある週から次の週まで続行する場合、PROC CALENDAR は、続行箇所に矢印(< >)を表示します。

アクティビティラインの長さは、使用可能な横のスペースの量によって異なります。次の変数を指定すると、長さを増やせます。

- OPTIONS ステートメントで LINESIZE=オプションを使用してページサイズを大きくします。
- WEEKDAYS オプションで土曜日と日曜日の印刷を抑制して、月曜日から金曜日までのスペースを増やします。

カレンダ表示のカスタマイズ

PROC CALENDAR は、20 のうち 17 の SAS フォーマット文字を使用して、カレンダの外枠の構成、アクティビティラインの印刷、休日の識別を行います。FORMCHAR=オプションを使用すると、PROC CALENDAR 出力の表示をカスタマイズして、デフォルトの代わりに固有の文字を使用できます。図 8.1 (190 ページ) および 表 8.10 (190 ページ)を参照してください。

プリンタで*拡張文字セット*(通常の英数字に加えてグラフィック文字も含む文字セット)がサポートされている場合は、FORMCHAR=オプションを使用してフォーマット文字を 16 進文字で再定義することにより、出力の表示を大幅に改善できます。どの 16 進コードをどの文字に使用するかについては、ハードウェアのドキュメントを参照してください。16 進値の割り当て例については、*"formatting-character(s)"* (189 ページ)を参照してください。

PROC CALENDAR による ODS 出力のポータビリティ

特定の状況で PROC CALENDAR を Output Delivery System と使用すると、ポータブルでないファイルが生成されます。SAS セッションの SAS システムオプション FORMCHAR=で非標準の線描文字が使用されると、SAS Monospace フォントがインストールされていない動作環境では線の代わりに不正な文字が出力に含まれていることがあります。この問題を回避するため、PROC CALENDAR を実行する前に次の OPTIONS ステートメントを指定します。

```
options formchar="|----|+|---+=|-/\<>*";
```

例: CALENDAR プロシジャ

例 1: 休日表示付きスケジュールカレンダ:週 5 日

要素: PROC CALENDAR ステートメントオプション
- DATA=
- HOLIDATA=
- WEEKDAYS

その他のステートメント
- DUR statement
- HOLISTART statement
- HOLIVAR statement
- HOLIDUR statement
- START statement

その他の機能
- PROC SORT ステートメント
- BY ステートメント
- 週 5 日のデフォルトカレンダ

詳細

この例では、次を行います。

- スケジュールカレンダを作成します。
- 2 つのデフォルト勤務パターン(1 日 8 時間、週 5 日)のうち 1 つを使用します。
- アクティビティを休日を避けてスケジュールします。
- 週 5 日を表示します。

プログラム

```
data allacty;
   input date : date7. event $ 9-36 who $ 37-48 long;
   datalines;
01JUL02 Dist. Mtg.                  All          1
17JUL02 Bank Meeting                1st Natl     1
02JUL02 Mgrs. Meeting               District 6   2
11JUL02 Mgrs. Meeting               District 7   2
03JUL02 Interview                   JW           1
08JUL02 Sales Drive                 District 6   5
15JUL02 Sales Drive                 District 7   5
08JUL02 Trade Show                  Knox         3
22JUL02 Inventors Show              Melvin       3
11JUL02 Planning Council            Group II     1
18JUL02 Planning Council            Group III    1
25JUL02 Planning Council            Group IV     1
12JUL02 Seminar                     White        1
19JUL02 Seminar                     White        1
```

```
18JUL02 NewsLetter Deadline         All          1
05JUL02 VIP Banquet                 JW           1
19JUL02 Co. Picnic                  All          1
16JUL02 Dentist                     JW           1
24JUL02 Birthday                    Mary         1
25JUL02 Close Sale                  WYGIX Co.    2
;

data hol;
   input date : date7. holiday $ 11-25 holilong @27;
   datalines;
05jul02   Vacation        3
04jul02   Independence    1
;

proc sort data=allacty;
   by date;
run;

options formchar="|----|+|---+=|-/\<>*";

proc calendar data=allacty holidata=hol weekdays;

   start date;
   dur long;

   holistart date;
   holivar holiday;
   holidur holilong;

   title1 'Summer Planning Calendar:  Julia Cho';
   title2 'President, Community Bank';
run;
```

プログラムの説明

ACTIVITIES データセットを作成します。 Allacty には、銀行頭取について個人アクティビティとビジネスアクティビティの両方の情報が含まれています。

```
data allacty;
   input date : date7. event $ 9-36 who $ 37-48 long;
   datalines;
01JUL02 Dist. Mtg.                  All          1
17JUL02 Bank Meeting                1st Natl     1
02JUL02 Mgrs. Meeting               District 6   2
11JUL02 Mgrs. Meeting               District 7   2
03JUL02 Interview                   JW           1
08JUL02 Sales Drive                 District 6   5
15JUL02 Sales Drive                 District 7   5
08JUL02 Trade Show                  Knox         3
22JUL02 Inventors Show              Melvin       3
11JUL02 Planning Council            Group II     1
18JUL02 Planning Council            Group III    1
25JUL02 Planning Council            Group IV     1
12JUL02 Seminar                     White        1
19JUL02 Seminar                     White        1
18JUL02 NewsLetter Deadline         All          1
05JUL02 VIP Banquet                 JW           1
19JUL02 Co. Picnic                  All          1
```

```
16JUL02 Dentist                    JW          1
24JUL02 Birthday                   Mary        1
25JUL02 Close Sale                 WYGIX Co.   2
;
```

HOLIDAYS データセットを作成します。

```
data hol;
   input date : date7. holiday $ 11-25 holilong @27;
   datalines;
05jul02   Vacation        3
04jul02   Independence    1
;
```

開始日を含む変数を基準にして ACTIVITIES データセットを並べ替えます。HOLIDAYS データセットは並べ替える必要はありません。

```
proc sort data=allacty;
   by date;
run;
```

FORMCHAR オプションを設定します。FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

スケジュールカレンダを作成します。DATA=は ACTIVITIES データセットを識別し、HOLIDATA=は HOLIDAYS データセットを識別します。WEEKDAYS は、1 週間が 5 日間の 8 時間勤務で構成されるように指定します。

```
proc calendar data=allacty holidata=hol weekdays;
```

アクティビティ開始日の変数とアクティビティ期間の変数を指定します。START ステートメントはアクティビティの開始日を含む ACTIVITIES データセットの変数を指定し、DUR は各アクティビティの期間を含む変数を指定します。スケジュールカレンダの作成には、START と DUR が必須です。

```
   start date;
   dur long;
```

休日情報を取得します。HOLISTART、HOLIVAR および HOLIDUR ステートメントは、各休日の開始日、名前および期間のそれぞれを含む HOLIDAYS データセットの変数を指定します。HOLIDAYS データセットを使用する場合は、HOLISTART が必須になります。少なくとも 1 つの休日が 2 日以上続くため、HOLIDUR が必須になります。

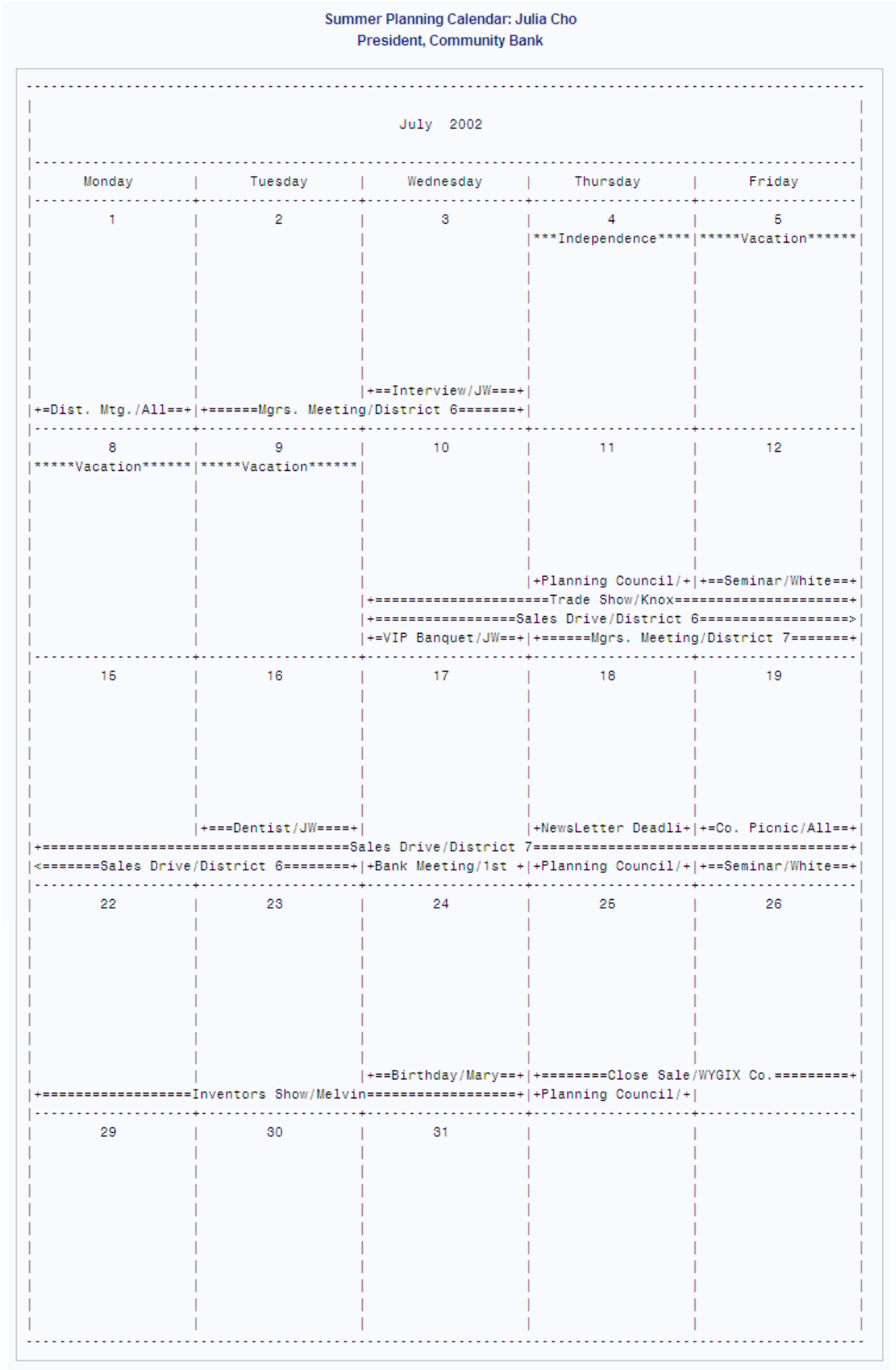
```
   holistart date;
   holivar holiday;
   holidur holilong;
```

タイトルを指定します。

```
   title1 'Summer Planning Calendar:  Julia Cho';
   title2 'President, Community Bank';
run;
```

出力:出力:HTML

アウトプット 8.4 *夏期計画カレンダ:Julia Cho*



Summer Planning Calendar: Julia Cho
President, Community Bank

July 2002

Monday	Tuesday	Wednesday	Thursday	Friday
1	2	3	4 ***Independence****	5 *****Vacation******
+=Dist. Mtg./All==+	+======Mgrs. Meeting/District 6=======+	+==Interview/JW===+		
8 *****Vacation******	9 *****Vacation******	10	11	12
			+Planning Council/+	+==Seminar/White==+
		+VIP Banquet/JW==+	+======Mgrs. Meeting/District 7=======+	+======Trade Show/Knox======+ <=======Sales Drive/District 6========+
15	16	17	18	19
	+===Dentist/JW====+		+NewsLetter Deadli+	+=Co. Picnic/All==+
<=======Sales Drive/District 6========+		+Bank Meeting/1st +	+Planning Council/+	+==Seminar/White==+
22	23	24	25	26
		+==Birthday/Mary==+	+========Close Sale/WYGIX Co.=========+	
+==========Inventors Show/Melvin==========+			+Planning Council/+	
29	30	31		

+=========Sales Drive/District 7=========+

例 2: 複数のカレンダを含むスケジュールカレンダ

要素: CALID statement
_CAL_変数
OUTPUT=COMBINE オプション
その他
DUR statement
(1 日 24 時間、週 7 日)のうち 1 つを使用します。

詳細

この例では、例 1 に基づいて、さらにアクティビティがビジネス用と個人用の 2 つのカレンダのうちどちらに属しているかを識別します。この例では、次を行います。

- スケジュールカレンダレポートを生成します。
- 2 つのカレンダを同じ出力ページに印刷します。
- アクティビティを休日を避けてスケジュールします。
- 2 つのデフォルト勤務パターン(1 日 24 時間、週 7 日)のうち 1 つを使用します。
- カレンダ名によってアクティビティと休日を識別します。

プログラム

```
data allacty2;
  input date:date7. happen $ 10-34 who $ 35-47 _CAL_ $ long;
  datalines;
01JUL02  Dist. Mtg.               All          CAL1   1
02JUL02  Mgrs. Meeting            District 6   CAL1   2
03JUL02  Interview                JW           CAL1   1
05JUL02  VIP Banquet              JW           CAL1   1
06JUL02  Beach trip               family       CAL2   2
08JUL02  Sales Drive              District 6   CAL1   5
08JUL02  Trade Show               Knox         CAL1   3
09JUL02  Orthodontist             Meagan       CAL2   1
11JUL02  Mgrs. Meeting            District 7   CAL1   2
11JUL02  Planning Council         Group II     CAL1   1
12JUL02  Seminar                  White        CAL1   1
14JUL02  Co. Picnic               All          CAL1   1
14JUL02  Business trip            Fred         CAL2   2
15JUL02  Sales Drive              District 7   CAL1   5
16JUL02  Dentist                  JW           CAL1   1
17JUL02  Bank Meeting             1st Natl     CAL1   1
17JUL02  Real estate agent        Family       CAL2   1
18JUL02  NewsLetter Deadline      All          CAL1   1
18JUL02  Planning Council         Group III    CAL1   1
19JUL02  Seminar                  White        CAL1   1
22JUL02  Inventors Show           Melvin       CAL1   3
24JUL02  Birthday                 Mary         CAL1   1
25JUL02  Planning Council         Group IV     CAL1   1
25JUL02  Close Sale               WYGIX Co.    CAL1   2
27JUL02  Ballgame                 Family       CAL2   1
;
```

```
data vac;
   input hdate:date7.  holiday $ 11-25 _CAL_ $ ;
   datalines;
29JUL02   vacation               CAL2
04JUL02   Independence           CAL1
;

proc sort data=allacty2;
   by date;
run;

options formchar="|----|+|---+=|-/\<>*";

proc calendar data=allacty2 holidata=vac;

   calid _CAL_ / output=combine;

   start date ;
   dur long;

   holistart hdate;
   holivar holiday;

   title1 'Summer Planning Calendar:  Julia Cho';
   title2 'President, Community Bank';
   title3 'Work and Home Schedule';
run;
```

プログラムの説明

ACTIVITIES データセットを作成し、個々のカレンダを識別します。 Allacty2 には、銀行頭取の個人アクティビティとビジネスアクティビティの両方が含まれています。_CAL_ 変数によって、イベントがどのカレンダに属しているかが識別されます。

```
data allacty2;
  input date:date7. happen $ 10-34 who $ 35-47 _CAL_ $ long;
  datalines;
01JUL02  Dist. Mtg.              All          CAL1   1
02JUL02  Mgrs. Meeting           District 6   CAL1   2
03JUL02  Interview               JW           CAL1   1
05JUL02  VIP Banquet             JW           CAL1   1
06JUL02  Beach trip              family       CAL2   2
08JUL02  Sales Drive             District 6   CAL1   5
08JUL02  Trade Show              Knox         CAL1   3
09JUL02  Orthodontist            Meagan       CAL2   1
11JUL02  Mgrs. Meeting           District 7   CAL1   2
11JUL02  Planning Council        Group II     CAL1   1
12JUL02  Seminar                 White        CAL1   1
14JUL02  Co. Picnic              All          CAL1   1
14JUL02  Business trip           Fred         CAL2   2
15JUL02  Sales Drive             District 7   CAL1   5
16JUL02  Dentist                 JW           CAL1   1
17JUL02  Bank Meeting            1st Natl     CAL1   1
17JUL02  Real estate agent       Family       CAL2   1
18JUL02  NewsLetter Deadline     All          CAL1   1
18JUL02  Planning Council        Group III    CAL1   1
19JUL02  Seminar                 White        CAL1   1
22JUL02  Inventors Show          Melvin       CAL1   3
24JUL02  Birthday                Mary         CAL1   1
```

```
25JUL02  Planning Council         Group IV     CAL1   1
25JUL02  Close Sale               WYGIX Co.    CAL1   2
27JUL02  Ballgame                 Family       CAL2   1
;
```

HOLIDAYS データセットを作成し、休日がどのカレンダに影響するかを識別します。 _CAL_ 変数によって、休日がどのカレンダに属しているかが識別されます。

```
data vac;
   input hdate:date7.  holiday $ 11-25 _CAL_ $ ;
   datalines;
29JUL02   vacation               CAL2
04JUL02   Independence           CAL1
;
```

開始日を含む変数を基準にして ACTIVITIES データセットを並べ替えます。 結合出力でカレンダを作成する場合は、CALID 変数は使用せずにアクティビティ開始日のみを基準にして並べ替えます。HOLIDAYS データセットは並べ替える必要はありません。

```
proc sort data=allacty2;
   by date;
run;
```

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

スケジュールカレンダを作成します。 DATA=は ACTIVITIES データセットを識別し、HOLIDATA=は HOLIDAYS データセットを識別します。デフォルトでは、出力カレンダは週 7 日の表示になります。

```
proc calendar data=allacty2 holidata=vac;
```

すべてのイベントと休日を単一カレンダに結合します。 CALID ステートメントは、イベントがどのカレンダに属するかを指定します。OUTPUT=COMBINE は、すべてのイベントと休日を同じカレンダに配置します。

```
   calid _CAL_ / output=combine;
```

アクティビティ開始日の変数とアクティビティ期間の変数を指定します。 START ステートメントはアクティビティの開始日を含む ACTIVITIES データセットの変数を指定し、DUR は各アクティビティの期間を含む変数を指定します。スケジュールカレンダの作成には、START と DUR が必須です。

```
   start date ;
   dur long;
```

休日情報を取得します。 HOLISTART ステートメントと HOLIVAR ステートメントは、各休日の開始日と名前のそれぞれを含む HOLIDAYS データセットの変数を指定します。HOLIDAYS データセットを使用する場合は、HOLISTART が必須になります。

```
   holistart hdate;
   holivar holiday;
```

タイトルを指定します。

```
   title1 'Summer Planning Calendar:  Julia Cho';
   title2 'President, Community Bank';
```

```
   title3 'Work and Home Schedule';
run;
```

出力:出力:HTML

アウトプット 8.5 *夏期計画カレンダ:勤務スケジュールと家庭スケジュール*

Summer Planning Calendar: Julia Cho
President, Community Bank
Work and Home Schedule

July 2002

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		1	2	3	4	5	6
CAL2							+Beach trip>
CAL1		+Dist. Mtg.+	+Mgrs. Meeting/District +	+Interview/+	Independence	+VIP Banque+	
	7	8	9	10	11	12	13
CAL2	<Beach trip+		+Orthodonti+				
CAL1		+==========Trade Show/Knox==========+			+Planning C+	+Seminar/Wh+	
		+====================Sales Drive/District 6====================+			+Mgrs. Meeting/District +		
	14	15	16	17	18	19	20
CAL2	+==Business trip/Fred===+			+Real estat+			
CAL1			+Dentist/JW+	+Bank Meeti+	+Planning C+	+Seminar/Wh+	
	+Co. Picnic+	+====================Sales Drive/District 7====================+			+NewsLetter+		
	21	22	23	24	25	26	27
CAL2							+Ballgame/F+
CAL1		+=======Inventors Show/Melvin========+		+Birthday/M+	+=Close Sale/WYGIX Co.==+		
					+Planning C+		
	28	29	30	31			
CAL2		**vacation**					

例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)

要素: PROC CALENDAR ステートメントオプション
- CALEDATA=
- DATETIME
- WORKDATA=

CALID statement
- _CAL_変数
- OUTPUT=SEPARATE オプション

その他のステートメント
- DUR statement
- OUTSTART statement
- OUTFIN statement

詳細

この例では、次を行います。

- 単一 PROC ステップで、各カレンダの個別出力ページを生成します。
- アクティビティを休日を避けてスケジュールします。
- 1 日 8 時間、週 5 1/2 日を表示します。
- カレンダごとに別々の勤務パターンと休日を使用します。

複数のカレンダに対する異なる出力の生成

この例と “例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)” (222 ページ) では、複数のカレンダに対して同じ入力データを使用して、異なる出力を生成します。これらのプログラムの相違点は、ACTIVITIES データセットの並べ替え方法と、OUTPUT=オプションの設定方法のみです。

表 8.12 *並べ替えと OUTPUT= 設定*

印刷オプション	並べ替え変数	OUTPUT=設定	例
各カレンダを別々のページに印刷	カレンダ ID と開始日	SEPARATE	3, 8
すべてのアクティビティを同じページに印刷し、各カレンダを識別	開始日	COMBINE	4, 2
すべてのアクティビティを同じページに印刷するが、各カレンダの識別はなし	開始日	MIX	4

プログラム

```
libname well 'SAS-library';
```

```
data well.act;
   input task & $16. dur : 5. date : datetime16.  _cal_ $ cost;
   datalines;
Drill Well          3.50  01JUL02:12:00:00  CAL1   1000
Lay Power Line      3.00  04JUL02:12:00:00  CAL1   2000
Assemble Tank       4.00  05JUL02:08:00:00  CAL1   1000
Build Pump House    3.00  08JUL02:12:00:00  CAL1   2000
Pour Foundation     4.00  11JUL02:08:00:00  CAL1   1500
Install Pump        4.00  15JUL02:14:00:00  CAL1    500
Install Pipe        2.00  19JUL02:08:00:00  CAL1   1000
Erect Tower         6.00  20JUL02:08:00:00  CAL1   2500
Deliver Material    2.00  01JUL02:12:00:00  CAL2    500
Excavate            4.75  03JUL02:08:00:00  CAL2   3500
;

data well.hol;
   input date date. holiday $ 11-25 _cal_ $;
   datalines;
09JUL02   Vacation            CAL2
04JUL02   Independence        CAL1
;

data well.cal;
   input _sun_ $ _sat_ $ _mon_ $ _tue_ $ _wed_ $ _thu_ $
         _fri_ $ _cal_ $;
   datalines;
Holiday Holiday  Workday Workday Workday Workday Workday CAL1
Holiday Halfday  Workday Workday Workday Workday Workday CAL2
;

data well.wor;
   input halfday time5.;
   datalines;
08:00
12:00
;

proc sort data=well.act;
   by _cal_ date;
run;

options formchar="|----|+|---+=|-/\<>*";

proc calendar data=well.act
              holidata=well.hol
              caledata=well.cal
              workdata=well.wor
              datetime;

   calid _cal_ / output=separate;

   start date;
   dur dur;

   holistart date;
   holivar holiday;

   outstart Monday;
   outfin Saturday;

   title1 'Well Drilling Work Schedule: Separate Calendars';
```

```
   format cost dollar9.2;
run;
```

プログラムの説明

アクティビティデータセットを永久保存できるようにライブラリを指定します。

```
libname well 'SAS-library';
```

ACTIVITIES データセットを作成し、個々のカレンダを識別します。 Well.Act は、井戸建設プロジェクトのアクティビティを含む永久 SAS データセットです。_CAL_変数によって、アクティビティの属しているカレンダが識別されます。

```
data well.act;
   input task & $16. dur : 5. date : datetime16.  _cal_ $ cost;
   datalines;
Drill Well          3.50  01JUL02:12:00:00  CAL1   1000
Lay Power Line      3.00  04JUL02:12:00:00  CAL1   2000
Assemble Tank       4.00  05JUL02:08:00:00  CAL1   1000
Build Pump House    3.00  08JUL02:12:00:00  CAL1   2000
Pour Foundation     4.00  11JUL02:08:00:00  CAL1   1500
Install Pump        4.00  15JUL02:14:00:00  CAL1    500
Install Pipe        2.00  19JUL02:08:00:00  CAL1   1000
Erect Tower         6.00  20JUL02:08:00:00  CAL1   2500
Deliver Material    2.00  01JUL02:12:00:00  CAL2    500
Excavate            4.75  03JUL02:08:00:00  CAL2   3500
;
```

HOLIDAYS データセットを作成します。 _CAL_変数によって、休日の属しているカレンダが識別されます。

```
data well.hol;
   input date date. holiday $ 11-25 _cal_ $;
   datalines;
09JUL02   Vacation            CAL2
04JUL02   Independence        CAL1
;
```

CALENDAR データセットを作成します。 各オブザベーションによって、丸一週間の勤務シフトが定義されます。_CAL_変数によって、勤務シフトがどのカレンダに適用されるかが識別されます。CAL1 では、月曜日から金曜日までデフォルトの 8 時間勤務シフトが使用されます。CAL2 では、土曜日が半日で、月曜日から金曜日まではデフォルトの 8 時間勤務シフトが使用されます。

```
data well.cal;
   input _sun_ $ _sat_ $ _mon_ $ _tue_ $ _wed_ $ _thu_ $
         _fri_ $ _cal_ $;
   datalines;
Holiday Holiday  Workday Workday Workday Workday Workday CAL1
Holiday Halfday  Workday Workday Workday Workday Workday CAL2
;
```

WORKDAYS データセットを作成します。 このデータセットでは、CALENDAR データセットで指定した日次勤務シフトが定義されます。(オブザベーションではなく)各変数に、業務期間と業務外期間が交互に並ぶ日次スケジュールが 1 つずつ含まれます。HALFDAY 勤務シフトは 4 時間続きます。

```
data well.wor;
   input halfday time5.;
   datalines;
08:00
12:00
;
```

カレンダ ID と開始日のそれぞれを含む変数を基準にして ACTIVITIES データセットを並べ替えます。HOLIDAYS データセットは並べ替える必要はありません。

```
proc sort data=well.act;
   by _cal_ date;
run;
```

FORMCHAR オプションを設定します。FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

スケジュールカレンダを作成します。DATA=は ACTIVITIES データセット、HOLIDATA=は HOLIDAYS データセット、CALEDATA=は CALENDAR データセット、WORKDATA=は WORKDAYS データセットを識別します。DATETIME は、START ステートメントで指定した変数に SAS 日時出力形式の値が含まれるように指定します。

```
proc calendar data=well.act
              holidata=well.hol
              caledata=well.cal
              workdata=well.wor
              datetime;
```

各カレンダを別々のページに印刷します。CALID ステートメントは、_CAL_変数によってカレンダが識別されるように指定します。OUTPUT=SEPARATE は、各カレンダの情報を別々のページに印刷します。

```
   calid _cal_ / output=separate;
```

アクティビティ開始日の変数とアクティビティ期間の変数を指定します。START ステートメントはアクティビティ開始日を含む ACTIVITIES データセットの変数を指定し、DUR はアクティビティ期間を含む変数を指定します。スケジュールカレンダには START と DUR が必須です。

```
   start date;
   dur dur;
```

休日情報を取得します。HOLISTART と HOLIVAR は、各休日の開始日と名前のそれぞれを含む HOLIDAYS データセットの変数を指定します。HOLIDAYS データセットを使用する場合は、HOLISTART が必須になります。

```
   holistart date;
   holivar holiday;
```

カレンダ表示をカスタマイズします。OUTSTART と OUTFIN は、カレンダに月曜日から土曜日まで週 6 日を表示するように指定します。

```
   outstart Monday;
   outfin Saturday;
```

タイトルを指定して、COST 変数をフォーマットします。

```
   title1 'Well Drilling Work Schedule: Separate Calendars';
   format cost dollar9.2;
run;
```

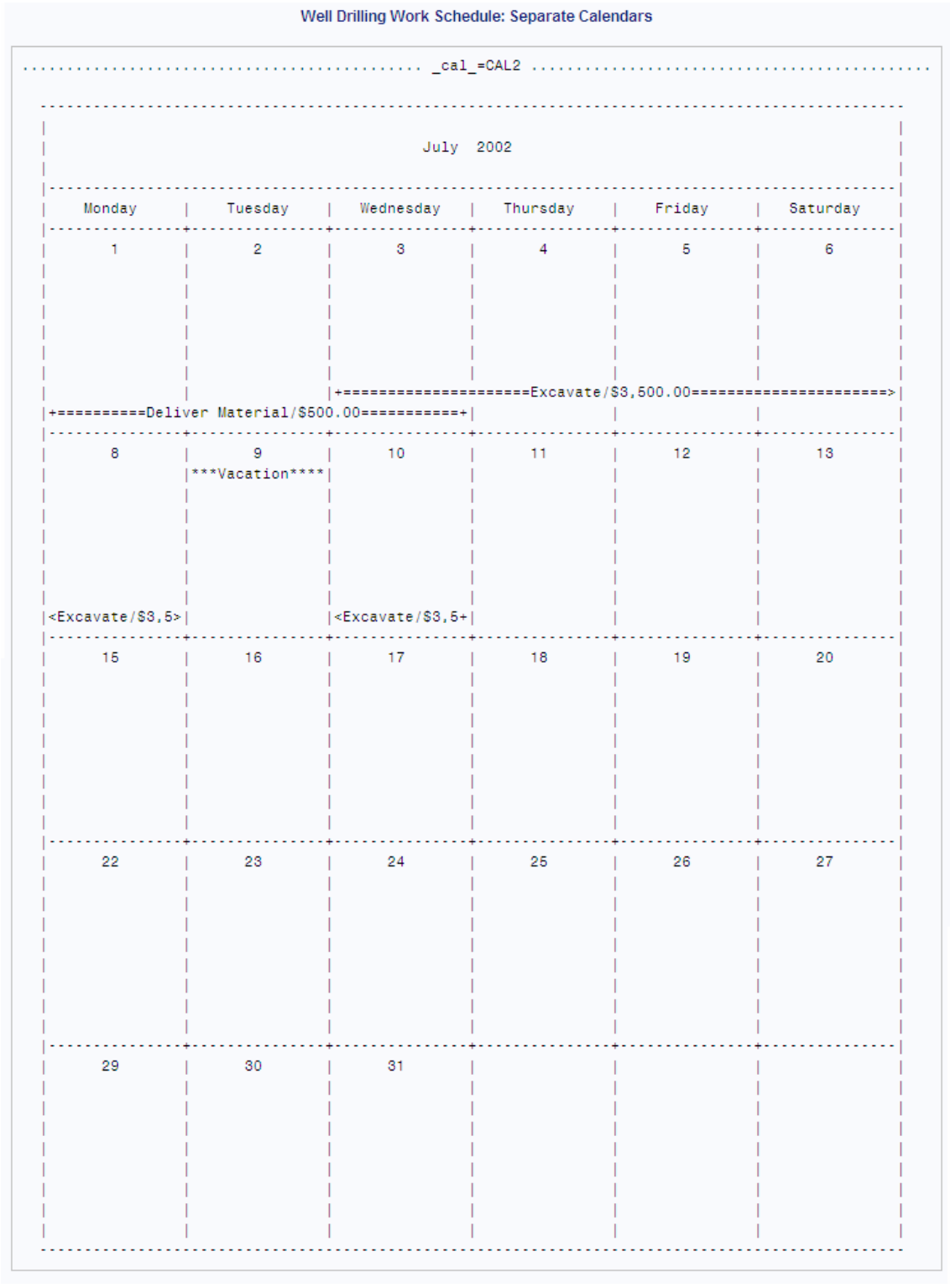
出力:出力:HTML

アウトプット 8.6 *井戸掘り作業スケジュール(その 1)*

Well Drilling Work Schedule: Separate Calendars

```
..................................... _cal_=CAL1 .....................................

 ---------------------------------------------------------------------------------------
|                                                                                       |
|                                      July  2002                                       |
|                                                                                       |
|---------------------------------------------------------------------------------------|
|    Monday     |    Tuesday    |   Wednesday   |   Thursday    |    Friday     |   Saturday    |
|---------------+---------------+---------------+---------------+---------------+---------------|
|       1       |       2       |       3       |       4       |       5       |       6       |
|               |               |               |*Independence**|               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |+Assemble Tank>|               |
|               |               |               |               |+Lay Power Lin>|               |
|+============Drill Well/$1,000.00=============>|               |<Drill Well/$1+|               |
|---------------+---------------+---------------+---------------+---------------+---------------|
|       8       |       9       |      10       |      11       |      12       |      13       |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|+=================Build Pump House/$2,000.00==================+|               |               |
|<====================Assemble Tank/$1,000.00====================+|               |               |
|<==Lay Power Line/$2,000.00===+|               |+==Pour Foundation/$1,500.00==>|               |
|---------------+---------------+---------------+---------------+---------------+---------------|
|      15       |      16       |      17       |      18       |      19       |      20       |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|+============================Install Pump/$500.00============================+|               |
|<==========Pour Foundation/$1,500.00==========+|               |+Install Pipe/>|               |
|---------------+---------------+---------------+---------------+---------------+---------------|
|      22       |      23       |      24       |      25       |      26       |      27       |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|+============================Erect Tower/$2,500.00============================>|               |
|<===Install Pipe/$1,000.00====+|               |               |               |               |
|---------------+---------------+---------------+---------------+---------------+---------------|
|      29       |      30       |      31       |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|               |               |               |               |               |               |
|<Erect Tower/$+|               |               |               |               |               |
 ---------------------------------------------------------------------------------------
```

アウトプット 8.7 *井戸掘り作業スケジュール(その 2)*



例 4: 例外的な勤務シフトの複数のスケジュールカレンダ(結合または混合出力)

要素: PROC CALENDAR ステートメントオプション
CALEDATA=

DATETIME
WORKDATA=

CALID statement
_CAL_変数
OUTPUT=COMBINE オプション
OUTPUT=MIXED オプション

その他のステートメント
DUR statement
OUTSTART statement
OUTFIN statement

データセット: Well.Aact
Well.Hol
Well.Cal
Well.Wor

詳細

この例では、次を行います。

- スケジュールカレンダを生成します。
- アクティビティを休日を避けてスケジュールします。
- カレンダごとに別々の勤務パターンと休日を使用します。
- 1 日 8 時間、週 5 1/2 日勤務を使用します。
- 各カレンダページに複数のカレンダを表示して識別します(結合出力)。
- 各カレンダページに複数のカレンダを表示しますが*識別はしません*(混合出力)。

この例では、結合出力と混合出力の両方を作成します。PROC CALENDAR ステップを 1 か所変更するだけで、結合カレンダ出力または混合カレンダ出力を生成できます。変更点は CALID ステートメントの OUTPUT=オプションの設定です。まず結合出力が生成され、次に混合出力が生成されます。

この例と “例 3: 例外的な勤務シフトの複数のスケジュールカレンダ(個別出力)” (216 ページ) では、複数のカレンダに対して同じ入力データを使用して、異なる出力を生成します。これらのプログラムの相違点は、ACTIVITIES データセットの並べ替え方法と、OUTPUT=オプションの設定方法のみです。

表 8.13 *並べ替えと OUTPUT=設定*

印刷オプション	並べ替え変数	OUTPUT=設定	例
各カレンダを別々のページに印刷	カレンダ ID と開始日	SEPARATE	3, 8
すべてのアクティビティを同じページに印刷し、各カレンダを識別	開始日	COMBINE	4, 2
すべてのアクティビティを同じページに印刷するが、各カレンダの識別はなし	開始日	MIX	4

結合文字のプログラム

```
libname well
'SAS-library';

proc sort data=well.act;
   by date;
run;

options formchar="|----|+|---+=|-/\<>*";

proc calendar data=well.act
              holidata=well.hol
              caledata=well.cal
              workdata=well.wor
              datetime;

   calid _cal_ / output=combine;

   start date;
   dur dur;

   holistart date;
   holivar holiday;

   title1 'Well Drilling Work Schedule: Combined Calendars';
   format cost dollar9.2;
run;
```

プログラムの説明

ACTIVITIES データセットが格納されている SAS ライブラリを指定します。

```
libname well
'SAS-library';
```

開始日を含む変数を基準にして ACTIVITIES データセットを並べ替えます。 結合カレンダ出力の生成時には、CALID 変数による並べ替えは行いません。

```
proc sort data=well.act;
   by date;
run;
```

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

スケジュールカレンダを作成します。 DATA=は ACTIVITIES データセット、HOLIDATA=は HOLIDAYS データセット、CALEDATA=は CALENDAR データセット、WORKDATA=は WORKDAYS データセットを識別します。DATETIME は、START ステートメントで指定した変数に SAS 日時出力形式の値が含まれるように指定します。

```
proc calendar data=well.act
              holidata=well.hol
              caledata=well.cal
              workdata=well.wor
              datetime;
```

すべてのイベントと休日を単一カレンダに結合します。 CALID ステートメントは、_CAL_ 変数によってカレンダが識別されるように指定します。OUTPUT=COMBINE は複数のカレンダを同じページに印刷して、各カレンダを識別します。

```
calid _cal_ / output=combine;
```

アクティビティ開始日の変数とアクティビティ期間の変数を指定します。 START ステートメントはアクティビティの開始日を含む ACTIVITIES データセットの変数を指定し、DUR は各アクティビティの期間を含む変数を指定します。スケジュールカレンダには START と DUR が必須です。

```
start date;
dur dur;
```

休日情報を取得します。 HOLISTART と HOLIVAR は、各休日の開始日と名前のそれぞれを含む HOLIDAYS データセットの変数を指定します。HOLIDAYS データセットを使用する場合は、HOLISTART が必須になります。

```
holistart date;
holivar holiday;
```

タイトルを指定して、COST 変数をフォーマットします。

```
title1 'Well Drilling Work Schedule: Combined Calendars';
format cost dollar9.2;
run;
```

出力:出力:HTML

アウトプット 8.8 *井戸掘り作業スケジュール:混合カレンダ*

Well Drilling Work Schedule: Combined Calendars

July 2002

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		1	2	3	4	5	6
CAL1					Independence	+Assemble T>	
						+Lay Power >	
		+========Drill Well/$1,000.00========>				<Drill Well+	
CAL2				+================Excavate/$3,500.00================>			
		+======Deliver Material/$500.00======+					
	7	8	9	10	11	12	13
CAL1		+===========Build Pump House/$2,000.00============+					
		<=============Assemble Tank/$1,000.00=============+					
		<Lay Power Line/$2,000.0+			+Pour Foundation/$1,500.>		
CAL2		<Excavate/$>	**Vacation**	<Excavate/$+			
	14	15	16	17	18	19	20
CAL1		+====================Install Pump/$500.00====================+					
		<=====Pour Foundation/$1,500.00======+				+Install Pi>	
	21	22	23	24	25	26	27
CAL1		+====================Erect Tower/$2,500.00====================>					
		<Install Pipe/$1,000.00=+					
	28	29	30	31			
CAL1		<Erect Towe+					

混合カレンダのプログラム

結合出力の代わりに混合出力を生成するには、同じプログラムを使用して、OUTPUT=オプションの設定を OUTPUT=MIX に変更します。

```
proc calendar data=well.act
              holidata=well.hol
              caledata=well.cal
              workdata=well.wor
              datetime;
   calid _cal_ / output=mix;
   start date;
   dur dur;
   holistart date;
   holivar holiday;
   outstart Monday;
   outfin Saturday;
   title1 'Well Drilling Work Schedule: Mixed Calendars';
   format cost dollar9.2;
run;
```

出力:出力:HTML

アウトプット 8.9 *井戸掘り作業スケジュール:混合カレンダ*

Well Drilling Work Schedule: Mixed Calendars

July 2002

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2	3	4	5	6
				+Assemble Tank>	
		+=========Excavate/$3,500.00=========>			
+==========Deliver Material/$500.00==========+			*Independence**	+Lay Power Lin>	
+============Drill Well/$1,000.00============>			*Independence**	<Drill Well/$1+	
8	9	10	11	12	13
+=================Build Pump House/$2,000.00=================+					
<==================Assemble Tank/$1,000.00==================+					
<==Lay Power Line/$2,000.00===+					
<Excavate/$3,5>	***Vacation****	<Excavate/$3,5+	+==Pour Foundation/$1,500.00==>		
15	16	17	18	19	20
+============================Install Pump/$500.00============================+					
<==========Pour Foundation/$1,500.00==========+				+Install Pipe/>	
22	23	24	25	26	27
+============================Erect Tower/$2,500.00============================>					
<===Install Pipe/$1,000.00====+					
29	30	31			
<Erect Tower/$+					

例 5: ブランク表示または休日表示付きのスケジュールカレンダ

要素: PROC CALENDAR ステートメントオプション

FILL
HOLIDATA=
INTERVAL=WORKDAY

その他のステートメント
DUR statement
HOLIDUR statement
HOLISTART statement
HOLIVAR statement

詳細

この例では、休日のみを表示するスケジュールカレンダを生成します。これと同じコードを使用して、PROC CALENDAR ステップから HOLIDATA=オプションならびに HOLISTART、HOLIVAR および HOLIDUR ステートメントを削除すると、ブランクカレンダセットを生成できます。

プログラム

```
data acts;
   input sta : date7. act $ 11-30 dur;
   datalines;
01JAN03   Start                0
31DEC03   Finish               0
;

data holidays;
   input sta : date7. act $ 11-30 dur;
   datalines;
01JAN03   New Year's           1
30MAR03   Good Friday          1
28MAY03   Memorial Day         1
04JUL03   Independence Day     1
03SEP03   Labor Day            1
22NOV03   Thanksgiving         2
25DEC03   Christmas Break      5
;

options formchar="|----|+|---+=|-/\<>*";

proc calendar data=acts holidata=holidays fill interval=workday;

   start sta;
   dur dur;

   holistart sta;
   holivar act;
   holidur dur;

   title1 'Calendar of Holidays Only';
run;
```

プログラムの説明

ACTIVITIES データセットを作成します。 最初の月と最後の月に 1 つずつアクティビティを指定して、各アクティビティの期間を 0 にします。PROC CALENDAR は、期間がゼロのアクティビティは出力に印刷しません。

```
data acts;
   input sta : date7. act $ 11-30 dur;
   datalines;
01JAN03   Start                0
31DEC03   Finish               0
;
```

HOLIDAYS データセットを作成します。

```
data holidays;
   input sta : date7. act $ 11-30 dur;
   datalines;
01JAN03   New Year's           1
30MAR03   Good Friday          1
28MAY03   Memorial Day         1
04JUL03   Independence Day     1
03SEP03   Labor Day            1
22NOV03   Thanksgiving         2
25DEC03   Christmas Break      5
;
```

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

カレンダを作成します。 DATA=は ACTIVITIES データセットを識別し、HOLIDATA=は HOLIDAYS データセットを識別します。FILL は、アクティビティがなくても、すべての月を表示します。デフォルトでは、アクティビティのある月のみレポートに表示されます。INTERVAL=WORKDAY は、アクティビティと休日を 1 日 8 時間単位で測定し、PROC CALENDAR がアクティビティを月曜日から金曜日までの間にのみスケジュールするように指定します。

```
proc calendar data=acts holidata=holidays fill interval=workday;
```

アクティビティ開始日の変数とアクティビティ期間の変数を指定します。 START ステートメントはアクティビティの開始日を含む ACTIVITIES データセットの変数を指定し、DUR は各アクティビティの期間を含む変数を指定します。スケジュールカレンダの作成には、START と DUR が必須です。

```
   start sta;
   dur dur;
```

休日情報を取得します。 HOLISTART、HOLIVAR および HOLIDUR ステートメントは、各休日の開始日、名前および期間のそれぞれを含む HOLIDAYS データセットの変数を指定します。HOLIDAYS データセットを使用する場合は、HOLISTART が必須になります。少なくとも 1 つの休日が 2 日以上続くため、HOLIDUR(または HOLIFIN)が必須になります。

```
   holistart sta;
   holivar act;
   holidur dur;
```

タイトルを指定します。

```
   title1 'Calendar of Holidays Only';
run;
```

出力:出力:HTML

次の出力は、出力の 12 月部分を示しています。INTERVAL=WORKDAY を指定しなければ、5 日間のクリスマス休暇が週末にもスケジュールされます。

アウトプット 8.10 *休日のみのカレンダ(12 月部分)*

Calendar of Holidays Only

December 2003

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25 Christmas Br	26 Christmas Br	27
28	29 Christmas Br	30 Christmas Br	31 Christmas Br			

例 6: 前のタスクの完了に基づき、スケジュールを計算する

要素: PROC CALENDAR プロシジャ機能
- PROC CALENDAR ステートメント
- CALID statement
- FIN statement
- VAR ステートメント

他の要素: PROC CPM ステップ

PROC SORT ステップ

詳細

プログラム説明

この例では、次を行います。

- 複数のカレンダを含むプロジェクトスケジュールを計算します(PROC CPM)。
- PROC CPM 出力データセットのリストを生成します(PROC PRINT)。
- スケジュールをカレンダ出力形式で表示します(PROC CALENDAR)。

この例では主に、次の基準を満たすスケジュールを計算する PROC CPM の機能を取り上げます。

- 初期開始日に基づいていること。
- 各従業員のスケジュールに個人の休暇を適用するなど、異なるカレンダに異なる業務外期間を適用すること。
- マイルストーン(期間が 0 のアクティビティ)を含めること。

SAS/OR ソフトウェアによるスケジュールタスクの自動化

PROC CALENDAR を使用してスケジュールカレンダを生成した場合は、スケジュールに変更が発生すると、アクティビティ開始日を手動で調整する必要があります。その代わりに、SAS/OR ソフトウェアで PROC CPM を使用すると、日付変更時に再スケジュールができます。さらに重要なことに、プロジェクトの初期開始日を提供するだけで、PROC CPM が、識別された後続タスク、すなわち先行タスクが終了するまで開始できないタスクに基づいて、アクティビティの開始日を計算します。

PROC CPM を使用するには、次の操作を実行する必要があります。

1. アクティビティと期間を含む ACTIVITIES データセットを作成します。(休日、カレンダおよび勤務シフトデータセットを使用して、勤務外日、週次勤務スケジュールおよび勤務シフトを指定できます)。
2. どのアクティビティが他のアクティビティの後続になるかを示します(先行関係)。
3. スケジュールで考慮する場合は、リソース制限を定義します。
4. 初期開始日を提供します。

PROC CPM は、データを処理して、各アクティビティの開始日と終了日を含むデータセットを生成できます。PROC CPM は、期間情報、週次勤務パターン、勤務シフト、ならびにスケジュールに割り込む休日および勤務外日に基づいて、アクティビティをスケジュールします。PROC CPM によって計算されたスケジュールについて、開始日と終了日の単純なリストから、カレンダ、ガントチャート、ネットワーク図に至るまで、複数のビューを生成できます。

参照

この例では、PROC CALENDAR のユーザーに、より高度な SAS スケジュールツールを紹介します。プロジェクト管理のタスクとツールといくつかの例については、*Project Management Using the SAS System* を参照してください。その他の例については、*SAS/OR Software:Project Management Examples* を参照してください。詳細な参照ドキュメントについては、*SAS/OR(R) 9.3 User's Guide:Mathematical Programming* を参照してください。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

data grant;
   input jobnum Task $ 4-22 Days Succ1 $ 27-45 aldate : date7. altype $
         _cal_ $;
   format aldate date7.;
   datalines;
1  Run Exp 1          11  Analyze Exp 1       .       .   Student
2  Analyze Exp 1       5  Send Report 1       .       .   Prof.
3  Send Report 1       0  Run Exp 2           .       .   Prof.
4  Run Exp 2          11  Analyze Exp 2       .       .   Student
5  Analyze Exp 2       4  Send Report 2       .       .   Prof.
6  Send Report 2       0  Write Final Report  .       .   Prof.
7  Write Final Report  4  Send Final Report   .       .   Prof.
8  Send Final Report   0                      .       .   Student
9  Site Visit          1                    18jul07 ms  Prof.
;

data nowork;
   format holista date7. holifin date7.;
   input holista : date7. holifin : date7. name $ 17-32 _cal_ $;
   datalines;
04jul07 04jul07 Independence Day Prof.
03sep07 03sep07 Labor Day        Prof.
04jul07 04jul07 Independence Day Student
03sep07 03sep07 Labor Day        Student
16jul07 17jul07 PROF Vacation    Prof.
16aug07 17aug07 STUDENT Vacation Student
;

proc cpm data=grant
         date='01jul07'd
         interval=weekday
         out=gcpm1
         holidata=nowork;
   activity task;
   successor succ1;
   duration days;
   calid _cal_;
   id task;
   aligndate aldate;
   aligntype altype;
   holiday holista / holifin=holifin;
run;

proc print data=gcpm1;
   title 'Data Set GCPM1, Created with PROC CPM';
run;

proc sort data=gcpm1;
   by e_start;
run;

proc calendar data=gcpm1
              holidata=nowork
              interval=workday;
   start e_start;
```

```
   fin   e_finish;
   calid _cal_ / output=combine;
   holistart holista;
   holifin   holifin;
   holivar name;
   var task;
   title 'Schedule for Experiment X-15';
   title2 'Professor and Student Schedule';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

ACTIVITIES データセットを作成し、個々のカレンダを識別します。 このデータによって、教師用(_CAL_の値は *Prof.*)と学生用(_CAL_の値は *Student*)の 2 つのカレンダが識別されます。Succ1 変数によって、現在のアクティビティが終了するまで開始できないアクティビティが識別されます。たとえば、*Analyze Exp 1* は *Run Exp 1* が完成しなければ開始できません。JOBNUM*3*、*6* および *8* の DAYS 値 *0* は、これらのジョブがマイルストーンであることを示しています。

```
data grant;
   input jobnum Task $ 4-22 Days Succ1 $ 27-45 aldate : date7. altype $
         _cal_ $;
   format aldate date7.;
   datalines;
1  Run Exp 1          11  Analyze Exp 1       .       .   Student
2  Analyze Exp 1       5  Send Report 1       .       .   Prof.
3  Send Report 1       0  Run Exp 2           .       .   Prof.
4  Run Exp 2          11  Analyze Exp 2       .       .   Student
5  Analyze Exp 2       4  Send Report 2       .       .   Prof.
6  Send Report 2       0  Write Final Report  .       .   Prof.
7  Write Final Report  4  Send Final Report   .       .   Prof.
8  Send Final Report   0                      .       .   Student
9  Site Visit          1                    18jul07 ms  Prof.
;
```

HOLIDAYS データセットを作成し、勤務外日がどのカレンダに属しているかを識別します。 2 つの休日が、教授のカレンダと学生のカレンダで 1 回ずつ、あわせて 2 回リストされています。各人に別々のカレンダが関連付けられているので、PROC CPM は、個人の休暇を適切なカレンダに適用できます。

```
data nowork;
   format holista date7. holifin date7.;
   input holista : date7. holifin : date7. name $ 17-32 _cal_ $;
   datalines;
04jul07 04jul07 Independence Day Prof.
03sep07 03sep07 Labor Day        Prof.
04jul07 04jul07 Independence Day Student
03sep07 03sep07 Labor Day        Student
16jul07 17jul07 PROF Vacation    Prof.
16aug07 17aug07 STUDENT Vacation Student
```

```
;
```

PROC CPM でスケジュールを計算します。 PROC CPM は、ACTIVITIES データセットと HOLIDAYS データセットで提供された情報を使用して、各アクティビティの開始日と終了日を計算します。DATE=オプションは、プロジェクトの開始日を提供します。この例では、2 つのカレンダが含まれていても、CALID ステートメントは必要ありません。カレンダ ID 変数に_CAL_という特別な名前が付いているからです。

```
proc cpm data=grant
         date='01jul07'd
         interval=weekday
         out=gcpm1
         holidata=nowork;
   activity task;
   successor succ1;
   duration days;
   calid _cal_;
   id task;
   aligndate aldate;
   aligntype altype;
   holiday holista / holifin=holifin;
run;
```

PROC CPM で作成された出力データセットを印刷します。 このステップは必須ではありません。PROC PRINT は、PROC CPM による計算結果を表示するには便利な方法です。

```
proc print data=gcpm1;
   title 'Data Set GCPM1, Created with PROC CPM';
run;
```

GCPM1 を PROC CALENDAR で使用する前に、アクティビティ開始日を含む変数を基準にして並べ替えます。

```
proc sort data=gcpm1;
   by e_start;
run;
```

スケジュールカレンダを作成します。 GCPM1 アクティビティデータセットです。PROC CALENDAR は、PROC CPM で計算された S_START 日付と S_FINISH 日付を使用して、スケジュールを印刷します。VAR ステートメントは、変数 TASK のみをカレンダ出力の表示対象として選択しています。

```
proc calendar data=gcpm1
              holidata=nowork
              interval=workday;
   start e_start;
   fin   e_finish;
   calid _cal_ / output=combine;
   holistart holista;
   holifin   holifin;
   holivar name;
   var task;
   title 'Schedule for Experiment X-15';
   title2 'Professor and Student Schedule';
run;
```

出力:出力:HTML

PROC PRINT は、GCPM1 のオブザベーションを表示して、PROC CPM で作成されたスケジュール計算を示しています。

アウトプット 8.11 *データセット GCPM1(PROC CPM で作成)*

Data Set GCPM1, Created with PROC CPM

Obs	Task	Succ1	Days	_cal_	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Run Exp 1	Analyze Exp 1	11	Student	02JUL07	17JUL07	02JUL07	17JUL07	0	0
2	Analyze Exp 1	Send Report 1	5	Prof.	18JUL07	24JUL07	18JUL07	24JUL07	0	0
3	Send Report 1	Run Exp 2	0	Prof.	25JUL07	25JUL07	25JUL07	25JUL07	0	0
4	Run Exp 2	Analyze Exp 2	11	Student	25JUL07	08AUG07	25JUL07	08AUG07	0	0
5	Analyze Exp 2	Send Report 2	4	Prof.	09AUG07	14AUG07	09AUG07	14AUG07	0	0
6	Send Report 2	Write Final Report	0	Prof.	15AUG07	15AUG07	15AUG07	15AUG07	0	0
7	Write Final Report	Send Final Report	4	Prof.	15AUG07	20AUG07	15AUG07	20AUG07	0	0
8	Send Final Report		0	Student	21AUG07	21AUG07	21AUG07	21AUG07	0	0
9	Site Visit		1	Prof.	19JUL07	19JUL07	19JUL07	19JUL07	0	0

PROC CALENDAR は、PROC CPM で計算された S_START 日付と S_FINISH 日付を使用して、次のスケジュールカレンダを作成しました。7 月 25 日と 8 月 15 日のアクティビティはマイルストーンなので、これによって後続アクティビティの開始が遅延することはありません。7 月 19 日の Site Visit と同じ日に、Analyze Exp 1 が発生していることに注意してください。リソースの過剰割り当てを防ぐには、SAS/OR ソフトウェアで使用可能な*リソース制約スケジュール*を使用します。

アウトプット 8.12 実験 X-15 のスケジュール(7 月)

Schedule for Experiment X-15
Professor and Student Schedule

July 2007

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	1	2	3	4	5	6	7
PROF.				Independence			
STUDENT		+=======Run Exp 1=======>		Independence	<=======Run Exp 1=======>		
	8	9	10	11	12	13	14
STUDENT		<============================Run Exp 1============================>					
	15	16	17	18	19	20	21
PROF.		PROF Vacatio	PROF Vacatio		+Site Visit+		
				+===========Analyze Exp 1============>			
STUDENT		<=======Run Exp 1=======+					
	22	23	24	25	26	27	28
PROF.		<=====Analyze Exp 1=====+		+Send Repor+			
STUDENT				+==============Run Exp 2==============>			
	29	30	31				
STUDENT		<=======Run Exp 2=======>					

アウトプット 8.13 *実験 X-15 のスケジュール(8 月)*

Schedule for Experiment X-15
Professor and Student Schedule

August 2007

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1	2	3	4
STUDENT				<==========Run Exp 2==========			
	5	6	7	8	9	10	11
PROF.					+=====Analyze Exp 2=====>		
STUDENT		<==========Run Exp 2==========+					
	12	13	14	15	16	17	18
PROF.				+=========Write Final Report=========>			
		<=====Analyze Exp 2=====+		+Send Repor+			
STUDENT					STUDENT Vaca	STUDENT Vaca	
	19	20	21	22	23	24	25
PROF.		<Write Fina+					
STUDENT			+Send Final+				
	26	27	28	29	30	31	

例 7: オブザベーション別の MEAN 値を含むサマリーカレンダ

要素: CALID statement
- _CAL_変数
- OUTPUT=SEPARATE オプション

その他のステートメント
- FORMAT statement
- LABEL statement
- MEAN statement
- SUM statement

他の要素: PROC FORMAT
- PICTURE statement

詳細

この例では、次を行います。

- サマリーカレンダを生成します。
- 休日を表示します。
- 3 つの変数について営業日(オブザベーション)別に合計値と平均値を生成します。
- 凡例を印刷して、変数ラベルを使用します。
- ピクチャ形式を使用して値を表示します。

*カレンダ月の日数*に基づいて MEAN 値を生成するには、MEANTYPE=NDAYS を使用します。デフォルトでは、MEANTYPE=NOBS が使用され、*データの存在する日数*に従って MEAN 値が計算されます。

プログラム

```
data meals;
   input date : date7. Brkfst Lunch Dinner;
   datalines;
01Dec08       123 234 238
02Dec08       188 188 198
03Dec08       123 183 176
04Dec08       200 267 243
05Dec08       176 165 177
08Dec08       178 198 187
09Dec08       165 176 187
10Dec08       187 176 231
11Dec08       176 187 222
12Dec08       187 187 123
15Dec08       176 165 177
16Dec08       156   . 167
17Dec08       198 143 167
18Dec08       178 198 187
19Dec08       165 176 187
22Dec08       187 187 123
;

data closed;
```

```
   input date date. holiday $ 11-25;
   datalines;
26DEC08   Repairs
29DEC08   Repairs
30DEC08   Repairs
31DEC08   Repairs
23DEC08   Vacation
24DEC08   Christmas Eve
25DEC08   Christmas
;

proc sort data=meals;
   by date;
run;

proc format;
   picture bfmt other = '000 Brkfst';
   picture lfmt other = '000 Lunch ';
   picture dfmt other = '000 Dinner';
run;

options formchar="|----|+|---+=|-/\<>*";

proc calendar data=meals holidata=closed;
   start date;

   holistart date;
   holiname holiday;

   sum brkfst lunch dinner / format=4.0;
   mean brkfst lunch dinner / format=6.2;
   label brkfst = 'Breakfasts Served'
         lunch  = '   Lunches Served'
         dinner = '   Dinners Served';
   format brkfst bfmt.
          lunch lfmt.
          dinner dfmt.;

   title 'Meals Served in Company Cafeteria';
   title2 'Mean Number by Business Day';
run;

title;
```

プログラムの説明

Activities データセットを作成します。 MEALS は、食堂営業日の朝食、昼食および夕食の提供数を記録します。

```
data meals;
   input date : date7. Brkfst Lunch Dinner;
   datalines;
01Dec08       123 234 238
02Dec08       188 188 198
03Dec08       123 183 176
04Dec08       200 267 243
05Dec08       176 165 177
08Dec08       178 198 187
09Dec08       165 176 187
```

```
10Dec08       187 176 231
11Dec08       176 187 222
12Dec08       187 187 123
15Dec08       176 165 177
16Dec08       156   . 167
17Dec08       198 143 167
18Dec08       178 198 187
19Dec08       165 176 187
22Dec08       187 187 123
;
```

Holidays データセットを作成します。

```
data closed;
   input date date. holiday $ 11-25;
   datalines;
26DEC08   Repairs
29DEC08   Repairs
30DEC08   Repairs
31DEC08   Repairs
23DEC08   Vacation
24DEC08   Christmas Eve
25DEC08   Christmas
;
```

アクティビティ開始日を基準にして Activities データセットを並べ替えます。Holidays データセットは並べ替える必要はありません。

```
proc sort data=meals;
   by date;
run;
```

食事の提供数を示す変数のピクチャ形式を作成します。

```
proc format;
   picture bfmt other = '000 Brkfst';
   picture lfmt other = '000 Lunch ';
   picture dfmt other = '000 Dinner';
run;
```

FORMCHAR オプションと LINESIZE オプションを設定します。FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

サマリーカレンダを作成します。DATA=は ACTIVITIES データセットを識別し、HOLIDATA=は HOLIDAYS データセットを識別します。START ステートメントは、アクティビティ開始日を含む ACTIVITIES データセットの変数を指定します。START は必須です。

```
proc calendar data=meals holidata=closed;
   start date;
```

休日情報を取得します。HOLISTART ステートメントと HOLIVAR ステートメントは、各休日の開始日と名前のそれぞれを含む HOLIDAYS データセットの変数を指定します。Holidays データセットを使用する場合は、HOLISTART が必須になります。

```
   holistart date;
```

```
holiname holiday;
```

合計値と平均値の計算、ラベル付け、および出力形式の適用を行います。 SUM ステートメントと MEAN ステートメントは、3 つの変数の合計値と平均値を計算し、指定された出力形式で印刷します。LABEL ステートメントは、凡例を印刷し、変数名ではなくラベルを使用します。FORMAT ステートメントは、ピクチャ形式を 3 つの変数に関連付けます。

```
sum brkfst lunch dinner / format=4.0;
mean brkfst lunch dinner / format=6.2;
label brkfst = 'Breakfasts Served'
      lunch  = '   Lunches Served'
      dinner = '   Dinners Served';
format brkfst bfmt.
       lunch lfmt.
       dinner dfmt.;
```

タイトルを指定します。

```
   title 'Meals Served in Company Cafeteria';
   title2 'Mean Number by Business Day';
run;

title;
```

出力:出力:HTML

アウトプット 8.14 *社員食堂で提供される食事:営業日別食事数*

Meals Served in Company Cafeteria
Mean Number by Business Day

December 2008

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	1 123 Brkfst 234 Lunch 238 Dinner	2 188 Brkfst 188 Lunch 198 Dinner	3 123 Brkfst 183 Lunch 176 Dinner	4 200 Brkfst 267 Lunch 243 Dinner	5 176 Brkfst 165 Lunch 177 Dinner	6
7	8 178 Brkfst 198 Lunch 187 Dinner	9 165 Brkfst 176 Lunch 187 Dinner	10 187 Brkfst 176 Lunch 231 Dinner	11 176 Brkfst 187 Lunch 222 Dinner	12 187 Brkfst 187 Lunch 123 Dinner	13
14	15 176 Brkfst 165 Lunch 177 Dinner	16 156 Brkfst . 167 Dinner	17 198 Brkfst 143 Lunch 167 Dinner	18 178 Brkfst 198 Lunch 187 Dinner	19 165 Brkfst 176 Lunch 187 Dinner	20
21	22 187 Brkfst 187 Lunch 123 Dinner	23 **Vacation**	24 Christmas Ev	25 *Christmas**	26 **Repairs***	27
28	29 **Repairs***	30 **Repairs***	31 **Repairs***			

	Sum	Mean
Breakfasts Served	2763	172.69
Lunches Served	2830	188.67
Dinners Served	2990	186.88

例 8: 例外的な勤務シフトの複数のサマリーカレンダ(個別出力)

要素: PROC CALENDAR ステートメントオプション
- DATETIME
- LEGEND

CALID statement
- _CAL_変数
- OUTPUT=SEPARATE オプション

その他のステートメント
- OUTSTART statement
- OUTFIN statement
- SUM statement

データセット: Well.Act

Well.Hol

詳細

この例では、次を行います。

- 単一 PROC ステップで、複数のカレンダのサマリーカレンダを生成します。
- カレンダを別々のページに印刷します。
- 休日を表示します。
- カレンダごとに別々の勤務パターン、勤務シフトおよび休日を使用します。

複数のカレンダに対する異なる出力の生成

この例では、複数のカレンダの個別出力を生成します。次の 2 点を変更するだけで、このデータの結合出力または混合出力を生成できます。

- Activities データセットの並べ替え方法
- OUTPUT=オプションの設定方法

表 8.14 *並べ替えと OUTPUT=設定*

印刷オプション	並べ替え変数	OUTPUT=設定	例
各カレンダを別々のページに印刷	カレンダ ID と開始日	SEPARATE	3, 8
すべてのアクティビティを同じページに印刷し、各カレンダを識別	開始日	COMBINE	4, 2
すべてのアクティビティを同じページに印刷するが、各カレンダの識別はなし	開始日	MIX	4

プログラム

```
libname well
'SAS-library';
run;

proc sort data=well.act;
   by _cal_ date;
run;

options formchar="|----|+|---+=|-/\<>*" linesize=132;

proc calendar data=well.act
              holidata=well.hol
              datetime legend;

   calid _cal_ / output=separate;

   start date;
   holistart date;
   holivar holiday;

   sum cost / format=dollar10.2;

   outstart Monday;
   outfin Saturday;

   title 'Well Drilling Cost Summary';
   title2 'Separate Calendars';
   format cost dollar10.2;
run;
```

プログラムの説明

Activities データセットが格納されている SAS ライブラリを指定します。

```
libname well
'SAS-library';
run;
```

カレンダ ID と開始日のそれぞれを含む変数を基準にして Activities データセットを並べ替えます。

```
proc sort data=well.act;
   by _cal_ date;
run;
```

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。この例では、出力でのデータの切り捨てを防ぐために LINESIZE を設定する必要があります。

```
options formchar="|----|+|---+=|-/\<>*" linesize=132;
```

サマリーカレンダを作成します。 DATA=は Activities データセット、HOLIDATA=は Holidays データセット、CALEDATA=は Calendar データセット、WORKDATA=は Workdays データセットを識別します。DATETIME は、START ステートメントで指定した変数に SAS 日時値が含まれるように指定します。LEGEND は、変数を識別するテキストを印刷します。

```
proc calendar data=well.act
```

```
                holidata=well.hol
                datetime legend;
```

各カレンダを別々のページに印刷します。 CALID ステートメントは、_CAL_変数によってカレンダが識別されるように指定します。OUTPUT=SEPARATE は、各カレンダの情報を別々のページに印刷します。

```
      calid _cal_ / output=separate;
```

アクティビティ開始日変数を指定し、休日情報を取得します。 START ステートメントは、アクティビティ開始日を含む Activities データセットの変数を指定します。HOLISTART ステートメントと HOLIVAR ステートメントは、各休日の開始日と名前のそれぞれを含む Holidays データセットの変数を指定します。これらのステートメントは、Holidays データセットを使用する場合は必須になります。

```
      start date;
      holistart date;
      holivar holiday;
```

合計値を計算します。 SUM ステートメントは、各カレンダのすべてのオブザベーションの COST 変数を合計します。

```
      sum cost / format=dollar10.2;
```

週 6 日を表示します。 OUTSTART と OUTFIN は、カレンダに月曜日から土曜日まで週 6 日を表示するように指定します。

```
      outstart Monday;
      outfin Saturday;
```

タイトルを指定して、COST 変数をフォーマットします。

```
      title 'Well Drilling Cost Summary';
      title2 'Separate Calendars';
      format cost dollar10.2;
   run;
```

出力:出力:HTML

アウトプット 8.15 *井戸掘りコストサマリー(その 1)*

Well Drilling Cost Summary
Separate Calendars

cal=CAL1

July 2002

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1 Drill Well 3.5 $1,000.00	2	3	4 ***Independence*** Lay Power Line 3 $2,000.00	5 Assemble Tank 4 $1,000.00	6
8 Build Pump House 3 $2,000.00	9	10	11 Pour Foundation 4 $1,500.00	12	13
15 Install Pump 4 $500.00	16	17	18	19 Install Pipe 2 $1,000.00	20 Erect Tower 6 $2,500.00
22	23	24	25	26	27
29	30	31			

Legend	Sum
task dur cost	$11,500.00

アウトプット 8.16 *井戸掘りコストサマリー(その 2)*

Well Drilling Cost Summary
Separate Calendars

cal=CAL2

July 2002

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1 Deliver Material 2 $500.00	2	3 Excavate 4.75 $3,500.00	4	5	6
8	9 *****Vacation*****	10	11	12	13
15	16	17	18	19	20
22	23	24	25	26	27
29	30	31			

Legend	Sum
task	
dur	
cost	$4,000.00

9 章

CATALOG プロシジャ

概要: CATALOG プロシジャ

CATALOG プロシジャは、SAS カタログのエントリを管理します。PROC CATALOG は対話型のステートメント主導のプロシジャで、次を実行できます。

- カタログのコンテンツのリストの作成
- カタログ、またはカタログ内の選択エントリのコピー
- カタログ内のエントリの名前変更、交換、削除
- カタログエントリ名の変更
- カタログエントリの説明の変更または削除による修正

SAS ライブラリおよびカタログの詳細については、*SAS 言語リファレンス: 解説編*を参照してください。

概念: CATALOG プロシジャ

SAS カタログのエントリの管理に SAS ウィンドウ環境を使用する方法については、SAS オンラインヘルプの**エクスプローラ**ウィンドウを参照してください。PROC CATALOG を使用するかわりに**エクスプローラ**ウィンドウを使用することもできます。**エクスプローラ**ウィンドウでは、プロシジャが行うほとんどの操作を実行することができます。

構文: CATALOG プロシジャ

ヒント: CATALOG プロシジャは RUN グループ処理をサポートします。

SAS エクスプローラウィンドウと、SQL プロシジャで DICTIONARY テーブルを使用して、同様の機能を実行できます。**エクスプローラ** ウィンドウの詳細については、オンラインヘルプを参照してください。PROC SQL の詳細については、*SAS SQL プロシジャユーザーガイド*を参照してください。

参照項目: CATALOG Procedure under Windows, UNIX, z/OS

PROC CATALOG CATALOG=<*libref.*>*catalog* <ENTRYTYPE=*entry-type*> <FORCE> <KILL>;

CONTENTS <OUT=*SAS-data-set*> <FILE=*fileref*>;

COPY OUT=<*libref.*>*catalog* <*option(s)*>;

SELECT *entry-1* <*entry-2* …> </ ENTRYTYPE=*entry-type* >;

EXCLUDE *entry-1* <*entry-2* …> </ ENTRYTYPE=*entry-type*>;

CHANGE *old-name-1=new-name-1*
<*old-name-2=new-name-2* …>
</ ENTRYTYPE=*entry-type*>;

EXCHANGE *name-1=other-name-1*
<*name-2=other-name-2* …>
</ ENTRYTYPE=*entry-type*>;

DELETE *entry-1* <*entry-2* …> </ ENTRYTYPE=*entry-type*>;

MODIFY *entry* (DESCRIPTION=<<'>*entry-description*<'>>)
</ ENTRYTYPE=*entry-type*>;

SAVE *entry-1* <*entry-2* …> </ ENTRYTYPE=*entry-type* >;

ステートメント	タスク	例
“PROC CATALOG ステートメント”	SAS カタログ間でエントリをコピーします	Ex. 1, Ex. 2, Ex. 3
“CHANGE ステートメント”	カタログエントリの名前を変更します	Ex. 2

ステートメント	タスク	例
“CONTENTS ステートメント”	カタログのコンテンツを印刷します	Ex. 2
“COPY ステートメント”	カタログ間で一部のエントリまたはすべてのエントリをコピーします	Ex. 1
“DELETE ステートメント”	指定エントリを削除します	Ex. 1
“EXCHANGE ステートメント”	2 つのカタログエントリの名前を切り替えます	
“EXCLUDE ステートメント”	エントリをコピー対象から除外します	Ex. 1
“MODIFY ステートメント”	カタログエントリの説明を変更します	Ex. 2
“SAVE ステートメント”	指定されているエントリ以外はすべて削除します	
“SELECT ステートメント”	選択エントリのみコピーします	Ex. 1

PROC CATALOG ステートメント

エントリを SAS カタログ間でコピーします。

構文

PROC CATALOG CATALOG=<*libref.*>*catalog* <ENTRYTYPE=*entry-type*> <FORCE> <KILL>;

オプション引数の要約

ENTRYTYPE=*entry-type*
: 現在の PROC CATALOG ステップの処理を 1 つのエントリの種類に制限します。

FORCE
: 別のリソース環境で開かれているカタログに対してステートメントの実行を強制します。

KILL
: SAS カタログのすべてのエントリを削除します。

必須引数

CATALOG=<*libref.*>*catalog*
処理する SAS カタログを指定します。

別名 CAT=, C=

デフォルト PROC CATALOG はカタログのすべてのエントリを処理します。

例 “例 3: FORCE オプションと KILL オプションの併用” (271 ページ)

オプション引数

ENTRYTYPE=*entry-type*
現在の PROC CATALOG ステップの処理を 1 つのエントリの種類に制限します。

別名 ET=

デフォルト PROC CATALOG はカタログのすべてのエントリを処理します。

操作 指定したエントリの種類は、従属ステートメントで使用される 1 レベルのエントリ名に適用されます。この指定は従属ステートメントでオーバーライドできません。

ENTRYTYPE=オプションは KILL オプションの影響を制限しません。

ヒント 単一の PROC CATALOG ステップで複数のエントリの種類を処理するには、ENTRYTYPE=を PROC CATALOG ステートメントではなく、従属ステートメントで使用します。

参照項目 “エントリの種類の指定” (262 ページ)

例 “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (266 ページ)

“例 2: コンテンツの表示、名前の変更、説明の変更” (269 ページ)

FORCE
別のリソース環境で開かれているカタログに対してステートメントの実行を強制します。

カタログのコンテンツを根本的に変えるような一部の CATALOG ステートメントには、操作するカタログに対する排他アクセスが必要です。排他アクセスを取得できない場合、アクションは失敗します。FORCE オプションの影響を受けるステートメントとカタログは、次のとおりです。

KILL
指定したカタログに影響します。

COPY
OUT=カタログに影響します。

COPY MOVE
IN=カタログと OUT=カタログに影響します。

SAVE
指定したカタログに影響します。

ヒント 排他アクセスが取得できない場合にもステートメントを実行するには、FORCE オプションを使用します。

例 “例 3: FORCE オプションと KILL オプションの併用” (271 ページ)

KILL
SAS カタログのすべてのエントリを削除します。

注意:
KILL オプションの影響を制限しないでください。このオプションは、その他のオプションまたはステートメントが有効になる前に SAS カタログのすべてのエントリを削除します。

操作 ENTRYTYPE=が指定されている場合でも、KILL オプションはすべてのカタログエントリを削除します。

KILL オプションはその他のステートメントが処理される前に SAS カタログのすべてのエントリを削除するため、SAVE ステートメントは影響しません。

ヒント KILL オプションはすべてのエントリを削除しますが、SAS ライブラリからの空のカタログは削除しません。空の SAS カタログを削除するには、PROC DATASETS、**DIR** ウィンドウなどの別の方法を使用する必要があります。

例 “例 3: FORCE オプションと KILL オプションの併用” (271 ページ)

CHANGE ステートメント

1 つ以上のカタログエントリの名前を変更します。

ヒント: 単一の CHANGE ステートメントで複数の名前を変更することも、複数の CHANGE ステートメントを使用することもできます。

例: “例 2: コンテンツの表示、名前の変更、説明の変更” (269 ページ)

構文

CHANGE *old-name-1=new-name-1*
<old-name-2=new-name-2 …>
</ ENTRYTYPE=entry-type>;

必須引数

old-name=new-name
カタログエントリの現在の名前と、それに割り当てる新しい名前を指定します。有効な SAS 名を指定します。

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目 “ENTRYTYPE=オプション” (263 ページ)

“エントリの種類の指定” (262 ページ)

CONTENTS ステートメント

プロシジャ出力のカタログのコンテンツをリストするか、コンテンツのリストを SAS データセット、外部ファイル、またはその両方に書き込みます。

注: CONTENTS ステートメントには、ENTRYTYPE=オプションを使用できません。

例: “例 2: コンテンツの表示、名前の変更、説明の変更” (269 ページ)

構文

CONTENTS <CATALOG=<*libref.*>*catalog* > <OUT=*SAS-data-set*> <FILE=*fileref*>;

引数なし

出力はプロシジャ出力に送信されます。

オプション引数

CATALOG=<*libref.*>*catalog*
処理する SAS カタログを指定します。

別名 CAT=, C=

デフォルト なし

FILE=*fileref*
SAS ファイル参照名で識別される、外部ファイルにコンテンツを送信します。

操作 *fileref* に割り当て済みのファイルがない場合は、動作環境での命名規則に従った名前の外部ファイルが作成されます。

OUT=*SAS-data-set*
コンテンツを SAS データセットに送信します。ステートメントが実行されると、データセットが作成されたことを示すメッセージが SAS ログに書き込まれます。データセットには、次の順序で 6 つの変数が含まれています。

LIBNAME	ライブラリ参照名
MEMNAME	カタログ名
NAME	エントリ名
TYPE	エントリの種類
DESC	エントリの説明
DATE	エントリが最後に変更された日付

COPY ステートメント

カタログ間で一部のエントリまたはすべてのエントリをコピーします。

制限事項: COPY ステートメントの影響は RUN ステートメントで、または SELECT または EXCLUDE ステートメント以外のステートメントの開始時に終了します。

ヒント: COPY ステートメントの後に、SELECT または EXCLUDE ステートメントのいずれか(両方ではなく)を使用して、コピーするエントリを制限します。

単一の PROC ステップで(PROC CATALOG ステートメントで指定した 1 つのカタログだけでなく)複数のカタログからエントリをコピーできます。

COPY ステートメントでは、ENTRYTYPE=オプションにスラッシュ(/)は不要です。

例: “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (266 ページ)

構文

COPY OUT=<*libref.*>*catalog* <*option(s)*>;

必須引数

OUT=<*libref.*>*catalog*
エントリのコピー先のカタログに名前を付けます。

オプション引数

ENTRYTYPE=*entry-type*
現在の COPY ステートメントと後続の SELECT または EXCLUDE ステートメントについて、処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目 “ENTRYTYPE=オプション” (263 ページ)

“エントリの種類の指定” (262 ページ)

IN=<*libref.*>*catalog*
コピーするカタログを指定します。

操作 IN=オプションは、PROC CATALOG ステートメントで指定された CATALOG=引数をオーバーライドします。

例 “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (266 ページ)

LOCKCAT=EXCLUSIVE | SHARE
複数のユーザーが同時に同じカタログにコピーできるかどうかを指定します。LOCKCAT=SHARE を使用することにより、カタログ全体ではなく個別のエントリがロックされ、これによりスループットが高まります。デフォルトは LOCKCAT=EXCLUSIVE で、カタログ全体を 1 人のユーザーにロックします。LOCKCAT=SHARE オプションの使用により、単一ユーザー環境で使用される場合はパフォーマンスが低下する可能性があります。各エントリのロックとロック解除と関連付けられているオーバーヘッドのためです。

MOVE
新しいコピーの作成後に元のカタログまたはエントリを削除します。

操作 MOVE オプションによってカタログからすべてのエントリが削除されると、プロシジャはライブラリからカタログを削除します。

NEW
カタログ(OUT=オプションで指定)がすでに存在する場合は上書きします。NEW オプションを省略すると、PROC CATALOG はカタログを更新します。

参照項目 連結カタログでの NEW オプションの使用については、“カタログ連結” (264 ページ)を参照してください。

NOEDIT
次の SAS/AF エントリの種類のコピー済バージョンが BUILD プロシジャによって編集されないようにします。

CBT
FRAME
HELP
MENU
PROGRAM
SCL
SYSTEM

制限事項 これらの種類以外のエントリに NOEDIT オプションを指定しても、無視されます。

ヒント 他のユーザー用の SAS/AF アプリケーションを作成する場合は、NOEDIT オプションを使用して特定のカタログエントリが変更されないようにして、アプリケーションを保護します。

例 “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (266 ページ)

NOSOURCE
SAS/AF PROGRAM、FRAME または SCL エントリをコピーする場合にソース行のコピーを省略します。

別名 NOSRC

制限事項 このオプションを PROGRAM、FRAME、SCL エントリ以外のエントリに指定しても、無視されます。

DELETE ステートメント

エントリを SAS カタログから削除します。

ヒント: いくつかのエントリのみを削除するには、DELETE ステートメントを使用します。削除しないエントリを指定した方が効率的であれば、SAVE を使用します。

複数のエントリを指定できます。複数の DELETE ステートメントを使用することもできます。

参照項目: SAVE ステートメント (259 ページ)

例: “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (266 ページ)

構文

DELETE *entry-1 <entry-2 …> </ ENTRYTYPE=entry-type>*;

必須引数

entry-1 <entry-2 …>
SAS カタログエントリの名前を 1 つ以上指定します

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

参照項目 “ENTRYTYPE=オプション” (263 ページ)

“エントリの種類の指定” (262 ページ)

EXCHANGE ステートメント

2 つのカタログエントリの名前を交換します。

制限事項: EXCHANGE ステートメントの使用時には、カタログエントリの種類は同じである必要があります。

構文

EXCHANGE *name-1=other-name-1*
<name-2=other-name-2 …>
</ ENTRYTYPE=entry-type>;

必須引数

name=other-name
プロシジャが交換する 2 つのカタログエントリ名を指定します。

操作 ENTRYTYPE=オプションを PROC CATALOG ステートメントまたは EXCHANGE ステートメントで使用する場合、エントリの種類なしでエントリ名のみ指定できます。

参照項目 “エントリの種類の指定” (262 ページ)

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目 “ENTRYTYPE=オプション” (263 ページ)

“エントリの種類の指定” (262 ページ)

EXCLUDE ステートメント

COPY ステートメントがコピーしないエントリを指定します。

制限事項: EXCLUDE ステートメントには COPY ステートメントが必要です。
EXCLUDE ステートメントを SELECT ステートメントと一緒に使用しないでください。

ヒント: 単一の EXCLUDE ステートメントで複数のエントリを指定できます。
複数の EXCLUDE ステートメントを RUN グループ内の単一の COPY ステートメントと使用できます。

参照項目: COPY ステートメント (255 ページ) および SELECT ステートメント (260 ページ)

例: “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (266 ページ)

構文

EXCLUDE *entry-1 <entry-2 …> </ ENTRYTYPE=entry-type>*;

必須引数

entry-1 <entry-2 …>
SAS カタログエントリの名前を 1 つ以上指定します

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

参照項目 “エントリの種類の指定” (262 ページ)

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目 “ENTRYTYPE=オプション” (263 ページ)

“エントリの種類の指定” (262 ページ)

MODIFY ステートメント

カタログエントリの説明を変更します。

例: “例 2: コンテンツの表示、名前の変更、説明の変更” (269 ページ)

構文

MODIFY *entry* (DESCRIPTION=<<'>*entry-description*<'>>)
</ ENTRYTYPE=*entry-type*>;

必須引数

entry
SAS カタログエントリの名前を 1 つ指定します。エントリの名前と一緒に種類を指定できます(*entry-name.entry-type*)。

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

参照項目 “エントリの種類の指定” (262 ページ)

DESCRIPTION=<<'>*entry-description*<'>>
カタログエントリの説明を最大 256 文字の新しい説明に置き換えるか、すべて削除して変更します。説明を一重引用符または二重引用符で囲みます。

別名 DESC

ヒント MODIFY ステートメントを CATALOG プロシジャで使用している場合に現在の説明を削除するには、DESCRIPTION=オプションをテキストなしで使用します。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目 “ENTRYTYPE=オプション” (263 ページ)
“エントリの種類の指定” (262 ページ)

SAVE ステートメント

SAS カタログから削除しないエントリを指定します。

制限事項: SAVE ステートメントでは KILL オプションの影響を制限できません。

ヒント: SAVE ステートメントを使用して、カタログ内の一部を除くすべてのエントリを削除します。削除するエントリを指定した方が効率的であれば、DELETE ステートメントを使用します。
複数のエントリを指定することも、複数の SAVE ステートメントを使用することもできます。

参照項目: DELETE Statement (256 ページ)

構文

SAVE *entry-1 <entry-2 …> </ ENTRYTYPE=entry-type >*;

必須引数

entry-1 <entry-2…>
SAS カタログエントリの名前を 1 つ以上指定します

制限事項 SAVE ステートメントで名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目 "ENTRYTYPE=オプション" (263 ページ)

"エントリの種類の指定" (262 ページ)

SELECT ステートメント

COPY ステートメントがコピーするエントリを指定します。

制限事項: SELECT ステートメントには COPY ステートメントが必要です。
SELECT ステートメントは、EXCLUDE ステートメントと併用できません。

ヒント: 単一の SELECT ステートメントで複数のエントリを指定できます。
複数の SELECT ステートメントを RUN グループ内の単一の COPY ステートメントと使用できます。

参照項目: COPY ステートメント (255 ページ)および EXCLUDE ステートメント (258 ページ)

例: "例 1: 複数のカタログからカタログエントリをコピー、削除、移動する" (266 ページ)

構文

SELECT *entry-1 <entry-2 …> </ ENTRYTYPE=entry-type >*;

必須引数

entry-1 <entry-2 …>
SAS カタログエントリの名前を 1 つ以上指定します

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目 “ENTRYTYPE=オプション” (263 ページ)。
“エントリの種類の指定” (262 ページ).

CATALOG プロシジャの使用

RUN グループを使用した対話型処理

定義

CATALOG プロシジャは対話型です。PROC CATALOG ステートメントをサブミットすると、PROC CATALOG ステートメントを繰り返さずにステートメントまたはステートメントグループを続けてサブミット、実行できます。

RUN ステートメントで終わる一連のプロシジャステートメントは、*RUN グループ*と呼ばれます。指定ステートメントグループで指定される変更は、RUN ステートメントの使用時に有効になります。

PROC CATALOG ステップの終了方法

DATA ステップおよび、ほとんどの SAS プロシジャでは、RUN ステートメントはステップの境界で、ステップを終了します。次のリストには、PROC CATALOG ステップを強制終了するための方法を示します。ただし、単純な RUN ステートメントは対話型プロシジャを終了しません。次のリストには、PROC CATALOG ステップを強制終了するための方法を示します。

- QUIT ステートメントをサブミットします
- RUN ステートメントを CANCEL オプションでサブミットします
- 別の DATA または PROC ステートメントをサブミットします
- SAS セッションを終了します

注: QUIT、DATA、または PROC ステートメントを入力すると、最後の RUN グループの後のステートメントが CATALOG プロシジャの強制終了前に実行されます。RUN ステートメントを CANCEL オプションで入力すると、残りのステートメントはプロシジャの終了前に*実行されません。*

“例 2: コンテンツの表示、名前の変更、説明の変更” (269 ページ)を参照してください。

エラー処理と RUN グループ

エラー処理は、RUN グループへのステートメント分割の部分に基づいています。構文エラーが発生すると、現在の RUN グループのステートメントの 1 つが実行され、次の RUN グループの実行に進みます。

たとえば、次のステーテメントには、スペルミスのある DELETE ステートメントが含まれています。

```
proc catalog catalog=misc entrytype=help;
      copy out=drink;
         select coffee tea;
      del juices;        /* INCORRECT!!! */
      exchange glass=plastic;
   run;
      change calstats=nutri;
   run;
quit;
```

DELETE ステートメントが誤って DEL と指定されているため、PROC CATALOG ステートメント自体を除く、RUN グループのステートメントはすべて実行されません。CHANGE ステートメントは実行されますが、これは異なる RUN グループにあるためです。

注: 1 つの RUN グループのステートメントが前の RUN グループの影響に依存するバッチジョブの設定時は注意が必要です。特にエントリの削除時と名前変更時です。

エントリの種類の指定

エントリの種類を指定する 4 つの方法

エントリの種類にデフォルト値はないため、入力しないと PROC CATALOG はエラーを生成します。エントリの種類は、次の表の 4 つの方法のうちいずれかを使用して入力できます。

表 9.1 *エントリの種類の指定*

エントリの種類	例
エントリ名	`delete test1.program test1.log test2.log;`
ET= (かっこで囲む)	`delete test1 (et=program);`
ET=スラッシュの後ろ *	`delete test1 (et=program) test1 test2 / et=log;`
ENTRYTYPE= (スラッシュなし) **	`proc catalog catalog=mycat et=log; delete test1 test2;`

* 従属ステートメントに入力

** PROC CATALOG または COPY ステートメントに入力

注: CONTENTS ステートメント以外のすべてのステートメントは、ENTRYTYPE=オプションを受け入れます。

ENTRYTYPE=オプションの利点

同じ種類のエントリを複数処理する場合、ENTRYTYPE=オプションによってキー入力を省略することができます。

現在のステップのすべてのステートメントに対してエントリの種類のデフォルトを作成するには、ENTRYTYPE=オプションを PROC CATALOG ステートメントで使用します。現在のステートメントに対してのみデフォルトを設定するには、ENTRYTYPE=オプションを従属ステートメントで使用します。

ある種類のエントリを多数指定し、その他の種類のエントリを少数しか指定しない場合があります。ENTRYTYPE=オプションを使用してデフォルトを指定し、デフォルトを使用するエントリの後に、デフォルトをオーバーライドする個別のエントリを(ENTRYTYPE=)のように*かっこで囲んで*指定します。

よくあるエラーの回避

ENTRYTYPE=オプションは、PROC CATALOG ステートメントと従属ステートメントの両方で指定することはできません。たとえば、次のステートメントはエラーを生成し、エントリを削除しません。ENTRYTYPE=オプションの指定が相互に矛盾するためです。

```
   /* THIS IS INCORRECT CODE. */
proc catalog cat=sample et=help;
   delete a b c / et=program;
run;
quit;
```

ENTRYTYPE=オプション

ENTRYTYPE=オプションは、CATALOG プロシジャの CONTENTS ステートメントを除くすべてのステートメントで利用可能です。

ENTRYTYPE=*entry-type*
*(かっこで囲まない場合)*PROC CATALOG ステートメントで使用されると、全体の PROC ステップに対するデフォルトのエントリの種類を設定します。その他すべてのステートメントで、このオプションは現在のステートメントのデフォルトのエントリの種類を設定します。ENTRYTYPE=オプションを省略すると、PROC CATALOG はカタログのすべてのエントリを処理します。NEW オプションを省略すると、PROC CATALOG はカタログを更新します。

別名 ET=

デフォルト PROC CATALOG はカタログのすべてのエントリを処理します。

操作 ENTRYTYPE=オプションを PROC CATALOG ステートメントで指定する場合、従属ステートメントで ENTRYTYPE=または(ENTRYTYPE=)を指定しないでください。

エントリ名の直後にかっこで囲んだ(ENTRYTYPE=)を指定すると、同じステートメントの ENTRYTYPE=オプションをオーバーライドします。

ヒント PROC CATALOG と COPY ステートメント以外のすべてのステートメントで、ENTRYTYPE=オプションはスラッシュの後に続きます。

単一の PROC CATALOG ステップで複数のエントリの種類を処理するには、ENTRYTYPE=オプションを PROC CATALOG ステートメントではなく、従属ステートメントで使用します。

参照項目 “エントリの種類の指定” (262 ページ) および “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (266 ページ)

(ENTRYTYPE=*entry-type*)
*(かっこで囲んだ場合)*直前のエントリの種類を特定します。

別名 (ET=)

制限事項 (ENTRYTYPE=)オプションを従属ステートメントのエントリ名の直後に指定しても、PROC CATALOG ステートメントの ENTRYTYPE=オプションをオーバーライドできません。構文エラーが生成されます。

操作 エントリ名の直後に(ENTRYTYPE=)オプションを指定すると、同じステートメントの ENTRYTYPE=オプションをオーバーライドします。

ヒント この形式は主に、従属ステートメントで使用される ENTRYTYPE=オプションの例外を指定する場合に便利です。次のステートメントは、A.HELP、B.FORMAT および C.HELP を削除します。

```
delete a b (et=format) c / et=help;
```

CHANGE および EXCHANGE ステートメントの場合は、次の例にあるように、各名前ペアに対して 1 回のみ(2 つ目の名前の後ろに)、かっこで囲んだ(ENTRYTYPE=)オプションを指定します。例:

```
change old1=new1 (et=log)
       old1=new2 (et=help);
```

参照項目 “エントリの種類の指定” (262 ページ), “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (266 ページ) および “例 2: コンテンツの表示、名前の変更、説明の変更” (269 ページ)

カタログ連結

カタログ連結について

CATALOG 連結には 2 つの種類があります。1 つ目は LIBNAME ステートメントによって指定され、2 つ目はグローバル CATNAME ステートメントによって指定されます。単一(未連結)のカタログで使用可能なすべてのステートメントとオプションは、カタログ連結で使用できます。

制約

連結カタログのコピーに CATALOG プロシジャを使用し、NEW オプションを使用する場合、次のルールが適用されます。

- 入力カタログが連結カタログであり、出力カタログが入力の連結カタログのいずれかのレベルに存在する場合、コピーは実行できません。
- 出力カタログが連結カタログであり、入力カタログが出力の連結カタログの最初のレベルに存在する場合、コピーは実行できません。

たとえば、次のコードは、これらの 2 つのルールとコピーの失敗を示します。

```
libname first 'SAS-library-1';
libname second 'SAS-library-2';
   /* create concat.x */
libname concat (first second);

   /* fails rule #1 */
proc catalog c=concat.x;
   copy out=first.x new;
```

```
run;
quit;

   /* fails rule #2 */
proc catalog c=first.x;
   copy out=concat.x new;
run;
quit;
```

次の表に、コピーが可能な場合を示します。表では、A と B がライブラリで、それぞれにカタログ X が含まれています。カタログ C は A と B の自動連結で、カタログ D は B と A の自動連結です。

入力カタログ	出力カタログ	コピー可能か
C.X	B.X	いいえ
C.X	D.X	いいえ
D.X	C.X	いいえ
A.X	A.X	いいえ
A.X	B.X	はい
B.X	A.X	はい
C.X	A.X	いいえ
B.X	C.X	はい
A.X	C.X	いいえ

結果: CATALOG プロシジャ

CATALOG プロシジャは、CONTENTS ステートメントがオプションなしで実行されるときに出力を生成します。。このプロシジャ出力には、名前が割り当てられます。この名前を使用して、Output Delivery System (ODS)の使用時にテーブルを参照し、テーブルを選択して出力データセットを作成できます。詳細については、を 参照してください “ODS Table Names and the Base SAS Procedures That Produce Them” (*SAS Output Delivery System: Advanced Topics*).

表 9.2 *CATALOG プロシジャによって生成される ODS 出力*

テーブル名	ライブラリの種類
Catalog_Random	カタログがランダムアクセスライブラリにある場合
Catalog_Sequential	カタログが順次ライブラリにある場合

例: CATALOG プロシジャ

例 1: 複数のカタログからカタログエントリをコピー、削除、移動する

要素: PROC CATALOG ステートメントオプション
CAT=
COPY statement
DELETE statement

詳細

この例では、次のタスクについて説明します。

- 少数の除外対象エントリの指定によるエントリのコピー
- 少数のコピー対象エントリの指定によるエントリのコピー
- 編集からのエントリの保護
- エントリの移動
- エントリの削除
- 複数のカタログのエントリの処理
- 複数の RUN グループでのエントリの処理

SAS カタログ Perm.Sample には、次のエントリが含まれています。

```
DEFAULT     FORM        Default form for printing
FSLETTER    FORM        Standard form for letters (HP Laserjet)
LOAN        FRAME       Loan analysis application
LOAN        HELP        Information about the application
BUILD       KEYS        Function Key Definitions
LOAN        KEYS        Custom key definitions for application
CREDIT      LOG         credit application log
TEST1       LOG         Inventory program
TEST2       LOG         Inventory program
TEST3       LOG         Inventory program
LOAN        PMENU       Custom menu definitions for applicaticm
CREDIT      PROGRAM     credit application pgm
TEST1       PROGRAM     testing budget applic.
TEST2       PROGRAM     testing budget applic.
TEST3       PROGRAM     testing budget applic.
LOAN        SCL         SCL code for loan analysis application
PASSIST     SLIST       User profile
```

SAS カタログ Perm.Formats には、次のエントリが含まれています。

```
REVENUE     FORMAT      FORMAT:MAXLEN=16,16,12
DEPT        FORMATC     FORMAT:MAXLEN=1,1,14
```

プログラム

```
libname perm 'SAS-library';

proc catalog cat=perm.sample;
   delete credit.program credit.log;
run;

   copy out=tcatall;
run;

   copy out=testcat;
      exclude test1 test2 test3  passist (et=slist) / et=log;
run;

   copy out=logcat move;
      select test1 test2 test3 / et=log;
run;

   copy out=perm.finance noedit;
      select loan.frame loan.help loan.keys loan.pmenu;
run;


copy in=perm.formats out=perm.finance;
      select revenue.format dept.formatc;
run;
quit;
```

プログラムの説明

ライブラリ参照を SAS ライブラリに割り当てます。 LIBNAME ステートメントは、ライブラリ参照名 Perm を永続 SAS カタログを含む SAS ライブラリに割り当てます。

```
libname perm 'SAS-library';
```

Perm.Sample カタログから 2 つのエントリを削除します。

```
proc catalog cat=perm.sample;
   delete credit.program credit.log;
run;
```

Perm.Sample カタログのすべてのエントリを Work.TCatAll カタログにコピーします。

```
   copy out=tcatall;
run;
```

---

3 つの LOG エントリと Paassist.Slist 以外のすべてを Perm.Sample から Work.TestCat にコピーします。 EXCLUDE ステートメントは、コピーしないエントリを指定します。ET= は、デフォルトの種類を指定します。(ET=)は、デフォルトの種類の例外を指定します。

```
   copy out=testcat;
      exclude test1 test2 test3  passist (et=slist) / et=log;
run;
```

---

3 つの LOG エントリを Perm.Sample から Work.LogCat に移動させます。 SELECT ステートメントは、移動するエントリを指定します。ET=は、処理を LOG エントリに制限します。

```
   copy out=logcat move;
      select test1 test2 test3 / et=log;
```

```
run;
```

5 つの SAS/AF ソフトウェアエントリを Perm.Sample から Perm.Finance にコピーします。 NOEDIT オプションは、これらのエントリが Perm.Finance 内で PROC BUILD によって編集されないように保護します。

```
   copy out=perm.finance noedit;
      select loan.frame loan.help loan.keys loan.pmenu;
run;
```

2 つの出力形式を Perm.Formats から Perm.Finance にコピーします。 IN=オプションを使用して、PROC CATALOG ステートメントで指定されたものとは異なるカタログからコピーできます。。COPY および SELECT ステートメントは、QUIT ステートメントが PROC CATALOG ステップを終了する前に実行されます。数値の出力形式と文字の出力形式のエントリの種類に注意してください。REVENUE.FORMAT は数値の出力形式で、DEPT.FORMATC は文字の出力形式です COPY および SELECT ステートメントは、QUIT ステートメントが PROC CATALOG ステップを終了する前に実行されます。

```
copy in=perm.formats out=perm.finance;
      select revenue.format dept.formatc;
run;
quit;
```

SAS ログ

ログ 9.1 *PROC CATALOG を使用したエントリのコピー、保護、削除、処理*

```
1    libname perm 'SAS-library'; NOTE:Libref PERM was successfully assigned as follows:Engine:
V9 Physical Name:SAS-library\perm 2    proc catalog cat=perm.sample; NOTE:Writing HTML Body file:
sashtml.htm 3       delete credit.program credit.log; 4run; NOTE:Deleting entry CREDIT.PROGRAM in
catalog PERM.SAMPLE.NOTE:Deleting entry CREDIT.LOG in catalog PERM.SAMPLE.5       copy out=tcatall;
6    run; NOTE:Copying entry DEFAULT.FORM from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry FSLETTER.FORM from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog WORK.TCATALL.NOTE:opying
entry BUILD.KEYS from catalog PERM.SAMPLE to catalog WORK.TCATALL.NOTE:Copying entry LOAN.KEYS from
catalog PERM.SAMPLE to catalog WORK.TCATALL.NOTE:Copying entry TEST1.LOG from catalog PERM.SAMPLE to
catalog WORK.TCATALL.NOTE:Copying entry TEST2.LOG from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry TEST3.LOG from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry TEST1.PROGRAM from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry TEST2.PROGRAM from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry TEST3.PROGRAM from catalog PERM.SAMPLE to catalog
WORK.TCATALL.NOTE:Copying entry LOAN.SCL from catalog PERM.SAMPLE to catalog WORK.TCATALL.NOTE:Copying
entry PASSIST.SLIST from catalog PERM.SAMPLE to catalog WORK.TCATALL.7       copy out=testcat;
8          exclude test1 test2 test3  passist (et=slist) / et=log; 9    run; NOTE:Copying entry
DEFAULT.FORM from catalog PERM.SAMPLE to catalog WORK.TESTCATNOTE:Copying entry FSLETTER.FORM from
catalog PERM.SAMPLE to catalog WORK.TESTCAT.NOTE:Copying entry LOAN.FRAME from catalog PERM.SAMPLE to
catalog WORK.TESTCAT.NOTE:Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog
WORK.TESTCAT.NOTE:Copying entry BUILD.KEYS from catalog PERM.SAMPLE to catalog
WORK.TESTCAT.NOTE:Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog
WORK.TESTCAT.NOTE:Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog
WORK.TESTCAT.NOTE:Copying entry TEST1.PROGRAM from catalog PERM.SAMPLE to catalog
WORK.TESTCAT.NOTE:Copying entry TEST2.PROGRAM from catalog PERM.SAMPLE to catalog
WORK.TESTCAT.NOTE:Copying entry TEST3.PROGRAM from catalog PERM.SAMPLE to catalog
WORK.TESTCAT.NOTE:Copying entry LOAN.SCL from catalog PERM.SAMPLE to catalog WORK.TESTCAT.10      copy
out=logcat move; 11         select test1 test2 test3 / et=log; 12   run; NOTE:Moving entry TEST1.LOG
from catalog PERM.SAMPLE to catalog WORK.LOGCAT.NOTE:Moving entry TEST2.LOG from catalog PERM.SAMPLE
to catalog WORK.LOGCAT.NOTE:Moving entry TEST3.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.
13      copy out=perm.finance noedit; 14         select loan.frame loan.help loan.keys loan.pmenu;
15   run;
```

```
NOTE:Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog PERM.FINANCE.Copying entry
LOAN.FRAME from catalog PERM.SAMPLE to catalog PERM.FINANCE.NOTE:NOTE:Copying entry LOAN.HELP from
catalog PERM.SAMPLE to catalog PERM.FINANCE.NOTE:NOTE:Copying entry LOAN.KEYS from catalog PERM.SAMPLE
to catalog PERM.FINANCE.NOTE:NOTE:Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog
PERM.FINANCE.16      copy in=perm.formats out=perm.finance; 17         select revenue.format
dept.formatc; 18   quit; NOTE:Copying entry REVENUE.FORMAT from catalog PERM.FORMATS to catalog
PERM.FINANCE.NOTE:NOTE:Copying entry DEPT.FORMATC from catalog PERM.FORMATS to catalog PERM.FINANCE.
```

例 2: コンテンツの表示、名前の変更、説明の変更

要素: PROC CATALOG ステートメントオプション
CATALOGS=
CHANGE statement
CONTENTS ステートメント
MODIFY statement

他の要素: TITLE statement

詳細

この例では、次のタスクについて説明します。

- カタログのエントリをリストして出力をファイルに送る
- エントリの名前の変更
- エントリの説明の変更
- 複数の RUN グループでのエントリの処理

プログラム

```
libname perm 'SAS-library';

proc catalog catalog=perm.finance;
   contents;
title1 'Contents of PERM.FINANCE before changes are made';
run;

   change dept=deptcode (et=formatc);
run;

   modify loan.frame (description='Loan analysis app. - ver1');
   contents;
title1 'Contents of PERM.FINANCE after changes are made';
run;

quit;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME ステートメントは、ライブラリ参照を永続 SAS カタログを含む SAS ライブラリに割り当てます。

```
libname perm 'SAS-library';
```

カタログのエントリをリストし、ファイルに出力を送ります。 CONTENTS ステートメントは、SAS カタログ Perm.Finance のコンテンツのリストを作成し、ファイルに出力を送ります。

```
proc catalog catalog=perm.finance;
   contents;
title1 'Contents of PERM.FINANCE before changes are made';
run;
```

エントリ名を変更します。 CHANGE ステートメントは、ユーザー作成の文字の出力形式を含むエントリの名前を変更します。(ET=)は、エントリの種類を指定します。

```
   change dept=deptcode (et=formatc);
run;
```

複数の RUN グループでエントリを処理します。 MODIFY ステートメントはエントリの説明を変更します。すべての変更の適用後、CONTENTS ステートメントは Perm.Finance のコンテンツのリストを作成します。QUIT はプロシジャを終了します。

```
   modify loan.frame (description='Loan analysis app. - ver1');
   contents;
title1 'Contents of PERM.FINANCE after changes are made';
run;

quit;
```

出力例

アウトプット 9.1 *変更前後の Perm.Finance のコンテンツ*

Contents of PERM.FINANCE before changes are made

Contents of Catalog PERM.FINANCE					
#	Name	Type	Create Date	Modified Date	Description
1	REVENUE	FORMAT	10/16/1996 13:48:11	10/16/1996 13:48:11	FORMAT:MAXLEN=16,16,12
2	DEPTCODE	FORMATC	10/30/1996 13:40:42	10/30/1996 13:40:42	FORMAT:MAXLEN=1,1,14
3	LOAN	FRAME	10/30/1996 13:40:43	02/18/2014 13:31:25	Loan analysis app. - ver1
4	LOAN	HELP	10/16/1996 13:48:10	10/16/1996 13:48:10	Information about the application
5	LOAN	KEYS	10/16/1996 13:48:10	10/16/1996 13:48:10	Custom key definitions for application
6	LOAN	PMENU	10/16/1996 13:48:10	10/16/1996 13:48:10	Custom menu definitions for application
7	LOAN	SCL	10/16/1996 13:48:10	10/16/1996 13:48:10	SCL code for loan analysis application

Contents of PERM.FINANCE after changes are made

Contents of Catalog PERM.FINANCE					
#	Name	Type	Create Date	Modified Date	Description
1	REVENUE	FORMAT	10/16/1996 13:48:11	10/16/1996 13:48:11	FORMAT:MAXLEN=16,16,12
2	DEPTCODE	FORMATC	10/30/1996 13:40:42	10/30/1996 13:40:42	FORMAT:MAXLEN=1,1,14
3	LOAN	FRAME	10/30/1996 13:40:43	03/28/2014 10:26:50	Loan analysis app. - ver1
4	LOAN	HELP	10/16/1996 13:48:10	10/16/1996 13:48:10	Information about the application
5	LOAN	KEYS	10/16/1996 13:48:10	10/16/1996 13:48:10	Custom key definitions for application
6	LOAN	PMENU	10/16/1996 13:48:10	10/16/1996 13:48:10	Custom menu definitions for application
7	LOAN	SCL	10/16/1996 13:48:10	10/16/1996 13:48:10	SCL code for loan analysis application

例 3: FORCE オプションと KILL オプションの併用

要素: PROC CATALOG ステートメントオプション
CATALOG=
FORCE
KILL

他の要素: %MACRO statement
%MEND statement
%PUT statement

詳細

この例では、次のタスクについて説明します。

- リソース環境の作成
- KILL オプションによってすべてのカタログエントリの削除を試行するが、エラーを受け取る
- KILL オプションによってすべてのカタログエントリが正常に削除されるように、FORCE オプションを指定する

プログラム

```
 %macro matt;
    %put &syscc;
%mend matt;

proc catalog c=work.sasmacr kill;
run;
quit;

proc catalog c=work.sasmacr kill force;
run;
quit;
```

プログラムの説明

プロセスを開始します(リソース環境)。 これは、Work.Sasmacr カタログのカタログエントリ MATT を開いて実行します。

```
 %macro matt;
    %put &syscc;
%mend matt;
```

KILL オプションを指定して、Work.Sasmacr のすべてのカタログエントリを削除します。 リソース環境(カタログを使用したプロセス)があるため、KILL は機能せず、エラーがログに送信されます。

```
proc catalog c=work.sasmacr kill;
run;
quit;
```

カタログエントリを削除するために、KILL オプションに FORCE オプションを指定します。

```
proc catalog c=work.sasmacr kill force;
run;
quit;
```

ログの例

ログ 9.2 *KILL オプションによる SAS ログへのエラー送信*

```
1     %macro matt; 2       %put &syscc; 3       %mend matt; 4 5    proc catalog
c=work.sasmacr kill; NOTE:Writing HTML Body file: sashtml.htm 6    run;
ERROR:You cannot open WORK.SASMACR.CATALOG for update access because
WORK.SASMACR.CATALOG is in use by you in resource environment
_O_TAGS.WARNING:Command CATALOG not processed because of errors noted above.7
quit; NOTE:The SAS System stopped processing this step because of
errors.NOTE:PROCEDURE CATALOG used (Total process time): real time
6.46 seconds cpu time            0.62 seconds
```

ログ 9.3 *KILL オプションへの FORCE オプションの追加によるカタログエントリの削除*

```
8    proc catalog c=work.sasmacr kill force; 9    run; NOTE:Deleting entry
MATT.MACRO in catalog WORK.SASMACR.10    quit; NOTE:PROCEDURE CATALOG used
(Total process time): real time           0.01 seconds cpu time            0.01
seconds
```

10 章

CHART プロシジャ

概要:CHART プロシジャ

CHART プロシジャの動作について

CHART プロシジャは、縦棒グラフおよび横棒グラフ、ブロックチャート、円グラフ、スターチャートを作成します。これらの種類のチャートは、変数の値、またはこれらの値と関連する統計量をグラフ表示します。グラフ表示される変数には、数値または文字が可能です。

PROC CHART はデータを迅速にビジュアル化できる便利なツールですが、色やさまざまなフォントを含むプレゼンテーション品質のグラフを作成する必要がある場合は SAS/GRAPH ソフトウェアを使用します。SAS/GRAPH ソフトウェアの GCHART プロシジャは、PROC CHART が作成するものと同じ種類のチャートを作成します。また、PROC GCHART はドーナツチャートを作成できます。

PROC CHART が作成するチャートの種類について

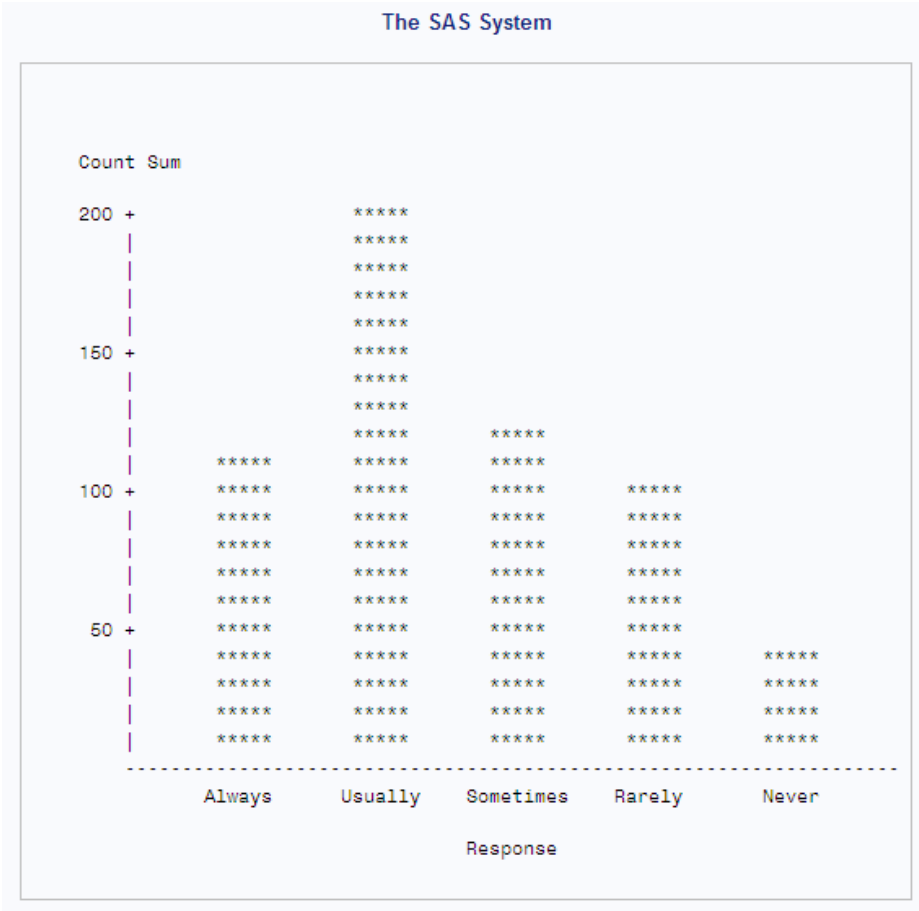
棒グラフ

横棒グラフと縦棒グラフは、データの大きさをバーで表します。バーはそれぞれデータのカテゴリを表しています。バーの長さや高さは、カテゴリごとの統計量の値を表します。

次の出力に、サーベイデータからの 5 つのカテゴリに対する回答数を表示する縦棒グラフを示します。次のステートメントでは、出力が生成されます。

```
proc chart data=survey;
   vbar response / sumvar=count
   axis=0 to 200 by 50
   midpoints='Always' 'Usually'
             'Sometimes' 'Rarely' 'Never';
run;
```

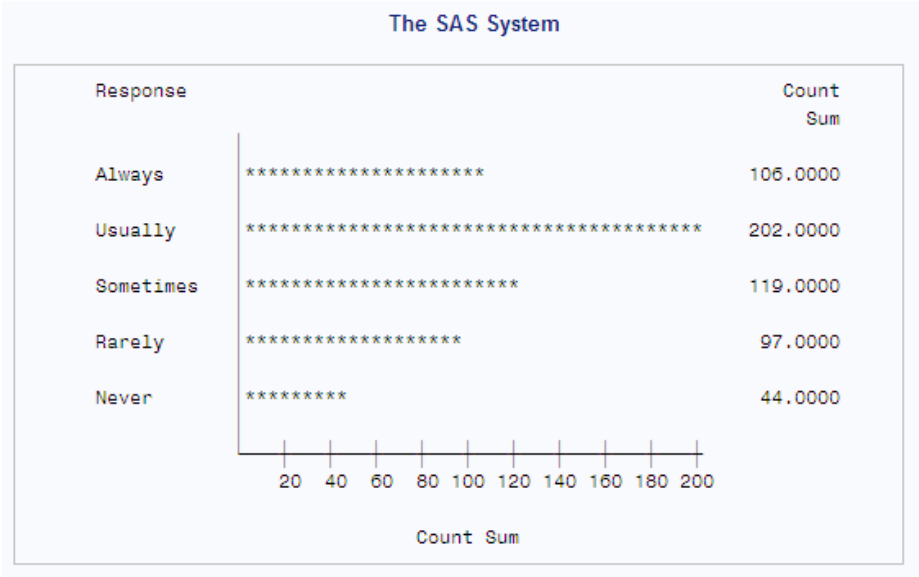
アウトプット 10.1 縦棒グラフ



次の出力は、同じデータを横棒グラフで表したものです。2 種類の棒グラフの特性は基本的に同じですが、横棒グラフにはデフォルトでバーの右側に統計量値の表が表示されます。次のステートメントでは、出力が生成されます。

```
proc chart data=survey;
   hbar response / sumvar=count
        midpoints='Always' 'Usually'
           'Sometimes' 'Rarely' 'Never';
run;
```

アウトプット 10.2　横棒グラフ

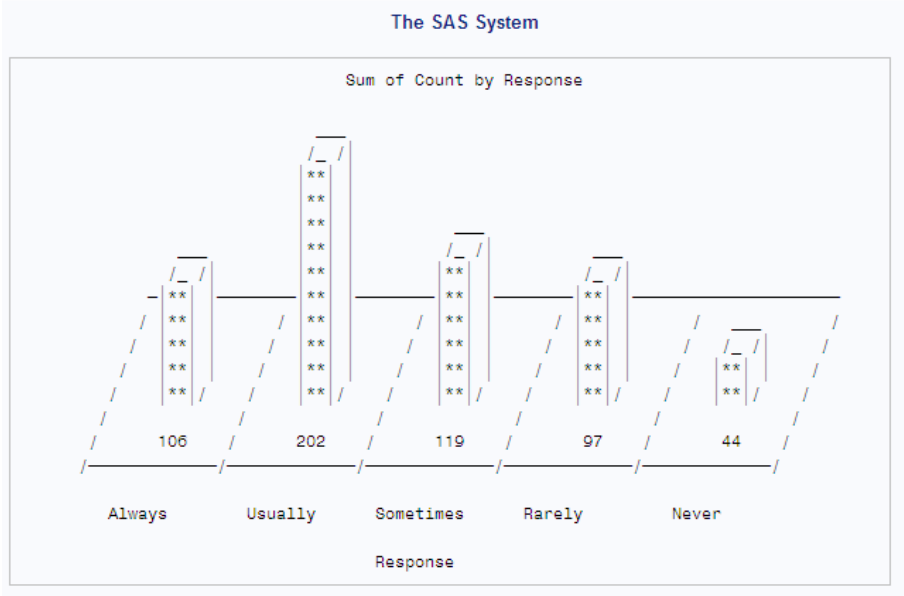


ブロックチャート

ブロックチャートは、データのカテゴリを表す四角形に異なる高さのブロックを配置して、データの相対的な大きさを表します。次の出力は、各サーベイ回答数をブロックチャートの形式で表したものです。

```
proc chart data=survey;
    block response / sumvar=count
    midpoints='Always' 'Usually'
              'Sometimes' 'Rarely' 'Never';
run;
```

アウトプット 10.3 *ブロックチャート*

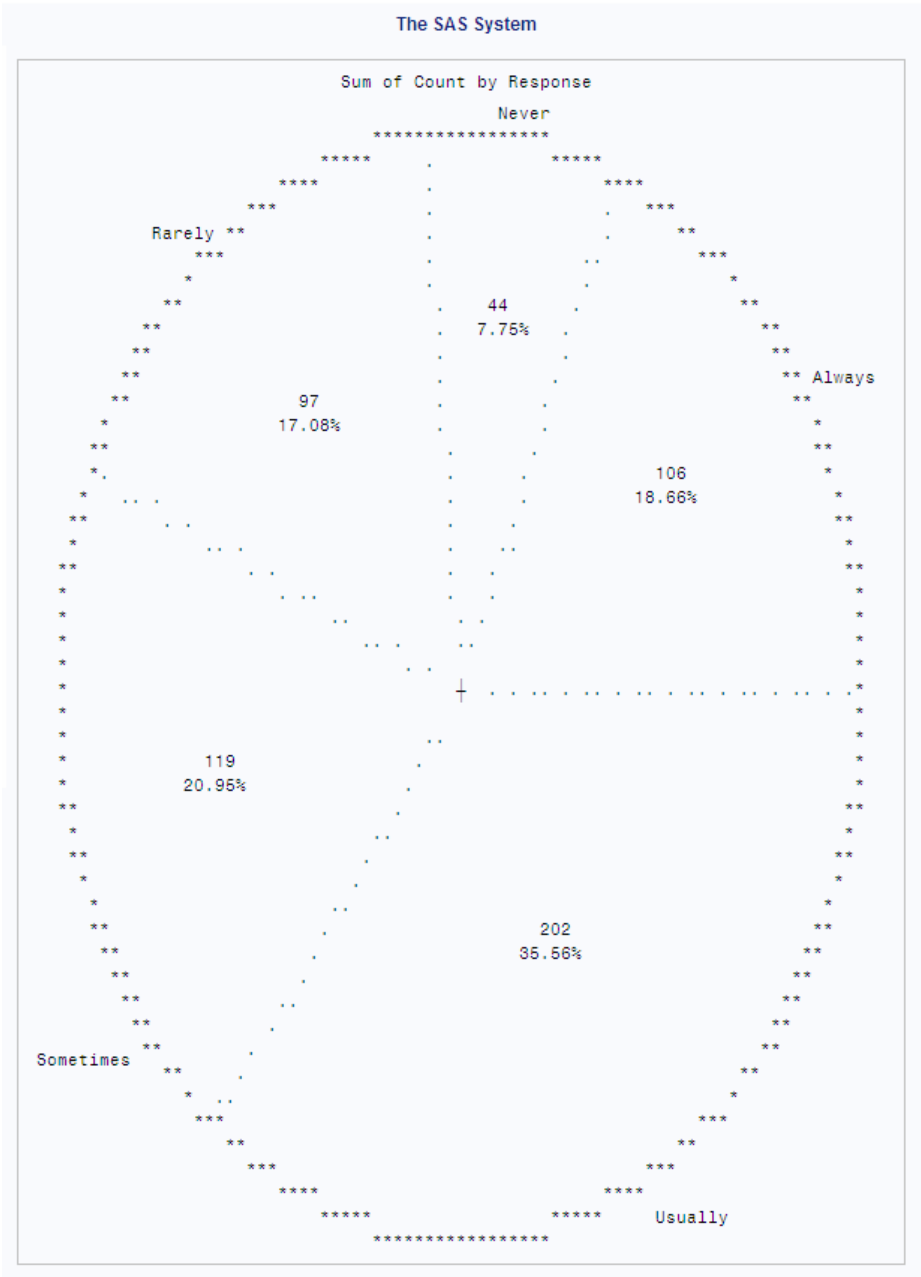


円グラフ

円グラフは、データを V 字のスライスとして表示することにより、すべてのカテゴリに対してそのカテゴリが占める割合を表します。スライスはそれぞれデータのカテゴリを表します。次の出力は、回答によって 5 つのスライスに分割されるサーベイ結果を表したものです。次のステートメントでは、出力が生成されます。

```
proc chart data=survey;
   pie response / sumvar=count;
run;
```

アウトプット 10.4 円グラフ



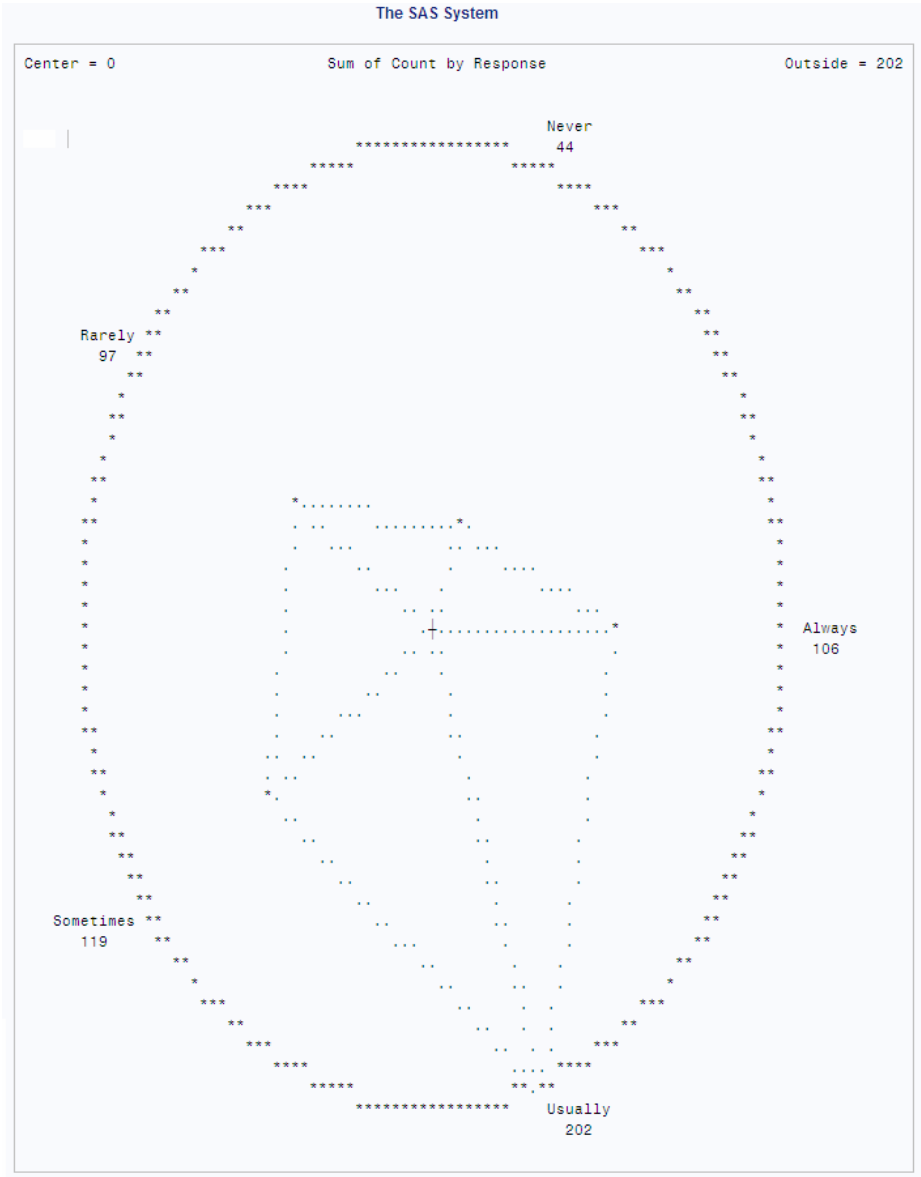
スターチャート

PROC CHART を使用して、グループ度数、合計、平均の値を表すスターチャートを生成できます。スターチャートは縦棒グラフに似ていますが、スターチャートのバーは、車輪のスポークのように中央点から放射状に広がっています。スターチャートは通常、月、日、時間ごとに取得される指標などの周期的なデータに使用されます。これは、カテゴリに固有の順序がある(“always”は“usually”よりも頻度が高く、“usually”は“sometimes”よりも頻度が高い)データにも使用されます。次の出力は、スターチャートに表示されるサーベイデータを表したものです。次のステートメントでは、出力が生成されます。

```
proc chart data=survey;
   star response / sumvar=count;
```

```
run;
```

アウトプット 10.5 *スターチャート*



概念:CHART プロシジャ

CHART プロシジャの変数の特性は、次のとおりです。

- 文字変数と出力形式の長さは 16 を超えることはできません。
- 連続した数値変数の場合、PROC CHART は自動的に表示間隔を選択します。ただし、間隔の中間点は定義可能です。
- 文字変数および、連続した範囲ではなく複数の重複しない値を含む不連続数値変数の場合、データ値自体がその間隔を定義します。

構文: CHART プロシジャ

要件 チャート生成ステートメントのうち少なくとも 1 つを使用する必要があります。

ヒント: ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

グローバルステートメントを使用することもできます。リストは、“グローバルステートメント” (24 ページ)および“Global Statements” (*SAS Statements: Reference*)を参照してください。

PROC CHART <*option(s)*>;
BLOCK *variable(s)* </ *option(s)*>;
BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* …>
<NOTSORTED>;
HBAR *variable(s)* </ *option(s)*>;
PIE *variable(s)* </ *option(s)*>;
STAR *variable(s)* </ *option(s)*>;
VBAR *variable(s)* </ *option(s)*>;

ステートメント	タスク	例
“PROC CHART ステートメント”	チャートを作成します	
“BLOCK ステートメント”	ブロックチャートを作成します	Ex. 6
“BY ステートメント”	BY グループごとに別のチャートを作成します	Ex. 6
“HBAR ステートメント”	横棒グラフを作成します	Ex. 5
“PIE ステートメント”	円グラフを作成します	
“STAR ステートメント”	スターチャートを作成します	
“VBAR ステートメント”	縦棒グラフを作成します	Ex. 1, Ex. 2, Ex. 3, Ex. 4

PROC CHART ステートメント

縦棒グラフおよび横棒グラフ、ブロックチャート、円グラフ、スターチャートを作成します。

構文

PROC CHART <*option(s)*>;

オプション引数

DATA=*SAS-data-set*
入力 SAS データセットを識別します。

制限事項 PROC CHART は、別のユーザーが同時にデータセットを更新している場合に同時アクセス権をサポートするエンジンと使用できません。

参照項目 “入力データセット” (25 ページ)

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
横軸と縦軸、参照線、およびチャートのその他の構造部分を構築するために使用する文字を定義します。また、出力にバー、ブロック、セクションを作成するために使用する記号を定義します。

position(s)
SAS フォーマット文字列における 1 つ以上の文字の位置を識別します。スペースまたはカンマで位置を区切ります。

デフォルト (*position(s)*)を省略すると、すべての可能な 20 の SAS フォーマット文字を指定することになります。

注 PROC CHART は、SAS が提供する 20 のフォーマット文字のうち 6 を使用します。表 10.1 (282 ページ)は、PROC CHART が使用するフォーマット文字を表示します。図 10.1 (283 ページ)は、PROC CHART で通常使用されるフォーマット文字の使用を示します。

formatting-character(s)
指定位置に使用する文字をリストします。PROC CHART は、*formatting-character(s)*の文字を表示されている順序で *position(s)*に割り当てます。たとえば、次のオプションではアスタリスク(*)を 2 番目のフォーマット文字に、番号記号(#)を 7 番目の文字に割り当てます。その他の文字は変更されません。

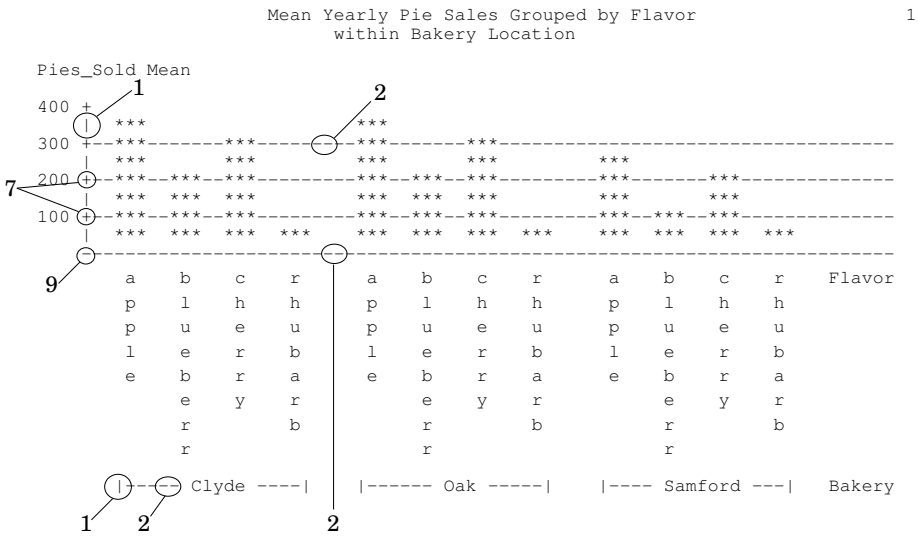
```
formchar(2,7)='*#'
```

表 10.1 *PROC CHART によって使用されるフォーマット文字*

Position	デフォルト	表示に使用
1	|	棒グラフの縦軸、ブロックチャートのブロックの側面、横棒グラフの参照線。並列棒グラフでは、最初と 2 番目のフォーマット文字がグループ変数の各値の辺り(チャートの上)に表示され、各グループの幅が示されます。
2	-	棒グラフの横軸、ブロックチャートのブロックを区切る水平線、縦棒グラフの参照線。並列棒グラフでは、最初と 2 番目のフォーマット文字がグループ変数の各値の辺り(チャートの上)に表示され、各グループの幅が示されます。
7	+	棒グラフの目盛、円グラフとスターチャートの中心。
9	-	棒グラフの軸のインターセクション
16	/	ブロックの終わりと、ブロックチャートのブロックを区切る対角線。

Position	デフォルト	表示に使用
20	*	円グラフとスターチャートの円

図 10.1 *PROC CHART 出力で通常使用されるフォーマット文字*



操作 SAS システムオプション FORMCHAR=では、デフォルトのフォーマット文字を指定します。システムオプションは、フォーマット文字の全体の文字列を定義します。プロシジャの FORMCHAR=オプションでは、選択した文字を再定義できます。

ヒント 16 進数文字を含む *formatting-characters* の文字を使用できます。16 進数文字を使用する場合、**x** を終了引用符の後に付ける必要があります。たとえば、次のオプションは 16 進数文字 2D を 2 番目のフォーマット文字に、16 進数文字 7C を 7 番目の文字に割り当てますが、その他の文字は変更しません。

```
formchar(2,7)='2D7C'x
```

参照項目 どの 16 進コードをどの文字に使用するかについては、ハードウェアのドキュメントを参照してください。

LPI=*value*
円グラフとスターチャートの割合を指定します。*value* は、次によって決定されます。

(lines per inch / columns per inch) * 10

たとえば、インチごとに 8 行、インチごとに 12 列のプリンタの場合は、LPI=6.6667 を指定します。

デフォルト 6

BLOCK ステートメント

ブロックチャートを作成します。

例: “例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

構文

BLOCK *variable(s)* </ *option(s)*>;

必須引数

variable(s)
PROC CHART がブロックチャートを変数ごとに 1 チャート作成する変数を指定します。

オプション引数

AXIS=*value-expression*
応答軸の値を指定します。*value-expression* は、それぞれスペースで区切られている個別値のリスト、または値の間の間隔が均等な範囲です。たとえば、次の範囲では、間隔 10 の 0 から 100 までの棒グラフの目盛を指定します。`hbar x / axis=0 to 100 by 10;`

制限事項 値は、個別に指定する場合でも、均等に間隔を空ける必要があります。

度数チャートの場合、値は整数である必要があります。

操作 ブロックチャートの場合、AXIS=は最も高いブロックのスケールを設定します。スケールを設定するため、PROC CHART は AXIS=リストの最大値を使用します。0 より大きい値がない場合、PROC CHART は AXIS=オプションを無視します。

AXIS=と BY ステートメントを使用すると、PROC CHART は BY グループに関して単一の軸を生成します。

注意 **value-expression の値は、データの範囲をオーバーライドします。** たとえば、データ範囲が 1 から 10 で、3 から 5 までの範囲を指定する場合、3 から 5 までの範囲のデータのみがチャートに表示されます。範囲外の値は、SAS ログの警告メッセージを作成します。

FREQ=*variable*
オブザベーションごとの度数カウントを表すデータセット変数を指定します。通常、各オブザベーションは、度数カウントに 1 つの値を提供します。FREQ=を使用して、各オブザベーションは FREQ=値のその値を提供します。

制限事項 FREQ=値が整数でない場合、PROC CHART によって切り捨てられます。

操作 SUMVAR=を使用する場合、PROC CHART は合計に FREQ=値を乗算します。

GROUP=*variable*
並列チャートが作成されます。各チャートは、GROUP=変数に対する共通値を持つオブザベーションを示します。GROUP=変数には文字または数値が可能で、非連続とみなされます。たとえば、次のステートメントは、各部門の男性と女性の度数棒グラフを作成します。

```
vbar gender / group=dept;
```

GROUP=変数の欠損値は、有効なレベルとして処理されます。

例 “例 4: 並列棒グラフの作成” (312 ページ)

“例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

“例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

G100
グループごとのパーセントの合計が 100 になるように指定します。デフォルトでは、PROC CHART は合計として 100%を使用します。たとえば、男性と女性を 3 つの年齢カテゴリに分ける棒グラフを作成する場合、デフォルトでは 6 つのバーの合計が 100%になります。ただし、G100 を使用すると、女性の 3 つのバーの合計が 100%になり、男性の 3 つのバーの合計が 100%になります。

操作 GROUP=を省略すると、PROC CHART は G100 を無視します。

LEVELS=*number-of-midpoints*
変数が連続している場合に各チャート変数を表すバーの数を指定します。

MIDPOINTS=*midpoint-specification* | OLD
バー、ブロック、セクションがそれぞれ表す値の範囲を範囲中間点を指定して定義します。

MIDPOINTS=の値は、次のうちいずれかになります。

midpoint-specification
中間点をそれぞれ別個に、または均等間隔の範囲にわたって指定します。たとえば、次のステートメントは、5 つのバーを含むチャートを作成します。最初のバーは、中間点が 10 の値範囲 X を表します。2 番目のバーは中間点が 20 の範囲、以降同様に表します。

```
vbar x / midpoints=10 20 30 40 50;
```

次に、文字変数に対する中間点指定の例を示します。

```
vbar x / midpoints='JAN' 'FEB' 'MAR';
```

次に、均等間隔の範囲にわたる中間点の指定例を示します。

```
vbar x / midpoints=10 to 100 by 5;
```

OLD
PROC CHART が SAS の前のバージョンで連続した変数の中間点の選択に使用したアルゴリズムを指定します。古いアルゴリズムは、Nelder (1976)に基づいていました。OLD を省略した場合に PROC CHART が使用する現在のアルゴリズムは、Terrell and Scott (1985)に基づいています。

デフォルト MIDPOINTS=が指定されていない場合、PROC CHART は値を SAS システムの通常の並べ替え順序で表示します。

MISSING
欠損値がチャート変数に有効なレベルになるように指定します。

NOHEADER
チャートの上部に印刷されるデフォルトのヘッダー行を非表示にします。

別名 NOHEADING

例 “例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

NOSYMBOL
サブグループ記号または凡例テーブルの印刷を行いません。

別名 NOLEGEND

操作 SUBGROUP=を省略すると、PROC CHART は NOSYMBOL を無視します。

SUBGROUP=*variable*
複数の文字を使用して、各バーまたはブロックを分割します。それぞれの文字数は、*variable* の値がバーまたはブロックに占める割合を表します。バーまたはブロックには、各値の最初の文字が、値が全体に占める分量だけ埋め込まれます。ただし、同じ文字で始まる値が複数ある場合は除きます。その場合、バーまたはブロックには、A、B、C などの文字が埋め込まれます。変数がフォーマットされると、PROC CHART はフォーマットされた値の最初の文字を使用します。

チャートで使用される文字、およびそれらが表す値は、チャートの下部の凡例に表示されます。サブグループ記号は、A から Z、0 から 9 まで、文字の昇順で並べ替えられます。

PROC CHART は、サブグループごとにそれぞれバーまたはブロックの高さを計算し、バー合計のパーセントの切り上げ、切り捨てを行います。そのため、バーの高さの合計は、SUBGROUP=オプションが使用されていない場合と同じバーよりも高い、または低い可能性があります。

操作 TYPE=MEAN と SUBGROUP=を両方使用する場合、PROC CHART は最初に SUMVAR=オプションでリストされる変数ごとに平均を計算します。次に、バーを各サブグループが占めるパーセントに分割します。

例 “例 3: バーをカテゴリに細分化する” (310 ページ)

SUMVAR=*variable*
PROC CHART が値または平均(TYPE=の値に依存)をチャートに表示する変数を指定します。

操作 SUMVAR=を使用し、TYPE=を MEAN、SUM 以外の値とともに使用する場合、TYPE=SUM は指定された TYPE=値をオーバーライドします。

ヒント HBAR と VBAR チャートの両方は、LABEL ステートメントを使用する場合、SUMVAR=変数のラベルを印刷できます。

例 “例 3: バーをカテゴリに細分化する” (310 ページ)

“例 4: 並列棒グラフの作成” (312 ページ)

“例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

“例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

SYMBOL=*character(s)*
SUBGROUP=オプションを使用しない場合、PROC CHART がチャートのバーまたはブロックで使用する文字を指定します。

デフォルト asterisk (*)

操作 SAS システムオプション OVP が有効で、印刷デバイスで重ね打ちがサポートされている場合、最大 3 文字を指定して重ね打ちされたチャートを作成できます。

例 “例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

TYPE=*statistic*
チャートのバーまたはセクションが何を表すかを指定します。*statistic* は、次のうちいずれかになります。

CFREQ
バー、ブロック、セクションがそれぞれ累積度数を表すように指定します。

CPERCENT
バー、ブロック、セクションがそれぞれ累積パーセントを表すように指定します。

別名 CPCT

FREQ
バー、ブロック、セクションがそれぞれ、データのチャート変数に対し値または範囲が発生する度数を表すように指定します。

MEAN
バー、ブロック、セクションがそれぞれ、そのバー、ブロック、セクションに属するすべてのオブザベーションにわたって SUMVAR=変数の平均を表すように指定します。

操作 TYPE=MEAN を使用して、MEAN および FREQ 統計量のみを計算できます。

例 “例 4: 並列棒グラフの作成” (312 ページ)

PERCENT
バー、ブロック、セクションがそれぞれ、指定値を含む、またはチャート変数の指定範囲内にあるオブザベーションのパーセントを表すように指定します。

別名 PCT

例 “例 2: パーセント棒グラフの作成” (307 ページ)

SUM
バー、ブロック、セクションがそれぞれ、各バー、ブロック、セクションに対応するオブザベーションに対する SUMVAR=変数の合計を表すように指定します。

デフォルト FREQ (SUM のデフォルトが発生する SUMVAR=を使用する場合を除く)

操作 TYPE=SUM を使用すると、SUM と FREQ 統計量のみ計算できます。

詳細

ステートメント結果
ブロックチャートはそれぞれ 1 出力ページに適合する必要があるため、BLOCK 変数、および GROUP=オプションで指定される変数に対しグラフ表示される値が多い場合は、SAS システムオプション LINESIZE=と PAGESIZE=を調整する必要があることがあります。

次の表に、66 行のページに適合可能な選択した LINESIZE= (LS=)指定に対する BLOCK 変数のグラフ表示される値の最大数を示します。

表 10.2 *BLOCK 変数のバーの最大数*

GROUP=値	LS= 132	LS= 120	LS= 105	LS= 90	LS= 76	LS= 64
0,1	9	8	7	6	5	4
2	8	8	7	6	5	4
3	8	7	6	5	4	3
4	7	7	6	5	4	3
5,6	7	6	5	4	3	2

GROUP=レベルの値が 3 文字を超える場合、適合可能な BLOCK 変数のグラフ表示される値の最大数が 1 減少することがあります。BLOCK レベル値は、12 文字に切り捨てます。この制限を超えると、PROC CHART はかわりに横棒グラフを作成します。

BY ステートメント

BY グループごとに個別のチャートを作成します。

参照項目: “BY” (68 ページ)

例: “例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* …>
<NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは指定するすべての変数別に並べ替えるか、適切にインデックス付けする必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
オブザベーションが BY ステートメントの文字 DESCENDING の直後に続く変数で降順に並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。オブザベーションは別の方法(時系列など)でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定した場合は、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

HBAR ステートメント

横棒グラフを作成します。

ヒント: HBAR チャートは、チャート変数の名前またはラベルのいずれかを印刷できます。

参照項目: “例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

構文

HBAR *variable(s)* </ *option(s)*>;

必須引数

variable(s)
PROC CHART が横棒グラフを変数ごとに 1 チャート作成する変数を指定します。

オプション引数

ASCENDING
バーと関連する統計量をグループ内のサイズの昇順で印刷します。

別名 ASC

AXIS=*value-expression*
応答軸の値を指定します。*value-expression* は、それぞれスペースで区切られている個別値のリスト、または値の間の間隔が均等な範囲です。たとえば、次の範囲では、間隔 10 の 0 から 100 までの棒グラフの目盛を指定します。`hbar x / axis=0 to 100 by 10;`

制限事項 値は、個別に指定する場合でも、均等に間隔を空ける必要があります。

度数チャートの場合、値は整数である必要があります。

操作 HBAR チャートと VBAR チャートの場合、AXIS=は応答軸の目盛を決定します。AXIS=指定に値が 1 つのみ含まれている場合、0 未満の値が最小目盛を決定する、または 0 より大きい値が最大目盛を決定します。

AXIS=と BY ステートメントを使用すると、PROC CHART は BY グループに関して単一の軸を生成します。

注意 **value-expression の値は、データの範囲をオーバーライドします。**たとえば、データ範囲が 1 から 10 で、3 から 5 までの範囲を指定する場合、3 から 5

までの範囲のデータのみがチャートに表示されます。範囲外の値は、SAS ログの警告メッセージを作成します。

CFREQ
累積度数を印刷します。

制限事項 HBAR ステートメントでのみ利用可能です

CPERCENT
累積パーセントを印刷します。

制限事項 HBAR ステートメントでのみ利用可能です

DISCRETE
数値チャート変数が連続ではなく非連続になるように指定します。DISCRETE がない場合、PROC CHART は MIDPOINTS=または LEVELS=を使用する場合を除いて、すべての数値変数が連続であるとみなし、それらの間隔を自動的に選択します。

FREQ
チャートの側面に対する各バーの度数を印刷します。

制限事項 HBAR ステートメントでのみ利用可能です

FREQ=*variable*
オブザベーションごとの度数カウントを表すデータセット変数を指定します。通常、各オブザベーションは、度数カウントに 1 つの値を提供します。FREQ=を使用して、各オブザベーションは FREQ=値のその値を提供します。

制限事項 FREQ=値が整数でない場合、PROC CHART によって切り捨てられます。

操作 SUMVAR=を使用する場合、PROC CHART は合計に FREQ=値を乗算します。

GROUP=*variable*
並列チャートが作成されます。各チャートは、GROUP=変数に対する共通値を持つオブザベーションを示します。GROUP=変数には文字または数値が可能で、非連続とみなされます。たとえば、次のステートメントは、各部門の男性と女性の度数棒グラフを作成します。

```
vbar gender / group=dept;
```

GROUP=変数の欠損値は、有効なレベルとして処理されます。

例 “例 4: 並列棒グラフの作成” (312 ページ)

“例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

“例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

GSPACE=*n*
バーのグループ間の間隔を指定します。GSPACE=0 を使用すると、バーの隣接グループ間の間隔はなくなります。

操作 GROUP=を省略すると、PROC CHART は GSPACE=を無視します

G100
グループごとのパーセントの合計が 100 になるように指定します。デフォルトでは、PROC CHART は合計として 100%を使用します。たとえば、男性と女性を 3 つの年齢カテゴリに分ける棒グラフを作成する場合、デフォルトでは 6 つのバーの合計が 100%になります。ただし、G100 を使用すると、女性の 3 つのバーの合計が 100%になり、男性の 3 つのバーの合計が 100%になります。

操作 GROUP=を省略すると、PROC CHART は G100 を無視します。

LEVELS=*number-of-midpoints*
変数が連続している場合に各チャート変数を表すバーの数を指定します。

MEAN
各バーによって表されるオブザベーションの平均を印刷します。

制限事項 SUMVAR=と TYPE=の使用時にのみ利用可能です

TYPE=CFREQ、CPERCENT、FREQ、PERCENT の場合は利用できません

MISSING
欠損値がチャート変数に有効なレベルになるように指定します。

NOSTATS
横棒グラフの統計量を非表示にします。

別名 NOSTAT

NOSYMBOL
サブグループ記号または凡例テーブルの印刷を行いません。

別名 NOLEGEND

操作 SUBGROUP=を省略すると、PROC CHART は NOSYMBOL を無視します。

NOZEROS
度数がゼロのバーを非表示にします。

PERCENT
チャート変数に対する指定値を持つオブザベーションのパーセントを印刷します。

REF=*value(s)*
指定した位置の応答軸に参照線を描きます。

ヒント REF=値は、TYPE=統計量の値に対応していなければなりません。

例 “例 4: 並列棒グラフの作成” (312 ページ)

SPACE=*n*
個別のバーの間の間隔を指定します。

ヒント SPACE=0 を使用して、隣接したバーの間の間隔をなくします。

GSPACE=オプションを使用して、各グループ内のバーの間の間隔を指定します。

SUBGROUP=*variable*
複数の文字を使用して、各バーまたはブロックを分割します。それぞれの文字数は、*variable* の値がバーまたはブロックに占める割合を表します。バーまたはブロ

ックには、各値の最初の文字が、値が全体に占める分量だけ埋め込まれます。ただし、同じ文字で始まる値が複数ある場合は除きます。その場合、バーまたはブロックには、A、B、C などの文字が埋め込まれます。変数がフォーマットされると、PROC CHART はフォーマットされた値の最初の文字を使用します。

チャートで使用される文字、およびそれらが表す値は、チャートの下部の凡例に表示されます。サブグループ記号は、A から Z、0 から 9 まで、文字の昇順で並べ替えられます。

PROC CHART は、サブグループごとにそれぞれバーまたはブロックの高さを計算し、バー合計のパーセントの切り上げ、切り捨てを行います。そのため、バーの高さの合計は、SUBGROUP=オプションが使用されていない場合と同じバーよりも高い、または低い可能性があります。

操作 TYPE=MEAN と SUBGROUP=を両方使用する場合、PROC CHART は最初に SUMVAR=オプションでリストされる変数ごとに平均を計算します。次に、バーを各サブグループが占めるパーセントに分割します。

例 “例 3: バーをカテゴリに細分化する” (310 ページ)

SUM
各バーが表すオブザベーションの合計数を印刷します。

制限事項 SUMVAR=と TYPE=の使用時にのみ利用可能です

TYPE=CFREQ、CPERCENT、FREQ、PERCENT の場合は利用できません

SUMVAR=*variable*
PROC CHART が値または平均(TYPE=の値に依存)をチャートに表示する変数を指定します。

操作 SUMVAR=を使用し、TYPE=を MEAN、SUM 以外の値とともに使用する場合、TYPE=SUM は指定された TYPE=値をオーバーライドします。

ヒント HBAR チャートは、LABEL ステートメントを使用する場合、SUMVAR=変数のラベルを印刷できます。

例 “例 3: バーをカテゴリに細分化する” (310 ページ)

“例 4: 並列棒グラフの作成” (312 ページ)

“例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

“例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

SYMBOL=*character(s)*
SUBGROUP=オプションを使用しない場合、PROC CHART がチャートのバーまたはブロックで使用する文字を指定します。

デフォルト asterisk (*)

操作 SAS システムオプション OVP が有効で、印刷デバイスで重ね打ちがサポートされている場合、最大 3 文字を指定して重ね打ちされたチャートを作成できます。

例 “例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

TYPE=*statistic*
チャートのバーまたはセクションが何を表すかを指定します。*statistic* は、次のうちいずれかになります。

CFREQ
バー、ブロック、セクションがそれぞれ累積度数を表すように指定します。

CPERCENT
バー、ブロック、セクションがそれぞれ累積パーセントを表すように指定します。

別名 CPCT

FREQ
バー、ブロック、セクションがそれぞれ、データのチャート変数に対し値または範囲が発生する度数を表すように指定します。

MEAN
バー、ブロック、セクションがそれぞれ、そのバー、ブロック、セクションに属するすべてのオブザベーションにわたって SUMVAR=変数の平均を表すように指定します。

操作 TYPE=MEAN を使用して、MEAN および FREQ 統計量のみを計算できます。

例 “例 4: 並列棒グラフの作成” (312 ページ)

PERCENT
バー、ブロック、セクションがそれぞれ、指定値を含む、またはチャート変数の指定範囲内にあるオブザベーションのパーセントを表すように指定します。

別名 PCT

例 “例 2: パーセント棒グラフの作成” (307 ページ)

SUM
バー、ブロック、セクションがそれぞれ、各バー、ブロック、セクションに対応するオブザベーションに対する SUMVAR=変数の合計を表すように指定します。

デフォルト FREQ (SUM のデフォルトが発生する SUMVAR=を使用する場合を除く)

操作 TYPE=SUM を使用すると、SUM と FREQ 統計量のみ計算できます。

WIDTH=*n*
棒グラフのバーの幅を指定します。

詳細

ステートメント結果
バーの数によって各チャートが 1 ページ以上の出力になる場合、各バーはデフォルトで 1 行を占めます。

デフォルトで、TYPE=FREQ、CFREQ、PCT または CPCT の横棒グラフの場合、PROC CHART は、度数、累積度数、パーセント、累積パーセントの統計量を印刷します。1 つ以上の統計量オプションを使用する場合、PROC CHART は要求される統計量および度数のみ印刷します。

PIE ステートメント

円グラフを作成します。

構文

PIE *variable(s)* </ *option(s)*>;

必須引数

variable(s)
PROC CHART が円グラフを変数ごとに 1 チャート作成する変数を指定します。

オプション引数

FREQ=*variable*
オブザベーションごとの度数カウントを表すデータセット変数を指定します。通常、各オブザベーションは、度数カウントに 1 つの値を提供します。FREQ=を使用して、各オブザベーションは FREQ=値のその値を提供します。

制限事項 FREQ=値が整数でない場合、PROC CHART によって切り捨てられます。

操作 SUMVAR=を使用する場合、PROC CHART は合計に FREQ=値を乗算します。

LEVELS=*number-of-midpoints*
変数が連続している場合に各チャート変数を表すバーの数を指定します。

MIDPOINTS=*midpoint-specification* | OLD
バー、ブロック、セクションがそれぞれ表す値の範囲を範囲中間点を指定して定義します。

MIDPOINTS=の値は、次のうちいずれかになります。

midpoint-specification
中間点をそれぞれ別個に、または均等間隔の範囲にわたって指定します。たとえば、次のステートメントは、5 つのバーを含むチャートを作成します。最初のバーは、中間点が 10 の値範囲 X を表します。2 番目のバーは中間点が 20 の範囲、以降同様に表します。

```
vbar x / midpoints=10 20 30 40 50;
```

次に、文字変数に対する中間点指定の例を示します。

```
vbar x / midpoints='JAN' 'FEB' 'MAR';
```

次に、均等間隔の範囲にわたる中間点の指定例を示します。

```
vbar x / midpoints=10 to 100 by 5;
```

OLD
PROC CHART が SAS の前のバージョンで連続した変数の中間点の選択に使用したアルゴリズムを指定します。古いアルゴリズムは、Nelder (1976)に基づいていました。OLD を省略した場合に PROC CHART が使用する現在のアルゴリズムは、Terrell and Scott (1985)に基づいています。

デフォルト MIDPOINTS=が指定されていない場合、PROC CHART は値を SAS システムの通常の並べ替え順序で表示します。

MISSING
欠損値がチャート変数に有効なレベルになるように指定します。

NOHEADER
チャートの上部に印刷されるデフォルトのヘッダー行を非表示にします。

別名 NOHEADING

例 “例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

SUMVAR=*variable*
PROC CHART が値または平均(TYPE=の値に依存)をチャートに表示する変数を指定します。

操作 SUMVAR=を使用し、TYPE=を MEAN、SUM 以外の値とともに使用する場合、TYPE=SUM は指定された TYPE=値をオーバーライドします。

ヒント HBAR と VBAR チャートの両方は、LABEL ステートメントを使用する場合、SUMVAR=変数のラベルを印刷できます。

例 “例 3: バーをカテゴリに細分化する” (310 ページ)

“例 4: 並列棒グラフの作成” (312 ページ)

“例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

“例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

TYPE=*statistic*
チャートのバーまたはセクションが何を表すかを指定します。*statistic* は、次のうちいずれかになります。

CFREQ
バー、ブロック、セクションがそれぞれ累積度数を表すように指定します。

CPERCENT
バー、ブロック、セクションがそれぞれ累積パーセントを表すように指定します。

別名 CPCT

FREQ
バー、ブロック、セクションがそれぞれ、データのチャート変数に対し値または範囲が発生する度数を表すように指定します。

MEAN
バー、ブロック、セクションがそれぞれ、そのバー、ブロック、セクションに属するすべてのオブザベーションにわたって SUMVAR=変数の平均を表すように指定します。

操作 TYPE=MEAN を使用して、MEAN および FREQ 統計量のみを計算できます。

例 “例 4: 並列棒グラフの作成” (312 ページ)

PERCENT
バー、ブロック、セクションがそれぞれ、指定値を含む、またはチャート変数の指定範囲内にあるオブザベーションのパーセントを表すように指定します。

別名 PCT

例 “例 2: パーセント棒グラフの作成” (307 ページ)

SUM
バー、ブロック、セクションがそれぞれ、各バー、ブロック、セクションに対応するオブザベーションに対する SUMVAR=変数の合計を表すように指定します。

デフォルト FREQ (SUM のデフォルトが発生する SUMVAR=を使用する場合を除く)

操作 TYPE=SUM を使用すると、SUM と FREQ 統計量のみ計算できます。

詳細

ステートメント結果

PROC CHART は、縦棒グラフのバーの数と同じ方法で円グラフのスライス数を決定します。3 つ未満の印刷位置から成る円グラフのスライスは、まとめて"OTHER"カテゴリにグループ化されます。

円グラフのサイズは、SAS システムオプション LINESIZE=と PAGESIZE=のみによって決定されます。デフォルトで、使用するプリンタがインチごとに 6 行、インチごとに 10 列を印刷しない場合、円グラフは楕円形になります。インチごとに 6 行、10 列を印刷しないプリンタで円グラフを作成するには、LPI=オプションを PROC CHART ステートメントで使用します。プリンタに適切な LPI=値を指定する式については、“LPI=*value*” (283 ページ) の説明を参照してください。

レベルが 50 を超える変数に対し円グラフを作成する場合、PROC CHART はかわりに横棒グラフを作成します。

STAR ステートメント

スターチャートを作成します。

構文

STAR *variable(s)* </ *option(s)*>;

必須引数

variable(s)
PROC CHART がスターチャートを変数ごとに 1 チャート作成する変数を指定します。

オプション引数

AXIS=*value-expression*
応答軸の値を指定します。*value-expression* は、それぞれスペースで区切られている個別値のリスト、または値の間の間隔が均等な範囲です。たとえば、次の範囲

では、間隔 10 の 0 から 100 までの棒グラフの目盛を指定します。**hbar x / axis=0 to 100 by 10;**

制限事項 値は、個別に指定する場合でも、均等に間隔を空ける必要があります。

度数チャートの場合、値は整数である必要があります。

操作 スターチャートの場合、単一の AXIS=値は値が 0 未満の場合は最小(チャートの中心)を設定し、値が 0 より大きい場合は最大(円の外)を設定します。AXIS=指定に複数の値が含まれる場合、PROC CHART はリストの最小値と最大値を使用します。

AXIS=と BY ステートメントを使用すると、PROC CHART は BY グループに関して単一の軸を生成します。

注意 **value-expression の値は、データの範囲をオーバーライドします。**たとえば、データ範囲が 1 から 10 で、3 から 5 までの範囲を指定する場合、3 から 5 までの範囲のデータのみがチャートに表示されます。範囲外の値は、SAS ログの警告メッセージを作成します。

FREQ=*variable*
オブザベーションごとの度数カウントを表すデータセット変数を指定します。通常、各オブザベーションは、度数カウントに 1 つの値を提供します。FREQ=を使用して、各オブザベーションは FREQ=値のその値を提供します。

制限事項 FREQ=値が整数でない場合、PROC CHART によって切り捨てられます。

操作 SUMVAR=を使用する場合、PROC CHART は合計に FREQ=値を乗算します。

LEVELS=*number-of-midpoints*
変数が連続している場合に各チャート変数を表すバーの数を指定します。

MIDPOINTS=*midpoint-specification* | OLD
バー、ブロック、セクションがそれぞれ表す値の範囲を範囲中間点を指定して定義します。

MIDPOINTS=の値は、次のうちいずれかになります。

midpoint-specification
中間点をそれぞれ別個に、または均等間隔の範囲にわたって指定します。たとえば、次のステートメントは、5 つのバーを含むチャートを作成します。最初のバーは、中間点が 10 の値範囲 X を表します。2 番目のバーは中間点が 20 の範囲、以降同様に表します。

```
vbar x / midpoints=10 20 30 40 50;
```

次に、文字変数に対する中間点指定の例を示します。

```
vbar x / midpoints='JAN' 'FEB' 'MAR';
```

次に、均等間隔の範囲にわたる中間点の指定例を示します。

```
vbar x / midpoints=10 to 100 by 5;
```

OLD
PROC CHART が SAS の前のバージョンで連続した変数の中間点の選択に使用したアルゴリズムを指定します。古いアルゴリズムは、Nelder (1976)に基づ

いていました。OLD を省略した場合に PROC CHART が使用する現在のアルゴリズムは、Terrell and Scott (1985)に基づいています。

デフォルト MIDPOINTS=が指定されていない場合、PROC CHART は値を SAS システムの通常の並べ替え順序で表示します。

MISSING
欠損値がチャート変数に有効なレベルになるように指定します。

NOHEADER
チャートの上部に印刷されるデフォルトのヘッダー行を非表示にします。

別名 NOHEADING

例 “例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

SUMVAR=*variable*
PROC CHART が値または平均(TYPE=の値に依存)をチャートに表示する変数を指定します。

操作 SUMVAR=を使用し、TYPE=を MEAN、SUM 以外の値とともに使用する場合、TYPE=SUM は指定された TYPE=値をオーバーライドします。

ヒント HBAR と VBAR チャートの両方は、LABEL ステートメントを使用する場合、SUMVAR=変数のラベルを印刷できます。

例 “例 3: バーをカテゴリに細分化する” (310 ページ)

“例 4: 並列棒グラフの作成” (312 ページ)

“例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

“例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

TYPE=*statistic*
チャートのバーまたはセクションが何を表すかを指定します。*statistic* は、次のうちいずれかになります。

CFREQ
バー、ブロック、セクションがそれぞれ累積度数を表すように指定します。

CPERCENT
バー、ブロック、セクションがそれぞれ累積パーセントを表すように指定します。

別名 CPCT

FREQ
バー、ブロック、セクションがそれぞれ、データのチャート変数に対し値または範囲が発生する度数を表すように指定します。

MEAN
バー、ブロック、セクションがそれぞれ、そのバー、ブロック、セクションに属するすべてのオブザベーションにわたって SUMVAR=変数の平均を表すように指定します。

操作 TYPE=MEAN を使用して、MEAN および FREQ 統計量のみを計算できます。

例 “例 4: 並列棒グラフの作成” (312 ページ)

PERCENT
バー、ブロック、セクションがそれぞれ、指定値を含む、またはチャート変数の指定範囲内にあるオブザベーションのパーセントを表すように指定します。

別名 PCT

例 “例 2: パーセント棒グラフの作成” (307 ページ)

SUM
バー、ブロック、セクションがそれぞれ、各バー、ブロック、セクションに対応するオブザベーションに対する SUMVAR=変数の合計を表すように指定します。

デフォルト FREQ (SUM のデフォルトが発生する SUMVAR=を使用する場合を除く)

操作 TYPE=SUM を使用すると、SUM と FREQ 統計量のみ計算できます。

詳細

ステートメント結果

スターの点の数は、横棒グラフのバーの数と同じように決定されます。

すべてのデータ値が正の場合、スターの中心はゼロを表し、円の外は最大値を表します。データ値が負の場合、中心は最小値を表します。最大値と最小値の指定方法については、AXIS=*value expression* の説明を参照してください。チャートの割合の指定方法については、次の説明を参照してください。“LPI=*value*” (283 ページ)

レベルが 24 を超える変数に対してスターチャートを作成する場合、PROC CHART はかわりに横棒グラフを作成します。

VBAR ステートメント

縦棒グラフを作成します。

例:
“例 1: 単純な度数カウントの作成” (305 ページ)
“例 2: パーセント棒グラフの作成” (307 ページ)
“例 3: バーをカテゴリに細分化する” (310 ページ)
“例 4: 並列棒グラフの作成” (312 ページ)

構文

VBAR *variable(s)* </ *option(s)*>;

必須引数

variable(s)
PROC CHART が縦棒グラフを変数ごとに 1 チャート作成する変数を指定します。

オプション引数

ASCENDING
バーと関連する統計量をグループ内のサイズの昇順で印刷します。

別名 ASC

AXIS=*value-expression*
応答軸の値を指定します。*value-expression* は、それぞれスペースで区切られている個別値のリスト、または値の間の間隔が均等な範囲です。たとえば、次の範囲では、間隔 10 の 0 から 100 までの棒グラフの目盛を指定します。`hbar x / axis=0 to 100 by 10;`

制限事項 値は、個別に指定する場合でも、均等に間隔を空ける必要があります。

度数チャートの場合、値は整数である必要があります。

操作 HBAR チャートと VBAR チャートの場合、AXIS=は応答軸の目盛を決定します。AXIS=指定に値が 1 つのみ含まれている場合、0 未満の値が最小目盛を決定する、または 0 より大きい値が最大目盛を決定します。

AXIS=と BY ステートメントを使用すると、PROC CHART は BY グループに関して単一の軸を生成します。

注意 **value-expression の値は、データの範囲をオーバーライドします。**たとえば、データ範囲が 1 から 10 で、3 から 5 までの範囲を指定する場合、3 から 5 までの範囲のデータのみがチャートに表示されます。範囲外の値は、SAS ログの警告メッセージを作成します。

DISCRETE
数値チャート変数が連続ではなく非連続になるように指定します。DISCRETE がない場合、PROC CHART は MIDPOINTS=または LEVELS=を使用する場合を除いて、すべての数値変数が連続であるとみなし、それらの間隔を自動的に選択します。

FREQ=*variable*
オブザベーションごとの度数カウントを表すデータセット変数を指定します。通常、各オブザベーションは、度数カウントに 1 つの値を提供します。FREQ=を使用して、各オブザベーションは FREQ=値のその値を提供します。

制限事項 FREQ=値が整数でない場合、PROC CHART によって切り捨てられます。

操作 SUMVAR=を使用する場合、PROC CHART は合計に FREQ=値を乗算します。

GROUP=*variable*
並列チャートが作成されます。各チャートは、GROUP=変数に対する共通値を持つオブザベーションを示します。GROUP=変数には文字または数値が可能で、非連続とみなされます。たとえば、次のステートメントは、各部門の男性と女性の度数棒グラフを作成します。

```
vbar gender / group=dept;
```

GROUP=変数の欠損値は、有効なレベルとして処理されます。

例 “例 4: 並列棒グラフの作成” (312 ページ)

“例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

“例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

GSPACE=*n*
バーのグループ間の間隔を指定します。GSPACE=0 を使用すると、バーの隣接グループ間の間隔はなくなります。

操作 GROUP=を省略すると、PROC CHART は GSPACE=を無視します

G100
グループごとのパーセントの合計が 100 になるように指定します。デフォルトでは、PROC CHART は合計として 100%を使用します。たとえば、男性と女性を 3 つの年齢カテゴリに分ける棒グラフを作成する場合、デフォルトでは 6 つのバーの合計が 100%になります。ただし、G100 を使用すると、女性の 3 つのバーの合計が 100%になり、男性の 3 つのバーの合計が 100%になります。

操作 GROUP=を省略すると、PROC CHART は G100 を無視します。

LEVELS=*number-of-midpoints*
変数が連続している場合に各チャート変数を表すバーの数を指定します。

MIDPOINTS=*midpoint-specification* | OLD
バー、ブロック、セクションがそれぞれ表す値の範囲を範囲中間点を指定して定義します。

MIDPOINTS=の値は、次のうちいずれかになります。

midpoint-specification
中間点をそれぞれ別個に、または均等間隔の範囲にわたって指定します。たとえば、次のステートメントは、5 つのバーを含むチャートを作成します。最初のバーは、中間点が 10 の値範囲 X を表します。2 番目のバーは中間点が 20 の範囲、以降同様に表します。

```
vbar x / midpoints=10 20 30 40 50;
```

次に、文字変数に対する中間点指定の例を示します。

```
vbar x / midpoints='JAN' 'FEB' 'MAR';
```

次に、均等間隔の範囲にわたる中間点の指定例を示します。

```
vbar x / midpoints=10 to 100 by 5;
```

OLD
PROC CHART が SAS の前のバージョンで連続した変数の中間点の選択に使用したアルゴリズムを指定します。古いアルゴリズムは、Nelder (1976)に基づいていました。OLD を省略した場合に PROC CHART が使用する現在のアルゴリズムは、Terrell and Scott (1985)に基づいています。

デフォルト MIDPOINTS=が指定されていない場合、PROC CHART は値を SAS システムの通常の並べ替え順序で表示します。

制限事項 VBAR 変数が数値の場合、中間点は昇順で指定する必要があります。

MISSING
欠損値がチャート変数に有効なレベルになるように指定します。

NOSYMBOL
サブグループ記号または凡例テーブルの印刷を行いません。

別名 NOLEGEND

操作 SUBGROUP=を省略すると、PROC CHART は NOSYMBOL を無視します。

NOZEROS
度数がゼロのバーを非表示にします。

REF=*value(s)*
指定した位置の応答軸に参照線を描きます。

ヒント REF=値は、TYPE=統計量の値に対応していなければなりません。

例 “例 4: 並列棒グラフの作成” (312 ページ)

SPACE=*n*
個別のバーの間の間隔を指定します。

ヒント SPACE=0 を使用して、隣接したバーの間の間隔をなくします。

GSPACE=オプションを使用して、各グループ内のバーの間の間隔を指定します。

SUBGROUP=*variable*
複数の文字を使用して、各バーまたはブロックを分割します。それぞれの文字数は、*variable* の値がバーまたはブロックに占める割合を表します。バーまたはブロックには、各値の最初の文字が、値が全体に占める分量だけ埋め込まれます。ただし、同じ文字で始まる値が複数ある場合は除きます。その場合、バーまたはブロックには、A、B、C などの文字が埋め込まれます。変数がフォーマットされると、PROC CHART はフォーマットされた値の最初の文字を使用します。

チャートで使用される文字、およびそれらが表す値は、チャートの下部の凡例に表示されます。サブグループ記号は、A から Z、0 から 9 まで、文字の昇順で並べ替えられます。

PROC CHART は、サブグループごとにそれぞれバーまたはブロックの高さを計算し、バー合計のパーセントの切り上げ、切り捨てを行います。そのため、バーの高さの合計は、SUBGROUP=オプションが使用されていない場合と同じバーよりも高い、または低い可能性があります。

操作 TYPE=MEAN と SUBGROUP=を両方使用する場合、PROC CHART は最初に SUMVAR=オプションでリストされる変数ごとに平均を計算します。次に、バーを各サブグループが占めるパーセントに分割します。

例 “例 3: バーをカテゴリに細分化する” (310 ページ)

SUMVAR=*variable*
PROC CHART が値または平均(TYPE=の値に依存)をチャートに表示する変数を指定します。

操作 SUMVAR=を使用し、TYPE=を MEAN、SUM 以外の値とともに使用する場合、TYPE=SUM は指定された TYPE=値をオーバーライドします。

ヒント VBAR チャートは、LABEL ステートメントを使用する場合、SUMVAR=変数のラベルを印刷できます。

例 “例 3: バーをカテゴリに細分化する” (310 ページ)

“例 4: 並列棒グラフの作成” (312 ページ)

“例 5: データサブセットごとに横棒グラフを作成する” (315 ページ)

“例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

SYMBOL=*character(s)*
SUBGROUP=オプションを使用しない場合、PROC CHART がチャートのバーまたはブロックで使用する文字を指定します。

デフォルト asterisk (*)

操作 SAS システムオプション OVP が有効で、印刷デバイスで重ね打ちがサポートされている場合、最大 3 文字を指定して重ね打ちされたチャートを作成できます。

例 “例 6: BY グループごとにブロックチャートを作成する” (316 ページ)

TYPE=*statistic*
チャートのバーまたはセクションが何を表すかを指定します。*statistic* は、次のうちいずれかになります。

CFREQ
バー、ブロック、セクションがそれぞれ累積度数を表すように指定します。

CPERCENT
バー、ブロック、セクションがそれぞれ累積パーセントを表すように指定します。

別名 CPCT

FREQ
バー、ブロック、セクションがそれぞれ、データのチャート変数に対し値または範囲が発生する度数を表すように指定します。

MEAN
バー、ブロック、セクションがそれぞれ、そのバー、ブロック、セクションに属するすべてのオブザベーションにわたって SUMVAR=変数の平均を表すように指定します。

操作 TYPE=MEAN を使用して、MEAN および FREQ 統計量のみを計算できます。

例 “例 4: 並列棒グラフの作成” (312 ページ)

PERCENT
バー、ブロック、セクションがそれぞれ、指定値を含む、またはチャート変数の指定範囲内にあるオブザベーションのパーセントを表すように指定します。

別名 PCT

例 “例 2: パーセント棒グラフの作成” (307 ページ)

SUM
バー、ブロック、セクションがそれぞれ、各バー、ブロック、セクションに対応するオブザベーションに対する SUMVAR=変数の合計を表すように指定します。

デフォルト FREQ (SUM のデフォルトが発生する SUMVAR=を使用する場合を除く)

操作	TYPE=SUM を使用すると、SUM と FREQ 統計量のみ計算できます。

WIDTH=*n*
棒グラフのバーの幅を指定します。

詳細

ステートメント結果

PROC CHART は、チャートごとに 1 ページ印刷します。PROC CHART は縦軸に沿って、チャート度数、累積度数、チャートパーセント、累積パーセント、合計、または平均を表します。各バーの下に、PROC CHART は TYPE=オプションの値(指定されている場合)に従って値を印刷します。文字変数または個別の数値変数の場合、この値はバーによって表される実績値になります。連続した数値変数の場合、値はバーによって表される間隔の中間点を表します。

PROC CHART は、縦軸の自動調整、バー幅の決定、バー間の間隔の選択を行うことができます。ただし、オプションを選択して、バー間隔とバー数の選択、欠損値をチャートに含める、並列チャートの作成、バーの分割を実行できます。行ごとの文字数(LINESIZE=)がすべての縦棒グラフの表示に十分でない場合、PROC CHART はかわりに横棒グラフを作成します。

結果:CHART プロシジャ

欠損値

PROC CHART は、欠損値の処理時に次のルールに従います。

- MISSING オプションを使用する場合、欠損値はチャート変数に有効なレベルとみなされます。
- GROUP=または SUBGROUP=変数の欠損値は、有効なレベルとして処理されます。
- PROC CHART は、FREQ=オプションと SUMVAR=オプションの欠損値を無視します。
- FREQ=変数の値が欠損値、ゼロ、または負である場合、オブザベーションはチャート変数の計算から除外されます。
- SUMVAR=変数の値が欠損値の場合、オブザベーションはチャート変数の計算から除外されます。

ODS テーブル名

CHART プロシジャは、作成する各テーブルに名前を割り当てます。これらの名前を使用して、Output Delivery System (ODS)を使用してテーブルを選択し、出力データセットを作成する際にそのテーブルを参照できます。詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

表 10.3 CHART プロシジャによって生成される ODS テーブル

Name	説明	使用されるステートメント
BLOCK	ブロックチャート	BLOCK
HBAR	横棒グラフ	HBAR
PIE	円グラフ	PIE
STAR	スターチャート	STAR
VBAR	縦棒グラフ	VBAR

PROC CHART による ODS 出力のポータビリティ

特定の状況で PROC CHART を Output Delivery System と使用すると、ポータブルでないファイルが生成されます。SAS セッションの SAS システムオプション FORMCHAR=で非標準の線描文字が使用されると、SAS Monospace フォントがインストールされていない動作環境では線の代わりに不正な文字が出力に含まれていることがあります。この問題を回避するため、PROC CHART を実行する前に OPTIONS ステートメントを指定します。

```
options formchar="|----|+|---+=|-/\<>*";
```

例: CHART プロシジャ

例 1: 単純な度数カウントの作成

要素: VBAR ステートメント

詳細

この例では、チャート変数の値に対する度数カウントを表す縦棒グラフを作成します。

プログラム

```
data shirts;
   input Size $ @@;
   datalines;
medium    large
large     large
large     medium
medium    small
small     medium
medium    large
small     medium
```

```
large     large
large     small
medium    medium
medium    medium
medium    large
small     small
;

proc chart data=shirts;
   vbar size;

   title 'Number of Each Shirt Size Sold';
run;
```

プログラムの説明

Shirts データセットを作成します。 Shirts には、衣料品店で 1 週間の間に販売された特定のシャツのサイズが含まれています。販売されたシャツごとに 1 つのオブザベーションが含まれています。

```
data shirts;
   input Size $ @@;
   datalines;
medium    large
large     large
large     medium
medium    small
small     medium
medium    large
small     medium
large     large
large     small
medium    medium
medium    medium
medium    large
small     small
;
```

度数カウントを含む縦棒グラフを作成します。 VBAR ステートメントは、Size 値の度数カウントに対し縦棒グラフを作成します。

```
proc chart data=shirts;
   vbar size;
```

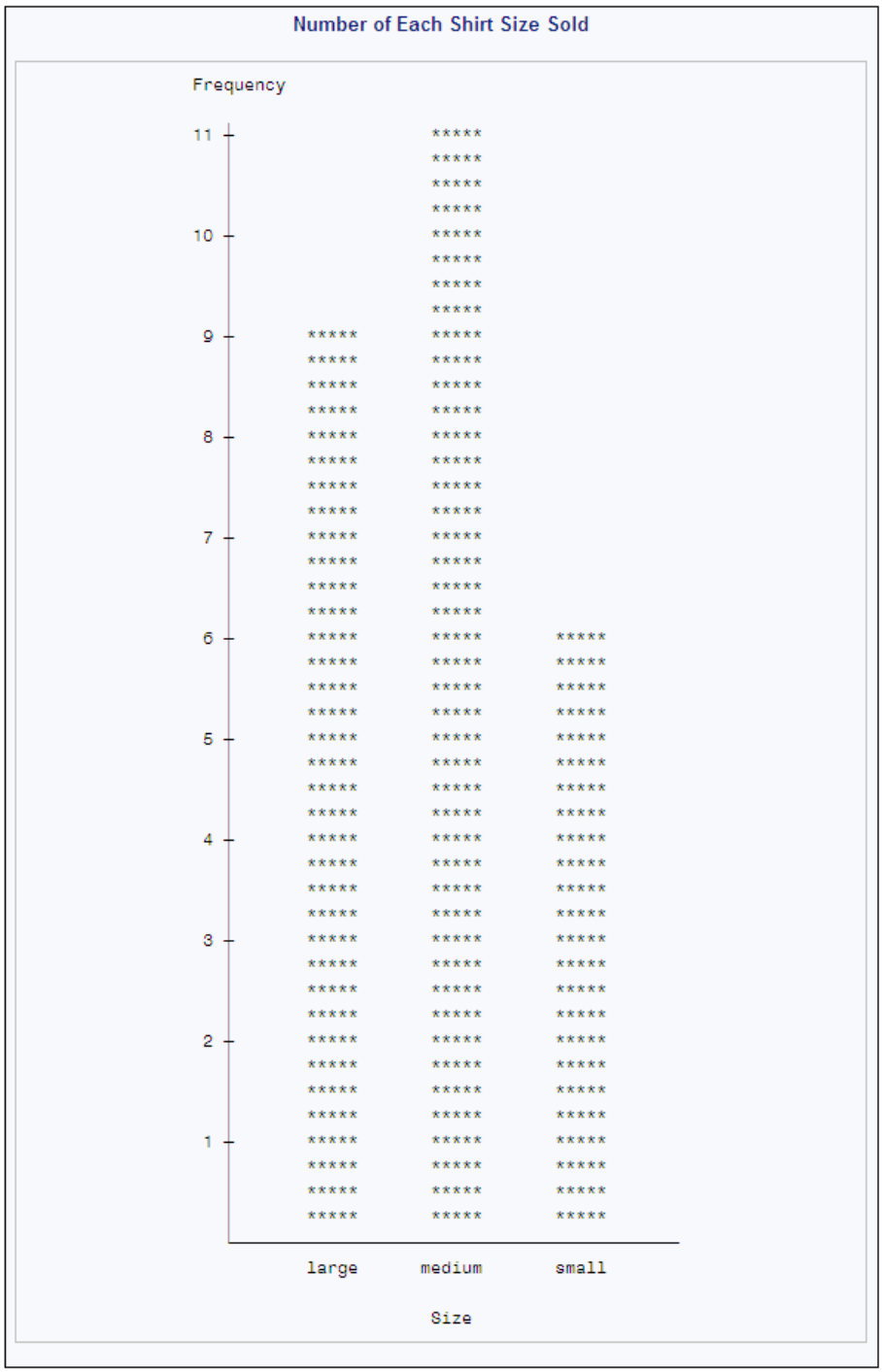
タイトルを指定します。

```
   title 'Number of Each Shirt Size Sold';
run;
```

出力:出力:HTML

次の度数チャートでは、1 週間の間の各シャツのサイズの店舗売上が表示されます。L サイズが 9 着、M サイズが 11 着、S サイズが 6 着です。

アウトプット 10.6 *販売された各シャツのサイズの数*



例 2: パーセント棒グラフの作成

要素: VBAR ステートメントオプション

TYPE=

データセット: SHIRTS

詳細

この例では、縦棒グラフを作成します。チャートの統計量は、販売されたシャツの合計数のカテゴリごとのパーセントです。

プログラム

```
proc chart data=shirts;
   vbar size / type=percent;

   title 'Percentage of Total Sales for Each Shirt Size';
run;
```

プログラムの説明

パーセントを含む縦棒グラフを作成します。 VBAR ステートメントは、縦棒グラフを作成します。TYPE=は、変数 Size に対するチャート統計量としてパーセントを指定します。

```
proc chart data=shirts;
   vbar size / type=percent;
```

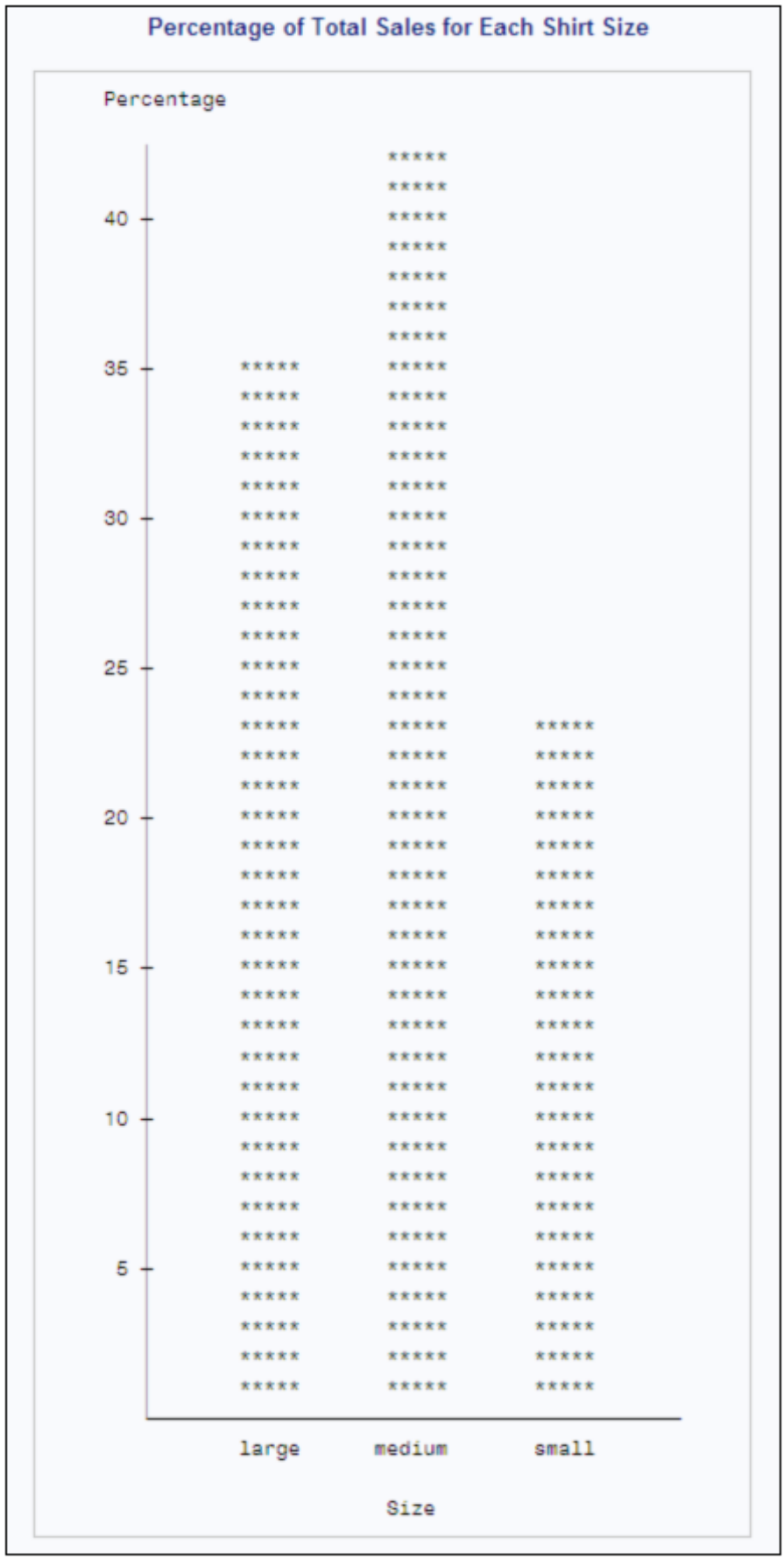
タイトルを指定します。

```
   title 'Percentage of Total Sales for Each Shirt Size';
run;
```

出力:出力:HTML

次のチャートには、シャツのサイズごとの合計売上のパーセントが表示されます。販売されたすべてのシャツのうち、42.3 パーセントが M、34.6 パーセントが L、23.1 パーセントが S でした。

アウトプット 10.7 シャツのサイズごとの合計売上のパーセント



例 3: バーをカテゴリに細分化する

要素: VBAR ステートメントオプション
SUBGROUP=
SUMVAR=

詳細

この例では、次を行います。

- 別の変数の値を表すバーの長さを含む 1 つの変数のカテゴリに対し縦棒グラフを作成します。
- 各バーを 3 番目の変数の値に基づいてカテゴリに分割します。

プログラム

```
data piesales;
   input Bakery $ Flavor $ Year Pies_Sold;
   datalines;
Samford  apple      2005  234
Samford  apple      2006  288
Samford  blueberry  2005  103
Samford  blueberry  2006  143
Samford  cherry     2005  173
Samford  cherry     2006  195
Samford  rhubarb    2005   26
Samford  rhubarb    2006   28
Oak      apple      2005  219
Oak      apple      2006  371
Oak      blueberry  2005  174
Oak      blueberry  2006  206
Oak      cherry     2005  226
Oak      cherry     2006  311
Oak      rhubarb    2005   51
Oak      rhubarb    2006   56
Clyde    apple      2005  213
Clyde    apple      2006  415
Clyde    blueberry  2005  177
Clyde    blueberry  2006  201
Clyde    cherry     2005  230
Clyde    cherry     2006  328
Clyde    rhubarb    2005   60
Clyde    rhubarb    2006   59
;

proc chart data=piesales;
   vbar flavor / subgroup=bakery

                 sumvar=pies_sold;

   title 'Pie Sales by Flavor Subdivided by Bakery Location';
run;
```

プログラムの説明

Piesales データセットを作成します。 Piesales には、同じ企業が所有する 3 つのベーカリーで 2 年間にわたって販売された各種パイの数が含まれます。ベーカリーはそれぞれ、Samford Avenue、Oak Street、Clyde Drive にあります。

```
data piesales;
   input Bakery $ Flavor $ Year Pies_Sold;
   datalines;
Samford  apple      2005  234
Samford  apple      2006  288
Samford  blueberry  2005  103
Samford  blueberry  2006  143
Samford  cherry     2005  173
Samford  cherry     2006  195
Samford  rhubarb    2005   26
Samford  rhubarb    2006   28
Oak      apple      2005  219
Oak      apple      2006  371
Oak      blueberry  2005  174
Oak      blueberry  2006  206
Oak      cherry     2005  226
Oak      cherry     2006  311
Oak      rhubarb    2005   51
Oak      rhubarb    2006   56
Clyde    apple      2005  213
Clyde    apple      2006  415
Clyde    blueberry  2005  177
Clyde    blueberry  2006  201
Clyde    cherry     2005  230
Clyde    cherry     2006  328
Clyde    rhubarb    2005   60
Clyde    rhubarb    2006   59
;
```

カテゴリに分割されるバーを含む縦棒グラフを作成します。 VBAR ステートメントは、パイの種類ごとに 1 つのバーを含む縦棒グラフを作成します。SUBGROUP=は、各バーをベーカリーごとの売上に分割します。

```
proc chart data=piesales;
   vbar flavor / subgroup=bakery
```

バーの長さの変数を指定します。 SUMVAR=は、その値がバーの長さによって表される変数として Pies_Sold を指定します。

```
                sumvar=pies_sold;
```

タイトルを指定します。

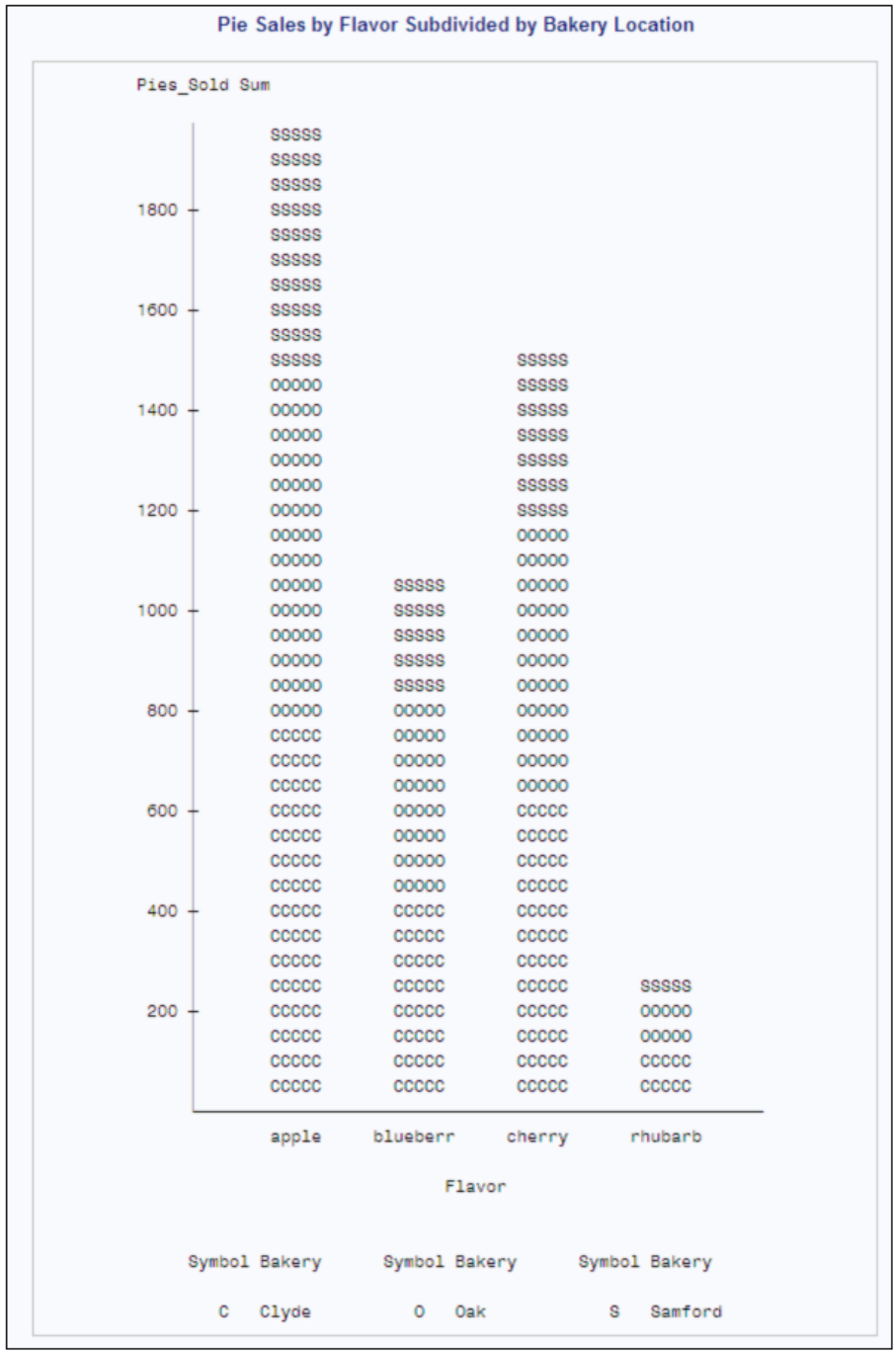
```
   title 'Pie Sales by Flavor Subdivided by Bakery Location';
run;
```

出力:出力:HTML

次の出力では、たとえばアップルパイの売上を表すバーは、3 つのすべてのベーカリーの 2 年間の合計数 1,940 個を表しています。Samford Avenue ベーカリーの記号は、上部に 522 個のパイを表します。Oak Street ベーカリーの記号は、真ん中に 690

個のパイを表します。Clyde Drive ベーカリーの記号は、アップルパイのバーの下部に 728 個のパイを表します。デフォルトで、横軸沿いのラベルは、8 文字に切り捨てられます。

アウトプット 10.8 *ベーカリーの場所別に分割された種類別パイの売上*



例 4: 並列棒グラフの作成

要素: VBAR ステートメントオプション
GROUP=

REF=
SUMVAR=
TYPE=

データセット: PIESALES

詳細

この例では、次を行います。

- 別の変数のカテゴリに対する変数の平均値をグラフ表示します
- 3 番目の変数のカテゴリに対する並列棒グラフを作成します
- チャートにわたる参照線を描きます

プログラム

```
proc chart data=piesales;
   vbar flavor / group=bakery
                 ref=100 200 300
                 sumvar=pies_sold
                 type=mean;
   title  'Mean Yearly Pie Sales Grouped by Flavor';
   title2 'within Bakery Location';
run;
```

プログラムの説明

並列縦棒グラフを作成します。 VBAR ステートメントは、GROUP=によって指定される Bakery の値にわたる売上を比較するための並列棒グラフを作成します。Bakery グループにはそれぞれ、Flavor 値ごとのバーが含まれています。

```
proc chart data=piesales;
   vbar flavor / group=bakery
```

参照線を作成します。 REF=は、100、200、300 でパイの売上をマーク付けする参照線を描きます。

```
                 ref=100 200 300
```

バーの長さの変数を指定します。 SUMVAR=は、バーの長さによって表される変数として Pies_Sold を指定します。

```
                 sumvar=pies_sold
```

統計量変数を指定します。 TYPE=は、ベーカリーと種類の組み合わせごとの 2005 年と 2006 年の売上の平均を計算します。

```
                 type=mean;
```

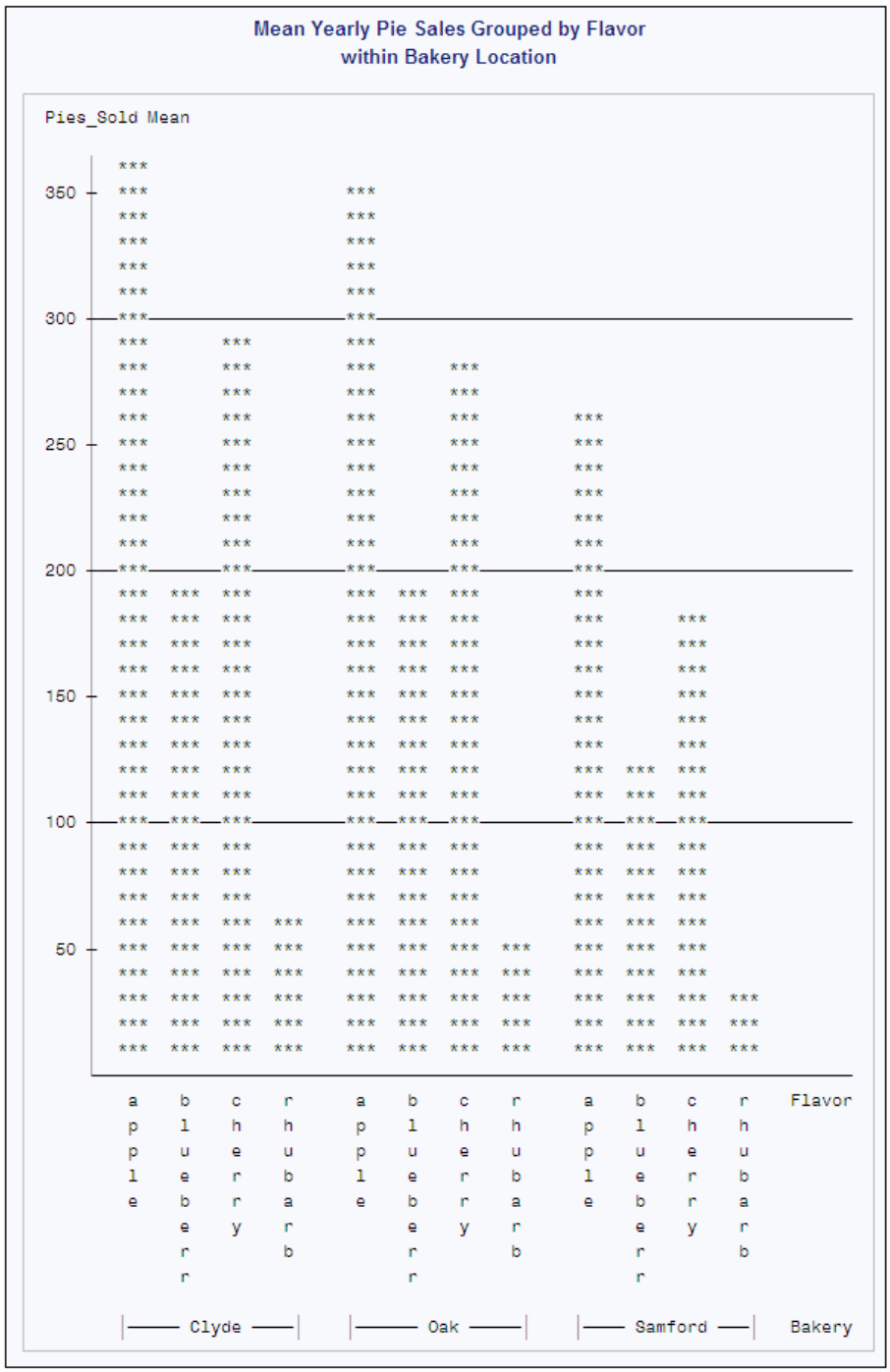
タイトルを指定します。

```
   title  'Mean Yearly Pie Sales Grouped by Flavor';
   title2 'within Bakery Location';
run;
```

出力:出力:HTML

次の並列棒グラフでは、ベーカリー間のパイの種類別売上を比較します。たとえば、アップルパイの売り上げに関して、Clyde Drive ベーカリーの平均は 364、Oak Street ベーカリーの平均は 345、Samford Avenue ベーカリーの平均は 261 です。

アウトプット 10.9 *ベーカリーの場所内の種類別にグループ化された年次パイ売上平均*



例 5: データサブセットごとに横棒グラフを作成する

要素: HBAR ステートメントオプション
GROUP=
SUMVAR=

他の要素: WHERE=データセットオプション

データセット: PIESALES

詳細

この例では、次を行います。

- 一般的な値を含むオブザベーションに対してのみ横棒グラフを作成します
- 別の変数のカテゴリに対する変数の値をグラフ表示します
- 3 番目の変数のカテゴリに対する並列棒グラフを作成します

プログラム

```
proc chart data=piesales(where=(year=2005));
   hbar bakery / group=flavor
   sumvar=pies_sold;
   title '2005 Pie Sales for Each Bakery According to Flavor';
run;
```

プログラムの説明

横棒グラフに対する変数値制限を指定します。 WHERE=は、チャートを 2005 年の売上合計のみに制限します。

```
proc chart data=piesales(where=(year=2005));
```

並列横棒グラフを作成します。 HBAR ステートメントは、GROUP=によって指定される Flavor の値にわたる売上を比較するための並列横棒グラフを作成します。Flavor グループにはそれぞれ、Bakery 値ごとのバーが含まれています。

```
   hbar bakery / group=flavor
```

バーの長さの変数を指定します。 SUMVAR=は、その値がバーの長さによって表される変数として Pies_Sold を指定します。

```
   sumvar=pies_sold;
```

タイトルを指定します。

```
   title '2005 Pie Sales for Each Bakery According to Flavor';
run;
```

出力:出力:HTML

アウトプット 10.10 *種類別のベーカリーごとの 2005 年のパイの売上*

2005 Pie Sales for Each Bakery According to Flavor

```
Flavor     Bakery                                Pies_Sold
                                                     Sum

                        |
apple      Clyde        |**********************    213.0000
           Oak          |**********************    219.0000
           Samford      |************************   234.0000
                        |
blueberr   Clyde        |******************         177.0000
           Oak          |******************         174.0000
           Samford      |**********                 103.0000
                        |
cherry     Clyde        |************************   230.0000
           Oak          |************************   226.0000
           Samford      |******************         173.0000
                        |
rhubarb    Clyde        |******                      60.0000
           Oak          |*****                       51.0000
           Samford      |***                         26.0000
                        |
                       -----+----+----+----+---
                            50   100   150   200

                             Pies_Sold Sum
```

例 6: BY グループごとにブロックチャートを作成する

要素: BLOCK ステートメントオプション
- GROUP=
- NOHEADER=
- SUMVAR=
- SYMBOL=

BY ステートメント

他の要素: PROC SORT

SAS システムオプション
- NOBYLINE
- OVP

TITLE statement
- #BYVAL 指定

データセット: PIESALES

詳細

この例では、次を行います。

- データセットを並べ替えます

- BY グループごとにブロックチャートを作成します
- ブロックを 3D グラフに編成します
- BY グループ固有タイトルを印刷します

プログラム

```
proc sort data=piesales out=sorted_piesales;
   by year;
run;

options nobyline;

proc chart data=sorted_piesales;
   by year;

   block bakery / group=flavor

                  sumvar=pies_sold

                  noheader

                  symbol='OX';

   title  'Pie Sales for Each Bakery and Flavor';
   title2 '#byval(year)';
run;

options byline;
```

プログラムの説明

入力データセット Piesales を並べ替えます。 PROC SORT は Piesales を年によって並べ替えます。並べ替えは、年ごとに別のチャートを作成するために必要です。

```
proc sort data=piesales out=sorted_piesales;
   by year;
run;
```

BY 行を非表示にし、ブロックチャートの重ね打ちされた文字を許可します。 NOBYLINE は、出力の通常の BY 行を非表示にします。

```
options nobyline;
```

複数のブロックチャートに対する BY グループを指定します。 BY ステートメントは、2005 年の売上と 2006 年の売上に対しそれぞれ 1 つのチャートを作成します。

```
proc chart data=sorted_piesales;
   by year;
```

ブロックチャートを作成します。 BLOCK ステートメントは、年ごとのブロックチャートを作成します。チャートにはそれぞれ、ブロックを含むセルのグリッド(下部の Bakery 値、側面の Flavor 値)が含まれます。

```
   block bakery / group=flavor
```

バーの長さの変数を指定します。 SUMVAR=は、その値がブロックの長さによって表される変数として Pies_Sold を指定します。

```
                  sumvar=pies_sold
```

デフォルトのヘッダー行を非表示にします。NOHEADER は、デフォルトのヘッダー行を非表示にします。

```
noheader
```

ブロック記号を指定します。SYMBOL=は、ブロックの記号を指定します。

```
symbol='OX';
```

タイトルを指定します。#BYVAL 指定は、年をタイトルの 2 番目の行に挿入します。

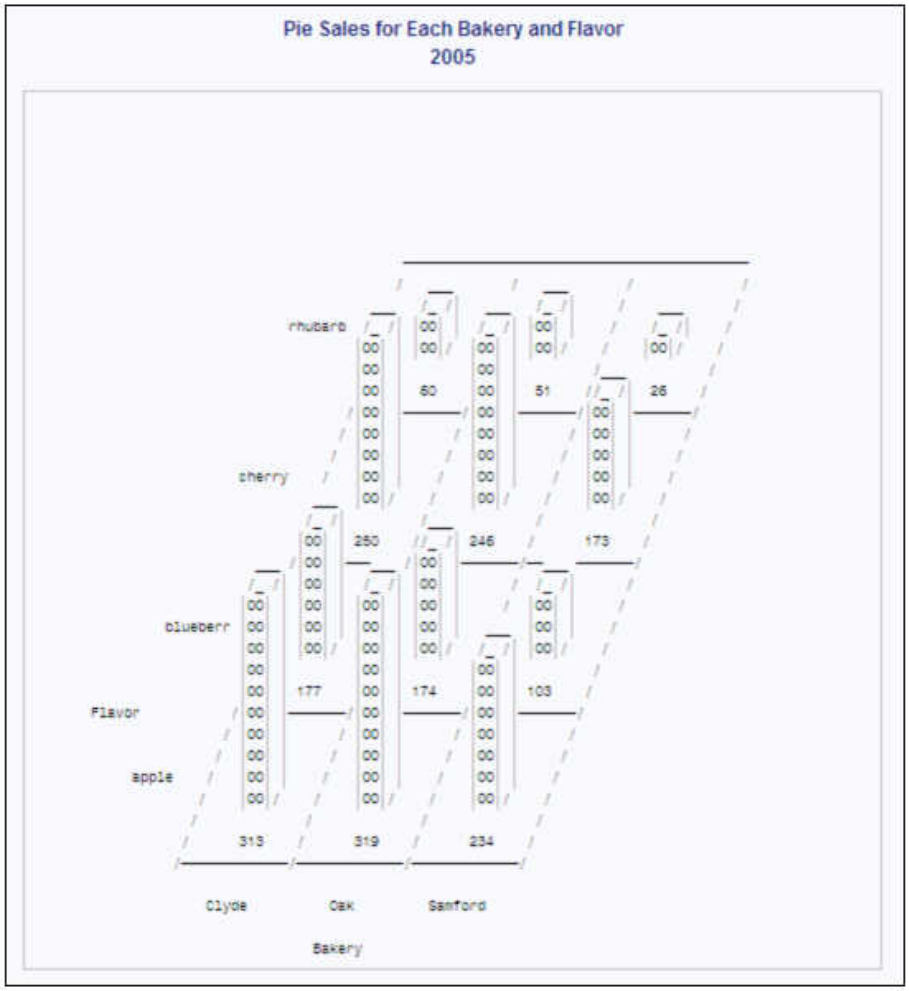
```
   title  'Pie Sales for Each Bakery and Flavor';
   title2 '#byval(year)';
run;
```

デフォルトの BY 行の印刷をリセットします。SAS システムオプション BYLINE は、デフォルトの BY 行の印刷をリセットします。

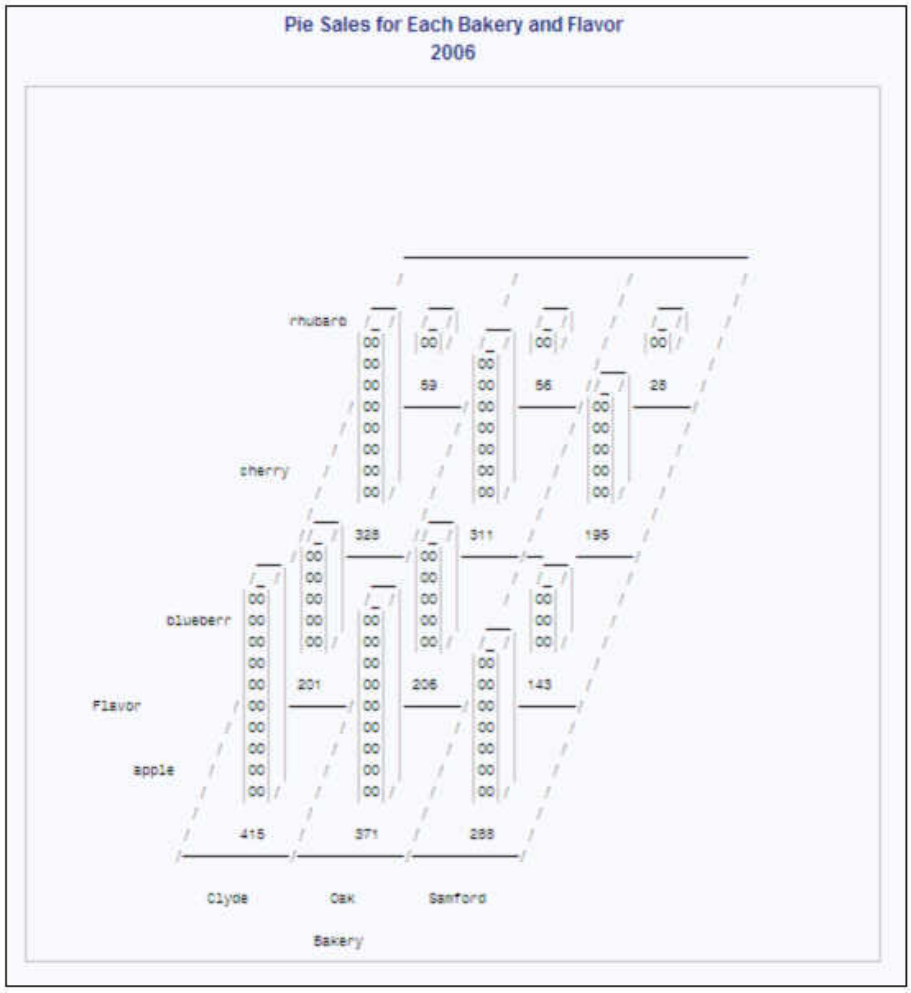
```
options byline;
```

出力:出力:HTML

アウトプット 10.11 *ベーカリーおよび種類ごとの 2005 年のパイ売上*



アウトプット 10.12 ベーカリーおよび種類ごとの 2006 年のパイ売上



参考文献

Nelder, J.A. 1976. “A Simple Algorithm for Scaling Graphs.” *Applied Statistics* 25 (1) London, England: The Royal Statistical Society: 94–96.

Terrell, G.R. and D.W.Scott. 1985. “Oversmoothed Nonparametric Density Estimates.” *Journal of the American Statistical Association* 80 (389): 209–214.

11 章

CIMPORT プロシジャ

概要: CIMPORT プロシジャ

CIMPORT プロシジャの動作について

CIMPORT プロシジャは、CPORT プロシジャによって作成(*エクスポート*)された移送ファイルを*インポート*します。PROC CIMPORT は、移送ファイルを SAS カタログ、SAS データセットまたは SAS ライブラリの元の形式に戻します。*移送ファイル*は SAS ライブラリ、SAS カタログ、または SAS データセットを移送出力形式で保持している順編成ファイルです。PROC CPORT が書き出す移送出力形式は、すべての環境およびすべての SAS リリースを通して同じです。

PROC CIMPORT は、SAS ファイルの*変換*も行います。つまり、SAS ファイルの出力形式を SAS のバージョン間で適切な SAS 出力形式に変更します。たとえば、PROC CPORT と PROC CIMPORT を使用して、ファイルを SAS の以前のリリースから以降のリリースに(SAS 6 から SAS®9 など) 、または同じバージョン間(SAS 9 動作環境間など)で移動できます。PROC CIMPORT は自動で移送ファイルを変換して、それをインポートします。

ただし、PROC CPORT と PROC CIMPORT では、新バージョンから旧バージョンへのファイル移送(これを回帰という)は許可されていません。たとえば、SAS® 9 から SAS 6 への移送は許可されません。

注: PROC CIMPORT と PROC CPORT を使用して、グラフィックカタログをバックアップできます。PROC COPY はグラフィックカタログのバックアップには使用できません。

PROC CIMPORT は出力を生成しませんが、ノートを SAS ログに書き込みます。

移送ファイルの作成処理と読み込み処理

次に、ソースコンピュータで移送ファイルを作成して、それをターゲットコンピュータで読むプロセスを示します。

1. 移送ファイルは、ソースコンピュータで PROC CPORT を使って作成されます。
2. 移送ファイルは、ソースコンピュータからターゲットコンピュータへ、通信ソフトまたは磁気メディアを使って移送されます。
3. 移送ファイルは、ターゲットコンピュータで PROC CIMPORT を使って読み込まれます。

   注: 移送ファイルは、PROC CPORT を使って作成されたものと、XPORT エンジンを使って作成されたものとで、互換性はありません。

移送ファイルの作成(PROC CPORT)、移送ファイルの移動、移送ファイルの復元(PROC CIMPORT)のステップの詳細については、*SAS ファイルの移動とアクセス*を参照してください。

構文: CIMPORT プロシジャ

ヒント: グラフィックカタログをバックアップするには、PROC CIMPORT または PROC CPORT を使用してください。PROC COPY はグラフィックカタログのバックアップには使用できません。

参照項目: Windows、UNIX 、z/OS での CIMPORT プロシジャ

移送ファイルの作成(PROC CPORT)、移送ファイルの移動、移送ファイルの復元(PROC CIMPORT)のステップの詳細については、次を参照してください。*SAS ファイルの移動とアクセス*

PROC CIMPORT *destination=libref | <libref.> member-name <option(s)>*;
EXCLUDE *SAS file(s) | catalog entry(s)* </ MEMTYPE=*mtype*> </ ENTRYTYPE=*entry-type*>;
SELECT *SAS file(s) | catalog entry(s)* </ MEMTYPE=*mtype*> </ ENTRYTYPE=*entry-type*>;

ステートメント	タスク	例
"PROC CIMPORT ステートメント"	移送ファイルをインポートします	Ex. 1, Ex. 2, Ex. 3
"EXCLUDE ステートメント"	1 つ以上の指定ファイルをインポートプロセスから除外します	

ステートメント	タスク	例
“SELECT ステートメント”	1 つ以上のファイルまたはエントリをインポートプロセスに指定します。	Ex. 2

PROC CIMPORT ステートメント

移送ファイルをインポートします。

例: “例 1: ライブラリ全体のインポート” (335 ページ)
“例 2: 個々のカタログエントリのインポート” (336 ページ)
“例 3: 単一インデックス付き SAS データセットのインポート” (337 ページ)

構文

PROC CIMPORT *destination=libref | <libref.> member-name <option(s)>*;

オプション引数の要約

FORCE
ロックされているカタログへのアクセスが可能になります。

NEW
インポートされた移送ファイルに対し新しいカタログを作成し、同じ名前の既存カタログを削除します。

NOEDIT
SAS/AF PROGRAM および SCL エントリを編集機能なしでインポートします。

NOSRC
コンパイルされた SCL コードを含む SAS/AF エントリに対するソースコードのインポートを行いません。

SORT
PROC CIMPORT を使用して並べ替えられたデータセットをインポートするときに必要に応じて出力データセットをもう一度並べ替えます。

Control the contents of the transport file

EXTENDSN=YES | NO
整数型(short-integer) (8 バイト未満)の長さをインポート時に 1 バイト拡張するかどうかを指定します。

UPCASE
アルファベット文字を大文字で出力ファイルに書き込みます。

Identify the input transport file

INFILE=*fileref* | '*filename*'
読み込む移送ファイルの事前に定義されているファイル参照名またはファイル名を指定します。

TAPE
入力移送ファイルをテープから読み込みます。

Look at the encoding of the transport file

ENCODINGINFO=ALL | *n*
データセットのエンコーディング値をログに出力するために読み込むデータセットヘッダーの数を指定します。

ISFILEUTF8=YES | NO ISFILEUTF8=TRUE | FALSE
ファイルが UTF-8 形式にエンコードされるかどうかを指定します。

Select files to import

AFTER=*date*
指定したエントリの種類をインポートプロセスから除外します。

ET=(*etype(s)*)
インポートするエントリの種類を指定します。

MEMTYPE=*mtype*
ライブラリのインポート時にデータセットのみ、カタログのみ、または両方が移動させるように指定します。

必須引数

destination=libref* | <*libref.*>*member-name
インポートするファイルの種類を特定し、インポートするカタログ、SAS データセット、または SAS ライブラリを指定します。

destination
単一カタログ、単一 SAS データセット、または SAS ライブラリのメンバとして移送ファイルのファイルを特定します。*destination* 引数は、次のうちいずれかになります。

CATALOG | CAT | C
DATA | DS | D
LIBRARY | LIB | L

libref | <*libref.*> *member-name*
移送ファイルの出力先として、特定のカタログ、SAS データセット、SAS ライブラリを指定します。*destination* 引数が CATALOG または DATA の場合、*libref* とメンバ名の両方を指定できます。*libref* が省略された場合、PROC CIMPORT はデフォルトのライブラリ(通常は WORK ライブラリ)を *libref* として使用します。*destination* 引数が LIBRARY の場合、*libref* のみを指定します。

参照項目 名前とメンバ名に使用できる命名規則については、“Names in the SAS Language” (*SAS Language Reference: Concepts*) を参照してください。

オプション引数

AFTER=*date*
指定したエントリの種類をインポートプロセスから除外します。*etype* が単一エントリの種類の場合は、かっこは省略できません。複数の値はスペースで区切ります。

操作 同じ PROC CIMPORT ステップで、EET=オプションと ET=オプションを同時に使用できません。

ENCODINGINFO=ALL | *n*
データセットのエンコーディング値をログに出力するために読み込むデータセットヘッダーの数を指定します。

ALL
すべてのデータセットヘッダーを読み込むことを指定します。データセットヘッダーの格納されたエンコーディング値は SAS ログに出力されます。

n
ゼロより大きい整数を指定します。この値は、読み込むデータセットヘッダーの数を表します。データセットヘッダーの格納されたエンコーディング値は SAS ログに出力されます。

範囲 ゼロより大きい整数を指定します。上限はシステムの制約によって異なります。*n* が移送ファイル

注 データセットヘッダーの読み取り中にライブラリにカタログが見つかった場合、カタログはスキップ

参照項目 移送ファイルの作成と復元については、*SAS ファイルの移動とアクセス*を参照してください。

例
```
proc cimport lib=out file='mixedencoding.tpt' encodinginfo=20;
run;

NOTE:The CPORTed data set ASCIIANYDS transport file encoding=us-ascii.NOTE:The CPORT
```

ET=(*etype(s)*)
インポートするエントリの種類を指定します。*etype* が単一エントリの種類の場合は、かっこは省略できません。複数の値はスペースで区切ります。

操作 同じ PROC CIMPORT ステップで、EET=オプションと ET=オプションを同時に使用できません。

EXTENDSN=YES | NO
整数型(short-integer) (8 バイト未満)の長さをインポート時に 1 バイト拡張するかどうかを指定します。拡張すると、整数型(short-integer)を IBM 形式または IEEE 形式で移送するときに精度の損失を回避できます。8 バイトの整数型(short-integer)の長さは拡張できません。

デフォルト YES

制限事項 このオプションは、データセットにのみ適用されます。

ヒント 小数値を整数型(short-integer)として保存しないでください。

ISFILEUTF8=YES | NO ISFILEUTF8=TRUE | FALSE
UTF-8 として移送ファイルに含まれるデータセットのエンコーディングを明示的に指定します。データセットエンコーディングは SAS 9.2 移送ファイルに記録(またはスタンプ)されますが、エンコーディングは 9.2 より前の SAS リリースを使用して作成された移送ファイルにはスタンプされません。そのため、UTF-8 エンコーディングの指定は、次の状況下で有用です。

- 移送ファイルのデータセットが、9.2 より前の SAS リリースを使用して作成された。
- データセットが UTF-8 としてエンコードされている。

ターゲット環境の移送ファイルを復元する場合、その復元作業の前に移送ファイルの説明がなければなりません。

YES | Y | yes | y | TRUE | true | T | t
移送ファイルのデータセットが UTF-8 としてエンコードされるように指定します。

NO | N | no | n | FALSE | false | F | f
移送ファイルのデータセットが UTF-8 としてエンコードされないように指定します。NO がデフォルトです。

ターゲット SAS セッションの移送ファイルを正常にインポートするために、移送ファイルに関する次の情報がなければなりません。

- ソース動作環境。たとえば、`Windows`
- SAS リリース。たとえば、`SAS 9.2`
- 移送ファイルの名前。たとえば、`tport.dat`
- 文字データのエンコーディング。たとえば、`wlatin1`
- 文字データの各国語。たとえば、アメリカ英語(または `en_US`)

デフォルト NO

制限事項 PROC CIMPORT がこのオプションを使用するのは、移送ファイルがデータセットのエンコーディングによってスタンプされない場合のみです。エンコーディングは、9.2 より前の SAS リリースでは記録されませんでした。エンコーディングが移送ファイルに記録され、ISFILEUTF8=オプションが PROC CIMPORT で指定されると、ISFILEUTF8=は無視されます。

参照項目 “CIMPORT の問題:移送ファイルのインポート移送ファイルのインポート ” (330 ページ)

移送ファイルの作成と復元については、*SAS ファイルの移動とアクセス*を参照してください。

FORCE
ロックされているカタログへのアクセスが可能になります。PROC CIMPORT はデフォルトで、更新中のカタログにその他のユーザーがアクセスしないようにするために、更新中のカタログをロックします。FORCE オプションはこのロックを無効にします。これにより、その他のユーザーはインポート中のカタログにアクセスできるようになります。また、その他のユーザーが現在アクセスしているカタログをインポートできるようになります。

注意:
FORCE オプションにより、予期せぬ結果が発生する可能性があります。 FORCE オプションを指定すると、複数のユーザーが同じカタログエントリに同時にアクセスできるようになります。

INFILE=*fileref* | '*filename*'
読み込む移送ファイルの事前に定義されているファイル参照名またはファイル名を指定します。INFILE=オプションを省略すると、PROC CIMPORT はファイル参照名 SASCAT で移送ファイルから読み込もうとします。ファイル参照名 SASCAT が存在しない場合、PROC CIMPORT は、SASCAT.DAT という名前のファイルから読み込もうとします。

別名 FILE=

例 “例 1: ライブラリ全体のインポート” (335 ページ)

MEMTYPE=*mtype*
データセットのみ、カタログのみ、または両方が移送ファイルからインポートされるように指定します。*mtype* に可能な値は、次のとおりです。

ALL　　カタログとデータセットの両方

CATALOG | CAT　カタログ

DATA | DS　SAS データセット

NEW
指定する出力先の名前が既存カタログと同じ場合に、インポートされた移送ファイルのコンテンツを含む新しいカタログを作成します。NEW は、インポートの出力先として指定する名前と同じ名前の既存カタログを削除します。NEW を指定せず、指定する出力先名が既存カタログ名と同じ場合、PROC CIMPORT はインポートされた移送ファイルを既存カタログに追加します。

NOEDIT
SAS/AF PROGRAM および SCL エントリを編集機能なしでインポートします。

SAS/AF ソフトウェアの BUILD プロシジャで MERGE ステートメントで NOEDIT オプションを使用して SCL コードを含む新規カタログを作成する場合と同じ結果が得られます。

注: NOEDIT オプションは SAS/AF PROGRAM と SCL エントリにのみ影響します。FSEDIT SCREEN および FSVIEW FORMULA エントリには影響しません。

別名 NEDIT

NOSRC
コンパイルされた SCL コードを含む SAS/AF エントリに対するソースコードのインポートを行いません。

SAS/AF ソフトウェアの BUILD プロシジャで MERGE ステートメントで NOSOURCE オプションを使用して SCL コードを含む新規カタログを作成する場合と同じ結果が得られます。

別名 NSRC

操作 NOSRC オプションは FRAME、PROGRAM、SCL 以外のエントリの種類と使用した場合、PROC CIMPORT に無視されます。

SORT
PROC CIMPORT を使用して並べ替えられたデータセットをインポートするときに必要に応じて出力データセットをもう一度並べ替えます。インポートするデータセットが 1 つ以上の文字変数によって並べ替えられ、ターゲット SAS セッションの文字値の順序がソースセッションと異なる場合、もう一度並べ替える必要があります。

注: CIMPORT SORT オプションは、並べ替え情報を含んでいないデータセットに影響しません。このオプションは、以前に並べ替えられたデータセットにのみ適用されます。

文字値の順序、つまり、照合順序は、SAS および SORT プロシジャを起動するときに使用されるオプションによって使用されるセッションエンコーディングと変換テーブルに応じて異なります。必要に応じて、インポート時にもう一度並べ替えを実行します。データセットのソートインジケータは保持されます。

詳細については、“Collating Sequence” (*SAS National Language Support (NLS): Reference Guide*)および 59 章, “SORT プロシジャ,” (1785 ページ)を参照してください。

PROC CIMPORT がデータセットをもう一度並べ替えると、次の情報が出力されます。

```
NOTE:PROC CIMPORT re-sorted the data set WORK.TEMP because it contained       character
```

例 `proc cimport 'transportfile.tpt' lib=library sort; run;`

z/OS のサンプルコードを次に示します。

```
proc cimport data=file.mwelect inf=CPO sort; run;
```

TAPE
入力移送ファイルをテープから読み込みます。

デフォルト PROC CIMPORT はディスクから読み込みます。

UPCASE
移送ファイルから読み込み、アルファベット文字を大文字で出力ファイルに書き込みます。

制限事項 UPCASE オプションは、ダブルバイト文字セット(DBCS)拡張が SAS に付随している場合のみ使用できます。

ヒント PROC CPORT は、すべての大文字を含む移送ファイルの作成に使用できます。詳細については、オプション“OUTTYPE=UPCASE” (424 ページ)を参照してください。

EXCLUDE ステートメント

指定したファイルまたはエントリをインポートプロセスから除外します。

操作: PROC CIMPORT ステップでは EXCLUDE ステートメントまたは SELECT ステートメントを使用できますが、両方を同時には使用できません。

ヒント: PROC CIMPORT 一回の起動で、使用できる EXCLUDE ステートメントの数に制限はありません。

構文

EXCLUDE *SAS file(s) | catalog entry(s)* </ MEMTYPE=*mtype*>
</ ENTRYTYPE=*entry-type*>;

必須引数

SAS file(s) | catalog entry(s)
インポートプロセスから除外する SAS ファイルまたはカタログエントリを指定します。SAS ライブラリのインポート時は SAS ファイル名を指定し、個別 SAS カタログのインポート時はカタログエントリ名を指定します。複数のファイル名またはエントリ名はスペースで区切ります。EXCLUDE ステートメントでは、ショートカットを使って、多数の似ている名前をリストすることができます。詳細については、“変数名リストを示すショートカット” (57 ページ)を参照してください。

オプション引数

ENTRYTYPE=*entry-type*
EXCLUDE ステートメントにリストされる 1 つ以上のカタログエントリに対し、単一のエントリの種類を指定します。カタログエントリの種類については、*SAS 言語リファレンス: 解説編*を参照してください。

別名 ETYPE=, ET=

制限事項 ENTRYTYPE=は個別の SAS カタログをインポートする時にのみ有効です。

MEMTYPE=*mtype*
EXCLUDE ステートメントにリストされている 1 つ以上の SAS ファイルに対して、単一のメンバの種類を指定します。*mtype* の値は、次のどれかになります。

ALL	カタログとデータセットの両方
CATALOG	カタログ
DATA	SAS データセット

ファイル名の後に、MEMTYPE= option をかっこで囲んで指定することもできます。かっこの中で、MEMTYPE=はそのすぐ前にあるファイル名の種類を識別します。オプションのこの形式を使用する場合は、EXCLUDE ステートメントでスラッシュの後に続く MEMTYPE=オプションを無効にしますが、PROC CIMPORT ステートメントの MEMTYPE=オプションと一致している必要があります。

別名 MTYPE=、MT=

デフォルト ALL

制限事項 MEMTYPE=は、SAS ライブラリのインポート時にのみ有効です。

SELECT ステートメント

インポートする個別のファイルまたはエントリを指定します。

操作: PROC CIMPORT ステップでは EXCLUDE ステートメントまたは SELECT ステートメントを使用できますが、両方を同時には使用できません。

ヒント: PROC CIMPORT 一回の起動で、使用できる SELECT ステートメントの数に制限はありません。

例: “例 2: 個々のカタログエントリのインポート” (336 ページ)

構文

SELECT *SAS file(s) | catalog entry(s)* </ MEMTYPE=*mtype*>
</ ENTRYTYPE=*entry-type*>;

必須引数

SAS file(s) | catalog entry(s)
インポートする SAS ファイルまたはカタログエントリを指定します。SAS ライブラリのインポート時は SAS ファイル名を指定し、個別 SAS カタログのインポート時はカタログエントリ名を指定します。複数のファイル名またはエントリ名はスペースで区切ります。SELECT ステートメントでは、ショートカットを使って、多数の似ている名前をリストすることができます。詳細については、“変数名リストを示すショートカット” (57 ページ) を参照してください。

オプション引数

ENTRYTYPE=*entry-type*
SELECT ステートメントにリストされる 1 つ以上のカタログエントリに対し、単一のエントリの種類を指定します。カタログエントリの種類については、*SAS 言語リファレンス: 解説編*を参照してください。

別名 ETYPE=, ET=

制限事項 ENTRYTYPE=は個別の SAS カタログをインポートする時にのみ有効です。

MEMTYPE=*mtype*
SELECT ステートメントにリストされている 1 つ以上の SAS ファイルに対して、単一のメンバの種類を指定します。有効な値は CATALOG または CAT、DATA、または ALL です。

ファイル名の後に、MEMTYPE= option をかっこで囲んで指定することもできます。かっこの中で、MEMTYPE=はそのすぐ前にあるファイル名の種類を識別します。オプションのこの形式を使用する場合は、SELECT ステートメントでスラッシュの後に続く MEMTYPE=オプションを無効にしますが、PROC CIMPORT ステートメントの MEMTYPE=オプションと一致している必要があります。

別名 MTYPE=、MT=

デフォルト ALL

制限事項 MEMTYPE=は、SAS ライブラリのインポート時にのみ有効です。

CIMPORT の問題:移送ファイルのインポート移送ファイルのインポート

移送ファイルとエンコーディングについて

移送ファイルの文字データは、次の 2 種類のエンコーディングのうちいずれかで作成されます。

- 移送ファイルが作成される SAS セッションの UTF-8 エンコーディング
- 移送ファイルが作成される SAS セッションのロケールに関連付けられている Windows エンコーディング

次の例に、エンコーディングの移送ファイルへの適用方法を示します。

表 11.1 *移送ファイルへのエンコーディングの割り当て*

移送ファイルのエンコーディング値	SAS 起動におけるエンコーディング適用例	説明
UTF-8	sas9 -encoding utf8;	SAS セッションは、UTF-8 エンコーディングを使用して起動されます。セッションエンコーディングは、移送ファイルに適用されます。

移送ファイルのエンコーディング値	SAS 起動におけるエンコーディング適用例	説明
wlatin2	sas9 -locale pl_PL;	SAS セッションは、Polish Poland ロケールに関連付けられているデフォルトの UNIX エンコーディング、latin2 を使用して起動されます。

各ロケールに関連付けられているエンコーディングのリストについては、ロケールテーブル(*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

ソースおよびターゲット SAS セッションのエンコーディングは、移送ファイルが正常にインポートされるように互換性がある必要があります。次に、互換性のあるソースおよびターゲット SAS セッションの例を示します。

表 11.2 *互換性のあるエンコーディング*

ソース SAS セッション			ターゲット SAS セッション	
ロケール	UNIX SAS セッションエンコーディング	移送ファイルエンコーディング	ロケール	Windows SAS セッションエンコーディング
es_MX (Spanish Mexico)	latin1	wlatin1	it_IT (Italian Italy)	wlatin1

ソースおよびターゲット SAS セッションのエンコーディングは、互換性があります。es_MX ロケールに対する Windows デフォルトエンコーディングが wlatin1 で、ターゲット SAS セッションのエンコーディングが wlatin1 であるためです。

ただし、ソースおよびターゲット SAS セッションのエンコーディングに互換性がない場合、移送ファイルは正常にインポートできません。次に、互換性のないエンコーディングの例を示します。

表 11.3 *互換性のないエンコーディング*

ソース SAS セッション			ターゲット SAS セッション	
ロケール	UNIX SAS セッションエンコーディング	移送ファイルエンコーディング	ロケール	z/OS エンコーディング
cs_CZ (Czech Czechoslovakia)	latin2	wlatin2	de_DE (German Germany)	open_ed-1141

ソースおよびターゲット SAS セッションのエンコーディングは、互換性がありません。cs_CZ ロケールの Windows デフォルトエンコーディングが wlatin2 で、ターゲット SAS セッションのエンコーディングが open_ed-1141 であるためです。移送ファイルは、これらのロケール間でインポートできません。

移送ファイルをインポートするとき、ENCODINGINFO=オプションを使用して移送ファイルのエンコーディング値を出力できます。それ以外の場合は、警告またはエラーメッセージによって互換性問題が通知されます。ENCODINGINFO=オプションの詳細については、“ENCODINGINFO=ALL | *n*” (324 ページ)を参照してください。

9.2 より前の SAS リリースを使用して作成された移送ファイルの問題

概要:9.2 より前の SAS リリース

9.2 より前の SAS リリースで作成された移送ファイルには、エンコーディング値がスタンプされません。そのため、CIMPORT プロシジャは、移送ファイルのエンコーディングの ID を認識せず、特定の警告やエラーの詳細をレポートできません。移送ファイルのエンコーディングは、復元アクションの実行時に推測する必要があります。

ただし、移送ファイルに関する知識を使用して、移送問題を解決できる必要があります。ターゲット SAS セッションでの移送ファイルのインポートに有用な情報については、Tips:for the ISFILEUTF8 option (325 ページ)を参照してください。移送ファイルの作成と復元については、*SAS ファイルの移動とアクセス*を参照してください。

次に、復元アクションを含む警告およびエラーメッセージを示します。

- “Error:移送ファイルエンコーディングが不明 ISFILEUTF8=オプションを使用します。” (332 ページ)
- “警告:移送ファイルエンコーディングが不明” (332 ページ)

Error:移送ファイルエンコーディングが不明 ISFILEUTF8=オプションを使用します。

このエラーメッセージでは、次の情報が提供されます。

- 移送ファイルが 9.2 より前の SAS リリースを使用して作成されている。
- エンコーディングが移送ファイルにスタンプされていないため、エンコーディングが不明である。
- ターゲット SAS セッションで UTF-8 エンコーディングが使用されている。

注: 復元ステップを実行するため、移送ファイルのエンコーディングがわかっている必要があります。

移送ファイルが UTF-8 としてエンコードされていることがわかっていれば、ファイルを再度インポートして、ISFILEUTF8=YES オプションを PROC CIMPORT で使用できます。

次に、UTF-8 移送ファイルと UTF-8 ターゲット SAS セッションの例を示します。UTF-8 移送ファイルが 9.2 より前の SAS リリースを使用して作成されている。

```
filename importin 'transport-file';
libname target 'sas-library';
proc cimport isfileutf8=yes infile=importin library=target memtype=data;
run;
```

構文の詳細については、ISFILEUTF8=オプション (325 ページ)を参照してください。

PROC CIMPORT が続きます。

警告:移送ファイルエンコーディングが不明

この警告メッセージでは、次の情報が提供されます。

- 移送ファイルが 9.2 より前の SAS リリースを使用して作成されている。
- エンコーディングが移送ファイルにスタンプされていないため、エンコーディングが不明である。

インポートされたデータセットから文字データを読み込みます。データを読み込めない場合、ターゲット SAS セッションのロケールが移送ファイルのエンコーディングと互換していないことが推測できます。

注: 復元ステップを実行するため、移送ファイルのエンコーディングがわかっている必要があります。

たとえば、Polish Poland ロケールを使用して作成された移送ファイルが、9.2 より前の SAS リリースを使用してソース SAS セッションで作成されたとします。ターゲット SAS セッションでは German ロケールを使用します。

1. ターゲット SAS セッションで、別の SAS セッションを開始し、移送ファイルを作成したソース SAS セッションのロケールにロケールを変更します。

   この例では、新しい SAS セッションを Polish Poland ロケールで開始します。

   ```
   sas9 -locale pl_PL;
   ```

2. ファイルを再度インポートします。次に例を示します。

   ```
   filename importin 'transport-file';
   libname target 'sas-library';
   proc cimport infile=importin library=target memtype=data;
   run;
   ```

PROC CIMPORT が続いて、Polish_Poland ロケールを使用した SAS セッションでデータが読み込み可能になります。

SAS のバージョン 9.2 以降を使用して作成した移送ファイルの問題

概要

文字データのエンコーディングは、SAS のバージョン 9.2 以降を使用して作成される移送ファイルにスタンプされます。そのため、CIMPORT プロシジャは、エンコード移送ファイルを UTF-8 エンコーディングを使用しない SAS セッションにインポートできないなどのエラー状況を検出できます。たとえば、UTF-8 移送ファイルは、Wlatin2 エンコーディングを使用する SAS セッションにインポートできません。

SAS のバージョン 9.2 以降は、移送ファイルのエンコーディングとターゲット SAS セッションのロケールの間の非互換性の状況を検出できます。一部の顧客の SAS アプリケーションが SAS 9.2 より前のリリースを使用して正常に実行されたため、PROC CIMPORT は警告のみレポートしますが、インポート処理の続行を許可します。

次に、復元アクションを含む警告およびエラーメッセージを示します。

- “Error:ターゲットセッションは UTF-8 以外:移送ファイルは UTF-8 を使用” (333 ページ)
- “警告:ターゲットセッションは UTF-8 以外:移送ファイルは UTF-8 以外” (334 ページ)

Error:ターゲットセッションは UTF-8 以外:移送ファイルは UTF-8 を使用

このエラーメッセージでは、次の情報が提供されます。

- ターゲットセッションでは、特定のエンコーディングを使用しています。
- 移送ファイルが UTF-8 としてエンコードされています。移送ファイルとターゲット SAS セッションのエンコーディングは、互換性がありません。

ターゲット SAS セッションのエンコーディングは、UTF-8 に変更する必要があります。

次に、SAS 9.2 UTF-8 移送ファイルと Wlatin1 ターゲット SAS セッションの例を示します。

1. 回復するには、ターゲット SAS セッションで、新しい SAS セッションを開始して、セッションエンコーディングを UTF-8 に変更します。次に例を示します。

   ```
   sas9 -encoding utf8;
   ```

2. ファイルを再度インポートします。次に例を示します。

   ```
   filename importin 'transport-file';
   libname target 'sas-library';
   proc cimport infile=importin library=target memtype=data;
   run;
   ```

PROC CIMPORT が続きます。

警告:ターゲットセッションは UTF-8 以外:移送ファイルは UTF-8 以外

この警告メッセージでは、次の情報が提供されます。

- ターゲット SAS セッションで、特定されたエンコーディングを使用している。
- 移送ファイルのエンコーディングが特定されている。移送ファイルとターゲット SAS セッションのエンコーディングは、互換性がありません。

次の表に、互換性のないソースおよびターゲット SAS セッションのロケールとエンコーディング値を示します。ソース SAS セッションで移送ファイルに割り当てられている wlatin2 Windows エンコーディングはターゲット SAS セッションの open_ed-1141 エンコーディングと互換性がありませんが、警告が表示され、インポートは続行されます。

表 11.4 *Czech および German ロケールの値のエンコード*

SAS セッション	Posix ロケール	Windows エンコーディング	UNIX エンコーディング	z/OS エンコーディング
ソース SAS セッション	cs_CZ (Czech Czechoslovakia)	wlatin2	latin2	open_ed-870
ターゲット SAS セッション	de_DE (German Germany)	wlatin1	latin9	open_ed-1141

移送ファイルはインポートされますが、ファイルのコンテンツには問題があります。メッセージは、互換性のないエンコーディング形式を特定します。回復するには、インポートされたファイルのコンテンツを読み込みます。ファイルが読み込めない場合、次のステップを実行します。

1. ターゲット SAS セッションで、新しい SAS セッションを開始し、ロケール(エンコーディングではなく)をソース SAS セッションで使用されるロケールに変更します。

   LOCALE=値は、ENCODING=値よりも優先されます。ENCODING=、DFLANG=、DATESTYLE=、PAPERSIZE=オプションに対するデフォルト値を自動的に設定するためです。

   ソースセッション(または移送ファイル)のロケールが不明な場合、移送ファイルの各言語から推測できます。

   たとえば、チェコ語が言語である場合、新しいターゲット SAS セッションで `cs_CZ` ロケールを指定します。

   新しい SAS セッションでの `cs_CZ` ロケールの指定例は、次のとおりです。

```
sas9 -locale cs_CZ;
```

ターゲット SAS セッションと移送ファイルは、互換性のあるエンコーディングを使用します。両方ともに wlatin2 を使用します。

詳細については、ロケールテーブル(*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

2. ファイルを再度インポートします。次に例を示します。

```
filename importin 'transport-file';
libname target 'sas-library';
proc cimport infile=importin library=target memtype=data;
run;
```

PROC CIMPORT が続きます。

数値精度の損失の問題

PROC CPORT と PROC CIMPORT では、極端に小さいまたは大きい数値では精度を欠く場合があります。詳細については、“Loss of Numeric Precision and Magnitude” (*SAS/CONNECT User's Guide*)を参照してください。

例: CIMPORT プロシジャ

例 1: ライブラリ全体のインポート

要素: PROC CIMPORT ステートメントオプション
INFILE=

詳細

この例では、PROC CPORT が別の動作環境で SAS ライブラリから作成した TRANFILE という名前の移送ファイルを PROC CIMPORT を使用してディスクから読み込む方法を示します。移送ファイルは、通信ソフトウェア、磁気媒体によって新しい動作環境に移動されます。PROC CIMPORT は、移送ファイルを新しい動作環境の NEWLIB という名前の SAS ライブラリにインポートします。

プログラム

```
libname newlib 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;

proc cimport library=newlib infile=tranfile;
run;
```

プログラムの説明

ライブラリ名とファイル名を指定します。 LIBNAME ステートメントは、新しい SAS ライブラリに対し LIBNAME を指定します。FILENAME ステートメントは、PROC CPORT が作成した移送ファイルのファイル名を指定し、ファイル特性に対し動作環境オプションを指定できるようにします。

```
libname newlib 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;
```

NEWLIB ライブラリの SAS ライブラリをインポートします。 PROC CIMPORT は、SAS ライブラリを NEWLIB という名前のライブラリにインポートします。

```
proc cimport library=newlib infile=tranfile;
run;
```

ログの例

ログ 11.1 ライブラリ全体のインポート

```
NOTE:Proc CIMPORT begins to create/update catalog NEWLIB.FINANCE NOTE:Entry LOAN.FRAME has been
imported.NOTE:Entry LOAN.HELP has been imported.NOTE:Entry LOAN.KEYS has been imported.NOTE:Entry
LOAN.PMENU has been imported.NOTE:Entry LOAN.SCL has been imported.NOTE:Total number of entries
processed in catalog NEWLIB.FINANCE:5 NOTE:Proc CIMPORT begins to create/update catalog NEWLIB.FORMATS
NOTE:Entry REVENUE.FORMAT has been imported.NOTE:Entry DEPT.FORMATC has been imported.NOTE:Total
number of entries processed in catalog NEWLIB.FORMATS:2
```

例 2: 個々のカタログエントリのインポート

要素: PROC CIMPORT ステートメントオプション
INFILE=
SELECT ステートメント

詳細

この例では、PROC CIMPORT を使用して個別のカタログエントリ LOAN.PMENU と LOAN.SCL を単一の SAS カタログから作成された移送ファイル TRANS2 からインポートする方法を示します。

プログラム

```
libname newlib 'sas-library';
filename trans2 'transport-file'

host-option(s)-for-file-characteristics;

proc cimport catalog=newlib.finance infile=trans2;
   select loan.pmenu loan.scl;
run;
```

プログラムの説明

ライブラリ名、ファイル名、動作環境オプションを指定します。 LIBNAME ステートメントは、新しい SAS ライブラリに対し LIBNAME を指定します。FILENAME ステートメントは、PROC CPORT が作成した移送ファイルのファイル名を指定し、ファイル特性に対し動作環境オプションを指定できるようにします。

```
libname newlib 'sas-library';
filename trans2 'transport-file'

host-option(s)-for-file-characteristics;
```

指定したカタログエントリを新しい SAS カタログにインポートします。 PROC CIMPORT は、個別のカタログエントリを TRANS2 移送ファイルからインポートして、NEWLIB.FINANCE という名前の新しい SAS カタログに保存します。SELECT ステートメントは、新しいカタログにインポートする移送ファイルから 2 つの指定されたエントリのみ選択します。

```
proc cimport catalog=newlib.finance infile=trans2;
   select loan.pmenu loan.scl;
run;
```

ログの例

ログ 11.2 *個々のカタログエントリのインポート*

```
NOTE:Proc CIMPORT begins to create/update catalog NEWLIB.FINANCE NOTE:Entry LOAN.PMENU has been
imported.NOTE:Entry LOAN.SCL has been imported.NOTE:Total number of entries processed in catalog
NEWLIB.FINANCE:2
```

例 3: 単一インデックス付き SAS データセットのインポート

要素: PROC CIMPORT ステートメントオプション
INFILE=

詳細

この例では、PROC CIMPORT を使用して、単一 SAS データセットからの PROC CPORT によって作成された移送ファイルからインデックス付けされた SAS データセットをインポートする方法を示します。

プログラム

```
libname newdata 'sas-library';
filename trans3 'transport-file'

host-option(s)-for-file-characteristics;

proc cimport data=newdata.times infile=trans3;
run;
```

プログラムの説明

ライブラリ名、ファイル名、動作環境オプションを指定します。 LIBNAME ステートメントは、新しい SAS ライブラリに対し LIBNAME を指定します。FILENAME ステートメントは、PROC CPORT が作成した移送ファイルのファイル名を指定し、ファイル特性に対し動作環境オプションを指定できるようにします。

```
libname newdata 'sas-library';
filename trans3 'transport-file'

host-option(s)-for-file-characteristics;
```

SAS データセットをインポートします。 PROC CIMPORT は、PROC CIMPORT ステートメントの DATA=指定によって特定する単一の SAS データセットをインポートします。PROC CPORT は、移送ファイル TRANS3 のデータセット NEWDATA.TIMES をエクスポートしました。

```
proc cimport data=newdata.times infile=trans3;
run;
```

ログの例

ログ 11.3 *単一インデックス付き SAS データセットのインポート*

```
NOTE:Proc CIMPORT begins to create/update data set NEWDATA.TIMES NOTE:The data set index x is
defined.NOTE:Data set contains 2 variables and 2 observations.Logical record length is 16
```

12 章

COMPARE プロシジャ

概要:COMPARE プロシジャ

COMPARE プロシジャの動作について

COMPARE プロシジャは、2 つの SAS データセットのコンテンツ、異なるデータセットの選択変数、または同一データセット内の変数を比較します。

PROC COMPARE は、基準データセットと比較データセットという 2 つのデータセットを比較します。プロシジャは、マッチング変数とマッチングオブザベーションを決定します。マッチング変数は、同名の変数か、または VAR および WITH ステートメントを使用してペアにする変数です。マッチング変数は同じ種類にする必要があります。マッチングオブザベーションは、指定したすべての ID 変数の値が同じオブザベーションで、ID ステートメントを使用していない場合は、データセットの同じ位置に発生するオブザベーションです。オブザベーションを ID 変数によってマッチングさせる場合は、すべての ID 変数を基準にして両方のデータセットを並べ替えておく必要があります。

PROC COMPARE から提供される情報について

PROC COMPARE は、比較対象の 2 つのデータセットについて次の情報を生成します。

- マッチング変数の値が異なるかどうか
- 一方のデータセットのオブザベーションが他方よりも多いかどうか
- 2 つのデータセットに共通する変数はどれか
- 一方のデータセットには存在するが他方のデータセットには存在しない変数はいくつあるか
- マッチング変数の出力形式、ラベルまたは種類が異なるかどうか
- マッチングオブザベーションの値の比較

さらに、PROC COMPARE は、比較する変数のオブザベーション間の差異について詳細を示す 2 種類の出力データセットを作成します。

次の例では、データセット Proclib.One と Proclib.Two を比較しています。このデータセットには、学生についての類似したデータが含まれています。

```
data proclib.one(label='First Data Set');
   input student year $ state $ gr1 gr2;
   label year='Year of Birth';
   format gr1 4.1;
   datalines;
1000 1990 NC 85 87
1042 1991 MS 90 92
1095 1989 TN 78 92
1187 1990 MA 87 94
;

data proclib.two(label='Second Data Set');
   input student $ year $ state $ gr1
         gr2 major $;
```

```
   label state='Home State';
   format gr1 5.2;
   datalines;
1000 1990 NC 85 87 Math
1042 1991 MS 90 92 History
1095 1989 TN 78 92 Physics
1187 1990 MA 87 94 Music
1204 1991 NC 82 96 English
;
```

概念:COMPARE プロシジャ

PROC COMPARE を使用した比較

PROC COMPARE はまず次の比較を行います。

- データセット属性(データセットオプション TYPE=および LABEL=により設定)。
- 変数。PROC COMPARE は、一方のデータセットの各変数をチェックして、他方のデータセットの変数と一致するかどうかを決定します。
- マッチング変数の属性(種類、長さ、ラベル、出力形式、入力形式)。
- オブザベーション。PROC COMPARE は、一方のデータセットの各オブザベーションをチェックして、他方のデータセットのオブザベーションと一致するかどうかを決定します。PROC COMPARE は、データセット内の位置かまたは ID 変数の値を基準にしてオブザベーションをマッチングします。

これらの比較を行った後で、PROC COMPARE はデータセットのマッチ部分の値を比較します。PROC COMPARE は、オブザベーションの位置かまたは ID 変数の値を基準にしてデータを比較します。

オブザベーションの位置の比較

次の図は、2 つのデータセットを示しています。網かけボックス内のデータは、データセットのうちプロシジャが比較する部分を示しています。同じ名前の変数は同じ種類であると仮定します。

図 12.1 オブザベーションの位置の比較

Data Set ONE

Obs	idnum	name	year	state	grade1	grade2
1	1000	Emily	1990	NC	85.0	87
2	1042	Dan	1991	MS	90.0	92
3	1095	Julia	1989	TN	78.0	92
4	1187	Robin	1990	MA	87.0	94

Data Set TWO

Obs	idnum	name	year	state	grade1	grade2	major
1	1000	Emily	1990	NC	85.00	87	Math
2	1042	Dan	1991	MS	90.00	92	History
3	1095	Julia	1989	TN	78.00	92	Physics
4	1187	Robin	1990	MA	87.00	94	Music
5	1204	Keith	1991	NC	82.00	96	English
6	1198	Ted	1990	PA	84.00	93	Music
7	1054	Lisa	1989	GA	81.00	94	French

PROC COMPARE を使用してデータセット TWO とデータセット ONE を比較する場合、プロシジャはデータセット ONE の 1 番目のオブザベーションとデータセット TWO の 1 番目のオブザベーションを比較してから、1 番目のデータセットの 2 番目のオブザベーションと 2 番目のデータセットの 2 番目のオブザベーションを比較し、その後も同じように処理を続けます。比較対象のオブザベーションごとに、プロシジャは idnum、name、year、state、grade1、grade2 値を比較します。

プロシジャは、データセット TWO の最後の 3 つのオブザベーションや変数 major の値についてはレポートしません。これはデータセット ONE に比較対象が存在しないからです。

ID 変数を使用した比較

単純な比較では、PROC COMPARE は、オブザベーション番号を使用して、どのオブザベーションを比較するかを決定します。ID 変数を使用する場合、PROC COMPARE は、ID 変数の値を使用して、どのオブザベーションを比較するかを決定します。ID 変数は、値が一意で、種類が同じである必要があります。

次の図で示されている 2 つのデータセットについては、IDNUM が ID 変数で、なおかつ両データセットの IDNUM の種類が同じだと仮定します。プロシジャは、IDNUM の値が同じオブザベーションを比較します。網かけボックス内のデータは、データセットのうちプロシジャが比較する部分を示しています。

図 12.2 ID 変数の値の比較

Data Set ONE

Obs	idnum	name	year	state	grade1	grade2
1	1000	Emily	1990	NC	85.0	87
2	1042	Dan	1991	MS	90.0	92
3	1095	Julia	1989	TN	78.0	92
4	1187	Robin	1990	MA	87.0	94

Data Set TWO

Obs	idnum	name	year	state	grade1	grade2	major
1	1000	Emily	1990	NC	85.00	87	Math
2	1042	Dan	1991	MS	90.00	92	History
3	1095	Julia	1989	TN	78.00	92	Physics
4	1187	Robin	1990	MA	87.00	94	Music
5	1204	Keith	1991	NC	82.00	96	English
6	1198	Ted	1990	PA	84.00	93	Music
7	1054	Lisa	1989	GA	81.00	94	French

データセットには、5 つのマッチング変数 name、year、state、grade1、grade2 が含まれています。また、idnum 値が `1000`、`1042`、`1095`、`1187` の 4 つのマッチングオブザベーションも含まれています。

データセット TWO には、データセット ONE にマッチングオブザベーションが含まれていない 3 つのオブザベーション(idnum=`1204`、idnum=`1198`、idnum=`1054`)が含まれています。同様に、データセット ONE には、データセット TWO の変数 YEAR とマッチする変数はありません。

ID 変数の使用例については、“例 5: ID 変数を利用し、オブザベーションを比較する” (388 ページ) を参照してください。

同等性の基準

CRITERION=オプションの使用

COMPARE プロシジャは、METHOD=オプションに従って測定された差異の大きさが CRITERION=オプションの値を上回る場合、数値が不等であると判断します。PROC COMPARE では、CRITERION=を適用するメソッドが 4 つあります。

- EXACT メソッドでは、完全に同等かが検証されます。
- ABSOLUTE メソッドでは、絶対差と CRITERION=で指定した値が比較されます。
- RELATIVE メソッドでは、絶対的な相対差と CRITERION=で指定した値が比較されます。
- PERCENT メソッドでは、絶対的なパーセント表示の差異と CRITERION=で指定した値が比較されます。

数値変数を比較する場合、基準データセットの値を *x*、比較データセットの値を *y* とします。*x* と *y* が両方とも欠損値ではない場合、次のように METHOD=の値と CRITERION=(γ)の値に従って値が不等と判断されます。

- METHOD=EXACT では、*y* と *x* が等しくない場合、値は不等です。
- METHOD=ABSOLUTE では、次の場合、値は不等です。

  $$\mathrm{ABS}(y - x) > \gamma$$

- METHOD=RELATIVE では、次の場合、値は不等です。

  $$\mathrm{ABS}(y - x)/((\mathrm{ABS}(x) + \mathrm{ABS}(y))/2 + \delta) > \gamma$$

  x=*y*=0 の場合、値は同等です。
- METHOD=PERCENT では、次の場合、値は不等です。

  $$100(\mathrm{ABS}(y - x)/\mathrm{ABS}(x)) > \gamma \quad \text{for} \quad x \neq 0$$

  または

  $$y \neq 0 \quad \text{for} \quad x = 0$$

x または *y* が欠損している場合、比較は NOMISSING オプションによって変わります。NOMISSING オプションが有効な場合、欠損値は常にいずれの値とも等しいと判断されます。NOMISSING オプションが無効な場合、欠損値は同じ種類の欠損値(すなわち、.=.、.^=.A、.A=.A、.A^=.B など)とのみ等しいと判断されます。

CRITERION=に対して指定した値が負の場合、実際に使用する基準 γ は、指定した基準の絶対値に、コンピュータの数値精度に応じたきわめて小さい数字 ε(イプシロン)を掛けた値と等しくなります。この数字 ε は、マシンで計算できる最小の正の浮動小数点値(1−ε<1<1+ε など)として定義されます。浮動小数点計算での四捨五入または切り捨てによる誤差は、通常 ε よりも数桁大きくなります。多くの場合、CRITERION=−1000 にすると、マシンレベルの精度で計算結果の同等性を合理的にテストできます。

RELATIVE メソッドで分母に加える値 δ は、METHOD=RELATIVE(δ)のように、メソッド名の後にかっこで囲んで指定します。METHOD=RELATIVE(δ)。METHOD=で指定をしない場合、δ はデフォルトで 0 になります。δ の値は、*x* と *y* の両方が非常に 0 に近い場合に、誤差の測定動作を制御するために使用されます。δ が与えられず、*x* と *y* の両方が非常に 0 に近い場合は、大きな相対誤差が生じます(限界は 2)。

δ の値を指定すると、RELATIVE メソッドが、小さな値に対して極度に反応することを避けられます。*x* と *y* が両方ともの絶対値よりもはるかに小さい場合に METHOD=RELATIVE(δ) CRITERION=γ を指定すると、METHOD=ABSOLUTE CRITERION=δγ を指定した場合と同じように比較されます。ただし、*x* または *y* のどちらかが δ の絶対値よりもはるかに大きい場合は、METHOD=RELATIVE CRITERION=γ の場合と同じように比較されます。*x* と *y*x と y の値が適度な場合は、METHOD=RELATIVE(δ) CRITERION=γ は、実質的には、METHOD=ABSOLUTE CRITERION=δ γ と METHOD=RELATIVE CRITERION=γ の比較になります。

文字変数については、一方の値が他方より長い場合、比較のために短い方の値にブランクが埋め込まれます。ブランク以外の文字値は、各文字が一致する場合のみ同等と判断されます。NOMISSING オプションが有効な場合、ブランクの文字値はいずれの値とも同等と判断されます。

差異とパーセント表示の差異の定義

PROC COMPARE は、値比較のレポートと OUT=データセットに、比較した数字の差異値とパーセント表示の差異値を表示します。この数量は、基準データセットの値を使用して、参照値として定義されます。数値変数を比較する場合、基準データセットの値

を x、比較データセットの値を y とします。x と y が両方とも欠損値ではない場合、差異およびパーセント表示の差異は、次のように定義されます。

- 差異 = $y - x$
- パーセント表示の差異 = $(y - x)/x * 100$ for $x \neq 0$
- パーセント表示の差異 = 欠損 $x = 0$.

PROC COMPARE での変数の出力形式の処理法

PROC COMPARE は、フォーマットされていない値を比較します。2 つのマッチング変数に異なるフォーマットされている場合、PROC COMPARE は変数の出力形式をリストにします。

構文: COMPARE プロシジャ

制限事項: WITH ステートメントの使用時には、VAR ステートメントも使用する必要があります。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化し、ログに適切に書き込まれるようにしてください。

LABEL、ATTRIB、FORMAT、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

PROC COMPARE <*option(s)*>;
  BY <DESCENDING> *variable-1*
    <<DESCENDING> *variable-2* …>
    <NOTSORTED>;
  ID <DESCENDING> *variable-1*
    <<DESCENDING> *variable-2* …>
    <NOTSORTED>;
  VAR *variable(s)*;
  WITH *variable(s)*;

ステートメント	タスク	例
“PROC COMPARE ステートメント”	SAS データセットのコンテンツを比較するか、または 2 つの変数を比較します。	Ex. 1, Ex. 2, Ex. 4, Ex. 6, Ex. 7
“BY ステートメント”	BY グループごとに個別の比較を行います。	
“ID ステートメント”	オブザベーションのマッチングに使用する変数を識別します。	Ex. 5
“VAR ステートメント”	比較を指定変数の値に制限します。	Ex. 2, Ex. 3, Ex. 4
“WITH ステートメント”	名前の異なる変数を比較します。	Ex. 2, Ex. 3, Ex. 4

ステートメント	タスク	例
“WITH ステートメント”	同一データセット内の 2 つの変数を比較します。	Ex. 4

PROC COMPARE ステートメント

2 つの SAS データセットのコンテンツ、異なるデータセットの選択変数、または同一データセット内の変数を比較します。

制限事項: COMPARE=を省略する場合は、WITH および VAR ステートメントを使用する必要があります。

PROC COMPARE は、比較するデータセットの片方または両方で RADIX によるアドレス指定ができない場合、別途エラーをレポートします。バージョン 6 の圧縮ファイルでは RADIX によるアドレス指定はできませんが、バージョン 7 からは圧縮ファイルで RADIX によるアドレス指定ができるようになりました。(データの整合性は損なわれません。本プロシジャは単にオブザベーションに異なる番号を付与するものです。

ヒント: BASE=オプションと COMPARE=オプションではデータセットオプションが使用できます。

例: “例 1: 差異についての包括的なレポートの作成” (375 ページ)
“例 2: 異なるデータセットでの変数の比較” (382 ページ)
“例 4: 同一データセット内での変数の比較” (386 ページ)
“例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)
“例 7: 統計量の出力データセット(OUTSTATS=)の作成” (397 ページ)

構文

PROC COMPARE <*option(s)*>;

オプション引数の要約

Control the details in the default report

ALLOBS
: すべてのマッチングオブザベーションの値を含めます。

ALLSTATS
: マッチング変数のすべてのペアについて要約統計量テーブルを印刷します。

ALLVARS
: すべてのマッチング変数の値と差異をレポートに含めます。

BRIEFSUMMARY
: 簡潔な比較要約のみを印刷します。

FUZZ=*number*
: 0 と 1 の間の数字のレポートを変更します。

MAXPRINT=*total* | (*per-variable, total*)
: 印刷される差異の数を制限します。

NODATE
: 作成日および最終変更日の印刷を抑制します。

NOPRINT
: すべての印刷出力を抑制します。

NOSUMMARY
データセット、変数、オブザベーション、値比較要約レポートを非表示にします。

NOVALUES
値比較結果レポートを非表示にします。

PRINTALL
値と差異の完全リストを作成します。

STATS
不等と判断されたマッチング数値変数のすべてのペアの要約統計量テーブルを印刷します。

TRANSPOSE
変数ごとではなくオブザベーションごとに値差異のレポートを印刷します。

Control the listing of variables and observations

LISTALL
一方のデータセットにしかない変数とオブザベーションをすべてリストに出力します。

LISTBASE
基準データセットにしかない変数とオブザベーションをすべてリストに出力します。

LISTBASEOBS
基準データセットにしかないオブザベーションをすべてリストに出力します。

LISTBASEVAR
一方のデータセットにしかない変数をすべてリストに出力します。

LISTCOMP
比較データセットにしかない変数とオブザベーションをすべてリストに出力します。

LISTCOMPOBS
比較データセットにしかないオブザベーションをすべてリストに出力します。

LISTCOMPVAR
比較データセットにしかない変数をすべてリストに出力します。

LISTEQUALVAR
値が同等と判断された変数をリストに出力します。

LISTOBS
一方のデータセットにしかないオブザベーションをすべてリストに出力します。

LISTVAR
一方のデータセットにしかない変数をすべてリストに出力します。

Control the output data set

OUT=*SAS-data-set*
出力データセットを作成します。

OUTALL
BASE=データセットと COMPARE=データセットのオブザベーションごとに 1 つずつオブザベーションを書き込みます。

OUTBASE
基準データセットの各オブザベーションに対するオブザベーションを書き込みます。

OUTCOMP
比較データセットのオブザベーションごとに 1 つずつオブザベーションを書き込みます。

OUTDIF
マッチングオブザベーションのペアごとに 1 つずつオブザベーションを出力データセットに書き込みます。

OUTNOEQUAL
すべての値が等しい場合、オブザベーションの書き込みを抑制します。

OUTPERCENT
マッチングオブザベーションのペアごとに 1 つずつオブザベーションを出力データセットに書き込みます。

Create an output data set that contains summary statistics

OUTSTATS=*SAS-data-set*
マッチング変数のすべてのペアの要約統計量を、指定した *SAS-data-set* に書き込みます。

Display a warning message in the SAS log

WARNING
差異が見つかると、SAS ログに警告メッセージを表示します。

Display an error message in the SAS log

ERROR
差異が見つかると、SAS ログにエラーメッセージを表示します。

Specify how the values are compared

CRITERION=γ
数値の同等性を判断する基準を指定します。

METHOD=ABSOLUTE | EXACT | PERCENT | RELATIVE<(δ)>
数値の同等性を判断するメソッドを指定します。

NOMISSBASE
基準データセットの欠損値をすべての値と同等であると判断します。

NOMISSCOMP
比較データセットの欠損値をすべての値と同等であると判断します。

NOMISSING
基準データセットと比較データセットの両方にある欠損値を、すべての値と同等であると判断します。

Specify the data sets to compare

BASE=*SAS-data-set*
基準データセットとして使用するデータセットを指定します。

COMPARE=*SAS-data-set*
比較データセットとして使用するデータセットを指定します。

Write notes to the SAS log

NOTE
比較結果を説明する注釈を SAS ログに表示します。

オプション引数

ALLOBS
すべてのマッチングオブザベーションについて、同等と判断された場合でも、値および(数値の場合は)差異を値比較結果のレポートに含めます。

デフォルト ALLOBS を省略すると、PROC COMPARE は不等と判断されたオブザベーションの値のみを印刷します。

操作 TRANSPOSE オプションと併せて指定すると、ALLOBS は ALLVARS オプションを呼び出して、すべてのマッチングオブザベーションとマッチング変数の値を表示します。

ALLSTATS
マッチング変数のすべてのペアについて要約統計量テーブルを印刷します。

参照項目 “要約統計量のテーブル” (369 ページ) (生成された統計量について)

ALLVARS
マッチング変数のすべてのペアについて、同等と判断された場合でも、値および(数値の場合は)差異を値比較結果のレポートに含めます。

デフォルト ALLVARS を省略すると、PROC COMPARE は不等と判断された変数の値のみを印刷します。

操作 TRANSPOSE オプションと併せて指定すると、ALLVARS は他のマッチング変数の値と前後関係のある不等値のみ表示します。TRANSPOSE オプションを省略すると、ALLVARS は ALLOBS オプションを呼び出して、すべてのマッチングオブザベーションとマッチング変数の値を表示します。

BASE=*SAS-data-set*
基準データセットとして使用するデータセットを指定します。

別名 DATA=

デフォルト 一番最近に作成された SAS データセット

ヒント WHERE=データセットオプションを BASE=オプションと併せて使用すると、比較に使用可能なオブザベーションを限定できます。

BRIEFSUMMARY
簡潔な比較要約のみを生成して、4 つのデフォルト要約レポート(データセット要約レポート、変数要約レポート、オブザベーション要約レポート、値比較要約レポート)を抑制します。

別名 BRIEF

ヒント デフォルトでは、要約レポートに伴って値差異のリストが作成されます。リストの作成を抑制するには、NOVALUES オプションを使用します。

例 “例 4: 同一データセット内での変数の比較” (386 ページ)

COMPARE=*SAS-data-set*
比較データセットとして使用するデータセットを指定します。

別名 COMP=、C=

デフォルト COMPARE=オプションを指定すると、比較データセットは基準データセットと同じになり、PROC COMPARE はデータセット内の変数を比較します。

制限事項 COMPARE=を省略する場合は、WITH ステートメントを使用する必要があります。

ヒント 比較に使用可能なオブザベーションを制限するには、WHERE=データセットオプションを COMPARE=とあわせて使用します。

CRITERION=γ
数値の同等性を判断する基準を指定します。通常、γ(ガンマ)の値は正です。この場合、その数自体が同等性の基準となります。γ に負の値を使用すると、PROC COMPARE は、SAS を実行しているコンピュータの精度に比例した同等性基準を使用します。

デフォルト 0.00001

参照項目 “同等性の基準” (343 ページ)

ERROR
差異が見つかると、SAS ログにエラーメッセージを表示します。

操作 このオプションは、WARNING オプションよりも優先されます。

FUZZ=*number*
number より小さい数字の値比較結果を変更します。PROC COMPARE は次のとおりに印刷します。

- *number* よりも小さいすべての変数値の代わりに 0 を印刷
- *number* よりも小さい差異またはパーセント表示の差異の代わりにブランクを印刷
- *number* よりも小さいすべての要約統計量のかわりに 0 を印刷

デフォルト 0

範囲 0 - 1

ヒント 些細な差異を多く含むレポートは、この形式の方が読みやすくなります。

LISTALL
一方のデータセットにしかない変数とオブザベーションをすべてリストに出力します。

別名 LIST

操作 LISTALL の処理は、LISTBASEOBS、LISTCOMPOBS、LISTBASEVAR、LISTCOMPVAR の 4 オプションを使用した場合と同じです。LISTBASEOBS、LISTCOMPOBS、LISTBASEVAR、LISTCOMPVAR。

LISTBASE
基準データセットにあって比較データセットにはないオブザベーションと変数をすべてリストに出力します。

操作 LISTBASE の処理は、LISTBASEOBS および LISTBASEVAR オプションを使用する場合と同じです。

LISTBASEOBS
基準データセットにあって比較データセットにはないオブザベーションをすべてリストに出力します。

LISTBASEVAR
基準データセットにあって比較データセットにはない変数をすべてリストに出力します。

LISTCOMP
比較データセットにあって基準データセットにはないオブザベーションと変数をすべてリストに出力します。

操作 LISTCOMP の処理は、LISTCOMPOBS および LISTCOMPVAR オプションを使用する場合と同じです。

LISTCOMPOBS
比較データセットにあって基準データセットにはないオブザベーションをすべてリストに出力します。

LISTCOMPVAR
比較データセットにあって基準データセットにはない変数をすべてリストに出力します。

LISTEQUALVAR
値が不等と判断された変数のデフォルトリストに加えて、すべてのオブザベーションで値が同等と判断された変数のリストを印刷します。

LISTOBS
一方のデータセットにしかないオブザベーションをすべてリストに出力します。

操作 LISTOBS の処理は、LISTBASEOBS および LISTCOMPOBS オプションを使用する場合と同じです。

LISTVAR
一方のデータセットにしかない変数をすべてリストに出力します。

操作 LISTVAR の処理は、LISTBASEVAR と LISTCOMPVAR の両オプションを使用する場合と同じです。

MAXPRINT=*total* | (*per-variable, total*)
印刷される差異の最大数を指定します。

total
印刷する差異の最大合計数です。デフォルト値は 500 ですが、ALLOBS オプション(または ALLVAR と TRANSPOSE の両オプション)を使用する場合は別です。その場合、デフォルトは 32000 になります。

per-variable
BY グループ内の変数ごとに印刷する差異の最大数です。デフォルト値は 50 ですが、ALLOBS オプション(または ALLVAR と TRANSPOSE の両オプション)を使用する場合は別です。その場合、デフォルトは 1000 になります。

MAXPRINT=オプションは、データセットが大きく異なる場合に、出力が極端に大きくならないようにします。

METHOD=ABSOLUTE | EXACT | PERCENT | RELATIVE<(δ)>
数値の同等性を判断するメソッドを指定します。定数 δ(デルタ)は、同等性測定の計算時に分母に加える値を指定する 0 と 1 の間の数です。デフォルトでは、δ は 0 です。

CRITERION=オプションを使用しなければ、デフォルトメソッドは EXACT です。CRITERION=オプションを使用する場合、デフォルトメソッドは RELATIVE(φ)です。この場合、φ(ファイ)は SAS を実行しているコンピュータの数値精度と CRITERION=の値に依存する小さい数です。

参照項目 “同等性の基準” (343 ページ)

NODATE
基準データセットと比較データセットの作成日と最終変更日を、データセット要約レポートに表示しないようにします。

NOMISSBASE
(デフォルトでは、欠損値は、.=.、.^=.A、.A=.A、.A^=.B などの同種の欠損値とのみ同等と判断されます)。

このオプションを使用すると、DATA ステップの UPDATE ステートメントで、基準データセットがマスタデータセットとして使用され、比較データセットがトランザクションデータセットとして使用されている場合に、比較データセットのオブザベーションに加える変更を決定できます。UPDATE ステートメントの詳細については、“UPDATE Statement” (*SAS Statements: Reference*)を参照してください。

NOMISSCOMP
比較データセットの欠損値をすべての値と同等であると判断します。(デフォルトでは、欠損値は、.=.、.^=.A、.A=.A、.A^=.B などの同種の欠損値とのみ同等と判断されます)。

このオプションを使用すると、DATA ステップの UPDATE ステートメントで、基準データセットがマスタデータセットとして使用され、比較データセットがトランザクションデータセットとして使用されている場合に、比較データセットのオブザベーションに加える変更を決定できます。UPDATE ステートメントの詳細については、“UPDATE Statement” (*SAS Statements: Reference*)を参照してください。

NOMISSING
基準データセットと比較データセットの両方にある欠損値を、すべての値と同等であると判断します。デフォルトでは、欠損値は、.=.、.^=.A、.A=.A、.A^=.B などの同種の欠損値とのみ同等と判断されます。

別名 NOMISS

操作 NOMISSING の処理は、NOMISSBASE と NOMISSCOMP の両方を使用する場合と同じです。

NOPRINT
すべての印刷出力を抑制します。

ヒント 出力データセットを 1 つ以上作成する場合に、このオプションを使用することがあります。

例 “例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)

NOSUMMARY
データセット、変数、オブザベーション、値比較要約レポートを非表示にします。

ヒント NOSUMMARY は、マッチング値に差異がない場合、出力を作成しません。

例 “例 2: 異なるデータセットでの変数の比較” (382 ページ)

NOTE
差異が見つかった場合、比較結果を説明する注釈を SAS ログに表示します。

NOVALUES
値比較結果レポートを非表示にします。

例 “概要:COMPARE プロシジャ” (340 ページ)

OUT=*SAS-data-set*
出力データセットを指定します。*SAS-data-set* が存在しない場合は、PROC COMPARE が作成します。*SAS-data-set* には、マッチング変数の差異が含まれます。

参照項目 “出力データセット(OUT=)” (373 ページ)

例 “例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)

OUTALL
基準データセットのオブザベーションと比較データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込みます。また、このオプションは、マッチングオブザベーションの値の差異およびパーセント表示の差異を含む出力データセットにもオブザベーションを書き込みます。

ヒント OUTALL の処理は、OUTBASE、OUTCOMP、OUTDIF、OUTPERCENT の 4 オプションを使用した場合と同じです。OUTBASE、OUTCOMP、OUTDIF、OUTPERCENT。

参照項目 “出力データセット(OUT=)” (373 ページ)

OUTBASE
基準データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込み、_TYPE_=BASE のオブザベーションを作成します。

参照項目 “出力データセット(OUT=)” (373 ページ)

例 “例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)

OUTCOMP
比較データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込み、_TYPE_=COMP のオブザベーションを作成します。

参照項目 “出力データセット(OUT=)” (373 ページ)

例 “例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)

OUTDIF
マッチングオブザベーションのペアごとに 1 つずつオブザベーションを出力データセットに書き込みます。オブザベーションの値には、オブザベーションのペアにおける値の差異を示す値が含まれます。各オブザベーションの_TYPE_の値は DIF です。

デフォルト OUTDIF オプションがデフォルトですが、OUTBASE、OUTCOMP または OUTPERCENT オプションを指定した場合は別です。これらのオプションのいずれかを使用する場合は、OUTDIF オプションを指定して、出力データセットに_TYPE_=DIF オブザベーションを作成する必要があります。

参照項目 “出力データセット(OUT=)” (373 ページ)

例 “例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)

OUTNOEQUAL
オブザベーションのすべての値が同等と判断された場合に、出力データセットへのオブザベーションの書き込みを抑制します。さらに、同等と判断された変数値も不等と判断された変数値も含まれるオブザベーションでは、OUTNOEQUAL オプションが特殊欠損値".E"を使用して、同等と判断された変数の差異とパーセント表示の差異を示します。

参照項目 “出力データセット(OUT=)” (373 ページ)

例 “例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)

OUTPERCENT
マッチングオブザベーションのペアごとに 1 つずつオブザベーションを出力データセットに書き込みます。オブザベーションの値には、オブザベーションのペアにおける値のパーセント表示の差異を示す値が含まれます。各オブザベーションの _TYPE_の値は PERCENT です。

参照項目 “出力データセット(OUT=)” (373 ページ)

OUTSTATS=*SAS-data-set*
マッチング変数のすべてのペアの要約統計量を、指定した *SAS-data-set* に書き込みます。

ヒント プロシジャ出力の統計量テーブルを印刷する場合は、STATS、ALLSTATS または PRINTALL オプションを使用します。

参照項目 “出力統計量データセット(OUTSTATS=)” (374 ページ)

“要約統計量のテーブル” (369 ページ)

例 “例 7: 統計量の出力データセット(OUTSTATS=)の作成” (397 ページ)

PRINTALL
オプション ALLVARS、ALLOBS、ALLSTATS、LISTALL、WARNING を呼び出します。ALLVARS、ALLOBS、ALLSTATS、LISTALL、WARNING。

例 “例 1: 差異についての包括的なレポートの作成” (375 ページ)

STATS
不等と判断されたマッチング数値変数のすべてのペアの要約統計量テーブルを印刷します。

参照項目 “要約統計量のテーブル” (369 ページ) (生成された統計量について)

TRANSPOSE
変数ごとではなくオブザベーションごとに値差異のレポートを印刷します。

操作 NOVALUES オプションも使用すると、TRANSPOSE オプションは、各オブザベーションについて値が不等と判断された変数の *names* のみをリストに出力し、値と差異は出力しません。

参照項目 “オブザベーションの比較結果(TRANSPOSE オプション使用)” (371 ページ).

WARNING
差異が見つかると、SAS ログに警告メッセージを表示します。

操作 ERROR オプションが WARNING オプションよりも優先されます。

BY ステートメント

BY グループごとに個別の比較を行います。

参照項目: “BY” (68 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* …>
<NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは指定するすべての変数によって並べ替える必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
オブザベーションが BY ステートメントの DESCENDING の直後に続く変数別に降順で並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。オブザベーションは別の方法(時系列など)でグループ化されます。

注 BY 変数の値によるオブザベーションの並べ替えの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

詳細

BY 処理と PROC COMPARE

BY ステートメントを PROC COMPARE とあわせて使用するには、基準データセットと比較データセットの両方を BY 変数によって並べ替える必要があります。すべての BY 変数が比較データセット内に存在するかどうか、また、存在する場合は、その属性

が基準データセット内の BY 変数の 1 つと一致するかどうかによって、比較の性質は異なります。次の表に、さまざまな状況下での PROC COMPARE の動作を示します。

表 12.1 さまざまな状況下における PROC COMPARE の動作

条件	PROC COMPARE の動作
すべての BY 変数が比較データセットに存在し、すべての属性が完全に一致する。	対応する BY グループを比較します。
比較データセットに BY 変数が 1 つもない。	基準データセットの各 BY グループを比較データセット全体と比較します。
一部の BY 変数が比較データセットに存在しない。	SAS ログにエラーメッセージを書き込んで、終了します。
一部の BY 変数が 2 つのデータセットでそれぞれ種類が異なる。	SAS ログにエラーメッセージを書き込んで、終了します。

ID ステートメント

オブザベーションのマッチングに使用する変数のリストを表示します。

参照項目: “ID 変数を使用した比較” (342 ページ)

例: “例 5: ID 変数を利用し、オブザベーションを比較する” (388 ページ)

構文

ID <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* …>
<NOTSORTED>;

必須引数

variable
プロシジャがオブザベーションのマッチングに使用する変数を指定します。複数の変数を指定できますが、データセットは、1 つまたは複数の指定変数順に並べ替える必要があります。この変数が *ID 変数*です。ID 変数は、印刷したレポートや出力データセットのオブザベーションも識別します。

オプション引数

DESCENDING
データセットが ID ステートメントで文字 DESCENDING の直後に続く変数別に降順で並べ替えられるように指定します。

DESCENDING オプションを使用する場合は、データセットを並べ替える必要があります。DESCENDING オプションを指定した ID ステートメントの処理には、インデックスは使用されません。さらに、ID 変数に対する DESCENDING の使用は、データセットを並べ替えるために使用した PROC SORT ステップにおける BY ステートメントの DESCENDING オプションの使用と対応している必要があります。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。データは別の方法(時系列など)でグループ化されます。

参照項目 “並べ替えられていないデータの比較” (357 ページ)

詳細

ID 変数の必要条件

- ID 変数が BASE=データセットに存在する必要があります。存在しなければ、PROC COMPARE は処理を停止します。
- ID 変数が COMPARE=データセットになければ、PROC COMPARE は SAS ログに警告メッセージを書き込み、その変数は使用せずに比較データセットのオブザベーションをマッチングします(ただし、OUT=データセットには書き込みます)。
- 両方のデータセットで ID 変数の種類を同一にする必要があります。
- NOTSORTED オプションを指定する場合を除いて、両方のデータセットを(もしあれば BY 変数内の)共通の ID 変数で並べ替える必要があります。

並べ替えられていないデータの比較

データを ID 変数で並べ替えない場合は、NOTSORTED オプションを使用できます。NOTSORTED オプションを指定した場合、または ID ステートメントを省略した場合、PROC COMPARE はオブザベーションを 1 対 1 でマッチングします。つまり、PROC COMPARE は、基準データセットの 1 番目のオブザベーションと比較データセットの 1 番目のオブザベーション、2 番目と 2 番目、というようにマッチングしていきます。NOTSORTED の使用時に、対応するオブザベーションの ID 値が同一ではなかった場合、PROC COMPARE はエラーメッセージを印刷して処理を停止します。

データセットを共通の ID 変数で並べ替えず、なおかつ NOTSORTED オプションを指定していない場合、PROC COMPARE は SAS ログに警告メッセージを書き込み、NOTSORTED の指定時と同様にデータセットの処理を続行します。

重複 ID 値の回避

各データセットのオブザベーションは、ID 変数値によって一意にラベル付けされている必要があります。PROC COMPARE は、データセットで同じ ID 値のオブザベーションを 2 つ連続で発見すると、次の処理を実行します。

- そのデータセットでの最初の発生時に警告 `Duplicate Observations` を印刷します。
- オブザベーション要約レポートに、データセットで発見された重複オブザベーションの合計数を印刷します。
- 基準データセットと比較データセットの重複オブザベーションを使用して、1 対 1 ベースでオブザベーションを比較します。

データセットが並べ替えられていない場合、PROC COMPARE は連続で発生した重複オブザベーションのみを検出します。

VAR ステートメント

変数値の比較を、VAR ステートメントで指定した変数に限定します。

例: “例 2: 異なるデータセットでの変数の比較” (382 ページ)
“例 3: 変数を複数回比較する” (384 ページ)

“例 4: 同一データセット内での変数の比較” (386 ページ)

構文

VAR *variable(s)*;

必須引数

variable(s)
BASE=データセットと COMPARE=データセットの両方か、または BASE=データセットにのみ出現する 1 つ以上の変数。

詳細

- VAR ステートメントを使用しない場合、PROC COMPARE は、BY ステートメントと ID ステートメントに記述された変数を除く、すべてのマッチング変数の値を比較します。
- VAR ステートメントの変数が COMPARE=データセットに存在しない場合、PROC COMPARE は SAS ログに警告メッセージを書き込み、その変数を無視します。
- VAR ステートメントの変数が BASE=データセットに存在しない場合、PROC COMPARE は処理を停止して、SAS ログにエラーメッセージを書き込みます。
- VAR ステートメントは、マッチング変数の値の比較のみを制限します。PROC COMPARE はそれでもマッチング変数の合計数とその属性の比較をレポートします。ただし、これらの変数についてはエラーメッセージも警告メッセージも生成されません。

WITH ステートメント

基準データセットの変数と比較データセットの異なる名前の変数を比較します。また、同一データセット内の異なる変数を比較します。

制限事項: WITH ステートメントの使用時には、VAR ステートメントも使用する必要があります。

例: “例 2: 異なるデータセットでの変数の比較” (382 ページ)
“例 3: 変数を複数回比較する” (384 ページ)
“例 4: 同一データセット内での変数の比較” (386 ページ)

構文

WITH *variable(s)*;

必須引数

variable(s)
VAR ステートメントの変数と比較する 1 つ以上の変数。

詳細

選択された変数の比較

基準データセットの変数を比較データセットの異なる名前の変数と比較する場合は、VAR ステートメントで基準データセットの変数名を指定し、WITH ステートメントでマッ

チング変数名を指定します。WITH 変数リストに指定する最初の変数と VAR 変数リストに指定する最初の変数、2 番目と 2 番目、というように対応します。WITH 変数リストが VAR 変数リストよりも短い場合、PROC COMPARE は、VAR ステートメントの余った変数が、比較データセットでも基準データセットでも同じ名前であるとみなします。WITH 変数リストが VAR 変数リストよりも長い場合、PROC COMPARE は余った変数を無視します。

VAR ステートメントまたは WITH ステートメントでは 1 つの変数名を何度でも記述できます。VAR 変数リストおよび WITH 変数リストの選択によって、変数をどんな順列でも比較できます。

PROC COMPARE ステートメントで COMPARE=オプションを省略する場合は、WITH ステートメントを使用する必要があります。この場合、PROC COMPARE は、BASE=データセットにある異なる名前の変数の値を比較します。

PROC COMPARE 出力のカスタマイズ

PROC COMPARE は非常に長い出力を生成します。オプションを 1 つ以上使用すると、比較の種類やレポートの詳細度を決定できます。たとえば、次の PROC COMPARE ステップの NOVALUES オプションは、マッチング変数の値の差異を示す出力部分を非表示にします。

```
options nodate pageno=1 linesize=80 pagesize=40;

title 'The SAS System';

proc compare base=proclib.one
             compare=proclib.two novalues;
run;
```

アウトプット 12.1 *WORK.ONE と WORK.TWO の比較(その 1)*

The SAS System

```
                        The COMPARE Procedure
                  Comparison of WORK.ONE with WORK.TWO
                             (Method=EXACT)

                           Data Set Summary

Dataset            Created          Modified  NVar    NObs  Label

WORK.ONE   16MAR11:16:43:16  16MAR11:16:43:16     5       4  First Data Set
WORK.TWO   16MAR11:16:43:16  16MAR11:16:43:16     6       5  Second Data Set


                           Variables Summary

            Number of Variables in Common: 5.
            Number of Variables in WORK.TWO but not in WORK.ONE: 1.
            Number of Variables with Conflicting Types: 1.
            Number of Variables with Differing Attributes: 3.
```

```
            Listing of Common Variables with Conflicting Types

                     Variable  Dataset   Type  Length

                     student   WORK.ONE  Num        8
                               WORK.TWO  Char       8


           Listing of Common Variables with Differing Attributes

          Variable  Dataset   Type  Length  Format  Label

          year      WORK.ONE  Char       8          Year of Birth
                    WORK.TWO  Char       8
          state     WORK.ONE  Char       8
                    WORK.TWO  Char       8          Home State
```

アウトプット 12.2 *WORK.ONE と WORK.TWO の比較(その 2)*

The SAS System

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Listing of Common Variables with Differing Attributes

Variable	Dataset	Type	Length	Format	Label
gr1	WORK.ONE	Num	8	4.1	
	WORK.TWO	Num	8	5.2	

Observation Summary

Observation	Base	Compare
First Obs	1	1
First Unequal	1	1
Last Unequal	4	4
Last Match	4	4
Last Obs	.	5

Number of Observations in Common: 4.
Number of Observations in WORK.TWO but not in WORK.ONE: 1.
Total Number of Observations Read from WORK.ONE: 4.
Total Number of Observations Read from WORK.TWO: 5.

Number of Observations with Some Compared Variables Unequal: 4.
Number of Observations with All Compared Variables Equal: 0.

アウトプット 12.3 *WORK.ONE と WORK.TWO の比較(その 3)*

The SAS System

```
                    The COMPARE Procedure
                Comparison of WORK.ONE with WORK.TWO
                          (Method=EXACT)

                    Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 3.
Total Number of Values which Compare Unequal: 6.
Maximum Difference: 20.


                 Variables with Unequal Values

      Variable  Type  Len   Compare Label  Ndif   MaxDif

      state     CHAR    8   Home State        2
      gr1       NUM     8                     2    1.000
      gr2       NUM     8                     2   20.000
```

“プロシジャの出力” (364 ページ) では、この 2 つのデータセットのデフォルト出力が示されます。“例 1: 差異についての包括的なレポートの作成” (375 ページ)では、この 2 つのデータセットの完全出力が示されます。

結果:COMPARE プロシジャ

結果レポート

PROC COMPARE は、次の方法で比較結果をレポートします。

- SAS ログ
- 自動マクロ SYSINFO に格納されたリターンコード
- プロシジャの出力
- 出力データセット

SAS ログ

WARNING、PRINTALL または ERROR オプションを使用する場合、PROC COMPARE は差異の説明を SAS ログに書き込みます。

マクロリターンコード(SYSINFO)

PROC COMPARE は自動マクロ変数 SYSINFO にリターンコードを格納します。リターンコードの値によって、比較結果についての情報が提供されます。SAS マクロは、PROC COMPARE の実行後、他のステップが開始される前に、SYSINFO の値をチェ

ックすることにより、PROC COMPARE ステップの結果を使用して、どんなアクションを取るべきか、または SAS プログラムのどの部分を実行するべきかを決定できます。

PROC COMPARE からの SYSINFO リターンコードを解釈するためのキーを次の表に示します。リストの各条件が真の場合、関連する値がリターンコードに追加されます。したがって、次の表に記載された該当条件のコードの合計が SYSINFO リターンコードになります。

表 12.2 *マクロリターンコード*

ビット	条件	コード	16 進	説明
1	DSLABEL	1	0001X	データセットラベルが異なります。
2	DSTYPE	2	0002X	データセットの種類が異なります。
3	INFORMAT	4	0004X	変数の入力形式が異なります。
4	FORMAT	8	0008X	変数の出力形式が異なります。
5	LENGTH	16	0010X	変数の長さが異なります。
6	LABEL	32	0020X	変数のラベルが異なります。
7	BASEOBS	64	0040X	基準データセットに、比較データセットにないオブザベーションがあります。
8	COMPOBS	128	0080X	比較データセットに、基準データセットにないオブザベーションがあります。
9	BASEBY	256	0100X	基準データセットに、比較データセットにない BY グループがあります。
10	COMPBY	512	0200X	比較データセットに、基準データセットにない BY グループがあります。
11	BASEVAR	1024	0400X	基準データセットに、比較データセットにない変数があります。
12	COMPVAR	2048	0800X	比較データセットに、基準データセットにない変数があります。
13	VALUE	4096	1000X	値の比較が不等でした。
14	TYPE	8192	2000X	変数の種類が一致しません。
15	BYVAR	16384	4000X	BY 変数が一致しません。
16	ERROR	32768	8000X	致命的なエラー:比較は行われません。

これらのコードは、データセットの差異の度合いを簡単にチェックできるよう段階的に並べられています。たとえば、同じ変数、オブザベーションおよび値を含む 2 つのデータセットをチェックするが、ラベルや出力形式などの差異について考慮しない場合は、次のステートメントを使用します。

```
proc compare base=SAS-data-set
             compare=SAS-data-set;
run;
```

```
%if &sysinfo >= 64 %then
   %do;
      handle error;
   %end;
```

DATA ステップのビットテスト機能を使用して SYSINFO 値の個々のビットを検証すると、特定の状況についてチェックできます。たとえば、比較データセットにはない基本データセットのオブザベーションの存在をチェックするには、次のステートメントを使用します。

```
proc compare base=SAS-data-set
             compare=SAS-data-set;
run;

%let rc=&sysinfo;
data _null_;
   if &rc='1......'b then
      put 'Observations in Base but not
           in Comparison Data Set';
run;
```

SYSINFO をチェックする前に PROC COMPARE を実行し、別の SAS ステップが開始する前に SYSINFO 値を取得する必要があります。SAS ステップはどれも SYSINFO をリセットするからです。

プロシジャの出力

プロシジャ出力の概要

次のセクションでは、“概要:COMPARE プロシジャ” (340 ページ)で示した 2 つのデータセットのデフォルト出力を示して説明します。PROC COMPARE は非常に長い出力を生成するので、出力は 7 つに分けて提示されます。

```
options nodate pageno=1 linesize=80 pagesize=60;
proc compare base=proclib.one compare=proclib.two;
  run;
```

データセット要約

このレポートは、比較されているデータセットの属性をリストにします。これらの属性には、次が含まれます。

- データセット名
- もしあればデータセットの種類
- もしあればデータセットラベル
- 作成日と最終変更日
- 各データセットの変数の数
- 各データセットのオブザベーションの数

Data Set Summary を閲覧するには、次の出力例を参照してください。データセット要約と変数要約の出力の一部を示しています。(COMPARE プロシジャは、ページサイズが小さすぎた場合は、データセットラベルを省略します)。

変数要約

このレポートでは、2 つのデータセットの変数が比較されます。レポートの最初の部分には、次が列挙されます。

- データセットに共通の変数の数
- 比較データセットにはなくて基準データセットにはある変数、およびその逆の変数の数
- 両データセットで種類が異なる変数の数
- その他の属性(長さ、ラベル、出力形式、入力形式)が異なる変数の数
- 比較のために指定した BY、ID、VAR、WITH 変数の数

レポートの 2 番目の部分には、属性の異なるマッチング変数のリストと、その属性の差異が示されます。(COMPARE プロシジャは、ページサイズが小さすぎた場合は、変数ラベルを省略します)。

次の出力は、データセット要約と変数要約を示しています。

アウトプット 12.4 *データセット要約と変数要約を示す出力の一部*

The SAS System

```
                        The COMPARE Procedure
                  Comparison of WORK.ONE with WORK.TWO
                             (Method=EXACT)

                           Data Set Summary

Dataset            Created          Modified  NVar    NObs  Label

WORK.ONE  18MAR11:11:22:21  18MAR11:11:22:21     5       4  First Data Set
WORK.TWO  18MAR11:11:22:22  18MAR11:11:22:22     6       5  Second Data Set


                           Variables Summary

            Number of Variables in Common: 5.
            Number of Variables in WORK.TWO but not in WORK.ONE: 1.
            Number of Variables with Conflicting Types: 1.
            Number of Variables with Differing Attributes: 3.
```

```
          Listing of Common Variables with Conflicting Types

                   Variable  Dataset   Type  Length

                   student   WORK.ONE  Num        8
                             WORK.TWO  Char       8


           Listing of Common Variables with Differing Attributes

          Variable  Dataset   Type  Length  Format  Label

          year      WORK.ONE  Char       8          Year of Birth
                    WORK.TWO  Char       8
          state     WORK.ONE  Char       8
                    WORK.TWO  Char       8          Home State
          gr1       WORK.ONE  Num        8  4.1
                    WORK.TWO  Num        8  5.2
```

オブザベーション要約

このレポートでは、基準データセットと比較データセットのオブザベーションについての情報が提供されます。最初に、各データセットの最初と最後のオブザベーション、最初と最後のマッチングオブザベーション、および最初と最後の異なるオブザベーションが示されます。それから、レポートに次が列挙されます。

- データセットに共通のオブザベーションの数
- 比較データセットにはなくて基準データセットにはあるオブザベーション、およびその逆のオブザベーションの数
- 各データセットのオブザベーションの合計数

- PROC COMPARE によって一部の変数が不等と判断されたマッチングオブザベーションの数
- PROC COMPARE によってすべての変数が同等と判断されたマッチングオブザベーションの数

次の出力は、オブザベーション要約を示しています。

アウトプット 12.5 *オブザベーション要約を示す出力の一部*

```
                        Observation Summary

                    Observation      Base  Compare

                    First Obs           1        1
                    First Unequal       1        1
                    Last  Unequal       4        4
                    Last  Match         4        4
                    Last  Obs           .        5

      Number of Observations in Common: 4.
      Number of Observations in WORK.TWO but not in WORK.ONE: 1.
      Total Number of Observations Read from WORK.ONE: 4.
      Total Number of Observations Read from WORK.TWO: 5.

      Number of Observations with Some Compared Variables Unequal: 4.
      Number of Observations with All Compared Variables Equal: 0.
```

値の比較の要約

このレポートにはまず、次が列挙されます。

- すべてのオブザベーションが同等な比較変数の数
- 一部のオブザベーションが不等な比較変数の数
- もしあれば欠損値を含む差異のある変数の数
- 不等と判断された値の合計数
- マッチング変数のすべてのペアの不等な値の最大差異測定(欠損値を含まない差異を対象とする)

さらに、一部のマッチングオブザベーションに不等な値がある変数について、レポートに次が列挙されます。

- 変数の名前
- 他の変数の属性
- PROC COMPARE がその変数を不等と判断した回数
- 値間に見られる最大差異測定(欠損値を含まない差異を対象とする)
- もしあれば、欠損値を含む比較によって起こる差異の数

次の出力は、値の比較の要約を示しています。

アウトプット 12.6 *値の比較の要約を示す出力の一部*

The SAS System

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 3.
Total Number of Values which Compare Unequal: 6.
Maximum Difference: 20.

Variables with Unequal Values

Variable	Type	Len	Compare Label	Ndif	MaxDif
state	CHAR	8	Home State	2	
gr1	NUM	8		2	1.000
gr2	NUM	8		2	20.000

値の比較の結果

このレポートは、1 つ以上のオブザベーションで不等と判断されたマッチング変数の各ペアの表で構成されています。PROC COMPARE は、文字値の比較時は、最初の 20 文字のみ表示します。TRANSPOSE オプションを使用した場合は、最初の 12 文字のみ表示します。各表には、次が表示されます。

- オブザベーション番号、または ID 変数の値(ID ステートメントを使用している場合)
- 基準データセットの変数の値
- 比較データセットの変数の値
- 2 つの値の差異(数値変数のみ)
- 2 つの値のパーセント表示の差異(数値変数のみ)

次の出力は、変数の値比較結果を示しています。

アウトプット 12.7 変数の値比較結果を示す出力の一部

Value Comparison Results for Variables

Obs	Home State Base Value state	Compare Value state
2	MD	MA
4	MA	MD

Obs	Base gr1	Compare gr1	Diff.	% Diff
1	85.0	84.00	-1.0000	-1.1765
3	78.0	79.00	1.0000	1.2821

Obs	Base gr2	Compare gr2	Diff.	% Diff
3	72.0000	73.0000	1.0000	1.3889
4	94.0000	74.0000	-20.0000	-21.2766

NOVALUES オプションを使用して値比較結果を非表示にできます。NOVALUES オプションと TRANSPOSE オプションの両方を使用すると、PROC COMPARE は、値が不等と判断された変数の名前をオブザベーションごとにリストにしますが、値および差異は表示しません。

要約統計量のテーブル

STATS、ALLSTATS または PRINTALL オプションを使用すると、変数の値比較結果セクションに、比較されている数値変数の要約統計量が含まれます。STATS オプションは、値が不等と判断された数値変数についてのみこの統計量を生成します。ALLSTATS および PRINTALL オプションは、すべての値が同等と判断されても、すべての数値変数についてこの統計量を生成します。

注: いかなる場合でも、PROC COMPARE は、不等な値を含むマッチングオブザベーションだけでなく、欠損値を含まないすべてのマッチングオブザベーションに基づいて、要約統計量を計算します。

次の出力には、次に示す基準データセット値、比較データセット値、差異、パーセント表示の差異の要約統計量が表示されます。

N
: 非欠損値の数。

MEAN
値の平均値。

STD
標準偏差。

MAX
最大値。

MIN
最小値。

MISSDIFF
基準データセットと比較データセットのどちらかの欠損値数。

STDERR
平均値の標準誤差。

T
T 比率(MEAN/STDERR)。

PROB> | T |
真の母平均が 0 の場合に、絶対 T 値がより大きくなる確率。

NDIF
不等と判断されたマッチングオブザベーションの数、および不等と判断されたマッチングオブザベーションの割合。

DIFMEANS
基準値の平均と比較値の平均の差異。この行には 3 つの数字が含まれます。1 番目は、基準値平均のパーセント表示平均。2 番目は、比較値平均のパーセント表示平均。3 番目は、2 つの平均値の差異(比較平均値から基準平均値を引く)。

R
どちらのデータセットでも非欠損なマッチングオブザベーションの基準値と比較値の相関係数。

RSQ
どちらのデータセットでも非欠損なマッチングオブザベーションの基準値と比較値の相関係数の 2 乗。

次の出力は、ALLSTATS オプションで“概要”に示した 2 つのデータセットを使用した結果です。

```
options nodate pageno=1
   linesize=80 pagesize=60;
proc compare base=proclib.one
     compare=proclib.two allstats;
   title 'Comparing Two Data Sets: Default Report';
run;
```

アウトプット 12.8 変数の値比較結果を示す出力の一部

Value Comparison Results for Variables

Obs	Home State Base Value state	Compare Value state
2	MD	MA
4	MA	MD

Obs	Base gr1	Compare gr1	Diff.	% Diff
1	85.0	84.00	-1.0000	-1.1765
3	78.0	79.00	1.0000	1.2821
N	4	4	4	4
Mean	85.5000	85.5000	0	0.0264
Std	5.8023	5.4467	0.8165	1.0042
Max	92.0000	92.0000	1.0000	1.2821
Min	78.0000	79.0000	-1.0000	-1.1765
StdErr	2.9011	2.7234	0.4082	0.5021
t	29.4711	31.3951	0.0000	0.0526
Prob>|t|	<.0001	<.0001	1.0000	0.9614
Ndif	2	50.000%		
DifMeans	0.000%	0.000%	0	
r, rsq	0.991	0.983		

注: PRINTALL でページサイズを大きくすると、PROC COMPARE は、文字変数の値比較結果を数値変数の結果の隣に印刷します。その場合、PROC COMPARE は、文字変数の NDIF のみ計算します。

オブザベーションの比較結果(TRANSPOSE オプション使用)

TRANSPOSE オプションは、比較結果を、変数別ではなくオブザベーション別に印刷します。比較結果は、オブザベーション要約レポートの前に置かれます。デフォルトでは、表の各行の値のソースが次のラベルで表示されます。

```
_OBS_1=number-1  _OBS_2=number-2
```

number-1 は変数の値が示されているデータベースセットのオブザベーションで、*number-2* は、比較データセットのオブザベーション数です。

次の出力では、変数ではなくオブザベーションに関する、PROCLIB.ONE と PROCLIB.TWO との差を示しています。

```
options nodate pageno=1
   linesize=80 pagesize=60;
proc compare base=proclib.one
     compare=proclib.two transpose;
   title 'Comparing Two Data Sets: Default Report';
run;
```

アウトプット 12.9 *オブザベーションの比較結果を示す出力の一部*

Comparing Two Data Sets: Default Report

```
                        The COMPARE Procedure
                  Comparison of WORK.ONE with WORK.TWO
                              (Method=EXACT)

                  Comparison Results for Observations

      _OBS_1=3 _OBS_2=3:
      Variable    Base Value       Compare          Diff.        % Diff
           gr1          78.0          79.00      1.000000      1.282051
           gr2     72.000000     73.000000      1.000000      1.388889

      _OBS_1=4 _OBS_2=4:
      Variable    Base Value       Compare          Diff.        % Diff
           gr2     94.000000     74.000000    -20.000000    -21.276596
         state            MA            MD
```

ID ステートメントを使用すると、ラベルの識別は次の形式になります。

ID-1=ID-value-1 ... ID-n=ID-value-n

この場合、*ID* は ID 変数名で、*ID-value* は ID 変数値です。

注: TRANSPOSE オプションを使用した場合、PROC COMPARE は値の最初の 12 文字のみ印刷します。

ODS テーブル名

COMPARE プロシジャは、作成する各テーブルに名前を割り当てます。これらの名前を使用して、Output Delivery System (ODS)を使用してテーブルを選択し、出力データセットを作成する際にそのテーブルを参照できます。詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

表 12.3 *COMPARE プロシジャによって生成される ODS テーブル*

テーブル名	説明	テーブル生成時の条件
CompareData sets	1 つまたは複数のデータセットについての情報	デフォルトでは、NOSUMMARY または NOVALUES オプションが指定されていない場合
CompareDetails (オブザベーションの比較結果)	基準データデットと比較データセットに共通しないオブザベーションのリスト	PRINTALL オプションが指定されている場合

テーブル名	説明	テーブル生成時の条件
CompareDetails (ID 変数の注記と警告)	重複 ID 変数値に関する注記と警告のリスト	ID ステートメントが指定され、重複 ID 変数値がどちらかのデータセットに存在する場合
CompareDifferences	変数値差異のレポート	デフォルトでは、NOVALUES オプションが指定されていない場合
CompareSummary	不等値のオブザベーション、値および変数の要約レポート	デフォルト
CompareVariables	基準データセットと比較データセットの変数の種類または属性の差異のリスト	デフォルトでは、値が同一ではないか、または NOSUMMARY オプションが指定されていない場合

出力データセット(OUT=)

デフォルトでは、OUT=データセットには、マッチングオブザベーションのペアごとに 1 つずつオブザベーションが含まれます。OUT=データセットには、比較しているデータセットから次の変数が含まれます。

- BY ステートメントで指定したすべての変数
- ID ステートメントで指定したすべての変数
- すべてのマッチング変数か、または(VAR ステートメントを使用した場合は)VAR ステートメントに記述したすべての変数

さらに、データセットには、PROC COMPARE がマッチング変数値のソースを識別するために作成する 2 つの変数_TYPE_および_OBS_も含まれます。_TYPE_ and _OBS_.

TYPE
は長さ 8 の文字変数です。この値によって、そのオブザベーションのマッチング(VAR)変数の値のソースが識別されます。(比較されない ID 変数と BY 変数については、元のデータセットからの値がこの値になります)。_TYPE_のラベルは **`Type of Observation`** です。この変数の 4 つの可能な値は次のとおりです。

BASE
このオブザベーションの値の出所は、基準データセットのオブザベーションです。PROC COMPARE は、OUTBASE オプションを指定すると、OUT=データセットにこの種類のオブザベーションを書き込みます。

COMPARE
このオブザベーションの値の出所は、比較データセットのオブザベーションです。PROC COMPARE は、OUTCOMP オプションを指定すると、OUT=データセットにこの種類のオブザベーションを書き込みます。

DIF
このオブザベーションの値は、基準データセットと比較データセットの値の差異です。文字変数の場合、PROC COMPARE は、同等な文字を表すにはピリオド(.)、不等な文字を表すには X を使用します。PROC COMPARE は、デフォルトで、OUT=データセットにこの種類のオブザベーションを書き込みます。ただし、OUTBASE、OUTCOMP または OUTPERCENT オプションを指定して他の種類のオブザベーションを要求する場合は、OUT=データセットにこの種類のオブザベーションを生成するには OUTDIF オプションを指定する必要があります。

PERCENT
このオブザベーションの値は、基準データセットと比較データセットの値のパーセント表示の差異です。文字変数の場合、種類 PERCENT のオブザベーションの値は、種類 DIF のオブザベーションの値と同じになります。

OBS
OUT=オブザベーションのソースをさらに識別する番号を含む数値変数です。

_TYPE_が BASE に等しいオブザベーションの場合、_OBS_は、VAR 変数値のコピー元の基準データセットのオブザベーション番号になります。同様に、_TYPE_が COMPARE に等しいオブザベーションの場合、_OBS_は、VAR 変数値のコピー元の比較データセットのオブザベーション番号になります。

_TYPE_が DIF または PERCENT に等しいオブザベーションの場合、_OBS_は、BY グループのマッチングオブザベーションに振られた シーケンス番号になります。

_OBS_のラベルは **Observation Number** です。

COMPARE プロシジャは、VAR 変数の長さを除き、ベースデータセットから OUT=データセットの変数名と属性を取得します。COMPARE プロシジャは VAR 変数のより長いほうの長さを使用します。これは、その長さがどのデータセットから取得されるかに関係ありません。この動作によって 2 つの重要な影響が生じます。

- VAR ステートメントと WITH ステートメントを使用すると、OUT=データセットの変数名は VAR ステートメントから取得されます。したがって、_TYPE_が **BASE** に等しいオブザベーションには VAR 変数の値が含まれ、一方、_TYPE_が **COMPARE** に等しいオブザベーションは WITH 変数の値が含まれます。
- 1 つの変数を複数の変数と比較するために、VAR ステートメントにその変数を複数含めた場合、PROC COMPARE が OUT=データセットに含められるのは最初の比較のみです。これは、各変数の名前を一意にする必要があるためです。その他の比較については、警告メッセージが生成されます。

OUT=オプションの例については、“例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)を参照してください。

出力統計量データセット(OUTSTATS=)

OUTSTATS=オプションを使用すると、PROC COMPARE は、比較する数値変数の各ペアに対して、ALLSTATS オプションと同じように要約統計量を計算します(“要約統計量のテーブル” (369 ページ)を参照)。OUTSTATS=データセットには、各変数ペアの各要約統計量ごとに 1 オブザベーションが含まれます。また、データセットには、比較に使用された BY 変数、および PROC COMPARE によって作成された複数の変数も含まれます。

VAR
オブザベーションの統計量の計算対象となった基準データセットの変数名が含まれる文字変数です。

WITH
オブザベーションの統計量の計算対象となった比較データセットの変数名が含まれる文字変数です。_WITH_変数は、WITH ステートメントを使用しない限り、OUTSTATS=データセットには含まれません。

TYPE
オブザベーションに含まれる統計量名を含む文字変数です。_TYPE_変数の値には、**N**、**MEAN**、**STD**、**MIN**、**MAX**、**STDERR**、**T**、**PROBT**、**NDIF**、**DIFMEANS**、**R**、**RSQ** があります。

BASE
比較データセットにマッチングオブザベーションがある基準データセットのオブザベーションの_VAR_によって指定された変数の値から計算された統計量の値が含まれる数値変数です。

COMP
基準データセットにマッチングオブザベーションがある比較データセットのオブザベーションの_VAR_変数によって(WITH ステートメントを使用している場合は _WITH_変数によって)指定された変数の値から計算された統計量の値が含まれる数値変数です。

DIF
基準データセットの_VAR_変数によって指定された変数と比較データセットの(_VAR_変数または_WITH_変数によって指定された)マッチング変数の値差異から計算された統計量の値が含まれる数値変数です。

PCTDIF
基準データセットの_VAR_変数によって指定された変数と比較データセットの(_VAR_変数または_WITH_変数によって指定された)マッチング変数の値のパーセント表示の差異から計算された統計量の値が含まれる数値変数です。

注: どちらの種類の出力データセットについても、PROC COMPARE は次のデータセットラベルのいずれかを割り当てます。

```
Comparison of base-SAS-data-set
with comparison-SAS-data-set

Comparison of variables in base-SAS-data-set
```

ラベルは 40 文字に制限されています。

OUTSTATS=データセットの例については、“例 7: 統計量の出力データセット (OUTSTATS=)の作成” (397 ページ) を参照してください。

例: COMPARE プロシジャ

例 1: 差異についての包括的なレポートの作成

要素: PROC COMPARE ステートメントオプション
BASE=
PRINTALL
COMPARE=

データセット: Proclib.One、Proclib.Two

詳細

この例では、PROC COMPARE がプロシジャ出力として生成する最も包括的なレポートを示します。

プログラム

```
libname proclib 'SAS-library';
```

```
options nodate pageno=1 linesize=80 pagesize=40;

proc compare base=proclib.one compare=proclib.two printall;
   title 'Comparing Two Data Sets: Full Report';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

SAS システムオプションを設定します。NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

2 つのデータセットの差異についての包括的なレポートを作成します。BASE=および COMPARE=は比較するデータセットを指定します。PRINTALL は差異についての完全なレポートを印刷します。

```
proc compare base=proclib.one compare=proclib.two printall;
   title 'Comparing Two Data Sets: Full Report';
run;
```

出力:出力:HTML

出力の A>は、完全なレポートにはあってデフォルトレポートにはない情報を示します。追加情報には、一方のデータセットにはあるが他方にはない変数のリスト、一方のデータセットにはあるが他方にはないオブザベーションのリスト、すべての同等な値を含む変数のリスト、要約統計量などがあります。統計量の説明については、“要約統計量のテーブル” (369 ページ)を参照してください。

アウトプット 12.10 2 つのデータセットの比較:完全なレポート(その 6)

Comparing Two Data Sets: Full Report

```
                        The COMPARE Procedure
                    Comparison of WORK.ONE with WORK.TWO
                              (Method=EXACT)

                           Data Set Summary

Dataset            Created          Modified  NVar    NObs  Label

WORK.ONE   16MAR11:17:52:15  16MAR11:17:52:15     5       4  First Data Set
WORK.TWO   16MAR11:17:52:15  16MAR11:17:52:15     6       5  Second Data Set


                           Variables Summary

              Number of Variables in Common: 5.
              Number of Variables in WORK.TWO but not in WORK.ONE: 1.
              Number of Variables with Conflicting Types: 1.
              Number of Variables with Differing Attributes: 3.
```

```
            Listing of Variables in WORK.TWO but not in WORK.ONE

                        Variable  Type  Length

                        major     Char       8


             Listing of Common Variables with Conflicting Types

                     Variable  Dataset   Type  Length

                     student   WORK.ONE  Num        8
                               WORK.TWO  Char       8
```

アウトプット 12.11 2 つのデータセットの比較:完全なレポート(その 6)

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Listing of Common Variables with Differing Attributes

Variable	Dataset	Type	Length	Format	Label
year	WORK.ONE	Char	8		Year of Birth
	WORK.TWO	Char	8		
state	WORK.ONE	Char	8		
	WORK.TWO	Char	8		Home State
gr1	WORK.ONE	Num	8	4.1	
	WORK.TWO	Num	8	5.2	

Comparison Results for Observations

Observation 5 in WORK.TWO not found in WORK.ONE.

Observation Summary

Observation	Base	Compare
First Obs	1	1
First Unequal	1	1
Last Unequal	4	4
Last Match	4	4
Last Obs	.	5

Number of Observations in Common: 4.
Number of Observations in WORK.TWO but not in WORK.ONE: 1.
Total Number of Observations Read from WORK.ONE: 4.
Total Number of Observations Read from WORK.TWO: 5.

Number of Observations with Some Compared Variables Unequal: 4.
Number of Observations with All Compared Variables Equal: 0.

アウトプット 12.12 2 つのデータセットの比較:完全なレポート(その 6)

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 3.
Total Number of Values which Compare Unequal: 6.
Maximum Difference: 20.

Variables with All Equal Values

Variable	Type	Len	Label
year	CHAR	8	Year of Birth

Variables with Unequal Values

Variable	Type	Len	Compare Label	Ndif	MaxDif
state	CHAR	8	Home State	2	
gr1	NUM	8		2	1.000
gr2	NUM	8		2	20.000

アウトプット 12.13 *2 つのデータセットの比較:完全なレポート(その 6)*

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Value Comparison Results for Variables

Obs	Year of Birth Base Value year	Compare Value year
1	1970	1970
2	1971	1971
3	1969	1969
4	1970	1970

Obs	Home State Base Value state	Compare Value state
1	NC	NC
2	MD	MA
3	PA	PA
4	MA	MD

アウトプット 12.14 2 つのデータセットの比較:完全なレポート(その 6)

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Value Comparison Results for Variables

Obs	Base gr1	Compare gr1	Diff.	% Diff
1	85.0	84.00	-1.0000	-1.1765
2	92.0	92.00	0	0
3	78.0	79.00	1.0000	1.2821
4	87.0	87.00	0	0
N	4	4	4	4
Mean	85.5000	85.5000	0	0.0264
Std	5.8023	5.4467	0.8165	1.0042
Max	92.0000	92.0000	1.0000	1.2821
Min	78.0000	79.0000	-1.0000	-1.1765
StdErr	2.9011	2.7234	0.4082	0.5021
t	29.4711	31.3951	0.0000	0.0526
Prob>|t|	<.0001	<.0001	1.0000	0.9614
Ndif	2	50.000%		
DifMeans	0.000%	0.000%	0	
r, rsq	0.991	0.983		

アウトプット 12.15 *2 つのデータセットの比較:完全なレポート(その 6)*

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Value Comparison Results for Variables

Obs	Base gr2	Compare gr2	Diff.	% Diff
1	87.0000	87.0000	0	0
2	92.0000	92.0000	0	0
3	72.0000	73.0000	1.0000	1.3889
4	94.0000	74.0000	-20.0000	-21.2766
N	4	4	4	4
Mean	86.2500	81.5000	-4.7500	-4.9719
Std	9.9457	9.4692	10.1776	10.8895
Max	94.0000	92.0000	1.0000	1.3889
Min	72.0000	73.0000	-20.0000	-21.2766
StdErr	4.9728	4.7346	5.0888	5.4447
t	17.3442	17.2136	-0.9334	-0.9132
Prob>|t|	0.0004	0.0004	0.4195	0.4285
Ndif	2	50.000%		
DifMeans	-5.507%	-5.828%	-4.7500	
r, rsq	0.451	0.204		

例 2: 異なるデータセットでの変数の比較

要素: PROC COMPARE ステートメントオプション
NOSUMMARY
VAR ステートメント
WITH ステートメント

データセット: Proclib.One、Proclib.Two

詳細

この例では、基準データセットの変数と比較データセットの変数を比較します。要約レポートはすべて非表示になります。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;
```

```
proc compare base=proclib.one compare=proclib.two nosummary;
   var gr1;
   with gr2;
   title 'Comparison of Variables in Different Data Sets';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

2 つのデータセットの差異についての要約レポートをすべて非表示にします。 BASE=は基準データセットを指定し、COMPARE=は比較データセットを指定します。NOSUMMARY はすべての要約レポートを非表示にします。

```
proc compare base=proclib.one compare=proclib.two nosummary;
```

基準データセットから 1 変数を指定して、比較データセットの 1 変数と比較します。 VAR ステートメントと WITH ステートメントは、比較する変数を指定します。この例では、基準データセットの GR1 と比較データセットの GR2 を比較しています。

```
   var gr1;
   with gr2;
   title 'Comparison of Variables in Different Data Sets';
run;
```

出力:出力:HTML

アウトプット 12.16 *異なるデータセットでの変数の比較*

Comparison of Variables in Different Data Sets

```
                        The COMPARE Procedure
                    Comparison of WORK.ONE with WORK.TWO
                              (Method=EXACT)

NOTE: Data set WORK.TWO contains 1 observations not in WORK.ONE.
NOTE: Values of the following 1 variables compare unequal: gr1^=gr2
```

Value Comparison Results for Variables

Obs	Base gr1	Compare gr2	Diff.	% Diff
1	85.0	87.0000	2.0000	2.3529
3	78.0	73.0000	-5.0000	-6.4103
4	87.0	74.0000	-13.0000	-14.9425

例 3: 変数を複数回比較する

要素: VAR ステートメント
WITH ステートメント

データセット: Proclib.One、Proclib.Two

詳細

この例では、基準データセットの 1 つの変数と比較データセットの 2 つの変数を比較します。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;

proc compare base=proclib.one compare=proclib.two nosummary;

   var gr1 gr1;
   with gr1 gr2;
   title 'Comparison of One Variable with Two Variables';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

2 つのデータセットの差異についての要約レポートをすべて非表示にします。 BASE=は基準データセットを指定し、COMPARE=は比較データセットを指定します。NOSUMMARY はすべての要約レポートを非表示にします。

```
proc compare base=proclib.one compare=proclib.two nosummary;
```

基準データセットから 1 つの変数を指定して、比較データセットの 2 つの変数と比較します。 VAR ステートメントと WITH ステートメントは、比較する変数を指定します。この例では、基準データセットの GR1 を、比較データセットの GR1 および GR2 と比較しています。

```
   var gr1 gr1;
   with gr1 gr2;
   title 'Comparison of One Variable with Two Variables';
run;
```

出力:出力:HTML

値比較結果セクションは、比較結果を示しています。

アウトプット 12.17 *1 変数と 2 変数の比較*

Comparison of One Variable with Two Variables

```
                        The COMPARE Procedure
                   Comparison of WORK.ONE with WORK.TWO
                              (Method=EXACT)

NOTE: Data set WORK.TWO contains 1 observations not in WORK.ONE.
NOTE: Values of the following 2 variables compare unequal: gr1^=gr1 gr1^=gr2
```

Value Comparison Results for Variables

Obs	Base gr1	Compare gr1	Diff.	% Diff
1	85.0	84.00	-1.0000	-1.1765
3	78.0	79.00	1.0000	1.2821

Obs	Base gr1	Compare gr2	Diff.	% Diff
1	85.0	87.0000	2.0000	2.3529
3	78.0	73.0000	-5.0000	-6.4103
4	87.0	74.0000	-13.0000	-14.9425

例 4: 同一データセット内での変数の比較

要素: PROC COMPARE ステートメントオプション
- ALLSTATS
- BRIEFSUMMARY

VAR ステートメント

WITH ステートメント

データセット: Proclib.One

詳細

この例では、PROC COMPARE が同一データセット内の 2 つの変数を比較できることを示します。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;

proc compare base=proclib.one allstats briefsummary;

   var gr1;
   with gr2;
   title 'Comparison of Variables in the Same Data Set';
run;
```

プログラムの説明

Proclib SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

---

1 つのデータセット内の差異についての簡潔な要約レポートを作成します。 ALLSTATS は要約統計量を印刷します。BRIEFSUMMARY は簡潔な比較要約のみを印刷します。

```
proc compare base=proclib.one allstats briefsummary;
```

比較する 2 つの変数を基準データセットから指定します。 VAR ステートメントと WITH ステートメントは、比較する基準データセットの変数を指定します。この例では、GR1 と GR2 を比較しています。比較データセットがないので、変数 GR1 および GR2 が基準データセットに存在している必要があります。

```
   var gr1;
   with gr2;
   title 'Comparison of Variables in the Same Data Set';
run;
```

出力:出力:HTML

アウトプット 12.18 *1 変数と 2 変数の比較*

Comparison of One Variable with Two Variables

```
                        The COMPARE Procedure
                    Comparisons of variables in WORK.ONE
                              (Method=EXACT)

NOTE: Values of the following 1 variables compare unequal: gr1^=gr2
```

Value Comparison Results for Variables

Obs	Base gr1	Compare gr2	Diff.	% Diff
1	85.0	87.0000	2.0000	2.3529
3	78.0	72.0000	-6.0000	-7.6923
4	87.0	94.0000	7.0000	8.0460
N	4	4	4	4
Mean	85.5000	86.2500	0.7500	0.6767
Std	5.8023	9.9457	5.3774	6.5221
Max	92.0000	94.0000	7.0000	8.0460
Min	78.0000	72.0000	-6.0000	-7.6923
StdErr	2.9011	4.9728	2.6887	3.2611
t	29.4711	17.3442	0.2789	0.2075
Prob>|t|	<.0001	0.0004	0.7984	0.8489
Ndif	3	75.000%		
DifMeans	0.877%	0.870%	0.7500	
r, rsq	0.898	0.807		

例 5: ID 変数を利用し、オブザベーションを比較する

要素: ID ステートメント

データセット: Proclib.Emp95
Proclib.Emp96

詳細

この例では、PROC COMPARE は、ID 変数のマッチング値があるオブザベーションのみを比較します。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;

data proclib.emp95;
   input #1 idnum $4. @6 name $15.
         #2 address $42.
         #3 salary 6.;
   datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
... more data lines...
3888 Kim Siu
5662 Magnolia Blvd Southeast Cary NC 27513
77558
;

data proclib.emp96;
   input #1 idnum $4. @6 name $15.
         #2 address $42.
         #3 salary 6.;
   datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
...more data lines...
6544 Roger Monday
3004 Crepe Myrtle Court Raleigh NC 27604
47007
;

proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
   by idnum;
run;

proc compare base=emp95_byidnum compare=emp96_byidnum;
   id idnum;
   title 'Comparing Observations that Have Matching IDNUMs';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Proclib.Emp95 データセットと Proclib.Emp96 データセットを作成します。 Proclib.Emp95 と Proclib.Emp96 には従業員データが含まれます。IDNUM の値は一意なので、ID 変数に適しています。最初の DATA ステップで、Proclib.Emp95 が作成されます。次の DATA ステップで Proclib.Emp96 が作成されます。

```
data proclib.emp95;
   input #1 idnum $4. @6 name $15.
         #2 address $42.
         #3 salary 6.;
   datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
... more data lines...
3888 Kim Siu
5662 Magnolia Blvd Southeast Cary NC 27513
77558
;

data proclib.emp96;
   input #1 idnum $4. @6 name $15.
         #2 address $42.
         #3 salary 6.;
   datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
...more data lines...
6544 Roger Monday
3004 Crepe Myrtle Court Raleigh NC 27604
47007
;
```

データセットを ID 変数順に並べ替えます。 両方のデータセットを、PROC COMPARE ステップで ID 変数として使用する変数順に並べ替える必要があります。OUT=は並べ替え後のデータの場所を指定します。

```
proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
```

```
   by idnum;
run;
```

ID 変数のマッチング値を含むオブザベーションを比較する要約レポートを作成します。 ID ステートメントは、IDNUM を ID 変数として指定します。

```
proc compare base=emp95_byidnum compare=emp96_byidnum;
   id idnum;
   title 'Comparing Observations that Have Matching IDNUMs';
run;
```

出力:出力:HTML

PROC COMPARE は IDNUM の値によって特定のオブザベーションを識別します。PROC COMPARE は、**`Value Comparison Results for Variables`** セクションに、一致しない住所と一致しない給与を印刷します。給与については、PROC COMPARE は数値の差異とパーセント表示の差異を計算します。ADDRESS は文字変数なので、PROC COMPARE は最初の 20 文字のみを表示します。オブザベーションの IDNUM が **`0987`**、**`2776`**、**`3888`** の住所については、20 文字目より後ろに差異が発生しているので、出力には差異が表示されていません。出力のプラス記号は、値の一部が表示されていないことを示します。値全体を確認するには、出力データセットを作成します。“例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する” (394 ページ)を参照してください。

アウトプット 12.19 *マッチング IDNUM を含むオブザベーションの比較(その 1)*

Comparing Observations that Have Matching IDNUMs

```
                          The COMPARE Procedure
          Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
                                (Method=EXACT)

                             Data Set Summary

     Dataset                      Created          Modified  NVar    NObs

     WORK.EMP95_BYIDNUM   17MAR11:08:58:10  17MAR11:08:58:10     4      10
     WORK.EMP96_BYIDNUM   17MAR11:08:58:10  17MAR11:08:58:10     4      12


                             Variables Summary

                    Number of Variables in Common: 4.
                    Number of ID Variables: 1.
```

```
                             Observation Summary

                  Observation      Base  Compare  ID

                  First Obs           1        1  idnum=0987
                  First Unequal       1        1  idnum=0987
                  Last  Unequal      10       12  idnum=9857
                  Last  Obs          10       12  idnum=9857

Number of Observations in Common: 10.
Number of Observations in WORK.EMP96_BYIDNUM but not in WORK.EMP95_BYIDNUM: 2.
Total Number of Observations Read from WORK.EMP95_BYIDNUM: 10.
Total Number of Observations Read from WORK.EMP96_BYIDNUM: 12.

Number of Observations with Some Compared Variables Unequal: 5.
Number of Observations with All Compared Variables Equal: 5.
```

アウトプット 12.20 マッチング IDNUM を含むオブザベーションの比較(その 2)

Comparing Observations that Have Matching IDNUMs

```
                    The COMPARE Procedure
      Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
                          (Method=EXACT)

                   Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 2.
Total Number of Values which Compare Unequal: 8.
Maximum Difference: 2400.


                Variables with Unequal Values

             Variable  Type  Len  Ndif   MaxDif

             address   CHAR   42     4
             salary    NUM     8     4     2400
```

```
            Value Comparison Results for Variables

_____________________________________________________________
        ||  Base Value           Compare Value
 idnum  ||  address                address
 _____  ||  ___________________+  ___________________+
        ||
 0987   ||  2344 Persimmons Bran  2344 Persimmons Bran
 2776   ||  12988 Wellington Far  12988 Wellington Far
 3888   ||  5662 Magnolia Blvd S  5662 Magnolia Blvd S
 9857   ||  1000 Taft Ave. Morri  100 Taft Ave. Morris
_____________________________________________________________
```

アウトプット 12.21 *マッチング IDNUM を含むオブザベーションの比較(その 3)*

Comparing Observations that Have Matching IDNUMs

The COMPARE Procedure
Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
(Method=EXACT)

Value Comparison Results for Variables

idnum	Base salary	Compare salary	Diff.	% Diff
0987	44010	45110	1100	2.4994
3286	87734	89834	2100	2.3936
3888	77558	79958	2400	3.0945
9857	38756	40456	1700	4.3864

例 6: 出力データセット(OUT=)を使用し、オブザベーションの値を比較する

要素: PROC COMPARE ステートメントオプション
NOPRINT
OUT=
OUTBASE
OUTBASE
OUTCOMP
OUTDIF
OUTNOEQUAL

他の要素: PRINT プロシジャ

データセット: Proclib.Emp95
Proclib.Emp96

詳細

この例では、マッチングオブザベーションの差異を示す出力データセットの作成と印刷を行います。

“例 5: ID 変数を利用し、オブザベーションを比較する” (388 ページ), では、20 文字目より後の差異は出力に表示されません。この例の出力データセットには、完全な値が表示されます。さらに、一方のデータセットにのみ出現するオブザベーションが表示されます。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=120 pagesize=40;

proc sort data=proclib.emp95 out=emp95_byidnum;
```

```
 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
   by idnum;
run;

proc compare base=emp95_byidnum compare=emp96_byidnum

             out=result outnoequal outbase outcomp outdif
     noprint;

   id idnum;
run;

proc print data=result noobs;
   by idnum;
   id idnum;
   title 'The Output Data Set RESULT';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=120 pagesize=40;
```

データセットを ID 変数順に並べ替えます。 両方のデータセットを、PROC COMPARE ステップで ID 変数として使用する変数順に並べ替える必要があります。OUT=は並べ替え後のデータの場所を指定します。

```
proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
   by idnum;
run;
```

比較するデータセットを指定します。 BASE=および COMPARE=は比較するデータセットを指定します。

```
proc compare base=emp95_byidnum compare=emp96_byidnum
```

出力データセット Result を作成し、すべての不等なオブザベーションとその差異を含めます。 OUT=は、出力データセットの命名と作成を行います。NOPRINT は、プロシジャ出力の印刷を抑制します。OUTNOEQUAL は、不等と判断されたオブザベーションのみを含めます。OUTBASE は、基準データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込みます。OUTCOMP は、比較データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込みます。OUTDIF は、2 つのオブザベーションの差異を含む出力データセットにオブザベーションを書き込みます。

```
                out=result outnoequal outbase outcomp outdif
      noprint;
```

ID 変数を指定します。 ID ステートメントは、IDNUM を ID 変数として指定します。

```
   id idnum;
run;
```

出力データセット RESULT を印刷し、BY および ID ステートメントで ID 変数を指定します。 PROC PRINT は、出力データセットを印刷します。BY ステートメントと ID ステートメントで同じ変数を指定すると、出力が読み取りやすくなります。この方法の詳細については、PRINT プロシジャを参照してください。

```
proc print data=result noobs;
   by idnum;
   id idnum;
   title 'The Output Data Set RESULT';
run;
```

出力:出力:HTML

文字変数の差異は、X またはピリオド(.)で示されます。X は文字が一致していないことを示します。ピリオドは文字が一致していることを示します。数値変数の場合、E は差異がないことを意味します。それ以外の場合は、数値の差異が表示されます。デフォルトでは、出力データセットは、比較データセットの 2 つのオブザベーションについて、基準データセットにマッチングオブザベーションがないことを示しています。これらのオブザベーションを出力データセットに表示するためにオプションを使用する必要はありません。

アウトプット 12.22 *出力データセット RESULT(その 1)*

The Output Data Set RESULT

idnum	_TYPE_	_OBS_	name	address	salary
0987	BASE	1	Dolly Lunford	2344 Persimmons Branch Apex NC 27505	44010
	COMPARE	1	Dolly Lunford	2344 Persimmons Branch Trail Apex NC 27505	45110
	DIF	1	.....	.....XXXXX.XXXXXXXXXXXXX	1100

idnum	_TYPE_	_OBS_	name	address	salary
2776	BASE	5	Robert Jones	12988 Wellington Farms Ave. Cary NC 27512	29025
	COMPARE	5	Robert Jones	12988 Wellington Farms Ave. Cary NC 27511	29025
	DIF	5	.....	.....X.	E

idnum	_TYPE_	_OBS_	name	address	salary
3278	COMPARE	6	Mary Cravens	211 N. Cypress St. Cary NC 27512	35362

idnum	_TYPE_	_OBS_	name	address	salary
3286	BASE	6	Hoa Nguyen	2818 Long St. Cary NC 27513	87734
	COMPARE	7	Hoa Nguyen	2818 Long St. Cary NC 27513	89834
	DIF	6	.....	.....	2100

アウトプット 12.23 出力データセット RESULT(その 2)

idnum	_TYPE_	_OBS_	name	address	salary
3888	BASE	7	Kim Siu	5662 Magnolia Blvd Southeast Cary NC 27513	77558
	COMPARE	8	Kim Siu	5662 Magnolia Blvd Southwest Cary NC 27513	79958
	DIF	7	.....	.....XX.....	2400

idnum	_TYPE_	_OBS_	name	address	salary
6544	COMPARE	9	Roger Monday	3004 Crepe Myrtle Court Raleigh NC 27604	47007

idnum	_TYPE_	_OBS_	name	address	salary
9857	BASE	10	Kathy Krupski	1000 Taft Ave. Morrisville NC 27508	38756
	COMPARE	12	Kathy Krupski	100 Taft Ave. Morrisville NC 27508	40456
	DIF	10	.....	...XXXXXXXXXXXXX.XXXXX.XXXXXXXXXX.....	1700

例 7: 統計量の出力データセット(OUTSTATS=)の作成

要素: PROC COMPARE ステートメントオプション
NOPRINT
OUTSTATS=

データセット: Proclib.Emp95
Proclib.Emp96

詳細

この例では、比較された数値変数の要約統計量を含む出力データセットを作成します。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;

proc sort data=proclib.emp95 out=emp95_byidnum;
   by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
   by idnum;
run;

proc compare base=emp95_byidnum compare=emp96_byidnum
             outstats=diffstat noprint;
   id idnum;
run;

proc print data=diffstat noobs;
   title 'The DIFFSTAT Data Set';
```

```
run;
```

プログラムの説明

Proclib SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

データセットを ID 変数順に並べ替えます。 両方のデータセットを、PROC COMPARE ステップで ID 変数として使用する変数順に並べ替える必要があります。OUT=は並べ替え後のデータの場所を指定します。

```
proc sort data=proclib.emp95 out=emp95_byidnum;
   by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
   by idnum;
run;
```

統計量の出力データセットを作成し、ID 変数のマッチング値があるオブザベーションを比較します。 BASE=および COMPARE=は比較するデータセットを指定します。OUTSTATS=は、出力データセット Diffstat.Noprint を作成し、プロシジャ出力を抑制します。ID ステートメントは、IDNUM を ID 変数として指定します。PROC COMPARE は、IDNUM の値を使用してオブザベーションのマッチングを行います。

```
proc compare base=emp95_byidnum compare=emp96_byidnum
             outstats=diffstat noprint;
   id idnum;
run;
```

出力データセット Diffstat を印刷します。 PROC PRINT は、出力データセット Diffstat を印刷します。

```
proc print data=diffstat noobs;
   title 'The DIFFSTAT Data Set';
run;
```

出力:出力:HTML

変数については “出力統計量データセット(OUTSTATS=)” (374 ページ)で説明しています。

アウトプット 12.24 *Diffstat データセット*

The DIFFSTAT Data Set

VAR	_TYPE_	_BASE_	_COMP_	_DIF_	_PCTDIF_
salary	N	10.00	10.00	10.00	10.0000
salary	MEAN	52359.00	53089.00	730.00	1.2374
salary	STD	24143.84	24631.01	996.72	1.6826
salary	MAX	92100.00	92100.00	2400.00	4.3864
salary	MIN	29025.00	29025.00	0.00	0.0000
salary	STDERR	7634.95	7789.01	315.19	0.5321
salary	T	6.86	6.82	2.32	2.3255
salary	PROBT	0.00	0.00	0.05	0.0451
salary	NDIF	4.00	40.00	.	.
salary	DIFMEANS	1.39	1.38	730.00	.
salary	R,RSQ	1.00	1.00	.	.

13 章

CONTENTS プロシジャ

概要: CONTENTS プロシジャ

CONTENTS プロシジャは、SAS データセットの内容を表示したり、SAS ライブラリのディレクトリを出力したりします。

大まかに言えば、CONTENTS プロシジャは DATASETS プロシジャの CONTENTS ステートメントと同様に機能します。CONTENTS プロシジャと PROC DATASETS の CONTENTS ステートメントの違いを次に示します。

- PROC CONTENTS の DATA=オプションの *libref* のデフォルトは、Work です。CONTENTS ステートメントの場合、デフォルトはプロシジャ入力ライブラリのライブラリ参照名です。
- PROC CONTENTS は、順編成ファイルを読み込めます。CONTENTS ステートメントは、読めません。

概念: CONTENTS プロシジャ

CONTENTS ステートメントの概念 (439 ページ)を参照してください。

注意:

CONTENTS の OUT=データセットの GENNUM 変数値と DICTIONARY テーブルからの GEN 変数値を混同しないでください。 CONTENTS プロシジャまたはステートメントからの GENNUM は、データセットの特定の世代を参照します。DICTIONARY テーブルからの GEN は、データセットの世代の合計数を参照します。

構文: CONTENTS プロシジャ

注: ATTRIB ステートメントは、PROC CONTENTS 出力には影響を与えません。PROC CONTENTS により、実際のメンバのラベル、インフォーマット、フォーマットがレポートされます。

PROC CONTENTS はオブザベーションを処理しないため、出力に作用する WHERE オプションは使用できません。

ヒント: CONTENTS ステートメントの詳細なドキュメントは、CONTENTS ステートメント (472 ページ)にあります。これらのオプションの説明については、下の表のリンクをクリックして DATASETS プロシジャのドキュメントを参照してください。

ATTRIB、FORMAT、および LABEL ステートメントが使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

PROC CONTENTS の使用時には、DATA=オプション、OUT=オプション、OUT2=オプションとともにデータセットオプションを使用できます。

ORDER=オプションは、OUT=と OUT2=データセットの順序には影響しません。

参照項目: 各 OS 上の CONTENTS プロシジャ:Windows、UNIX、z/OS

PROC CONTENTS <*option(s)*>;

ステートメント	タスク	例
“CONTENTS ステートメント”	SAS データセット(複数可)の内容を表示したり、SAS ライブラリのディレクトリを出力したりします。	Ex. 1

例: SAS データセットの説明

要素: PROC CONTENTS ステートメントオプション
- DATA=
- OUT=

他の要素: SAS データセットオプション
- OPTIONS ステートメント
- TITLE statement

詳細

この例は、GROUP データセットに対する CONTENTS プロシジャからの出力を示しています。出力には、“例 4: SAS データセットの変更” (555 ページ) で GROUP データセットに行われた変更と、GRPOUT データセットの内容が示されています。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME health
```

```
'SAS-library';

proc datasets library=health nolist;
run;

proc contents data=health.group (read=green) out=health.grpout;
   title  'The Contents of the GROUP Data Set';
run;

proc contents data=health.grpout;
   title  'The Contents of the GRPOUT Data Set';
run;
```

プログラムの説明

システムオプションを設定します。 PAGESIZE=オプションで、SAS ログと SAS 出力を構成する行の数が指定されます。LINESIZE=オプションは、SAS ログと SAS プロシジャ出力のページサイズを指定します。NODATE オプションは、日時を印刷しないことを指定します。PAGENO=オプションで、出力の次ページの開始ページ番号を指定します。

```
options pagesize=40 linesize=80 nodate pageno=1;
```

ライブラリ参照名を設定します。

```
LIBNAME health
'SAS-library';
```

Health をプロシジャの入力ライブラリに指定し、ディレクトリリストの表示を抑制します。

```
proc datasets library=health nolist;
run;
```

データセット GROUP から出力データセット GRPOUT を作成します。 記述するデータセットとして GROUP を指定し、GROUP データセットへの読み込みアクセス権を付与して、出力データセット GRPOUT を作成します。

```
proc contents data=health.group (read=green) out=health.grpout;
   title  'The Contents of the GROUP Data Set';
run;
```

GRPOUT データセットの内容が示されます。

```
proc contents data=health.grpout;
   title  'The Contents of the GRPOUT Data Set';
run;
```

出力例

アウトプット 13.1 *Group データセットの内容*

The Contents of the GROUP Data Set

The CONTENTS Procedure

Data Set Name	HEALTH.GROUP	Observations	148
Member Type	DATA	Variables	11
Engine	V9	Indexes	0
Created	09/03/2014 10:35:02	Observation Length	96
Last Modified	09/03/2014 10:35:02	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	3
First Data Page	1
Max Obs per Page	84
Obs in First Data Page	63
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\\procdatasets\health\group.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Informat
9	BIRTH	Num	8		
4	CITY	Char	15		
3	FNAME	Char	15		
10	HIRED	Num	8	DATE7.	DATE7.
11	HPHONE	Char	12		
1	IDNUM	Char	4		
7	JOBCODE	Char	4		
2	LNAME	Char	15		
8	SALARY	Num	8	COMMA8.	
6	SEX	Char	2		
5	STATE	Char	3		

アウトプット 13.2 *GRPOUT データセットの内容*

The Contents of the GRPOUT Data Set

The CONTENTS Procedure

Data Set Name	WORK.GRPOUT	Observations	11
Member Type	DATA	Variables	41
Engine	V9	Indexes	0
Created	05/15/2015 11:57:53	Observation Length	888
Last Modified	05/15/2015 11:57:53	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	73728
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	82
Obs in First Data Page	11
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\\procdatasets\health\grpout.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
32	CHARSET	Char	8		Host Character Set
33	COLLATE	Char	8		Collating Sequence
28	COMPRESS	Char	8		Compression Routine
20	CRDATE	Num	8	DATETIME16.	Create Date
22	DELOBS	Num	8		Deleted Observations in Data Set
36	ENCRYPT	Char	8		Encryption Routine
19	ENGINE	Char	8		Engine Name
27	FLAGS	Char	3		Update Flags (Protect Contribute Add)
10	FORMAT	Char	32		Variable Format
12	FORMATD	Num	8		Number of Format Decimals
11	FORMATL	Num	8		Format Length
38	GENMAX	Num	8		Maximum Number of Generations
40	GENNEXT	Num	8		Next Generation Number
39	GENNUM	Num	8		Generation Number
25	IDXCOUNT	Num	8		Number of Indexes for Data Set
23	IDXUSAGE	Char	9		Use of Variable in Indexes
13	INFORMAT	Char	32		Variable Informat
15	INFORMD	Num	8		Number of Informat Decimals
14	INFORML	Num	8		Informat Length
16	JUST	Num	8		Justification
9	LABEL	Char	256		Variable Label
7	LENGTH	Num	8		Variable Length

次に、変数と属性のアルファベット順のリストの続きを示します。

1	LIBNAME	Char	8		Library Name
3	MEMLABEL	Char	256		Data Set Label
2	MEMNAME	Char	32		Library Member Name
24	MEMTYPE	Char	8		Library Member Type
21	MODATE	Num	8	DATETIME16.	Last Modified Date
5	NAME	Char	32		Variable Name
18	NOBS	Num	8		Observations in Data Set
34	NODUPKEY	Char	3		Sort Option: No Duplicate Keys
35	NODUPREC	Char	3		Sort Option: No Duplicate Records
17	NPOS	Num	8		Position in Buffer
37	POINTOBS	Char	3		Point to Observations
26	PROTECT	Char	3		Password Protection (Read Write Alter)
29	REUSE	Char	3		Reuse Space
30	SORTED	Num	8		Sorted and/or Validated
31	SORTEDBY	Num	8		Position of Variable in Sortedby Clause
41	TRANSCOD	Char	3		Character Variables Transcoded
6	TYPE	Num	8		Variable Type
4	TYPEMEM	Char	8		Special Data Set Type (From TYPE=)
8	VARNUM	Num	8		Variable Number

Sort Information	
Sortedby	LIBNAME MEMNAME
Validated	YES
Character Set	ANSI

14 章

COPY プロシジャ

概要: COPY プロシジャ

COPY プロシジャは SAS ライブラリから 1 つ以上の SAS ファイルをコピーします。

大まかに言えば、COPY プロシジャは DETASETS プロシジャの COPY ステートメントと同様に機能します。ただし、相違点が 2 つあります。

- PROC COPY には IN=引数が必須です。COPY ステートメントでは IN=はオプションです。省略された場合、デフォルト値はプロシジャの入力ライブラリのライブラリ参照名です。
- PROC DATASETS は、順次データアクセスのみを許可するライブラリには使用できません。

注: MIGRATE プロシジャは、特に SAS ライブラリを前のリリースから最新のリリースに移行するためのものです。移行に関して、PROC MIGRATE は PROC COPY にはない利点を提供できます。詳細については、MIGRATE プロシジャ (1176 ページ)を参照してください。

構文: COPY プロシジャ

制限事項: PROC COPY は、カタログとの連結は無視します。連結カタログをコピーする場合は、PROC CATALOG COPY を使用します。

PROC COPY では、データセットオプションはサポートされていません。

PROC COPY では、グラフィックカタログはバックアップされません。グラフィックカタログでバックアップするには、PROC CPORT または PROC CIMPORT を使用してください。

操作: 言語照合順序があるデータセットをソートするには、ユニコード用国際化コンポーネント(ICU)バージョンを使用します。言語的に並べ替えられたデータセットに、現在の SAS セッションの ICU バージョン番号とは異なる番号が含まれている場合、PROC COPY はデータセットのソート順を OUT=出力先ライブラリに保持します。ただし、そのデータセットからソート済みのマーキングがなくなり、SAS ログに警告メッセージが書き込まれます。言語ソートの詳細については、59 章, “SORT プロシジャ,” (1785 ページ)を参照してください。

ヒント: COPY プロシジャの詳細なドキュメントは、COPY ステートメント (480 ページ)にあります。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

PROC COPY OUT=*libref-1*
<CLONE | NOCLONE>
<CONSTRAINT=YES | NO>
<DATECOPY>
<ENCRYPTKEY=*key-value*>
<FORCE>
IN=*libref-2*
<INDEX=YES | NO>
<MEMTYPE=(*member-type(s)*)>
<MOVE <ALTER=*alter-password*>>
<OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2 …>*) >;
SELECT *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*> <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

ステートメント	タスク	例
“COPY ステートメント”	1 つ以上のファイルをコピーします	Ex. 1, Ex. 3
“EXCLUDE ステートメント”	ファイルや種類を除外します	Ex.
“SELECT ステートメント”	ファイルや種類を選択します	Ex. 1, Ex. 2

COPY プロシジャの使用

ファイルの大規模ディレクトリからの選択ファイルのコピー

COPY プロシジャを使用すると、ライブラリのメモリ内ディレクトリが得られます。ライブラリに数千ものメンバが含まれ、コピーされるのはその中のいくつかのメンバのみである場合、パフォーマンスの問題が生じることがあります。こうしたパフォーマンスの問題を解決するには、COPY ステートメントの MEMTYPE=オプションと SELECT ステートメントの組み合わせを使用します。次にこのプロセスの例を示します。

```
proc copy in=work out=mylib memtype=(data catalog);
```

```
   select mydata x1-x10 data2;
run;
```

注: MEMTYPE=ALL またはワイルドカード指定(":")のいずれかを使用する場合、パフォーマンスコードは使用できません。

ヒント MSGLEVEL=I オプションを設定し、SELECT パフォーマンスコードを使用できる場合、次のメッセージが SAS ログに送信されます:`INFO:COPY with SELECT performance is in use.`

ホスト間での SAS データセットの移送

XPORT エンジンおよび XML エンジンとともに使用すると、COPY プロシジャで、ホスト間で移動可能な移送ファイルを作成し、読み込むことができます。PROC COPY では、SAS データセットでのみ移送ファイルを作成できます。カタログや他の種類の SAS ファイルでは作成できません。

移送は次の 3 段階のプロセスです。

1. PROC COPY を使用して1つ以上の SAS データセットを、トランスポート(XPORT)エンジンまたは XML エンジンで作成されたファイルにコピーします。このファイルは移送ファイルと呼ばれ、常に順次ファイルになります。
2. ファイルを作成したら、FTP などの通信ソフトやテープを介して、別の動作環境にそれを移動することもできます。通信ソフトを使用する場合には、変換を防ぐためにバイナリー形式で移動してください。ファイルをメインフレームに移動する場合は、そのファイルに特定の属性が必要です。詳細は動作環境に応じた SAS ドキュメントと、SAS テクニカルサポートのウェブページを参照してください。
3. 受信ホストにファイルが正常に移動したら、PROC COPY を使用して移送ファイルのデータセットを SAS ライブラリにコピーします。

例については、“例 1: ホスト間での SAS データセットのコピー” (412 ページ)を参照してください。

ファイルの移送の詳細については、*SAS ファイルの移動とアクセス*を参照してください。

CPORT および CIMPORT プロシジャを使用しても、SAS ファイルを移送できます。詳細については、15 章, “CPORT プロシジャ,” (417 ページ)および 11 章, “CIMPORT プロシジャ,” (321 ページ)を参照してください。

SAS ライブラリを前のリリースの SAS から移行する必要がある場合は、移行フォーカスエリア http://support.sas.com/migration を参照してください。

詳細については、PROC DATASETS の CONTENT ステートメントの詳細 (486 ページ)セクションを参照してください。

AES で暗号化されたデータファイルのコピー

AES で暗号化されたデータファイルをコピーするときは、ENCRYPTKEY=データセットオプションを使用する必要があります。AES 暗号化をサポートしていないライブラリに、AES で暗号化されたファイルをコピーするとエラーが発生します。AES 暗号化の詳細については、“AES Encryption” (*SAS Language Reference: Concepts*)および“AES 暗号化データファイルのコピー” (490 ページ)を参照してください。

ENCRYPTKEY=データセットオプションを使用する例を次に示します。

```
proc copy in=Lib1 out=Lib2; select My-Data1 (encryptkey=key-value1) <My-Data2 (encryptke
```

参照一貫性制約を含む、AES で暗号化されたデータファイルをコピーするには、“参照用一貫性制約を含む AES 暗号化データファイルのコピー” (491 ページ)を参照してください。

出力データファイルの圧縮

COPY プロシジャでは、データセットオプションはサポートされていません。このため、COMPRESS=データセットオプションを PROC COPY や PROC DATASETS の COPY ステートメントで使用することはできません。PROC COPY で生成された OUTPUT データセットを圧縮するには、COMPRESS=YES システムオプションを設定してから、NOCLONE オプションを設定して PROC COPY ステートメントを使用できます。

```
options compress=yes;
proc copy in=work out=new noclone;
select x;
run;
```

例: COPY プロシジャ

例 1: ホスト間での SAS データセットのコピー

要素: PROC COPY ステートメントオプション
- IN=
- MEMTYPE=
- OUT=
- SELECT ステートメント

他の要素: XPORT エンジン

詳細

この例は、ホストで移送ファイルを作成し、別のホストでそれを読み取る方法が示されています。

この例が正しく動作するには、使用している動作環境に対応する SAS ドキュメントに記載されているように、移送ファイルが特定の特性を持っている必要があります。また、移送ファイルは受信動作環境にバイナリー形式で移動する必要があります。

プログラム

```
libname source 'SAS-library-on-sending-host';

libname xptout xport 'filename-on-sending-host';


proc copy in=source out=xptout memtype=data;
   select bonus budget salary;
run;
```

```
libname insource xport 'filename-on-receiving-host';

proc copy in=insource out=work;
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 SOURCE などのライブラリ参照名を、移送する SAS データセットを含む SAS ライブラリに割り当てます。また、ライブラリ参照名を移送ファイルに割り当て、XPORT キーワードを使用して XPORT エンジンを指定します。

```
libname source 'SAS-library-on-sending-host';

libname xptout xport 'filename-on-sending-host';
```

SAS データセットを移送ファイルにコピーします。

```
proc copy in=source out=xptout memtype=data;
   select bonus budget salary;
run;
```

プロシジャで移送ファイルからのデータを読み取れるようにします。 LIBNAME ステートメントの XPORT エンジンによって、プロシジャは移送ファイルからデータを読み取れるようになります。

```
libname insource xport 'filename-on-receiving-host';
```

SAS データセットを受信ホストにコピーします。 ファイルをコピーしたら、PROC COPY を使用して SAS データセットを受信ホストの作業データライブラリにコピーします。Windows ホストに対しては、バイナリモードで FTP を使用できます。

```
proc copy in=insource out=work;
run;
```

ログの例

ログ 14.1 *ソースライブラリログ*

```
1    LIBNAME source 'SAS-library-on-sending-host '; NOTE:Libref SOURCE was
successfully assigned as follows:Engine:        V9 Physical Name:SAS-library-on-
sending-host 2    LIBNAME xptout xport 'filename-on-sending-host'; NOTE:Libref
XPTOUT was successfully assigned as follows:Engine:        XPORT Physical Name:
filename-on-sending-host 3    proc copy in=source out=xptout memtype=data; 4
select bonus budget salary; 5    run; NOTE:Copying SOURCE.BONUS to XPTOUT.BONUS
(memtype=DATA).NOTE:The data set XPTOUT.BONUS has 1 observations and 3
variables.NOTE:Copying SOURCE.BUDGET to XPTOUT.BUDGET (memtype=DATA).NOTE:The
data set XPTOUT.BUDGET has 1 observations and 3 variables.NOTE:Copying
SOURCE.SALARY to XPTOUT.SALARY (memtype=DATA).NOTE:The data set XPTOUT.SALARY
has 1 observations
```

ログ 14.2 *インソースライブラリログ*

```
1     LIBNAME insource xport 'filename-on-receiving-host'; NOTE:Libref INSOURCE
was successfully assigned as follows:Engine:      XPORT Physical Name: filename-
on-receiving-host 2     proc copy in=insource out=work; 3     run; NOTE:Input
library INSOURCE is sequential.NOTE:Copying INSOURCE.BUDGET to WORK.BUDGET
(memtype=DATA).NOTE:BUFSIZE is not cloned when copying across different
engines.System Option for BUFSIZE was used.NOTE:The data set WORK.BUDGET has 1
observations and 3 variables.NOTE:Copying INSOURCE.BONUS to WORK.BONUS
(memtype=DATA).NOTE:BUFSIZE is not cloned when copying across different
engines.System Option for BUFSIZE was used.NOTE:The data set WORK.BONUS has 1
observations and 3 variables.NOTE:Copying INSOURCE.SALARY to WORK.SALARY
(memtype=DATA).NOTE:BUFSIZE is not cloned when copying across different
engines.System Option for BUFSIZE was used.NOTE:The data set WORK.SALARY has 1
observations and 3 variables.
```

例 2: SAS データセットエンコーディングの変換

要素: PROC COPY ステートメントオプション
- IN=
- NOCLONE
- OUT=
- SELECT ステートメント

他の要素: CVP engine

詳細

この例は、エンコーディングの種類を別の種類に変換する方法を示しています。この例が正しく動作するには、2つのエンコーティングに互換性がある必要があります。詳細については、“Compatible and Incompatible Encodings ” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

プログラム

```
LIBNAME inlib cvp 'SAS-library';
LIBNAME outlib 'SAS-library' outencoding="encoding value for output";
proc copy noclone in=inlib out=outlib;
```

```
   select car;
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 2 つのエンコーディングに互換性がある必要があります。

```
LIBNAME inlib cvp 'SAS-library';
LIBNAME outlib 'SAS-library' outencoding="encoding value for output";
proc copy noclone in=inlib out=outlib;
   select car;
run;
```

ログの例

ログ 14.3 *InLib ライブラリログ*

```
1    LIBNAME inlib cvp  'SAS-library'; NOTE:Libref INLIB was successfully
assigned as follows:Engine:        V9 Physical Name:SAS-library 2    LIBNAME
outlib 'SAS-library' outencoding="encoding value for output"; NOTE:Libref OUTLIB
was successfully assigned as follows:Engine:        V9 Physical Name:SAS-library
3    proc copy noclone in=inlib out=outlib; 4    select car; 5    run;
NOTE:Copying INLIB.CAR to OUTLIB.CAR (memtype=DATA).NOTE:System Options for
BUFSIZE and REUSE were used at user's request.NOTE:Libname and/or system options
for compress, pointobs, data representation and encoding attributes were used at
user's request.NOTE:Data file OUTLIB.CAR.DATA is in a format that is native to
another host, or the file encoding does not match the session encoding.Cross
Environment Data Access will be used, which might require additional CPU
resources and might reduce performance.NOTE:There were 25 observations read from
the data set INLIB.CAR.NOTE:The data set OUTLIB.CAR has 25 observations and 2
variables.
```

例 3: PROC COPY を使用して 32 ビットマシンから 64 ビットマシンに移行する

要素: PROC COPY ステートメントオプション
- IN=
- OUT=
- NOCLONE
- SELECTstatement

他の要素: OUTREP=データセットオプション

詳細

この例は、PROC COPY を使用して 32 ビット環境から 64 ビット環境に移行する方法を示しています。PROC MIGRATE は、32 ビット環境から 64 ビット環境に移行する場合に、アイテムストアをサポートしていません。

プログラム

```
libname source 'SAS-library';
libname target 'SAS-library'
```

```
    outrep=windows_64;

proc copy in=source out=target NOCLONE;
    select data-set-name;
run;
```

プログラムの説明

ライブラリリソースを割り当てます。 32 ビットマシンから 64 ビットマシンに変更する場合、OUTREP=オプションを使用します。

```
libname source 'SAS-library';
libname target 'SAS-library'
    outrep=windows_64;
```

データセットを 32 ビットマシンから 64 ビットマシンにコピーします。

```
proc copy in=source out=target NOCLONE;
    select data-set-name;
run;
```

15 章

CPORT プロシジャ

概要: CPORT プロシジャ

CPORT プロシジャの動作について

CPORT プロシジャは SAS データセット、SAS カタログ、または SAS ライブラリを順編成ファイル出力形式(移送ファイル)に書き出します。PROC CPORT を CIMPORT プロシジャと一緒に使うと、ある環境から他の環境にファイルを移動できます。*移送ファイル*は SAS ライブラリ、SAS カタログ、または SAS データセットを移送出力形式で保持している順編成ファイルです。PROC CPORT が書き出す移送出力形式は、すべての環境およびすべての SAS リリースを通して同じです。PROC CPORT では、*エクスポート*は SAS ライブラリ、SAS カタログまたは SAS データセットを移送出力形式にすることを意味します。PROC CPORT はカタログやデータセットを単一または SAS ライブラリをしてエクスポートできます。PROC CIMPORT は、移送ファイルを SAS カタログ、SAS データセットまたは SAS ライブラリの元の形式に戻します(*imports*)。

また、PROC CPORT は、SAS ファイルを*変換*します。変換とは、SAS ファイルの出力形式を SAS のあるバージョンに適しているものから、別のバージョンに適しているもの

に変更するということです。たとえば、PROC CPORT と PROC CIMPORT を使って、前のリリースの SAS からもっと新しいリリースの SAS にファイルを移動できます。PROC CIMPORT は自動で移送ファイルを変換して、それをインポートします。

注: PROC CIMPORT と PROC CPORT を使用して、グラフィックカタログをバックアップできます。PROC COPY はグラフィックカタログのバックアップには使用できません。

PROC CPORT は出力を作成しませんが(移送ファイル以外)、SAS ログに NOTES を書き出します。

移送ファイルの作成処理と読み込み処理

次に、ソースコンピュータで移送ファイルを作成して、それをターゲットコンピュータで読むプロセスを示します。

1. 移送ファイルは、ソースコンピュータで PROC CPORT を使って作成されます。
2. 移送ファイルは、ソースコンピュータからターゲットコンピュータへ、通信ソフトまたは磁気メディアを使って移送されます。
3. 移送ファイルは、ターゲットコンピュータで PROC CIMPORT を使って読み込まれます。

   注: 移送ファイルは、PROC CPORT を使って作成されたものと、XPORT エンジンを使って作成されたものとで、互換性はありません。

移送ファイルの作成(PROC CPORT)、移送ファイルの移動、移送ファイルの復元(PROC CIMPORT)のステップの詳細については、*SAS ファイルの移動とアクセス*を参照してください。

構文: CPORT プロシジャ

ヒント: グラフィックカタログをバックアップするには、PROC CIMPORT または PROC CPORT を使用してください。PROC COPY はグラフィックカタログのバックアップには使用できません。

参照項目: Windows、UNIX 、z/OS での CPORT プロシジャ

移送ファイルの作成(PROC CPORT)、移送ファイルの移動、移送ファイルの復元(PROC CIMPORT)のステップの詳細については、次を参照してください。*SAS ファイルの移動とアクセス*

PROC CPORT *source-type=libref* | <*libref.*>*member-name* <*option(s)*>;
EXCLUDE *SAS file(s)* | *catalog entry(s)* </ MEMTYPE=*mtype*>
</ ENTRYTYPE=*entry-type*>;
SELECT *SAS file(s)* | *catalog entry(s)* </ MEMTYPE=*mtype*>
</ ENTRYTYPE=*entry-type*> ; ;
TRANTAB NAME=*translation-table-name*
<*option(s)*>;

ステートメント	タスク	例
“PROC CPORT ステートメント”	移送ファイルを作成します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
“EXCLUDE ステートメント”	移送ファイルから 1 つ以上の指定ファイルを除外します	
“SELECT ステートメント”	移送ファイルに含める 1 つ以上のファイルやエントリを指定します	Ex. 2
“TRANTAB ステートメント”	エクスポートするカタログエントリの文字用の変換テーブル(複数可)を指定します。	Ex. 4

PROC CPORT ステートメント

移送ファイルを作成します。

例: “例 1: 複数のカタログのエクスポート” (429 ページ)
“例 2: 個々のカタログエントリのエクスポート” (430 ページ)
“例 3: 単一 SAS データセットのエクスポート” (431 ページ)
“例 4: 変換テーブルの適用” (432 ページ)
“例 5: 変更日に基づくエントリのエクスポート” (433 ページ)

構文

PROC CPORT *source-type=libref | <libref.>member-name <option(s)>*;

オプション引数の要約

NOEDIT
インポートした場合は編集不可で SAS/AF PROGRAM と SCL エントリをエクスポートします。

NOSRC
エクスポートされたカタログエントリはコンパイルされた SCL コードは含むがソースコードは含まないと指定します。

OUTLIB=*libref*
SAS ライブラリに関連付けされているライブラリ参照名を指定します。

Control the contents of the transport file

ASIS
表示される文字データの移送形式への変換を行いません。

CONSTRAINT=YES | NO
整合性の制約のエクスポートをコントロールします。

DATECOPY
作成日時と変更日時を移送ファイルにコピーします。

INDEX=YES|NO
インデックスとインデックス付きの SAS データセットのエクスポートをコントロールします。

INTYPE=*DBCS-type*
エクスポート対象の SAS ファイルに保存される DBCS データの種類を指定します。

NOCOMPRESS
移送ファイルでバイナリゼロとブランクの圧縮を抑制します。

OUTTYPE=UPCASE
すべてのアルファベット文字を大文字で移送ファイルに書き込みます。

TRANSLATE=(*translation-list*)
特定の文字を ASCII の一種または EBCDIC 値から他種へ変換します。

Identify the transport file

FILE=*fileref* | '*filename*'
書き込む移送ファイルを指定します。

TAPE
PROC CPORT の出力をテープに指定します。

Select files to export

AFTER=*date*
更新日が指定した日付以降の全データセットまたはカタログエントリのコピーをエクスポートします。

EET=(*etype(s)*)
移送ファイルから特定のエントリの種類を除外します。

ET=(*etype(s)*)
移送ファイルに特定のエントリの種類を含めます。

GENERATION=YES | NO
データセットのすべての世代をエクスポートするかどうか指定します。

MEMTYPE=*mtype*
ライブラリのエクスポート時にデータセットのみ、カタログのみ、または両方を移動させるかを指定します。

必須引数

source-type=libref* | < *libref.*>*member-name
エクスポートするファイルの種類を特定し、エクスポートするカタログ、SAS データセット、または SAS ライブラリを指定します。

source-type
単一カタログ、単一 SAS データセット、または SAS ライブラリのメンバとしてエクスポートする 1 つ以上のファイルを特定します。*source-type* 引数は、次のうちのどれかになります。

CATALOG | CAT | C

DATA | DS | D

LIBRARY | LIB | L

注 パスワード保護がかかっているデータセットをソースの種類として指定した場合、移送ファイル作成時にはパスワードも含まなくてはなりません。詳細については、“PROC CPORT ステートメントの READ=データセットオプション ” (428 ページ)を参照してください。

libref | *<libref.>member-name*
エクスポートする特定のカタログ、SAS データセット、または SAS ライブラリを指定します。*source-type* が CATALOG または DATA の場合、ライブラリ参照名とメンバ名の両方を指定できます。*libref* が省略された場合、PROC CPORT

はデフォルトのライブラリ(通常は WORK ライブラリ)を *libref* として使用します。*source-type* 引数が LIBRARY の場合、*libref* のみを指定します。ライブラリを指定した場合、PROC CPORT はライブラリからデータセットとカタログのみをエクスポートします。他の種類のファイルをエクスポートすることはできません。

参照項目 名前とメンバ名に使用できる命名規則については、“Names in the SAS Language” (*SAS Language Reference: Concepts*) を参照してください。

オプション引数

AFTER=*date*
更新日が指定した日付以降の全データセットまたはカタログエントリのコピーをエクスポートします。変更日時は、一番最近でデータセットやカタログの内容が変更になった日付です。日付を SAS 日付リテラルまたは SAS 日付値(数値)で指定します。

ヒント CATALOG プロシジャを使うと、カタログエントリの変更日時を特定できます。

例 “例 5: 変更日に基づくエントリのエクスポート” (433 ページ)

ASIS
表示される文字データの移送形式への変換を行いません。このオプションは、DBCS(ダブルバイト文字セット)データを含むファイルを、1 つのオペレーション環境から同じ種類の DBCS データを使う別の環境に移動する時に使います。

操作 ASIS オプションは NOCOMPRESS オプションを起動します。

同じ PROC CPORT ステップで ASIS オプションと OUTTYPE=オプションを同時に使用できません。

CONSTRAINT=YES | NO
データセットに定義されている整合性の制約のエクスポートをコントロールします。CONSTRAINT=YES を指定すると、ライブラリではすべての種類の整合性の制約がエクスポートされ、単一データセットでは一般整合性の制約のみエクスポートされます。CONSTRAINT=NO を指定した場合、一貫性制約なしで作成されたインデックスはポートされますが、一貫性制約も一貫性制約ありで作成されたインデックスもポートされません。一貫性制約については、*SAS 言語リファレンス: 解説編*の SAS ファイルのセクションを参照してください。

別名 CON=

デフォルト YES

操作 同じ PROC CPORT で CONSTRAINT=と INDEX=の両方を設定することはできません。

INDEX=NO を指定した場合、一貫性制約はエクスポートされません。

DATECOPY
SAS ファイルが最初に作成された時と最後に変更された時のそれぞれの SAS 内部日時を、結果として生じる移送ファイルにコピーします。動作環境の日時は保持されません。

制限事項 DATECOPY は出力ファイルが V8 または V9 エンジンを使用する場合のみ使用できます。

注 移送中のファイルに追加処理が必要な属性が含まれている場合、最終変更日は現在の日時に変更される可能性があります。

ヒント タイムゾーンオフセットを使用してデータセットを移送するには DATECOPY オプションを指定する必要があります。

作成日時は、PROC DATASETS ステップの MODIFY ステートメントの DTC=オプションで変更できます。詳細については、“MODIFY ステートメント” (508 ページ)を参照してください。

EET=(*etype(s)*)
移送ファイルから特定のエントリの種類を除外します。*etype* が単一エントリの種類の場合は、かっこは省略できません。複数の値はスペースで区切ります。

操作 同じ PROC CPORT ステップで、EET=オプションと ET=オプションを同時に使用できません。

ET=(*etype(s)*)
移送ファイルに特定のエントリの種類を含めます。*etype* が単一エントリの種類の場合は、かっこは省略できません。複数の値はスペースで区切ります。

操作 同じ PROC CPORT ステップで、EET=オプションと ET=オプションを同時に使用できません。

FILE=*fileref* | '*filename*'
書き込む移送ファイルの事前に定義されているファイル参照名またはファイル名を指定します。FILE=オプションを省略した場合、定義されていれば、PROC CPORT はファイル参照名 SASCAT に書き込みます。ファイル参照名 SASCAT が定義されていない場合、PROC CPORT は現在のディレクトリの SASCAT.DAT に書き込みます。

注 SASCAT が定義されていない場合の PROC CPORT の動作は、動作環境によって違います。詳細については、動作環境に関する SAS のドキュメントを参照してください。

例 すべての例

GENERATION=YES | NO
SAS データセットのすべての世代をエクスポートするかどうか指定します。データセットの基本生成のみエクスポートするには、GENERATION=NO を PROC CPORT ステートメントで指定します。特定の世代番号をエクスポートするには、PROC CPORT ステートメントでデータセットを指定する際に GENNUM=データセットオプションを使います。世代データセットの詳しい情報については、*SAS 言語リファレンス: 解説編*を参照してください。

別名 GEN=

デフォルト ライブラリは YES で単一データセットは NO

注 PROC CIMPORT は移送ファイルにあるデータセットのすべての世代をインポートします。同じ名前の前の世代セットを削除し、インポートされた世代セットと世代数が同じでなくても置き換えます。

INDEX=YES|NO
インデックスとインデックス付きのデータセットをエクスポートするか指定します。

デフォルト YES

操作 同じ PROC CPORT で CONSTRAINT=と INDEX=の両方を設定することはできません。

INDEX=NO を指定した場合、一貫性制約はエクスポートされません。

INTYPE=*DBCS-type*
エクスポート対象の SAS ファイルに保存される DBCS データの種類を指定します。ダブルバイト文字セット(DBCS)データはセットの各文字に最大 2 バイト使用します。*DBCS-type* は次の値のうちの 1 つになります。

IBM | HITAC | FACOM	z/OS の場合
IBM	VSE の場合
DEC | SJIS	OpenVMS の場合
PCIBM | SJIS	OS/2 の場合

デフォルト INTYPE=オプションを使用しない場合、DBCS は SAS システムオプションの DBCSTYPE=の値になります。

制限事項 INTYPE=オプションは、ダブルバイト文字セット(DBCS)拡張が SAS に付随している場合のみ使用できます。これらの拡張は膨大な計算リソースを使用するため、それを必要とするサイト用の特別なディストリビューションがあります。このオプションが DBCS 拡張が有効になっていないサイトで使われた場合、エラーがでます。

操作 INTYPE=オプションを OUTTYPE=オプションと同時に使用して、DBCS データの種類を他の種類に変更します。

INTYPE=オプションは NOCOMPRESS オプションを起動します。

同じ PROC CPORT ステップで INTYPE=オプションと ASIS オプションを同時に使用できません。

ヒント 構成ファイルの SAS システムオプション DBCSTYPE=の値を設定できます。

MEMTYPE=*mtype*
PROC CPORT が移送ファイルに書き込む SAS ファイルの種類を限定します。MEMTYPE= は処理を 1 つのメンバの種類に限定します。*mtype* の値は、次のどれかになります。

ALL
カタログとデータセットの両方

CATALOG | CAT
カタログ

DATA | DS
SAS データセット

別名 MT=

デフォルト ALL

例 “例 1: 複数のカタログのエクスポート” (429 ページ)

NOCOMPRESS
移送ファイルでバイナリゼロとブランクの圧縮を抑制します。

別名 NOCOMP

デフォルト PROC CPORT はスペースを節約するために、デフォルトでバイナリゼロとブランクを圧縮します。

操作 ASIS と INTYPE=と OUTTYPE=オプションは NOCOMPRESS オプションを起動します。

注 移送ファイルを圧縮しても、カタログやデータセットにある元のファイルが圧縮されていたかどうかのフラッグは変更されません。

NOEDIT
インポートした場合は編集不可で SAS/AF PROGRAM と SCL エントリをエクスポートします。

NOEDIT オプションでは、SAS/AF ソフトウェアの BUILD プロシジャで MERGE ステートメントで NOEDIT オプションを使用して SCL コードを含む新規カタログを作成するのと同じ結果が得られます。

別名 NEDIT

注 NOEDIT オプションは SAS/AF PROGRAM と SCL エントリにのみ影響します。FSEDIT SCREEN または FSVIEW FORMULA エントリには影響しません。

NOSRC
エクスポートされたカタログエントリはコンパイルされた SCL コードは含むがソースコードは含まないと指定します。

NOSRC オプションでは、SAS/AF ソフトウェアの BUILD プロシジャで MERGE ステートメントで NOSRC オプションを使用して SCL コードを含む新規カタログを作成するのと同じ結果が得られます。

別名 NSRC

OUTLIB=*libref*
SAS ライブラリに関連付けされているライブラリ参照名を指定します。OUTLIB= option を指定した場合、PROC CIMPORT は自動的に起動され、指定ライブラリ内の入力ライブラリ、データセット、またはカタログを再作成します。

別名 OUT=

ヒント 元のデータを完全なまま保持したい場合は、OUTLIB=オプションを使用して同じ動作環境内で SAS ファイルの DBCS の種類を別の種類に変更します。

OUTTYPE=UPCASE
すべての表示されている文字を大文字で移送ファイルと OUTLIB=ファイルに書き込みます。

操作 OUTTYPE=オプションは NOCOMPRESS オプションを起動します。

TAPE
PROC CPORT の出力をテープに指定します。

デフォルト PROC CPORT からの出力がディスクに送信されます。

TRANSLATE=(*translation-list*)
特定の文字を ASCII の一種または EBCDIC 値から他種へ変換します。*translation-list* の各要素は次の形式になります。

- *ASCII-value-1* TO *ASCII-value-2*
- *EBCDIC-value-1* TO *EBCDIC-value-2*

ASCII の値には 16 進数表現も 10 進数表現も使用できます。16 進数表現を使用する場合、値は数字で始まり、x で終わらなければなりません。16 進数の値がアルファベット文字で始まる場合は、先頭にゼロを追加します。

例えば、すべての左かっこを左中かっこに変換するには、TRANSLATE=オプションを次のように指定します(ASCII 文字の場合)。

```
translate=(5bx to 7bx)
```

次の例はすべての左かっこを左中かっこに、そしてすべての右かっこを右中かっこに変換します。

```
translate=(5bx to 7bx 5dx to 7dx)
```

EXCLUDE ステートメント

指定ファイルまたはエントリを移送ファイルから除外します。

操作: PROC CPORT ステップでは EXCLUDE ステートメントまたは SELECT ステートメントを使用できますが、両方を同時には使用できません。

ヒント: PROC CPORT 一回の起動で、使用できる EXCLUDE ステートメントの数に制限はありません。

構文

EXCLUDE *SAS file(s) | catalog entry(s)* </ MEMTYPE=*mtype*>
</ ENTRYTYPE=*entry-type*>;

必須引数

SAS file(s) | catalog entry(s)
移送ファイルから除外する SAS ファイルまたはカタログを指定します。SAS ライブラリのエクスポート時は SAS ファイル名を指定し、個別 SAS カタログのエクスポート時はカタログエントリ名を指定します。複数のファイル名またはエントリ名はスペースで区切ります。EXCLUDE ステートメントでは、ショートカットを使って、多数の似ている名前をリストすることができます。詳細については、“変数名リストを示すショートカット” (57 ページ)を参照してください。

オプション引数

ENTRYTYPE=*entry-type*
EXCLUDE ステートメントにリストされているカタログエントリには、単一のエントリの種類を指定します。カタログエントリの種類については、*SAS 言語リファレンス: 解説編*を参照してください。

別名 ETYPE=, ET=

制限事項 ENTRYTYPE=は個別の SAS カタログをエクスポートする時にのみ有効です。

MEMTYPE=*mtype*
EXCLUDE ステートメントにリストされている 1 つ以上の SAS ファイルに対して、単一のメンバの種類を指定します。有効な値は CATALOG または CAT、DATA、または ALL です。EXCLUDE ステートメントで MEMTYPE=オプションを指定しない場合、処理は PROC CPORT ステートメントの MEMTYPE=オプションで指定したメンバの種類に限定されます。

ファイル名の後に、MEMTYPE= option をかっこで囲んで指定することもできます。かっこの中で、MEMTYPE=はそのすぐ前にあるファイル名の種類を識別します。オプションのこの形式を使用する場合は、EXCLUDE ステートメントでスラッシュの後に続く MEMTYPE=オプションを無効にします。

別名 MTYPE=、MT=

デフォルト PROC CPORT ステートメントまたは EXCLUDE ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

制限事項 MEMTYPE=は SAS ライブラリをエクスポートする時にのみ有効です。

PROC CPORT ステートメントの MEMTYPE=でメンバの種類を指定する場合、EXCLUDE ステートメントの MEMTYPE=で指定するメンバの種類と一致しなければなりません。

参照項目 名前とメンバ名に使用できる命名規則については、“Names in the SAS Language” (*SAS Language Reference: Concepts*) を参照してください。

SELECT ステートメント

指定ファイルまたはエントリを移送ファイルに含めます。

操作: PROC CPORT ステップでは EXCLUDE ステートメントまたは SELECT ステートメントを使用できますが、両方を同時には使用できません。

ヒント: PROC CPORT 一回の起動で、使用できる SELECT ステートメントの数に制限はありません。

例: “例 2: 個々のカタログエントリのエクスポート” (430 ページ)

構文

SELECT *SAS file(s) | catalog entry(s)* </ MEMTYPE=*mtype*>
</ ENTRYTYPE=*entry-type*> ; ;

必須引数

SAS file(s) | catalog entry(s)
移送ファイルに含む SAS ファイルまたはカタログを指定します。SAS ライブラリのエクスポート時は SAS ファイル名を指定し、個別 SAS カタログのエクスポート時はカタログエントリ名を指定します。複数のファイル名またはエントリ名はスペースで区切ります。SELECT ステートメントでは、ショートカットを使って、多数の似ている

名前をリストすることができます。詳細については、“変数名リストを示すショートカット” (57 ページ)を参照してください。

オプション引数

ENTRYTYPE=*entry-type*
SELECT ステートメントにリストされているカタログエントリには、単一のエントリの種類を指定します。カタログエントリの種類については、*SAS 言語リファレンス: 解説編*を参照してください。

別名 ETYPE=, ET=

制限事項 ENTRYTYPE=は個別の SAS カタログをエクスポートする時にのみ有効です。

MEMTYPE=*mtype*
SELECT ステートメントにリストされている 1 つ以上の SAS ファイルに対して、単一のメンバの種類を指定します。有効な値は CATALOG または CAT、DATA、または ALL です。SELECT ステートメントで MEMTYPE=オプションを指定しない場合、処理は PROC CPORT ステートメントの MEMTYPE=オプションで指定したメンバの種類に限定されます。

メンバの名前の後に、MEMTYPE= option をかっこで囲んで指定することもできます。かっこの中で、MEMTYPE=はそのすぐ前にあるメンバの名前の種類を識別します。オプションのこの形式を使用する場合は、SELECT ステートメントでスラッシュの後に続く MEMTYPE=オプションを無効にします。

別名 MTYPE=、MT=

デフォルト PROC CPORT ステートメントまたは SELECT ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

制限事項 MEMTYPE=は SAS ライブラリをエクスポートする時にのみ有効です。

PROC CPORT ステートメントの MEMTYPE=でメンバの種類を指定する場合、SELECT ステートメントの MEMTYPE=で指定するメンバの種類と一致しなければなりません。

参照項目 名前とメンバ名に使用できる命名規則については、“Names in the SAS Language” (*SAS Language Reference: Concepts*) を参照してください。

TRANTAB ステートメント

エクスポートするカタログエントリの文字に対し翻訳テーブルを指定します。

ヒント: 各 TRANTAB ステートメントでは 1 つの変換テーブルしか指定できません。ただし、PROC CPORT の一回の起動で複数の変換テーブルを使用することができます。

参照項目: CPORT プロシジャと UPLOAD と DOWNLOAD プロシジャの TRANTAB ステートメントに関しては、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

例: “例 4: 変換テーブルの適用” (432 ページ)

構文

TRANTAB NAME=*translation-table-name <option(s)>*;

PROC CPORT ステートメントの READ=データセットオプション

読み取り保護されているデータセットの移送ファイルを作成するには、パスワード(クリアテキストまたはエンコードされているもの)を提供しなくてはなりません。パスワードが含まれていない場合、移送ファイルは作成できません。

パスワード保護付きのデータセットを扱っている場合は、READ=オプションでパスワードを提供できます。READ=オプションで読み取り保護されたデータセットのパスワードを提供しない場合、パスワードの入力を求めるメッセージが表示されます。

読み取り保護のデータセットの移送ファイルを作成するには、READ=データセットオプションを使って、適切なパスワードを提供します。例 1 では、PROC CPORT は SOURCE.GRADES という名前の入力ファイルを コピーし、データセットとパスワード ADMIN を提供し、GRADESOUT という名前の移送ファイルを作成します。

例 1:クリアテキスト パスワード:

```
proc cport data=source.grades(read=admin) file=gradesout;
```

例 2 では、エンコードされたパスワードは READ=オプションで指定されます。エンコードされたパスワードは、PWENCODE プロシジャによって生成されます。詳細については、51 章, “PWENCODE プロシジャ,” (1495 ページ)を参照してください。

例 2:エンコードされたパスワード

```
proc cport
data=source.grades(read={sas003}6EDB396015B96DBD9E80F0913A543819A8E5)
   file=gradesout;
```

パスワード保護さらたデータセットを参照するケースでパスワードが省略されている場合、SAS はパスワード入力を促す画面を表示します。有効でないパスワードが入力された場合、ログにエラーメッセージが出力されます。エラーの例は次になります。

```
ERROR: Invalid or missing READ password on member WORK.XYZ.DATA
```

データセットがライブラリの一部として移送される場合や SELECT ステートメントで指定されている場合は、パスワードは必要ありません。ただし、パスワード保護のデータセットが移送されてもターゲット SAS エンジンがパスワードをサポートしていない場合、移送ファイルのインポートはできません。

READ=データセットオプションの詳細については、*SAS データセットオプション: リファレンス*を、そしてパスワード保護されたデータセットの詳細については、*SAS 言語リファレンス: 解説編*を参照してください。

注: PROC CIMPORT は移送先の環境で移送ファイルの復元でパスワードを必要としません。ただし、パスワード保護のデータを扱う他の SAS プロシジャではパスワードが必要になります。

CPORT での問題:移送ファイルの作成

データ制御ブロックの特性

z/OS 環境下での移送ファイルの作成およびインポートでのよくある問題は、データ制御ブロック(DCB)の特性の指定もれです。移送ファイルを参照する場合は、必ず次の DCB の特性を指定してください。

- LRECL=80
- BLKSIZE=8000
- RECFM=FB
- DSORG=PS

他のよくある問題は、他の環境から z/OS にファイルを移動する際に通信ソフトウェアを使う場合に起こります。移送ファイルが z/OS に来たときには正しいデータ制御ブロックの特性を持っていないことがあります。通信ソフトウェアでファイル特性の指定ができない場合、z/OS では次のアプローチを試してください。

1. z/OS でファイルを正しいデータ制御ブロックの特性付きで作成し、そして初期化します。
2. 移送ファイルを他の環境から z/OS で新規に作成されたファイルにバイナリで移動します。

数値精度の損失

PROC CPORT と PROC CIMPORT では、極端に小さいまたは大きい数値では精度を欠く場合があります。詳細については、“Loss of Numeric Precision and Magnitude” (*SAS/CONNECT User's Guide*)を参照してください。

例: CPORT プロシジャ

例 1: 複数のカタログのエクスポート

要素: PROC CPORT ステートメントオプション
FILE=
MEMTYPE=

詳細

この例では、PROC CPORT を使って指定する SAS ライブラリのすべての SAS カタログにあるエントリーをエクスポートする方法を示します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;

proc cport library=source file=tranfile memtype=catalog;
run;
```

プログラムの説明

エクスポート対象のソースファイルを含む SAS ライブラリのライブラリ参照と、出力移送ファイルが書き込まれるファイル参照を指定します。 LIBNAME ステートメントは、SAS ライブラリのライブラリ参照名を割り当てます。FILENAME ステートメントは、PROC CPORT が作成する移送ファイルのファイル特性に対するファイル参照名と動作環境オプションを割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;
```

移送ファイルを作成します。 PROC CPORT ステップは、ソースライブラリがある動作環境で実行します。MEMTYPE=CATALOG は、ソースライブラリのすべての SAS カタログを移送ファイルに書き込みます。

```
proc cport library=source file=tranfile memtype=catalog;
run;
```

ログの例

ログ 15.1 *複数のカタログのエクスポート*

```
NOTE:Proc CPORT begins to transport catalog SOURCE.FINANCE NOTE:The catalog has 5 entries and its
maximum logical record length is 866.NOTE:Entry LOAN.FRAME has been transported.NOTE:Entry LOAN.HELP
has been transported.NOTE:Entry LOAN.KEYS has been transported.NOTE:Entry LOAN.PMENU has been
transported.NOTE:Entry LOAN.SCL has been transported.NOTE:Proc CPORT begins to transport catalog
SOURCE.FORMATS NOTE:The catalog has 2 entries and its maximum logical record length is 104.NOTE:Entry
REVENUE.FORMAT has been transported.NOTE:Entry DEPT.FORMATC has been transported.
```

例 2: 個々のカタログエントリのエクスポート

要素: PROC CPORT ステートメントオプション
FILE=
SELECT ステートメント

詳細

この例では、PROC CPORT を使って、カタログにあるすべてのエントリではなく、個々のカタログエントリをエクスポートする方法を示します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;

proc cport catalog=source.finance file=tranfile;
select loan.scl;
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME と FILENAME ステートメントは、ソースライブラリのライブラリ参照名と、移送ファイルのファイル参照名をそれぞれ割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;
```

エントリを移送ファイルに書き込みます。 SELECT は、LOAN.SCL エントリのみをエクスポート対象の移送ファイルに書き込みます。

```
proc cport catalog=source.finance file=tranfile;
select loan.scl;
run;
```

ログの例

ログ 15.2 個々のカタログエントリのエクスポート

```
NOTE:Proc CPORT begins to transport catalog SOURCE.FINANCE NOTE:The catalog has 5 entries and its
maximum logical record length is 866.NOTE:Entry LOAN.SCL has been transported.
```

例 3: 単一 SAS データセットのエクスポート

要素: PROC CPORT ステートメントオプション
FILE=

詳細

この例では、PROC CPORT を使って単一 SAS データセットをエキスポートする方法を示します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;

proc cport data=source.times file=tranfile;
```

```
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME と FILENAME ステートメントは、ソースライブラリのライブラリ参照名と、移送ファイルのファイル参照名をそれぞれ割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;
```

エクスポート中のファイルの種類を指定します。 PROC CPORT ステートメントの DATA=指定により、プロシジャはライブラリやカタログではなく、SAS データセットがエクスポート中であることを認識します。

```
proc cport data=source.times file=tranfile;
run;
```

ログの例

ログ 15.3 単一 SAS データセットのエクスポート

```
NOTE:Proc CPORT begins to transport data set SOURCE.TIMES NOTE:The data set contains 2 variables and 2
observations.Logical record length is 16.NOTE:Transporting data set index information.
```

例 4: 変換テーブルの適用

要素: PROC CPORT ステートメントオプション
FILE=
TRANTAB ステートメントオプション
TYPE=

詳細

この例では、PROC CPORT のエクスポート前に移送ファイルにカスタマイズされた変換テーブルを適用する方法を示します。この例では、TTABLE1 というカスタマイズされた変換テーブルを作成したと仮定します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;

proc cport catalog=source.formats file=tranfile;
   trantab name=ttable1 type=(format);
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME と FILENAME ステートメントは、ソースライブラリのライブラリ参照名と、移送ファイルのファイル参照名をそれぞれ割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;
```

翻訳固有を適用します。 TRANTAB ステートメントは、カスタマイズした翻訳テーブル TTABLE1 で指定する翻訳を適用します。TYPE=は、翻訳を FORMAT エントリに制限します。

```
proc cport catalog=source.formats file=tranfile;
   trantab name=ttable1 type=(format);
run;
```

ログの例

ログ 15.4 *変換テーブルの適用*

```
NOTE:Proc CPORT begins to transport catalog SOURCE.FORMATS NOTE:The catalog has 2 entries and its
maximum logical record length is 104.NOTE:Entry REVENUE.FORMAT has been transported.NOTE:Entry
DEPT.FORMATC has been transported.
```

例 5: 変更日に基づくエントリのエクスポート

要素: PROC CPORT ステートメントオプション
AFTER=
FILE=

詳細

この例は、PROC CPORT で AFTER=オプションで指定する日付以降に変更されたカタログエントリのみを移送する方法を示します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;

proc cport catalog=source.finance file=tranfile
           after='09sep1996'd;
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME と FILENAME ステートメントは、ソースライブラリのライブラリ参照名と、移送ファイルのファイル参照名をそれぞれ割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file'

host-option(s)-for-file-characteristics;
```

移送ファイルに書き込まれるカタログエントリを指定します。 AFTER=は、変更日が 1996 年 9 月 9 日以降のカタログエントリのみが移送ファイルに書き込まれるように指定します。

```
proc cport catalog=source.finance file=tranfile
           after='09sep1996'd;
run;
```

ログの例

PROC CPORT は指定カタログのすべてのエントリのエクスポートプロセスが開始したと知らせるメッセージを SAS ログに出力します。しかし、PROC CPORT は FINANCE カタログの LOAN.FRAME と LOAN.HELP エントリのみを移送ファイルに書き出しています。なぜなら、それら 2 つのエントリのみの変更日時が 1996 年 9 月 9 日以降だったからです。すなわち、指定カタログ内のすべてのエントリで、その 2 つのみが AFTER=オプションの要件を満たしていたからです。

ログ 15.5 *変更日に基づくエントリのエクスポート*

```
NOTE:Proc CPORT begins to transport catalog SOURCE.FINANCE NOTE:The catalog has 5 entries and its
maximum logical record length is 866.NOTE:Entry LOAN.FRAME has been transported.NOTE:Entry LOAN.HELP
has been transported.
```

16 章

DATASETS プロシジャ

概要: DATASETS プロシジャ

DATASETS プロシジャの動作について

DATASETS プロシジャは SAS ファイルを管理するユーティリティプロシジャです。PROC DATASETS を使って、次の事ができます。

- SAS ファイルを 1 つの SAS ライブラリから他のライブラリにコピーする
- SAS ファイルの名前を変更する
- SAS ファイルを修復する
- SAS ファイルを削除する
- SAS ライブラリにある SAS ファイルをリストする
- SAS データセットの属性をリストする。属性の例は、
  - データが最後に変更された日付
  - データが圧縮されているかどうか
  - データがインデックス付きかどうか

- SAS ファイルのパスワードを操作する
- SAS データセットに追加する
- SAS データセットの属性やデータセット内の変数を変更する
- SAS データセットのインデックスを作成および削除する
- SAS データセットの監査ファイルを作成および管理する
- SAS データセットの一貫性制約を作成および削除する
- データセットの拡張属性を作成および管理する

PROC DATASETS のサンプル出力

DATASETS プロシジャには、次の関数が含まれます。

- CONTROL ライブラリのすべてのデータセットを HEALTH ライブラリにコピー
- HEALTH ライブラリの内容をリスト表示
- HEALTH ライブラリから SYNDROME データセットを削除
- PRENAT データセットの名前を INFANT に変更

SAS ログは次のように出力されます。

```
LIBNAME control 'SAS-library-1';
LIBNAME health 'SAS-library-2';

proc datasets memtype=data;
copy in=control out=health;
run;

proc datasets library=health memtype=data details;
delete syndrome;
change prenat=infant;
run;
quit;
```

ログ 16.1 *PROC DATASETS のログ*

```
744 proc datasets library=health memtype=data details;
Directory

Libref HEALTH
Engine V9
Physical Name c:\Documents and Settings\myfile\My Documents\procdatasets\health
Filename c:\Documents and Settings\myfile\My Documents\procdatasets\health


Member Obs, Entries File
# Name Type or Indexes Vars Label Size Last Modified

1 ALL DATA 23 17 13312 12Sep07:10:57:50
2 BODYFAT DATA 83 13 California Results 13312 12Sep07:10:57:54
3 CONFOUND DATA 8 4 5120 12Sep07:10:57:50
4 CORONARY DATA 39 4 5120 12Sep07:10:57:50
5 DRUG1 DATA 6 2 JAN2005 DATA 5120 12Sep07:10:57:50
6 DRUG2 DATA 13 2 MAY2005 DATA 5120 12Sep07:10:57:50
7 DRUG3 DATA 11 2 JUL2005 DATA 5120 12Sep07:10:57:50
8 DRUG4 DATA 7 2 JAN2002 DATA 5120 12Sep07:10:57:50
9 DRUG5 DATA 1 2 JUL2002 DATA 5120 12Sep07:10:57:50
10 GROUP DATA 148 11 Test Subjects 33792 12Sep07:13:01:16
GROUP INDEX 1 9216 12Sep07:13:01:16
11 GRPOUT DATA 11 40 17408 13Dec10:11:40:24
12 GRPOUT1 DATA 11 40 17408 13Dec10:10:28:32
13 INFANT DATA 149 6 17408 12Sep07:10:57:52
14 MLSCL DATA 32 4 Multiple Sclerosis Data 5120 12Sep07:10:57:52
15 NAMES DATA 7 4 5120 12Sep07:10:57:52
16 OXYGEN DATA 31 7 17408 12Sep07:13:01:16
17 PERSONL DATA 148 11 25600 12Sep07:10:57:52
18 PHARM DATA 6 3 Sugar Study 5120 12Sep07:10:57:52
19 POINTS DATA 6 6 5120 12Sep07:10:57:52
20 RESULTS DATA 10 5 5120 12Sep07:10:57:54
21 SLEEP DATA 108 6 9216 12Sep07:10:57:54
22 TEST2 DATA 15 5 5120 12Sep07:10:57:54
23 TRAIN DATA 7 2 5120 12Sep07:10:57:54
24 VISION DATA 16 3 5120 12Sep07:10:57:54
25 WEIGHT DATA 1 2 5120 12Sep07:10:57:50
26 WGHT DATA 83 13 13312 12Sep07:10:57:54
745 delete syndrome;
746 change prenat=infant;
747 run;
ERROR: The file HEALTH.PRENAT (memtype=DATA) was not found, but appears on a CHANGE statement.
748 quit;
749
750 proc datasets memtype=data;
```

注

- DATASETS プロシジャでもカタログに対していくつかの操作はできますが、通常はカタログ管理には CATALOG プロシジャが最適なユーティリティです。
- *member* という用語は、しばしば *SAS ファイル*と同意語で使われます。SAS ファイルと SAS ライブラリの詳細については、“SAS Files Concepts” (*SAS Language Reference: Concepts*)を参照してください。
- PROC DATASETS は順次データライブラリとは使用できません。
- LENGTH ステートメントや、ATTRIB ステートメントの LENGTH=オプションを使用して、変数の長さを変更することはできません。

- PROC DATASETS や PROC CONTENTS および SAS エクスプローラのような他の SAS のコンポーネントの間で、変更日時に差がでることがあります。2 つの変更日時には明らかな違いがあります。
  - オペレーション環境の変更日時は SAS エクスプローラと PROC DATASETS LIST オプションで表示されます。
  - CONTENTS ステートメントで使われる変更日付はデータセット内のデータが実際に変更された日時です。
- 大量のメンバを含むライブラリの場合、DATASETS プロシジャの処理時間が長くなる場合があります。パフォーマンスを向上するには、ライブラリを小さいライブラリに再編成するとよいでしょう。

概念

プロシジャの実行

ステートメントの実行

DATASETS プロシジャを開始する際、PROC DATASETS ステートメントでプロシジャ入力ライブラリを指定します。プロシジャ入力ライブラリを省略すると、プロシジャは現在のデフォルトの SAS ライブラリ(通常は WORK ライブラリ)を使います。新しいプロシジャ入力ライブラリを指定するには、DATASETS プロシジャを再度発行してください。

ステートメントは記述された順番で実行されます。データセットのコンテンツを確認し、データセットをコピーして、最初のデータセットのコンテンツと 2 番目のものを視覚的に比較する場合、CONTENTS、COPY、CONTENTS を使用します。

RUN グループ処理

PROC DATASETS は RUN グループ処理をサポートしています。RUN グループ処理はプロシジャを終了させることなく RUN グループのサブミットを可能にします。

DATASETS プロシジャは 4 種類の RUN グループをサポートしています。各 RUN グループはそれを構成および実行するステートメントで定義されます。

PROC DATASETS のいくつかのステートメントは、インプライド RUN ステートメントとしての役割を果たし、その前にある RUN グループを実行させます。

次のリストは、どのステートメントが RUN グループを構成して、何が各 RUN グループを実行させるかについて説明します。

- PROC DATASETS ステートメントは、いつでも即座に実行されます。PROC DATASETS ステートメントの実行に、他のステートメントは必要ありません。したがって、PROC DATASETS ステートメントは、それ単体で RUN グループになります。
- MODIFY ステートメントとそれに従属するステートメントは RUN グループを形成します。これらの RUN グループは、いつでも即座に実行されます。MODIFY RUN グループの実行に、他のステートメントは必要ありません。
- APPEND、CONTENTS、および COPY ステートメント(存在する場合は、EXCLUDE と SELECT も含む)は、それぞれに別の RUN グループを形成します。すべての APPEND ステートメントは単一ステートメント RUN グループ、すべての CONTENTS ステートメントも単一ステートメント RUN グループ、そしてすべての COPY ステップは RUN グループを形成します。プロシジャのその他のステートメン

ト(COPY ステートメントまたは NODIFY ステートメントのいずれかに従属しているステートメントを除く)により、RUN グループが実行されます。

- RUN グループは、次のうち 1 つ以上のステートメントから形成されます。
  - AGE
  - CHANGE
  - DELETE
  - EXCHANGE
  - REPAIR
  - SAVE

  これらのステートメントのうちいずれかが PROC ステップのシーケンスで記述されている場合、そのシーケンスは RUN グループを形成しています。たとえば、REPAIR ステートメントが SAVE ステートメントの直後に記述されている場合、REPAIR ステートメントは強制的に SAVE ステートメントを実行しません。同じ RUN グループの一部となります。RUN グループを実行するには、次のステートメントのうちいずれかをサブミットします。
  - PROC DATASETS
  - APPEND
  - CONTENTS
  - COPY
  - MODIFY
  - QUIT
  - RUN
  - 別の DATA ステップまたは PROC ステップ

1 つのタスクと関連付けられているプログラムステートメントが、RUN ステートメントまたはインプライド RUN ステートメントまで読み込まれます。その前のすべてのステートメントが即座に実行され、別の RUN ステートメントまたはインプライド RUN ステートメントまで読み込みが続行されます。最後のタスクを実行するには、RUN ステートメント、またはプロシジャを停止するステートメントを使用する必要があります。

次の PROC DATASETS ステップには、5 つの RUN グループが含まれています。

```
LIBNAME dest 'SAS-library';
/* RUN group */

proc datasets;
/* RUN group */
change nutr=fatg;
delete bldtest;
exchange xray=chest;
/* RUN group */
copy out=dest;
select report;
/* RUN group */
modify bp;
label dias='Taken at Noon';
rename weight=bodyfat;
/* RUN group */
append base=tissue data=newtiss;
```

```
quit;
```

注: 対話型ラインモードで実行している場合、RUN ステートメントをサブミットする前にステートメントがすでに実行されたというメッセージを受信できます。この環境を PROC DATASETS の実行に使用している場合、タスクの計画には注意してください。

エラー処理

通常、ステートメントでエラーが発生した場合、エラーを含む RUN グループは実行されません。エラーを含む RUN グループの前後の RUN グループは正常に実行されます。MODIFY RUN グループは例外です。MODIFY ステートメントに従属しているステートメントで構文エラーが発生した場合、エラーを含むステートメントのみ実行されません。RUN グループのその他のステートメントは実行されます。

注: ステートメント(ステートメント名)の最初の文字にエラーがあり、プロシジャでそれを認識できない場合、プロシジャはそのステートメントを前の RUN グループの一部として処理します。

パスワードエラー

ステートメントに正しくないパスワードまたは省略されたパスワードに関連するエラーがある場合、エラーはエラーを含むステートメントにのみ影響します。RUN グループのその他のステートメントは実行されます。

エラーのある RUN グループの強制実行

PROC DATASETS ステートメントの FORCE オプションは、1 つ以上のステートメントにエラーが含まれている場合でも RUN グループの実行を強制します。エラーのないステートメントのみ実行されます。

プロシジャの終了

DATASETS プロシジャを停止するには、QUIT ステートメント、RUN CANCEL ステートメント、新しい PROC ステートメントまたは DATA ステートメントのいずれかを発行する必要があります。QUIT ステートメントをサブミットすると、実行されなかったステートメントが実行されます。RUN CANCEL ステートメントをサブミットすると、実行されなかったステートメントがキャンセルされます。

DATASETS プロシジャでパスワードを使用する

DATASETS プロシジャのいくつかのステートメントでは、SAS ファイルのパスワードを操作するオプションがサポートされています。これらのオプション ALTER=、PW=、READ=、WRITE=もデータセットオプションです。[1]SAS ファイルへのパスワードの影響については、“Assigning Passwords” (*SAS Language Reference: Concepts*)を参照してください。

ステートメント AGE、CHANGE、DELETE、EXCHANGE、REPAIR、SELECT でパスワード保護されている SAS ファイルを使用している場合、PROC DATASETS ステートメントまたはそれに従属するステートメントで ALTER=および PW=パスワードオプションを指定できます。

注: ALTER=オプションは、COPY(ファイルの移動時)ステートメントおよび MODIFY ステートメントについては多少結果が異なります。詳細については、COPY ステートメント (480 ページ) および MODIFY ステートメント (508 ページ)を参照してください。

---

1 APPEND ステートメントと CONTENTS ステートメントでは、これらのオプションを SAS データセットオプションを使用するのと同じように、SAS データセット名の後にかっこで囲んで使用します。

パスワードの検索は、次の順序で実行されます。

1. 下位ステートメントの SAS ファイル名の後のかっこ内。かっこ内で使用される場合、オプションはそのオプションの直前の名前のみ参照します。データライブラリ内の複数の SAS ファイルを使用しており、SAS ファイルにそれぞれ異なるパスワードがある場合、個別の名前の後にパスワードオプションをかっこで囲んで指定する必要があります。

   次のステートメントで、ALTER=オプションは SAS ファイル BONES にのみパスワード RED を付与します。

   ```
   delete xplant bones(alter=red);
   ```

2. 下位ステートメントのスラッシュ(/)の後。スラッシュの後にパスワードオプションを使用する場合、同じオプションが SAS ファイル名の後のかっこ内に記述されない限り、オプションはステートメントで指定したすべての SAS ファイルを参照します。この方法は、使用している複数の SAS ファイルすべてのパスワードが同一である場合に使用します。

   次のステートメントで、かっこ内の ALTER=オプションが SAS ファイル CHEST にパスワード RED を、スラッシュの後の ALTER=オプションが SAS ファイル VIRUS にパスワード BLUE を付与します。

   ```
   delete chest(alter=red) virus / alter=blue;
   ```

3. PROC DATASETS ステートメント内。PROC DATASETS ステートメントでのパスワード指定は、ライブラリで使用しているすべての SAS ファイルのパスワードが同一である場合に便利です。この場合、オプションはかっこ内に指定しないでください。

   次の PROC DATASETS ステップで、PW=オプションは SAS ファイル INSULIN と ABNEG に対してパスワード RED を与えます。

   ```
   proc datasets pw=red;
   delete insulin;
   contents data=abneg;
   run;
   ```

   注: SELECT ステートメントの SAS ファイルに対するパスワードの場合、PROC DATASETS ステートメントの前に COPY ステートメントが検索されます。

処理するメンバの種類の制限

PROC DATASETS ステートメント内

下位ステートメントの複数のメンバの種類を参照し、メンバの種類を PROC DATASETS ステートメントで指定した場合、そのメンバの種類をすべて PROC DATASETS ステートメントに含めます。元の PROC DATASETS ステートメントのメンバの種類のみ有効です。次の例は、複数のメンバの種類をリストしたものです。

```
proc datasets lib=library memtype=(data view);
```

下位ステートメント内

次の下位ステートメントで MEMTYPE=オプションを使用し、処理可能なメンバの種類に制限します。

AGE CHANGE DELETE EXCHANGE

EXCLUDE REPAIR SAVE SELECT

注: MEMTYPE=オプションは、ステートメント CONTENTS、COPY、MODIFY については多少結果が異なります。詳細については、CONTENTS ステートメント (472 ページ)、COPY ステートメント (480 ページ)、および MODIFY ステートメント (508 ページ)を参照してください。

プロシジャでは、MEMTYPE=が次の順序で検索されます。

1. SAS ファイル名の直後のかっこ内。かっこ内で使用される場合、MEMTYPE= オプションはそのオプションの直前の SAS ファイルのみ参照します。たとえば、次のステートメントは、HOUSE.DATA、LOT.CATALOG、SALES.DATA を削除します。DELETE ステートメントのデフォルトのメンバの種類が DATA であるためです。(詳細については、表 16.1 (444 ページ)の各ステートメントのデフォルトタイプを参照してください)

```
delete house lot(memtype=catalog) sales;
```

2. ステートメントの最後のスラッシュ(/)の後。スラッシュの後に使用される場合、MEMTYPE= オプションは SAS ファイル名の後のかっこ内に記述されない限り、ステートメントで指定したすべての SAS ファイルを参照します。たとえば、次のステートメントは Lotpix.catalog、Regions.data、および Appl.catalog を削除します。

```
delete lotpix regions(memtype=data) appl / memtype=catalog;
```

3. PROC DATASETS ステートメント内。たとえば、この DATASETS プロシジャは APPL.CATALOG を削除します。

```
proc datasets memtype=catalog;
delete appl;
run;
```

   注: EXCLUDE ステートメントと SELECT ステートメントを使用する際は、プロシジャは MEMTYPE=オプションを PROC DATASETS ステートメントの前に COPY ステートメントで検索します。詳細については、"SAS ファイルのコピー/移動時のメンバの種類の指定" (487 ページ) を参照してください。

4. (デフォルト値について) MEMTYPE=オプションを下位ステートメントまたは PROC DATASETS ステートメントで指定しない場合、下位ステートメントのデフォルト値により、処理可能なメンバの種類が決定されます。

メンバの種類

MEMTYPE=オプションに指定できる値は、次のとおりです。

ACCESS
: アクセスディスクリプタファイル (SAS/ACCESS ソフトウェアによって作成)

ALL
: すべてのメンバの種類

CATALOG
: SAS カタログ

DATA
: SAS データファイル

FDB
: 財務データベース

MDDB
: 多次元データベース

PROGRAM
: 保存されたコンパイル済み SAS プログラム

VIEW
SAS ビュー

次の表に、各ステートメントで使用できるメンバの種類を示します。

表 16.1 *下位ステートメントと適切なメンバの種類*

ステートメント	適切なメンバの種類	デフォルトのメンバの種類
AGE	ACCESS、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	DATA
CHANGE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
CONTENTS	ALL、DATA、VIEW	DATA *
COPY	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
DELETE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	DATA
EXCHANGE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
EXCLUDE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
MODIFY	ACCESS、DATA、VIEW	DATA
REPAIR	ALL、CATALOG、DATA	ALL **
SAVE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
SELECT	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL

* CONTENTS ステートメントで DATA=_ALL_の場合、デフォルトは ALL です。ALL には、DATA と VIEW のみ含まれます。

** ALL には、DATA と CATALOG のみ含まれます。

世代データセットの処理制限

DATASETS プロシジャの複数のステートメントでは、世代データセットの処理を制限するための GENNUM=オプションがサポートされています。GENNUM=もデータセットオプションです。[1] 世代データセットの要求および使用方法については、“Understanding Generation Data Sets” (*SAS Language Reference: Concepts*)の、“Generation Data Sets”を参照してください。

1 APPEND ステートメントと CONTENTS ステートメントの場合、GENNUM=を SAS データセットオプションを使用するのと同じように SAS データセット名の後にかっこで囲んで使用します。

ステートメント AUDIT、CHANGE、DELETE、MODIFY、REPAIR の世代グループを使用している場合、PROC DATASETS ステートメントまたは下位ステートメントの処理を特定のバージョンに制限できます。

注: GENNUM=オプションは、MODIFY ステートメントについては多少結果が異なります。MODIFY ステートメント (508 ページ)を参照してください。

注: ステートメント AGE、COPY、EXCHANGE、SAVE については、処理を特定のバージョンに制限できません。これらのステートメントは、世代グループ全体に適用されます。

世代指定が次の順序で検索されます。

1. 下位ステートメントの SAS データセット名の後のかっこ内。かっこ内で使用される場合、オプションはそのオプションの直前の名前のみ参照します。使用しているデータライブラリ内の複数の SAS データセットのバージョンにそれぞれ異なる世代バージョンを指定する場合、個別の名前の後に GENNUM=をかっこで囲んで指定する必要があります。

   次のステートメントで、GENNUM=オプションは SAS データセット BONES に対してのみ世代グループのバージョンを指定します。

   ```
   delete xplant bones (gennum=2);
   ```

2. 下位ステートメントのスラッシュ(/)の後。スラッシュの後に GENNUM=オプションを使用する場合、同じオプションが SAS データセット名の後のかっこ内に記述されない限り、オプションはステートメントで指定したすべての SAS データセットを参照します。この方法は、使用している複数のファイルすべてのバージョンを同一にする場合に使用します。

   次のステートメントでは、かっこ内の GENNUM=オプションにより SAS データセット CHEST の世代バージョンが指定され、スラッシュの後の GENNUM=オプションにより SAS データセット VIRUS の世代バージョンが指定されます。

   ```
   delete chest (gennum=2) virus / gennum=1;
   ```

3. PROC DATASETS ステートメント内。ライブラリで使用しているすべての SAS データセットのバージョンを同一にする場合、PROC DATASETS ステートメントで世代バージョンを指定すると便利です。この場合、オプションはかっこ内に指定しないでください。

   次の PROC DATASETS ステップでは、GENNUM=オプションは SAS ファイル INSULIN と ABNEG に対して世代バージョンを指定します。

   ```
   proc datasets gennum=2;
   delete insulin;
   contents data=abneg;
   run;
   ```

   注: SELECT ステートメントの SAS ファイルに対する世代バージョンの場合、PROC DATASETS ステートメントの前に COPY ステートメントが検索されます。

拡張属性

拡張属性とは、SAS ファイル用にカスタマイズされたメタデータのことです。ユーザー定義の特性で、SAS データセットまたは変数と関連付けられます。データセットの変数長などの SAS 属性は事前に定義された SAS システム属性ですが、拡張属性は自分で定義する属性です。拡張属性は(名前, 値)ペアから構成されます。

MODIFY ステートメントを使用すると、拡張属性を追加、削除、設定、更新できます。COPY ステートメントを使用すると、OUT=ライブラリエンジンが拡張属性をサポートす

る場合、拡張属性がコピーされます。APPEND ステートメントを使用すると、BASE=データセットが存在しない場合に拡張属性は付加されません。

拡張属性は、カスタム属性を変数またはデータセットに関連付けるために必要なタスクの自動化に使用されます。

拡張属性には数値または文字値を指定できます。各著属性の文字値に最大長はありません。デフォルトでは、各値は 256 バイトのセグメントに保存されます。XATTR OPTIONS ステートメントの SEGLEN=オプションを使用すると、セグメント長を変更できます。このオプションは、文字属性値を保持するストレージ要素長を指定します。特定の文字属性値の中に大きさが足りないセグメントサイズがある場合、別のセグメントが割り当てられます。処理時間を最小限に抑えるため、データセットの大部分の属性値を調整する長さを選択します。

次の出力は、拡張属性を含むデータセットと変数を示します。

アウトプット 16.1 *拡張属性を持つデータセットのコンテンツ*

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
2	age	Num	8
5	cars	Num	8
3	income	Num	8
4	kids	Num	8
1	purchase	Char	3

Alphabetic List of Data Set Extended Attributes

Extended Attribute	Numeric Value	Character Value
attrib	.	table
role	.	train

Alphabetic List of Extended Attributes on Variables

Extended Attribute	Attribute Variable	Numeric Value	Character Value
level	income	.	interval
level	purchase	.	nominal
role	age	.	reject
role	income	.	input
role	purchase	.	target

拡張属性の使い方の詳細は、“XATTR ADD ステートメント” (523 ページ)、“XATTR DELETE ステートメント” (523 ページ)、“XATTR REMOVE ステートメント” (525 ペー

ジ)、“XATTR SET ステートメント” (525 ページ)、、“XATTR UPDATE ステートメント” (526 ページ)を参照してください。

構文: DATASETS プロシジャ

ヒント: RUN グループ処理をサポートします。

Output Delivery System をサポートします。詳細については、“Output Delivery System: Basic Concepts” (*SAS Output Delivery System: User's Guide*)を参照してください。

参照項目: 詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

Windows の DATASETS プロシジャ、UNIX、z/OS

PROC DATASETS <*option(s)*>;
AGE *current-name related-SAS-file(s)*
</ <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;
APPEND BASE=<*libref.*>*SAS-data-set*
<APPENDVER=V6>
<DATA=<*libref.*>*SAS-data-set*>
<ENCRYPTKEY=*key-value*>
<FORCE>
<GETSORT>
<NOWARN>;
AUDIT *SAS-file* (<*SAS-password*> <ENCRYPTKEY=*key-value*> <GENNUM=*integer*)>;
INITIATE <AUDIT_ALL=NO | YES>;
LOG <ADMIN_IMAGE=YES | NO>
<BEFORE_IMAGE=YES | NO>
<DATA_IMAGE=YES | NO>
<ERROR_IMAGE=YES | NO>;
<SUSPEND | RESUME | TERMINATE; >
<USER_VAR *variable(s)* >;
CHANGE *old-name-1=new-name-1*
<*old-name-2=new-name-2* …>
</ <ALTER=*alter-password*> <GENNUM=ALL | *integer*> <MEMTYPE=*member-type*>>;
CONTENTS <*option(s)*>;
COPY OUT=*libref-1*
<CLONE | NOCLONE>
<CONSTRAINT=YES | NO>
<DATECOPY>
<ENCRYPTKEY=*key-value*>
<FORCE>
<IN=*libref-2*>
<INDEX=YES | NO>
<MEMTYPE=(*member-type(s)*)>
<MOVE <ALTER=*alter-password*>>
<OVERRIDE=*(ds-option-1=value-1 <ds-option-2=value-2 … >)*>;
EXCLUDE *SAS-file(s)* < / MEMTYPE=*member-type*>;
SELECT *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*> <ALTER=*alter-password*>
<MEMTYPE= *member-type*>>;

DELETE *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*> <ALTER=*alter-password*>
<GENNUM=ALL | HIST | REVERT *integer*>
<MEMTYPE=*member-type*>>;

EXCHANGE *name-1=other-name-1*
<*name-2=other-name-2 …*>
</ <ALTER=*alter-password*> <MEMTYPE=*member-type*> >;

MODIFY *SAS-file* <(*option(s)*)>
</ <CORRECTENCODING=*encoding-value*> <DTC=*SAS-date-time*>
<GENNUM=*integer*> <MEMTYPE=*member-type*>>;

ATTRIB *variable-list(s) attribute-list(s)*;

FORMAT *variable-1 <format-1>*
<variable-2 <format-2> …>;

IC CREATE *<constraint-name=> constraint*
<MESSAGE='*message-string*' <MSGTYPE=USER>>;

IC DELETE *constraint-name(s)* | _ALL_ ;

IC REACTIVATE *foreign-key-name* REFERENCES *libref*;

INDEX CENTILES *index(s)*
</ <REFRESH> <UPDATECENTILES=ALWAYS | NEVER | *integer*>>;

INDEX CREATE *index-specification(s)*
</ <NOMISS> <UNIQUE> <UPDATECENTILES=ALWAYS | NEVER | *integer*>>;

INDEX DELETE *index(s)* | _ALL_;

INFORMAT *variable-1 <informat-1>*
<variable-2 <informat-2> …>;

LABEL *variable-1=<'label-1'* | ' '>
<variable-2=<'label-2' | ' '> …>;

RENAME *old-name-1=new-name-1*
<old-name-2=new-name-2 …>;

XATTR ADD DS *attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 …>;

or

XATTR ADD VAR *variable-name-1 (attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 …>)
<variable-name-2 (attribute-name-1=attribute-value-1
< attribute-name-2=attribute-value-2 …>)>;

XATTR DELETE;

XATTR OPTIONS <SEGLEN=*number-of-bytes*>;

XATTR REMOVE DS *attribute-name(s)*;

or

XATTR REMOVE VAR *variable-name-1* (*attribute-name(s)*)
<variable-name-2 (*attribute-name(s)*)>;

XATTR SET DS *attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 …>;

or

XATTR SET VAR *variable-name-1 (attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 …>)
<variable-name-2 (attribute-name-1=attribute-value-1
<attribute-name-2=attribute-value-2…>) >;

XATTR UPDATE DS *attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 …>;

または

XATTR UPDATE VAR *variable-name-1 (attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 …>)
<variable-name-2 (attribute-name-1=attribute-value-1
<attribute-name-2=attribute-value-2 …>)>;

REBUILD *SAS-file* < / <ENCRYPTKEY=*key-value*> <ALTER=*password*>
< GENNUM=*integer*> < MEMTYPE=*member-type* > <NOINDEX>>;

REPAIR *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*> <ALTER=*alter-password*>
<GENNUM=*integer*> <MEMTYPE=*member-type*>>;

SAVE *SAS-file(s)* </ MEMTYPE=*member-type*>;

ステートメント	タスク	例
“PROC DATASETS ステートメント”	SAS ファイルを管理する	
“AGE ステートメント”	関連 SAS ファイルグループの名前を変更する	Ex. 7
“APPEND ステートメント”	ある SAS データセットのオブザベーションを別の SAS データセットの末尾に追加する	Ex. 8, Ex. 6, Ex. 9
“ATTRIB ステートメント”	出力形式、入力形式、またはラベルを MODIFY ステートメントで指定した SAS データセットの変数と関連付ける	Ex. 1
“AUDIT ステートメント”	監査ファイルへのイベントの記録を開始、制御、中断、再開、終了する	
“CHANGE ステートメント”	1 つ以上の SAS ファイルの名前を変更する	Ex. 2
“CONTENTS ステートメント”	SAS データセット(複数可)の内容を記述し、SAS ライブラリのディレクトリを出力する	Ex. 9, Ex. 5, Ex. 10
“COPY ステートメント”	SAS ファイルのすべて、または一部をコピーする	Ex. 2
“DELETE ステートメント”	SAS ファイルを削除する	Ex. 2
“EXCHANGE ステートメント”	2 つの SAS ファイル名を交換する	Ex. 2
“EXCLUDE ステートメント”	SAS ファイルをコピーから除外する	Ex. 2
“FORMAT ステートメント”	変数の出力形式を常に割り当て、変更、削除する	Ex. 4
“IC CREATE ステートメント”	一貫性制約を作成する	
“IC DELETE ステートメント”	一貫性制約を削除する	
“IC REACTIVATE ステートメント”	外部キー一貫性制約を再アクティブ化する	

ステートメント	タスク	例
“INDEX CENTILES ステートメント”	インデックス付き変数のパーセント点数統計量を更新する	
“INDEX CREATE ステートメント”	単一インデックスまたは複合インデックスを作成する	Ex. 4
“INDEX DELETE ステートメント”	1 つ以上のインデックスを削除する	
“INFORMAT ステートメント”	変数の入力形式を常に割り当て、変更、削除する	Ex. 4
“INITIATE ステートメント”	SAS データファイルと同じ名前で、データセットの種類が AUDIT の監査ファイルを作成する	Ex. 11
“LABEL ステートメント”	変数ラベルを割り当て、変更、削除する	Ex. 4
“LOG ステートメント”	監査ファイルの設定を指定する	Ex. 11
“MODIFY ステートメント”	SAS ファイルの属性と変数の属性を変更する	Ex. 4
“REBUILD ステートメント”	無効化されたインデックスと一貫性制約を元に戻すか、削除するかどうかを指定する	
“RENAME ステートメント”	SAS データセットの変数の名前を変更する	Ex. 4
“REPAIR ステートメント”	破損した SAS データセットまたはカタログを元に戻す	
“RESUME ステートメント”	監査ファイルが中断された場合に、監査ファイルへのイベント記録を再開する	Ex. 11
“SAVE ステートメント”	SAVE ステートメントにリストされている以外のすべての SAS ファイルを削除する	Ex. 3
“SELECT ステートメント”	コピーする SAS ファイルを選択する	Ex. 2
“SUSPEND ステートメント”	監査ファイルへのイベントの記録を中断する	Ex. 11
“TERMINATE ステートメント”	イベントの記録を終了し、監査ファイルを削除する	Ex. 11
“USER_VAR ステートメント”	オブザベーションへの更新のたびに任意変数が監査ファイルに記録されるように定義する	Ex. 11
“XATTR ADD ステートメント”	拡張属性を変数またはデータセットに追加する	Ex. 12

ステートメント	タスク	例
“XATTR DELETE ステートメント”	変数またはデータセットの拡張属性を削除する	
“XATTR OPTIONS ステートメント”	拡張属性に必要なオプションを指定する	Ex. 12
“XATTR REMOVE ステートメント”	変数またはデータセットから拡張属性を削除する	
“XATTR SET ステートメント”	拡張属性を変数またはデータセットを更新または追加する	
“XATTR UPDATE ステートメント”	変数またはデータセットの拡張属性を更新する	Ex. 12

PROC DATASETS ステートメント

SAS ファイルを管理します。

構文

PROC DATASETS <*option(s)*>;

オプション引数の要約

ALTER=*alter-password*
: SAS ライブラリ内の変更保護された SAS ファイルへの変更アクセス権限を与えます。

DETAILS|NODETAILS
: オブザベーションの数、変数の数、インデックスの数、およびデータセットラベルに関する情報をログに含めます。

ENCRYPTKEY=*key-value*
: AES 暗号化のキー値を指定します。

FORCE
: エラーがある場合でも、RUN グループの実行または追加操作を強制的に行います。

GENNUM=ALL|HIST|REVERT|*integer*
: 世代データセットの処理を制限します。

KILL
: SAS ファイルを削除します。

LIBRARY=*libref*
: プロシジャ入力/出力ライブラリを指定します。

MEMTYPE=(*member-type(s)*)
: 処理を特定の種類の SAS ファイルに制限します。

NODETAILS
DETAILS|NODETAILS の説明を参照してください。

NOLIST
ディレクトリを出力しません。

NOPRINT
ログおよびリストへの出力を抑制します。

NOWARN
エラー処理を行いません。

PW= *password*
読み取り、書き込み、変更のアクセス権限を与えます。

READ=*read-password*
読み取りアクセス権限を与えます。

オプション引数

ALTER=*alter-password*
SAS ライブラリ内の変更保護された SAS ファイルに変更パスワードを与えます。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

DETAILS|NODETAILS
次の列がログに書き込まれるかどうかを決定します。

Obs、Entries、または Indexes
種類 AUDIT、DATA、VIEW の SAS ファイルのオブザベーション数、種類 CATALOG のエントリ数、およびデータファイルと関連付けられている種類 INDEX のファイル数(ある場合)を提供します。SAS データセットのオブザベーション数を決定できない場合、この列の値は欠損値に設定されます。たとえば、非常に大きなデータセットで、オブザベーションまたは削除されたオブザベーションの数が倍精度整数で保存可能な数を超えている場合、カウントは欠損値として表示されます。種類 CATALOG の値は、エントリの合計数です。その他の種類の場合、この列はブランクです。

ヒント 種類 INDEX のファイルの値は、ユーザー定義のインデックスと、一貫性制約によって作成されるインデックスを含みます。インデックス所有権と属性情報を表示するには、PROC DATASETS を CONTENTS ステートメントと OUT2 オプションと使用します。

Vars
種類 AUDIT、DATA、VIEW の変数の数を提供します。SAS データセットの変数の数を決定できない場合、この列の値は欠損値に設定されます。その他の種類の場合、この列はブランクです。

Label
SAS データセットと関連付けられているラベルが含まれます。この列は、種類 DATA に対してのみラベルを出力します。

DETAILS オプションが出力に影響するのは、ディレクトリが指定され、そのディレクトリに SAS ライブラリ内の読み取り保護されているすべての SAS ファイルへの読み取りアクセス権限が必要な場合だけです。読み取りパスワードを入力しない場合、ディレクトリリストには DETAILS オプションによって作成された列に対する欠損値が含まれます。

デフォルト DETAILS も NODETAILS も指定されない場合、デフォルトはシステムオプション設定となります。デフォルトシステムオプション設定は、NODETAILS です。

ヒント SAS ウィンドウ環境を使用し、読み取り保護されている SAS ファイルを含むライブラリに対し DETAILS オプションを指定する場合、ダイアログボックスで PROC DATASETS ステートメントで指定しない各読み取りパスワードの入力を求められます。そのため、同一の読み取りパスワードを同一の SAS ライブラリ内のすべてのファイルに割り当てます。

例 “例 2: SAS ファイルの操作” (548 ページ)

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

参照項目 “AES 暗号化データセットの追加” (463 ページ)

FORCE
2 つの別のアクションを実行します。

- RUN グループの 1 つ以上のステートメントにエラーがある場合でも、強制的に RUN グループを実行します。RUN グループ処理とエラー処理の詳細については、“RUN グループ処理” (439 ページ) を参照してください。
- データセット内の変数が同一でない場合でも、すべての APPEND ステートメントによる 2 つのデータセットの連結を強制的に行います。APPEND ステートメントは、NOWARN オプションが指定(APPEND ステートメントまたは PROC DATASETS によって)されていない限り余剰の変数をドロップし、警告メッセージを SAS ログに発行します。FORCE オプションの詳細については、APPEND ステートメント (458 ページ)を参照してください。

GENNUM=ALL|HIST|REVERT|*integer*
世代データセットの処理を制限します。有効な値は次のとおりです。

ALL
下位の CHANGE ステートメントと DELETE ステートメントについては、世代グループの基本バージョンおよびすべての履歴バージョンを参照します。

HIST
下位の DELETE ステートメントについては、すべての履歴バージョンを参照します。ただし、世代グループの基本バージョンは除きます。

REVERT|0
下位の DELETE ステートメントについては、世代グループの基本バージョンを参照し、(ある場合は)最新の履歴バージョンを基本バージョンに変更します。

integer
下位のステートメント AUDIT、CHANGE、MODIFY、DELETE、REPAIR については、世代グループの特定なバージョンを参照します。正の数の指定は、データセット名に追加される特定の世代番号の絶対参照です(つまり、`gennum=2` は MYDATA#002 を指定します) 。負の数の指定は、最新から最古への基本バージョンに関連する履歴バージョンの関連参照です(つまり、`gennum=-1` は最新の履歴バージョンを参照します)。

参照項目 “世代データセットの処理制限” (444 ページ)

KILL
SAS ライブラリ内の処理可能なすべての SAS ファイルを削除します。
MEMTYPE= オプションは、ステートメントが削除するメンバの種類をサブセットします。次の例では、WORK ライブラリ内のすべてのデータファイルを削除します。

```
proc datasets lib=work kill memtype=data; run; quit;
```

注意:

KILL オプションは、ステートメントのサブミット直後に SAS ファイルを削除します。 SAS ファイルに ALTER=パスワードが割り当てられている場合、そのパスワードを SAS ファイルを削除するために指定する必要があります。

LIBRARY=*libref*
プロシジャが処理するライブラリを指定します。このライブラリは、*プロシジャ入力/出力ライブラリ*です。

別名 DDNAME=, DD=, LIB=

デフォルト Work または User

注 順次エンジン(テープ形式エンジンなど)を介してアクセスされる SAS ライブラリは、LIBRARY=オプションの値として指定できません。

参照項目 WORK ライブラリと USER ライブラリの詳細については、“One-level SAS Data Set Names” (*SAS Language Reference: Concepts*)を参照してください。

例 “例 2: SAS ファイルの操作” (548 ページ)

MEMTYPE=(*member-type(s)*)
処理を 1 つ以上のメンバの種類に制限し、データライブラリディレクトリのリストを指定されているメンバの種類の SAS ファイルに制限します。たとえば、次の PROC DATASETS ステートメントは処理をデフォルトデータライブラリ内の SAS データセットに制限し、SAS ログ内のディレクトリリストをメンバの種類 DATA の SAS ファイルに制限します。

```
proc datasets memtype=data;
```

別名 MTYPE=, MT=

デフォルト ALL

参照項目 “処理するメンバの種類の制限” (442 ページ)

NODETAILS
“DETAILS|NODETAILS” (453 ページ)を参照してください。

NOLIST
SAS ログおよび開いている LISTING 以外の出力先で SAS ファイルのディレクトリの出力を抑制します。

注 ODS LISTING を有効化して LISTING ODS 以外の出力先を開く場合、PROC DATASETS 出力は SAS ログおよび ODS 出力先の両方に送信されます。NOLIST オプションはこのいずれにも出力しません。SAS ログのみで出力を確認するには、メンバおよびディレクトリ出力オブジェクトを指定し、ODS EXCLUDE ステートメントを使用します。たとえば、RTF および LISTING 出力先が両方開いており、ディレクトリとメンバ情報が **LOG** ウィンドウのみで必要な場合、次を使用します。

```
ods rtf file="listing_nolist.rtf";
ods trace on;
ods rtf exclude directory members;

proc datasets lib=work;
run;
quit;
```

```
ods rtf close;
```

例 “例 4: SAS データセットの変更” (555 ページ)

NOPRINT
SAS ログおよび開いている LISTING 以外の出力先で SAS ファイルのディレクトリの出力を抑制します。NOPRINT オプションは、CONTENTS ステートメントにおける NOLIST オプションと NOPRINT オプションの組み合わせです。

NOWARN
ステートメント SAVE、CHANGE、EXCHANGE、REPAIR、DELETE または COPY で指定される SAS ファイル、または AGE ステートメントで最初の SAS ファイルとしてリストされる SAS ファイルがプロシジャ入力ライブラリにない場合に発生するエラー処理を行いません。エラーが発生し、NOWARN オプションが有効な場合、PROC DATASETS はその RUN グループの処理を続行します。NOWARN が無効な場合、PROC DATASETS は RUN グループの処理を停止し、処理を停止しない DELETE 以外のすべての操作に対し警告を発行します。

PW= ***password***
SAS ライブラリ内の保護されている SAS ファイルに対するパスワードを与えます。PW=は、READ=、WRITE=または ALTER=に対する別名として機能します。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

READ=*read-password*
SAS ライブラリ内の読み取り保護されている SAS ファイルに対する読み取りパスワードを与えます。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

AGE ステートメント

ライブラリ内の関連 SAS ファイルグループの名前を変更します。

例: “例 7: SAS データセットのエージング” (563 ページ)

構文

AGE *current-name related-SAS-file(s)*
</ <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

必須引数

current-name
プロシジャが名前を変更する SAS ファイルです。*current-name* は、*related-SAS-file(s)*の最初の名前を受け取ります。

related-SAS-file(s)
SAS ライブラリ内の 1 つ以上の SAS ファイルです。

オプション引数

ALTER=*alter-password*
AGE ステートメントで指定した、変更保護されている SAS ファイルに対する変更パスワードを与えます。AGE ステートメントは SAS ファイルの名前変更および削除を行うため、AGE ステートメントを使用するには変更アクセス権限が必要です。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。AGE ステートメントで指定するすべての SAS ファイルは、同一のメンバの種類である必要があります。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

別名 MTYPE=、MT=

デフォルト PROC DATASETS ステートメントで MEMTYPE=を指定しない場合、デフォルトは DATA となります。

参照項目 “処理するメンバの種類の制限” (442 ページ)

詳細

AGE ステートメントは、*current-name* の名前を *related-SAS-files* の最初の名前に変更し、*related-SAS-files* の最初の名前を *related-SAS-files* の 2 番目の名前に変更し、以下、*related-SAS-files* の最後から 2 番目の SAS ファイルの名前が *related-SAS-files* の最後の名前に変更されるまで続きます。次に、AGE ステートメントは *related-SAS-files* の最後のファイルを削除します。

AGE ステートメントで指定した最初の SAS ファイルが SAS ライブラリ内にない場合、PROC DATASETS は AGE ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。AGE ステートメントは、*related-SAS-files* のいずれもエージングしません。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを使用します。

related-SAS-files のうちいずれも存在しない場合、プロシジャは SAS ログに警告メッセージを発行しますが、可能な SAS ファイルのエージングは続行します。

インデックスを含むデータセットをエージングする場合、インデックスは続行してそのデータセットに対応します。

エージングできるのは、世代グループ全体のみです。たとえば、データセット A と B に世代グループがある場合、次のステートメントが世代グループ B を削除し、世代グループ A を名前 B にエージング(名前変更)します。

```
age a b;
```

たとえば、データセット A の世代グループに 3 つの履歴バージョンがあり、データセット B の世代グループに 2 つの履歴バージョンがあるとします。この場合、A から B へのエージングがこの影響を受けます。

処理前の名前	バージョン	処理後の名前	バージョン
A	base	B	base
A	1	B	1

処理前の名前	バージョン	処理後の名前	バージョン
A	2	B	2
A	3	B	3
B	base	削除される	
B	1	削除される	
B	2	削除される	

APPEND ステートメント

ある SAS データセットの末尾に、別の SAS データセットのオブザベーションを追加します。

デフォルト: BASE=データセットが SAS サーバー経由でアクセスされる場合や、APPEND ステートメントが処理を開始する時にその他のユーザーがデータセットを開いていない場合、BASE=データセットのデフォルトは CNTLLEV=MEMBER (メンバレベルロック)となります。この動作が発生した場合、その他のユーザーはそのデータセットの処理時にファイルを更新できません。

要件 BASE=データセットは、更新処理をサポートする SAS ライブラリのメンバである必要があります。

ヒント: BASE=引数と DATA=オプションに対し、ほとんどのデータセットオプションを指定できます。ただし、データオプション DROP=、KEEP=または RENAME= を BASE=データセットに対して指定する場合、そのオプションは無視されます。グローバルステートメントも使用できます。

処理中にエラーが発生した場合、データセットは損傷としてマーク付けされ、次の REPAIR ステートメントで追加前の状態にリセットされます。データセットにインデックスがある場合、そのインデックスはオブザベーションごとに更新されませんが、最後に一度更新されます。(この動作は、APPENDVER=V6 が設定されていない限りバージョン 7 以降のものです。)

例: “例 6: 2 つの SAS データセットを連結する” (560 ページ)

構文

APPEND BASE=<*libref.*>*SAS-data-set*
<APPENDVER=V6>
<DATA=<*libref.*>*SAS-data-set*>
<ENCRYPTKEY=*key-value*>
<FORCE>
<GETSORT>
<NOWARN>;

必須引数

BASE=<*libref.*>*SAS-data-set*
オブザベーションの追加先となるデータセットを指定します。

libref
SAS データセットを含むライブラリを指定します。ライブラリ参照名を省略すると、デフォルトはプロシジャ入力ライブラリのライブラリ参照名となります。PROCAPPEND を使用している場合、ライブラリ参照名のデフォルトは WORK または USER のいずれかになります。

SAS-data-set
SAS データセットを指定します。APPEND ステートメントは、この名前の既存データセットが見つからない場合、ライブラリに新しいデータセットを作成します。つまり、APPEND ステートメントを使用して、BASE=引数で新しいデータセット名を指定することによりデータセットを作成できます。

新しいデータセットを作成する、既存データセットに追加するにかかわらず、BASE=データセットはすべての追加操作の後の最新の SAS データセットとなります。

別名 OUT=

例 “例 6: 2 つの SAS データセットを連結する” (560 ページ)

オプション引数

APPENDVER=V6
オブザベーションを BASE=データセットに追加するためのバージョン 6 の動作を使用します。一度に 1 つのオブザベーションが追加されます。バージョン 7 からは、パフォーマンスを向上させるため、データセットの処理後にすべてのオブザベーションが追加されるようにデフォルトの動作が変わりました。

参照項目 “インデックス付きデータセットへの追加 — 高速追加メソッド” (463 ページ)

DATA=<*libref.*>*SAS-data-set*
BASE=引数で指定される SAS データセットの最後に追加するオブザベーションを含む SAS データセットを指定します。

libref
SAS データセットを含むライブラリを指定します。ライブラリ参照名を省略すると、デフォルトはプロシジャ入力ライブラリのライブラリ参照名となります。DATA=データセットは、SAS ライブラリのものとなります。データセットがプロシジャ入力ライブラリ以外のライブラリに存在する場合、2 レベル名を使用する必要があります。

SAS-data-set
SAS データセットを指定します。APPEND ステートメントはこの名前の既存データセットが見つからなかった場合、処理を停止します。

別名 NEW=

デフォルト SAS ライブラリからの、最も新しく作成された SAS データセット

参照項目 “世代グループとの追加” (466 ページ)

例 “例 6: 2 つの SAS データセットを連結する” (560 ページ)

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

参照項目 “AES 暗号化データセットの追加” (463 ページ)

FORCE

DATA=データセットに次の基準のうちいずれかを満たす変数が含まれている場合、APPEND ステートメントによるデータセットの連結を強制的に行います。

- BASE=データセットにありません。
- BASE=データセットの変数と同じ種類がありません。
- BASE=データセットの変数より長いです。

ヒント GENNUM=データセットオプションを使用して、世代グループの特定のバージョン間での追加が可能です。次に、一部の例を示します。

```
/* appends historical version to base A */
proc datasets;
append base=a
data=a (gennum=2);

/* appends current version of A to historical version */
proc datasets;
append base=a (gennum=1)
data=a;
```

参照項目 “例 6: 2 つの SAS データセットを連結する” (560 ページ) および“属性が異なる変数を含むデータセットへの追加” (465 ページ)

例 “変数が異なるデータセットへの追加” (464 ページ)

GETSORT

ソートインジケータを DATA=データセットから BASE=データセットにコピーします。ソートインジケータは、次の基準が満たされる場合に PROC SQL の PROC SORT 句または ORDERBY 句によって作成されます。

- BASE=データセットは、次の基準を満たしていなければなりません。
  - SAS バージョン 7 以降
  - オブザベーションが含まれていない
  - ソートインジケータを受け入れる

注意:
DATA=データセットが並べ替えられない場合でも、BASE=データセット上の既存するソートインジケータが警告なしで上書きされます。

- DATA=データセットは、次の基準を満たしていなければなりません。
  - PROC SORT によって作成されたソートインジータを含む
  - BASE=データセットと同じデータ表現

制限事項 BASE=データセットが監査証跡と関連付けられている場合、GETSORT オプションはそのデータセットに影響しません。この制約は、APPEND プロセスが続行される間、出力における WARNING の原因となります。

DATA=データファイルにドロップ、保持、または名前変更された変数がある場合、GETSORT オプションはそのデータセットに影響しません。

例 “例 9: ソートインジケータの情報の取得” (570 ページ)

NOWARN
FORCE オプションと使用した場合、異なる変数を持つ 2 つのデータセットを連結する時に、警告を非表示にします。

詳細

並べ替え済みデータセットの追加

次のガイドラインを使用して、並べ替え済みデータセットを追加し、並べ替えを保持できます。

- DATA=データセットと BASE= データセットには、SORT プロシジャからのソートインジケータが含まれています。
- DATA=データセットと BASE= データセットは、同一の変数を使用して並べ替えられます。
- DATA=データセットから追加されたオブザベーションは、BASE=データセットの並べ替え順序に準拠しています。

BASE=データセットからのソートインジケータは保持されます。

ブロック I/O メソッドを使用して追加

ブロック I/O メソッドは、一度に 1 つのオブザベーションを追加するのではなく、データブロックを追加するために使用されます。大きなデータセットを追加する場合、このメソッドによりパフォーマンスが向上します。SAS によって、ブロック I/O メソッドを使用するかどうかが決定されます。すべてのデータセットで ブロック I/O メソッドが使用できるわけではありません。APPEND ステートメントと Base SAS エンジンによって設定された制約があります。

使用中の追加メソッドに関する情報を SAS ログに表示するには、MSGLEVEL=システムオプションを次のように指定できます。

```
options msglevel=i;
```

ブロック I/O メソッドが使用されていない場合、SAS ログに次のメッセージが出力されます。

```
INFO: Data set block I/O cannot be used because:
```

APPEND ステートメントがブロック I/O を使用しないと判断した場合、次の説明のうちの 1 つが SAS ログに出力されます。

INFO: - データセットは異なるエンジンを使用し、異なる変数または異なる可能性がある属性を持ちます。

INFO: - WHERE 句があります。

INFO: - メンバレベルロックがありません。

```
INFO: - The OBS option is active.
```

```
INFO: - The FIRSTOBS option is active.
```

Base SAS engine がブロック I/O メソッドを使用しないと判断した場合、次の説明のうちの 1 つが SAS ログに出力されます。

INFO: - 参照一貫性制約が存在します。

INFO: - クロス環境データアクセスが使用されています。

INFO: - ファイルが圧縮されます。

INFO: - ファイルに中断されない監査ファイルが含まれています。

追加されるオブザベーションの制限

追加されるオブザベーションを制限するため、WHERE= データセットオプションと DATA=データセットを使用できます。同様に、DATA=データセットからのオブザベーションを制限するため、WHERE ステートメントを使用できます。WHERE ステートメントは、BASE=データセットに影響しません。WHERE=データセットオプションと BASE=データセットを使用する場合、WHERE=は影響しません。

注意:

既存する BASE=データセットの場合: BASE=データセットに WHERE ステートメントがある場合、WHEREUP= オプションが YES に設定されている場合にのみ有効です。

注意:

存在しない BASE=データセットの場合: 存在しない BASE=データセットに WHERE ステートメントがある場合、WHEREUP オプション設定に関係なく、WHERE ステートメントを使用します。

注: WHERE=データセットオプションを使用して、データセットをそれ自体に追加することはできません。

SET ステートメントと APPEND ステートメント間の選択

DATA ステップで SET ステートメントを使用し、2 つのデータセットを連結する場合、両方のデータセットのすべてのオブザベーションを処理して新しく作成する必要があります。APPEND ステートメントは元のデータセットのデータ処理を行わずに、新しいオブザベーションを元のデータセットの最後に直接追加します。次のうちいずれかが発生した場合は、SET ステートメントよりも APPEND ステートメントを使用したほうが効率的です。

- BASE=データセットが大きい。
- BASE=データセットのすべての変数の長さと種類が DATA=データセット の変数と同じで、すべての変数が両方のデータセットに存在する場合。

  注: CONTENTS ステートメントを使用して、変数の長さと種類を確認できます。

オブザベーションを(データをジャーナル型のデータセットに継続的に追加している本稼働プログラムなどで)SAS データセットに頻繁に追加する場合、APPEND ステートメントは非常に便利です。

パスワード保護されている SAS データセットの追加

APPEND ステートメントを使用するため、DATA=データセットへの読み取りアクセス権限と BASE=データセットへの書き込みアクセス権限が必要です。アクセス権を得るには、APPEND ステートメント内でデータセット名の直後にかっこで囲み READ=および WRITE=データセットを使用します。パスワード保護されているデータセットを追加する場合、次のガイドラインを使用します。

- APPEND ステートメントで DATA=データセットに対する読み取りパスワードを与えない場合、デフォルトでプロシジャが PROC DATASETS ステートメントの DATA=データセットに対する読み取りパスワードを検索します。ただし、プロシジャは PROC DATASETS ステートメントの BASE=データセットに対する書き込みパスワードは検索しません。そのため、APPEND ステートメントの BASE=データセットに対する書き込みパスワードを指定する必要があります。
- BASE=データセットが読み取り保護だけされている場合、APPEND ステートメントでその読み取りパスワードを指定する必要があります。

AES 暗号化データセットの追加

2 つの AES 暗号化データセットを使用する場合、データセットにアクセスするために ENCRYPTKEY=データセットオプションを使用する必要があります。次は、両方のデータセットで ENCRYPTKEY=オプションを使用する例です。

```
proc datasets;
append base=a (encryptkey=secret)
data=a (encryptkey=jlgh56);
run;
```

暗号化されていないデータセットに追加する場合、DATA=データセットで ENCRYPTKEY=を指定する必要があります。なお、BASE=データセットエンジンで AES 暗号化がサポートされていない場合でも、データセットに追加できます。追加されたデータは暗号化されません。

```
proc datasets;
append base=a
data=a (encryptkey=key-value);
run;
```

AES 暗号化に関する詳細は、“AES Encryption” (*SAS Language Reference: Concepts*) を参照してください。ENCRYPTKEY=データセットオプションに関する詳細は、“ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)を参照してください。

圧縮データセットへの追加

圧縮されている SAS データセットを連結できます。BASE=データセット、DATA=データセットのいずれか、またはその両方を圧縮できます。BASE=データセットで削除されたオブザベーションからのスペースの再利用が可能な場合、APPEND ステートメントはオブザベーションを BASE=データセットの真ん中に挿入し、利用可能なスペースを利用します。

COMPRESS=データセットと REUSE=データセットおよびシステムオプションの詳細については、*SAS データセットオプション: リファレンス*と *SAS システムオプション: リファレンス*を参照してください。

インデックス付きデータセットへの追加 — 高速追加メソッド

バージョン 7 から、パフォーマンスを向上させるためにインデックス付きデータセットへの追加の動作が変わりました。

- バージョン 6 では、インデックス付きデータセットに追加する際に、インデックスは追加された各オブザベーションに対して更新されました。インデックス更新は無作為になる傾向にあります。そのため、ディスク I/O が高くなった可能性があります。
- 現在では、インデックスはすべてのオブザベーションがデータセットに追加されるまで更新されません。追加後、オブザベーションが内部的に並べ替えられ、データがインデックスに逐次挿入されます。この動作によりほとんどのディスク I/O が減少し、追加メソッドがさらに早くなります。

次の要件を満たしている場合、高速追加メソッドはデフォルトで使用されます。その他の場合、バージョン 6 メソッドが使用されます。

- BASE=データセットは、メンバレベルロックが可能です。CNTLLEV=が記録のために設定されている場合、高速追加メソッドは使用されていません。
- BASE=データセットには、参照一貫性制約が含まれていません。

- BASE=データセットにはクロス環境データアクセス(CEDA)機能を使用してアクセスしません。
- BASE=データセットは、WHERE=データセットオプションを使用していません。

使用中の追加メソッドに関する情報を SAS ログに表示するには、MSGLEVEL=システムオプションを次のように指定できます。

```
options msglevel=i;
```

高速追加メソッドが使用されている場合にメッセージが表示されるか、高速追加メソッドが使用されていない理由に関してメッセージが表示されます。

現在の追加メソッドは、インデックスによって決定される制約に関係なく、オブザベーションを BASE=データセットに内部的に追加します。たとえば、UNIQUEF オプションによって作成されたインデックスを含む変数の場合、その値はインデックスが更新されるまでその一意性について検証されません。一意でない値が検出されると、問題のあるオブザベーションはデータセットから削除されます。オブザベーションが追加されると、そのうちのいくつかが後で削除される場合があります。

簡単な例として、BASE=データセットに変数 ID に対し UNIQUE インデックスを含む、1 から 10 までの番号の付いた 10 つのオブザベーションが含まれているとします。1 から 5 までの 5 つのオブザベーションを含むデータセットを追加し、オブザベーション 3 と 4 に ID に対する同一の値が含まれているとします。次のアクションが発生します。

1. オブザベーションが追加されると、BASE=データセットには 1 から 15 までの番号の付いた 15 つのオブザベーションが含まれます。
2. ID に対するインデックスが更新され、値が検証され、オブザベーション 13 と 14 に ID に対する同一の値が含まれるよう指定されます。
3. BASE=データセットからオブザベーションのいずれかが削除され、その結果、1 から 15 までの番号の付いた 14 つのオブザベーションとなります。たとえば、オブザベーション 13 が削除されます。どのオブザベーションが削除されるかは予測できません。内部並べ替えではどちらかのオブザベーションが最初に識別される場合があるためです。(バージョン 6 では、オブザベーション 13 が追加され、オブザベーション 14 が削除されるということが予測可能でした。)

現在の動作(オブザベーションが削除される)を希望しない場合、または追加されるオブザベーションの予測を希望する場合は、APPENDVER=V6 オプションを指定して、バージョン 6 追加メソッドを要求してください。

```
proc datasets;
append base=a data=b appendver=v6;
run;
```

注: バージョン 6 では、インデックスを削除してから追加後に再度作成することにより、パフォーマンスの向上が可能でした。現在のメソッドではその必要がありません。ただし、パフォーマンスはデータの性質によって異なります。

変数が異なるデータセットへの追加

DATA=データセットに BASE=データセットにない変数が含まれている場合、APPEND ステートメントで FORCE オプションを使用し、2 つのデータセットの連結を強制的に行います。APPEND ステートメントは余剰の変数をドロップし、警告メッセージを発行します。NOWARN オプションを使用して、警告メッセージを非表示にできます。

BASE=データセットに DATA=データセットにない変数が含まれている場合、APPEND ステートメントはデータセットを連結しますが、DATA=データセットからのオブザベーションには、DATA=データセットになかった変数に対する欠損値が含まれます。この場合、FORCE オプションは不要です。

BASE=データセットでオプション DROP=、KEEP=、または RENAME=を使用する場合、オプション ONLY が APPEND 処理に影響し、追加された BASE=データセットの変数を変更しません。DROP=オプションと KEEP=オプションを使用してドロップされる、または保持されない変数は、追加された BASE= データセットに存在しています。RENAME=オプションを使用して名前が変更される変数は、追加された BASE=データセットにその元の名前で存在しています。

属性が異なる変数を含むデータセットへの追加

BASE=データセットの変数の属性が DATA= データセットの変数と異なる場合、BASE=データセットの属性が優先されます。

DATA=データセットの SAS 出力形式が BASE=データセットのものと異なる場合、BASE=データセットの SAS 出力形式が使用されます。ただし、BASE=データセットの SAS 出力形式と一貫性を取るために DATA=データセットのデータは変換されません。結果は、正しくないように見えるデータになります。警告が SAS ログに表示されます。次の例に、異なる SAS 出力形式を使用したデータの追加について示します。

```
data format1;
input Date date9.;
format Date date9.;
datalines;
24sep1975
22may1952
;

data format2;
input Date datetime20.;
format Date datetime20.;
datalines;
25aug1952:11:23:07.4
;

proc append base=format1 data=format2;
run;
```

次のメッセージが SAS ログに表示されます。

```
NOTE: Appending WORK.FORMAT2 to WORK.FORMAT1.
WARNING: Variable Date has format DATE9. on the BASE data set
and format DATETIME20. on the DATA data set. DATE9. used.
NOTE: There were 1 observations read from the data set WORK.FORMAT2.
NOTE: 1 observations added.
NOTE: The data set WORK.FORMAT1 has 3 observations and 1 variables.
```

DATA=データセットの変数の長さが BASE=データセットのものより長い場合、または同じ変数が 1 つのデータセットでは文字変数で、もう 1 つのデーセットでは数値変数の場合、FORCE オプションを使用します。FORCE を使用した結果は、次のとおりです。

- BASE=データセットの変数の長さが優先します。DATA=データセットの値は切り捨てられ、BASE=データセットで指定されている長さに合うように調整されます。
- BASE=データセットの変数の種類が優先します。APPEND ステートメントは、正しくない種類の値(DATA=データセットの変数に対するすべての値) を欠損値と置き換えます。

注: 文字変数のトランスコーディング属性が BASE=データセットと DATA=データセットで逆の場合 (一方が YES、もう一方が NO など)、警告が発行されます。トランスコーディング属性を決定するには、各データセットに対し CONTENTS プロシジャを

使用します。トランスコーディング属性は、ATTRIB ステートメントの TRANSCODE= オプション、または PROC SQL の TRANSCODE=列修飾子によって設定します。

一貫性制約を含むデータセットの追加

DATA=データセットに一貫性制約が含まれ、BASE=データセットが存在しない場合、APPEND ステートメントは一般制約をコピーします。参照制約はコピーされません。BASE=データセットが存在する場合、APPEND アクションはオブザベーションのみコピーします。

一貫性制約またはインデックスを含む 0 オブザベーションデータセットへの追加

PROC APPEND または APPEND ステートメントの使用は、特にデータセットに一貫性制約またはインデックスが含まれる場合、データセットを追加する際の最も効率的な方法とは限りません。オブザベーションを追加する前にデータセットとそのすべての属性を定義してください。一貫性制約またはインデックスを追加してからデータを追加する方法よりも、次の方法の方が迅速です。

1. 0 オブザベーションかつ一貫性制約またはインデックスを含まないデータセットを作成します。
2. データを追加します。
3. インデックスと一貫性制約を作成します。

次は、既存データセットからインデックスと一貫性制約を取得し、別のデータセットに適用する際の簡単な方法です。Work.Model がすでに存在し、オブザベーションがないことにご注意ください。

```
Proc contents data=original out2=icsidx;
run;

proc sql noprint;
select recreate into :recreate from icsidx;
quit;

proc datasets lib=to_lib nolist;
modify model;
recreatm;
quit;

proc contents data=to_lib.model;
run;
```

世代グループとの追加

GENNUM= データセットオプションを使用して、世代グループの特定のバージョンに追加します。次に例を示します。

SAS ステートメント	結果

`proc datasets;`
`append base=a`
`data=b(gennum=2);`
	履歴バージョン B#002 を base A に追加

SAS ステートメント	結果

`proc datasets;`
`append base=a(gennum=2)`
`data=b(gennum=2);`
	履歴バージョン B#002 を履歴バージョン A#002 に追加

APPEND ステートメントの代わりに APPEND プロシジャを使用

APPEND プロシジャと PROC DATASETS の APPEND ステートメントの違いは、BASE=引数と DATA=引数のライブラリ参照名のデフォルト値のみです。PROC APPEND のデフォルトは WORK か USER です。APPEND ステートメントのデフォルトは、プロシジャの入力ライブラリのライブラリ参照名です。

システム障害

プロシジャの実行時にシステム障害またはその他の種類の障害が発生した場合、追加操作が正常に行われないことがあります。オブザベーションの一部、またはすべてが BASE=に追加されない可能性があります。また、BASE=データセットが破損することがあります。APPEND 操作はかわりに更新を実行します。つまり、オブザベーションの追加を開始する前に元のデータセットのコピーを作成しません。元のオブザベーションを復元する場合、基本データファイルの監査証跡を開始し、更新前のオブザベーションの保存を選択できます。次に、DATA ステップを書き込み、元のオブザベーションをデータファイルに抽出して再度適用できます。監査証跡の開始の詳細については、PROC DATASETS (468 ページ)を参照してください。

ATTRIB ステートメント

出力形式、入力形式またはラベルを MODIFY ステートメントで指定した SAS データセットの変数と関連付けます。

制限事項: ATTRIB ステートメントは、MODIFY RUN グループに表示されるはずです

注: ATTRIB ステートメントは、CONTENTS ステートメント出力に影響を与えません。CONTENTS は、実際のメンバに関するラベル、入力形式、および出力形式を報告します。

例: AUDIT ステートメント (543 ページ)

構文

ATTRIB *variable-list(s) attribute-list(s)*;

必須引数

variable-list(s)
属性と関連付ける変数を指定します。SAS で許可される形式の変数をリストできます。

attribute-list(s)
variable-list に割り当てる属性を 1 つ以上指定します。ATTRIB ステートメントで次の属性のうち 1 つ以上を指定します。

FORMAT=*format*
出力形式を *variable-list の変数と関連付けます。*

ヒント 出力形式は標準 SAS 出力形式、または FORMAT プロシジャで定義される出力形式のいずれかです。

INFORMAT=*informat*
入力形式を *variable-list* の変数と関連付けます。

ヒント 入力形式は標準 SAS 入力形式、または FORMAT プロシジャで定義される入力形式のいずれかです。

LABEL='*label*'
ラベルを *variable-list* の変数と関連付けます。

詳細

DATASETS プロシジャ内で、ATTRIB ステートメントは MODIFY RUN グループで使用する必要があり、オプション FORMAT、INFORMAT、LABEL のみ使用できます。ATTRIB ステートメントは、キーワード_ALL_を使用した、データセット内のすべての変数ラベル、出力形式または入力形式を削除、変更するための最も簡単な方法です。例については、“例 1: データセットのすべてのラベルと出力形式の削除” (543 ページ)を参照してください。

一部の属性を削除または変更している場合、LABEL ステートメント (507 ページ)については、FORMAT ステートメント (498 ページ)、および INFORMAT ステートメント (505 ページ)を使用するとさらに簡単です。

AUDIT ステートメント

監査ファイルへのイベントの記録を開始、制御し、監査ファイルのイベントの記録を中断、再開、終了します。

ヒント: AUDIT ステートメントは、監査証跡を開始するかどうか、監査ファイルのイベントの記録を中断、再開、終了するかどうかによって、2 種類あります。

PROC DATASETS MODIFY ステートメントを使用して、出力形式、入力形式などの属性をデータファイルのユーザー変数に対し定義できます。

参照項目: “Understanding an Audit Trail” (*SAS Language Reference: Concepts*)

構文

AUDIT *SAS-file* <(*SAS-password* <ENCRYPTKEY=*key-value*> <GENNUM=*integer*>)>;

INITIATE <AUDIT_ALL=NO | YES>;

LOG<ADMIN_IMAGE=YES | NO>
<BEFORE_IMAGE=YES | NO>
<DATA_IMAGE=YES | NO>
<ERROR_IMAGE=YES | NO>;

<SUSPEND | RESUME | TERMINATE; >

<USER_VAR *variable(s)* >;

必須引数

SAS-file
監査するプロシジャ入力ライブラリの SAS データファイルを指定します。

オプション引数

SAS-password
SAS データファイルのパスワードを指定します(存在する場合)。かっこは必須です。

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

GENNUM=*integer*
アクション SUSPEND、RESUME または TERMINATE が世代ファイルの監査証跡で実行されるように指定します。世代ファイルで監査証跡を開始できません。GENNUM=に有効な値は *integers* で、世代ファイルの特定バージョンを参照する番号です。正の数の指定は、データセット名に追加される特定の世代番号の絶対参照です(つまり、**gennum=2** は MYDATA#002 を指定します)。負の数の指定は、最新から最古への基本バージョンに関連する履歴バージョンの関連参照です(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。デフォルトの 0 を指定すると、基本バージョンが参照されます。かっこは必須です。

制限事項 GENNUM=オプションは、INTIATE ステートメントまたは USER_VAR ステートメントの前に指定できません。

詳細

監査ファイルの作成

次の例では、監査ファイル MYLIB.MYFILE.AUDIT を作成し、更新をデータファイル MYLIB.MYFILE.DATA に書き込み、すべての利用可能なレコードイメージを保存します。

```
proc datasets library=mylib;
audit myfile (alter=password);
initiate;
run;
```

次の例では、同じ監査ファイルを作成しますが、エラーレコードイメージのみを保存します。

```
proc datasets library=mylib;
audit myfile (alter=password);
initiate;
log data_image=no
before_image=no
data_image=no;
run;
```

次の例では、AUDIT_ALL=YES を使用して監査ファイルを開始します。

```
proc datasets lib=mylib; /* all audit image types will be logged
and the file cannot be suspended */
audit myfile (alter=password);
initiate audit_all=yes;
quit;
```

次の例では、監査ファイルを終了します。

```
proc datasets lib=mylib;
audit myfile (alter=password);
terminate;
```

```
quit;
```

AUDIT ステートメントは、ファイルの*監査実行グループ*を開始します。ファイルの複数の監査実行グループは、次の方法でサブミット可能です。

- 同一の PROC DATASETS ステップで
- 別の PROC DATASETS ステップで
- 別の SAS セッションで

すべての監査ファイル関連ステートメント(INITIATE、USER_VAR、LOG、SUSPEND、RESUME、TERMINATE)の前に、それらが適用されるファイルを識別する AUDIT ステートメントを配置する必要があります。

INITIATE ステートメントは監査ファイルを作成し、最初の AUDIT ステートメントでサブミットされる必要があります。USER_VAR、LOG、SUSPEND、RESUME、TERMINIATE などのその他の監査関連ステートメントは、INITIATE ステートメントがサブミットされるまで監査ファイルに有効になりません。世代データセットで監査ファイルを開始できますが、これは世代グループの最新世代である必要があります。GENNUM を指定して最新世代を識別できません。最新世代はデフォルトで取得されます。

次は、指定ファイルに対する最初の AUDIT 実行グループの AUDIT ステートメントの例です。

```
AUDIT file <(SAS-password)>;
```

監査ファイルが一度開始されると、そのファイルに対して INITIATE ステートメントの後に続く AUDIT ステートメントは GENNUM を指定できます。

```
 AUDIT file <(<SAS-password><GENNUM=integer>)>;
```

USER_VAR ステートメントは、同じ AUDIT 実行グループの INITIATE ステートメントの直後に指定する必要があります。

CHANGE ステートメント

同一の SAS ライブラリの 1 つ以上の SAS ファイルの名前を変更します。

例: “例 2: SAS ファイルの操作” (548 ページ)

構文

CHANGE *old-name-1=new-name-1*
<*old-name-2=new-name-2* …>
</ <ENCRYPTKEY=*key-value*>
<ALTER=*alter-password*> <GENNUM=ALL | *integer*> <MEMTYPE=*member-type*>>;

必須引数

old-name=new-name
入力データライブラリの SAS ファイル名を変更します。*old-name* は、入力データライブラリの既存する SAS ファイルの名前である必要があります。

例 “例 2: SAS ファイルの操作” (548 ページ)

オプション引数

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。このオプションが必要なのは、RELATIVE GENNUM が指定されている場合に限られます。詳細については、“ライブラリコンテンツと AES 暗号化” (477 ページ)を参照してください。

ALTER=*alter-password*
CHANGE ステートメントで指定した、変更保護されている SAS ファイルに対する変更パスワードを与えます。CHANGE ステートメントは SAS ファイル名を変更するため、*new-name* に CHANGE ステートメントを使用するために変更アクセス権限が必要です。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

GENNUM=ALL|*integer*
世代データセットの処理を制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。次のリストに、有効値を示します。

ALL | 0
世代グループの基本バージョンとすべての履歴バージョンを参照します。

integer
世代グループの特定バージョンを参照します。正の数の指定は、データセット名に追加される特定の世代番号の絶対参照です(つまり、**gennum=2** は MYDATA#002 を指定します)。負の数の指定は、最新から最古への基本バージョンに関連する履歴バージョンの関連参照です(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。

たとえば、次のステートメントはバージョン名 A#003 を base B に変更します。

```
proc datasets;
change A=B / gennum=3;

proc datasets;
change A(gennum=3)=B;
```

次の CHANGE ステートメントではエラーが発生します。

```
proc datasets;
change A(gennum=3)=B(gennum=3);
```

参照項目 “世代データセットの処理制限” (444 ページ) および“Understanding Generation Data Sets” (*SAS Language Reference: Concepts*)

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

別名 MTYPE=、MT=

デフォルト PROC DATASETS ステートメントで MEMTYPE=を指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目 “処理するメンバの種類の制限” (442 ページ)

詳細

CHANGE ステートメントは、CHANGE ステートメントで変更をリストする順序ではなく、*old-names* がディレクトリリストで発生する順序によって名前を変更します。

old-name の SAS ファイルが SAS ライブラリに存在しない場合、PROC DATASETS は CHANGE ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを使用します。

インデックスを含むデータセットの名前を変更する場合、インデックスは続行してそのデータセットに対応します。

CONTENTS ステートメント

1 つ以上の SAS データセットの内容を記述し、SAS ライブラリのディレクトリを出力します。

制限事項: PROC CONTENTS はオブザベーションを処理しないため、出力に作用する WHERE オプションは使用できません。

注: ATTRIB ステートメントは、CONTENTS ステートメント出力に影響を与えません。CONTENTS は、実際のメンバに関するラベル、入力形式、および出力形式を報告します。

ヒント: DATA=、OUT=および OUT2=オプションではデータセットオプションが使用できます。グローバルステートメントも使用できます。

例: “例 5: SAS データセットの説明” (557 ページ)

構文

CONTENTS <*option(s)*>;

オプション引数の要約

CENTILES
: インデックス付き変数に関するパーセント点情報を出力します。

DATA=*SAS-file-specification*
: 入力データセットを指定します。

DETAILS|NODETAILS
: オブザベーションの数、変数の数、インデックスの数、およびデータセットラベルに関する情報を出力に含めます。

DIRECTORY
: SAS ライブラリの SAS ファイルのリストを出力します。

ENCRYPTKEY=*key-value*
: AES 暗号化のキー値を指定します。

FMTLEN
: 変数の入力形式または出力形式の長さを出力します。

MEMTYPE=(*member-type(s)*)
: 処理を 1 つ以上の種類の SAS ファイルに制限します。

NODETAILS
: DETAILS|NODETAILS の説明を参照してください。

NODS
: 個々のファイルを出力しません。

NOPRINT
出力しません。

ORDER= COLLATE | CASECOLLATE | IGNORECASE | VARNUM
変数のリストを指定した順序で出力します。

OUT=*SAS-data-set*
出力データセット名を指定します。

OUT2=*SAS-data-set*
インデックスと一貫性制約に関する情報を含む出力データセットの名前を指定します。

SHORT
省略出力を行います。

VARNUM
データセット内の位置による変数のリストを出力します。デフォルトでは、CONTENTS ステートメントは変数をアルファベット順でリストします。

オプション引数

CENTILES
インデックス付き変数に関するパーセント点情報を出力します。

CENTILES オプションが選択され、インデックスがデータセット上に存在する場合、次の追加フィールドが PROC CONTENTS のデフォルトレポートに出力されます。追加フィールドは、インデックスが単一または複合かどうかににによって異なります。

#
データセット上のインデックスの数。

Index
インデックス名。

Update Centiles
インデックス付き変数に対し CENTILES の前に変更が必要なデータ値のパーセントが自動的に更新されます。

Current Update Percentage
CENTILES が更新されてから更新されたインデックスのパーセント。

# of Unique Values
一意のインデックス付き値の数です。

Variables
インデックスの作成に使用される変数の名前。パーセント点情報は、変数の下にリストされます。

DATA=*SAS-file-specification*
ライブラリ全体またはライブラリ内の特定の SAS データセットを指定します。*SAS-file-specification* は、次の形式のうちいずれかが可能です。

<libref.>SAS-data-set
処理する SAS データセットを 1 つ指定します。ライブラリ参照名のデフォルトは、プロシジャ入力ライブラリのライブラリ参照名です。たとえば、プロシジャ入力ライブラリから SAS データセット HTWT の内容を取得するには、次の CONTENTS ステートメントを使用します。

```
contents data=HtWt;
```

世代グループから特定のバージョンの内容を取得するには、次の CONTENTS ステートメントにあるように GENNUM=データセットオプションを使用します。

```
contents data=HtWt(gennum=3);
```

*<libref.>*_ALL_
MEMTYPE=オプションによって指定した種類(複数可)を含むすべての SAS データセットに関する情報を提供します。*libref* は、SAS ライブラリを参照します。ライブラリ参照名のデフォルトは、プロシジャ入力ライブラリのライブラリ参照名です。

- _ALL_キーワードを使用している場合、SAS ライブラリの読み取り保護されているすべての SAS データセットへの読み取りアクセス権限が必要です。
- DATA=_ALL_は、SAS ライブラリに含まれる SAS ファイルのリストを自動的に出力します。SAS ビューの場合、そのビューと関連付けられているすべてのライブラリ参照名は、リストに対して処理されるように現在のセッションで割り当てられる必要があります。

デフォルト SAS ライブラリからの、ジョブまたはセッションで最も新しく作成されたデータセット。

ヒント DATA= オプションで読み取り保護されているデータセットを指定し、読み取りパスワードを与えない場合、デフォルトでプロシジャが PROC DATASETS ステートメントで読み取りパスワードを検索します。ただし、DATA=オプションを指定せず、デフォルトのデータセット(セッションで最後に作成されたもの)が読み取り保護されている場合、プロシジャは PROC DATASETS ステートメントで読み取りパスワードを検索しません。

例 “例 5: SAS データセットの説明” (557 ページ)

DETAILS|NODETAILS
DETAILS には出力に情報の追加列が含まれますが、DIRECTORY も指定されている場合だけです。

デフォルト DETAILS も NODETAILS も指定されていない場合、デフォルトは次のようになります。CONTENTS プロシジャの場合、デフォルトはシステムオプション設定の NODETAILS です。CONTENTS ステートメントの場合、デフォルトは PROC DATASETS ステートメントで指定されている値で、これもシステムオプション設定となります。

参照項目 PROC DATASETS ステートメント (452 ページ)の Optional Argument セクションの追加列の説明

DIRECTORY
指定した SAS ライブラリのすべての SAS ファイルのリストを出力します。DETAILS も指定されている場合、DIRECTORY を使用すると DETAILS|NODETAILS (474 ページ)に記述されている追加列が出力されます。

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。詳細については、“ライブラリコンテンツと AES 暗号化” (477 ページ)を参照してください。

FMTLEN
入力形式または出力形式の長さを出力します。入力形式または出力形式の長さを変数と関連付けるときに指定しない場合、長さは FMTLEN オプションを使用しない限り CONTENTS ステートメントの出力に表示されません。長さは出力データセットの FORMATL 変数または INFORML 変数にも表示されます。

MEMTYPE=(*member-type(s)*)
処理を 1 つ以上のメンバの種類に制限します。CONTENTS ステートメントは、メンバの種類 DATA、VIEW および ALL(DATA と VIEW が含まれます)に対してのみ出力を生成します。

CONTENTS ステートメントの MEMTYPE=は、DATASETS プロシジャのその他の大部分のステートメントの MEMTYPE=と次の点が異なります。

- スラッシュはオプションの前に指定できません。
- MEMTYPE= オプションをかっこで囲み、その影響をその直前の SAS ファイルにのみ制限することはできません。

MEMTYPE=は結果として、DATA=メンバが置かれているライブラリのディレクトリとなります。ただし、MEMTYPE=は、_ALL_キーワードが DATA=オプションで使用されない限り、その内容が表示されるメンバの種類を制限しません。たとえば、次のステートメントは、メンバの種類 DATA の SAS データセットのみの内容を生成します。

```
proc datasets memtype=data;
contents data=_all_;
run;
```

別名 MT=, MTYPE=

デフォルト DATA

NODS
DATA= オプションで_ALL_を指定する際に、個々のファイルの内容を出力しません。CONTENTS ステートメントは、SAS ライブラリディレクトリのみを出力します。DATA=オプションで SAS データセットを 1 つだけ指定する際に、NODS オプションは使用できません。

NODETAILS
“DETAILS|NODETAILS” (474 ページ)を参照してください。

NOPRINT
CONTENTS ステートメントの出力を行いません。

ORDER= COLLATE | CASECOLLATE | IGNORECASE | VARNUM

COLLATE
変数のリストを大文字名、次に小文字名で始まるアルファベット順で出力します。

CASECOLLATE
変数の一覧を、大文字と小文字が混在する名前と数値が含まれている場合でも、アルファベット順で出力します。

IGNORECASE
変数のリストを大文字と小文字に関係なくアルファベット順で出力します。

VARNUM
VARNUM オプションと同じです。

参照項目 “VARNUM” (476 ページ)

注 ORDER=オプションは、OUT=と OUT2=データセットの順序には影響しません。

例 ORDER=のデフォルトと 4 つのオプションを比較するには、“例 10: ORDER= オプションを CONTENTS ステートメントとともに使用する” (573 ページ)を参照してください。

OUT=*SAS-data-set*
出力 SAS データセットを指定します。

ヒント OUT=は、ステートメントからの出力を非表示にしません。出力を非表示にする場合、NOPRINT オプションを使用する必要があります。

参照項目 OUT=データセットの変数の説明については、“OUT=データセット” (534 ページ)を参照してください。CONTENTS 出力を ODS データセットに処理のため取得する方法については、“例 8: ODS 出力 ” (564 ページ)を参照してください。

OUT2=*SAS-data-set*
インデックスと一貫性制約に関する情報を含む出力データセットを指定します。

ヒント UPDATECENTILES がインデックス定義で指定されなかった場合、デフォルト値 5 が OUT2 データセットの RECREATE 変数で使用されます。

OUT2=は、ステートメントからの出力を非表示にしません。出力を非表示にするには、NOPRINT オプションを使用する必要があります。

参照項目 OUT2=データセットの変数の説明については、“OUT2=データセット” (542 ページ)を参照してください。

一貫性制約およびインデックスをデータセットから削除するには、“OUT2=データファイルの置換” (476 ページ)を参照してください。

SHORT
SAS データセットの変数名のリスト、インデックス情報、並べ替え情報のみ出力します。

制限事項 変数のリストが 32,767 文字を超える場合、リストは切り捨てられ、WARNING が SAS ログに書き込まれます。変数の完全リストを取得するには、変数のアルファベット順リストを要求します。

VARNUM
変数名のリストをデータセットの論理位置の順序で出力します。デフォルトでは、CONTENTS ステートメントは変数をアルファベット順でリストします。データセットの変数の物理位置は、エンジン依存です。

詳細

OUT2= データファイルの置換

データセットで一貫性制約とインデックスに関する情報を持つデータセットが必要な場合、OUT2=データセットオプションを使用します。データセットからインデックスと一貫性制約を削除し、PROC CONTENTS をもう一度実行して OUT2=データセットを作成する場合、次のメッセージが表示されます。

```
NOTE: データセット USER.MYINFO んいは、1 つのオブザベーションと 19 の変数が含まれます。
```

既存の OUT2=ファイルを置換するには、新しいファイルにオブザベーションが含まれる必要があります。PROC CONTENTS の実行時に一貫性制約またはインデックスが存在しない場合、オブザベーションが作成され、OUT2 ファイルに書き込まれます。このプロセスによって、前の PROC CONTENTS 出力が置換されます。

変数の出力

CONTENTS ステートメントは、デフォルトで変数のアルファベット順リストを出力します。番号範囲リスト形式の変数は除きます。x1–x100 などの番号範囲リストは、増分

順で x1–x100 と出力されます。詳細については、“変数と属性のアルファベット順リスト” (530 ページ)を参照してください。

注: ビューが変数ラベルを含むデータセットから作成された後にラベルが変更される場合、CONTENTS プロシジャ出力または DATASETS プロシジャ出力に元のラベルが表示されます。CONTENTS プロシジャ出力または DATASETS プロシジャ出力が新しい変数ラベルを反映するために、ビューを再コンパイルする必要があります。

ICU 改訂番号の表示

CONTENTS ステートメントは、データセットで言語ソートの SORT プロシジャを使用する場合、Unicode 用国際化コンポーネント (ICU)改訂番号を出力します。言語ソートに関する詳細は、59 章, “SORT プロシジャ,” (1785 ページ)を参照してください。

ライブラリコンテンツと AES 暗号化

ライブラリ内の全データのコンテンツを要求する場合は、_ALL_オプションを使用します。ライブラリに AES 暗号化されたデータファイルが含まれる場合、データファイルへのアクセスには ENCRYPTKEY=データセットオプションを使用する必要があります。次は、ENCRYPTKEY=オプションを使用する例です。

```
proc contents data=MyLib._all_ (encryptkey=key-value);
run;
```

キー値がライブラリ内の特定のデータファイルのキー値に一致しない場合、正しいキー値を有力するよう求められます。

AES 暗号化に関する詳細は、“AES Encryption” (*SAS Language Reference: Concepts*) を参照してください。ENCRYPTKEY=データセットオプションに関する詳細は、“ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)を参照してください。

CONTENTS ステートメントの代わりに CONTENTS プロシジャを使用

CONTENTS プロシジャと PROC DATASETS の CONTENTS ステートメントの唯一の違いは、DATA=オプションのライブラリ参照名のデフォルトです。PROC CONTENTS の場合、デフォルトは WORK です。CONTENTS ステートメントの場合、デフォルトはプロシジャ入力ライブラリのライブラリ参照名です。

SAS データセットのオブザベーションの長さ、配置、埋め込み

配置には 3 つの異なるケースがあります。

- SAS データセット内のオブザベーションは、可能な場合は 2 バイト境界上で配置されます。結果として、8 バイト数値変数と 4 バイト数値変数はデータセットの前の 8 バイト境界に配置され、その後に文字変数が発生順に続きます。データセットに 4 バイト数値データのみ含まれている場合、配置は 4 バイト境界に基づいて行われます。数値倍精度は比較または増分の前に移動し配置するよりも直接操作可能なため、境界によってパフォーマンスがさらに向上します。

  指定されているディスクデータページバッファ内に多くのオブザベーションが含まれているため、各オブザベーションがダブルバイト境界で配置されるためにオブザベーション間に埋め込みがある場合があります。次の例を参照してください。

```
data a;
length aa 7 bb 6 cc $10 dd 8 ee 3;
aa = 1;
bb = 2;
cc = 'abc';
```

```
dd = 3;
ee = 4;
ff = 5;
output;
run;

proc contents data=a out=a1;
run;

proc print data=a1(keep=name length varnum npos);
run;
```

図 16.1 オブザベーション長

The CONTENTS Procedure

Data Set Name	WORK.A	Observations	1
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	04/11/2014 11:05:35	Observation Length	48
Last Modified	04/11/2014 11:05:35	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

PROC CONTENTS はオブザベーション長 48 を表示します。PROC PRINT は、NPOS が各変数に対しゼロ基準オフセットであるオブザベーション内の変数の内部レイアウトを表示します。

図 16.2 オブザベーションおよび変数の境界

Obs	NAME	LENGTH	VARNUM	NPOS
1	aa	7	1	16
2	bb	6	2	23
3	cc	10	3	32
4	dd	8	4	0
5	ee	3	5	29
6	ff	8	6	8

変数 DD と FF は、実際の数値倍精度であり、それぞれオフセット 0 と 8 のため、自動的に配置されます。その他のオブザベーションには、残りの数値変数と文字変数が含まれます。

このレイアウトの最後の物理変数は、オフセット 32、長さ 10 の CC です。これにより、PROC CONTENTS がオブザベーション長を 48 とレポートする場合でも内部長 42 が指定されます。差異は埋め込みの 6 バイトのため、次のオブザベーションはディスクページバッファ内のダブルバイト境界で配置されます。

- 次の例にあるように、オブザベーションに 8 バイトの数値変数が含まれていない場合は配置は行われません。この場合、オブザベーション長 7 が指定され、ディスクページバッファ内のオブザベーション間に埋め込みはありません。

```
data b;
length aa 6 cc $1;
aa = 1;
cc = 'x';
output;
run;

proc contents data=b out=b1;
run;

proc print data=b1(keep=name length varnum npos);
run;
```

図 16.3 変数と属性

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	aa	Num	6
2	cc	Char	1

- 圧縮データセットのオブザベーションはディスクページバッファ内で配置されませんが、同じアルゴリズムがオブザベーション内の変数の配置に使用されます。圧縮オ

ブザベーションは解凍し、作業バッファに移動する必要があります。8 バイトの数値は配置され、解凍後すぐに使用可能です。PROC CONTENTS 出力のオブザベーション長は、オペレーティングシステム固有のオーバーヘッドのため、長くなることがあります。

COPY ステートメント

SAS ライブラリのすべての、または一部の SAS ファイルをコピーします。

制限事項: COPY ステートメントでは、データセットオプションはサポートしていません。

ヒント: 32 ビットマシンから 64 ビットマシンに移行するには、PROC COPY (415 ページ) の例を参照してください。

COPY ステートメントは、SAS/SHARE、SAS/CONNECT などのリモートライブラリサービス(RLS)の使用時は出力ライブラリのエンコーディングおよびデータ表現をデフォルトとして使用します。RLS を使用していない場合、PROC COPY オプション NOCLONE を出力ファイルに使用して、出力ライブラリのエンコーディングとデータ表現を求める必要があります。NOCLONE オプションを使用すると、データライブラリのデータ表現 (OUTREP= LIBNAME オプションで指定されている場合)、または動作環境のネイティブなデータ表現を含むコピーとなります。

例: “例 2: SAS ファイルの操作” (548 ページ)

構文

COPY OUT=*libref-1*
<CLONE | NOCLONE>
<CONSTRAINT=YES | NO>
<DATECOPY>
<ENCRYPTKEY=*key-value*>
<FORCE>
IN=*libref-2*
<INDEX=YES | NO>
<MEMTYPE=(*member-type(s)*)>
<MOVE <ALTER=*alter-password*> >
<OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2 …>*) >;

必須引数

OUT=*libref-1*
SAS ファイルのコピー先となる SAS ライブラリを指定します。

別名 OUTLIB=と OUTDD=

例 “例 2: SAS ファイルの操作” (548 ページ)

オプション引数

CLONE | NOCLONE
次のデータセット属性をコピーするかどうかを指定します。

- 入力/出力バッファのサイズ
- データセットの圧縮
- 空きスペースの再利用

- 入力データセットのデータ表現、ライブラリ、動作環境
- 値のエンコーディング
- 圧縮されているデータセットにオブザベーション番号で無作為にアクセス可能かどうか

これらの属性は、データセットオプション、SAS システムオプション、LIBNAME ステートメントオプションで指定されます。

- BUFSIZE= (入力/出力バッファのサイズの値)
- COMPRESS= (データセットを圧縮するかどうかの値)
- REUSE= (空きスペースを再利用するかどうかの値)
- OUTREP= (データ表現の値)
- ENCODING=または INENCODING= (値のエンコーディング)
- POINTOBS= (圧縮されたデータセットにオブザベーション番号で無作為にアクセスできるかどうかの値)

次の表に、BUFSIZE=属性に関する COPY ステートメントの動作を要約します。

表 16.2 *CLONE とバッファページサイズ属性*

オプション	COPY ステートメント
CLONE	入力データセットからの BUFSIZE=値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで BUFSIZE=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	SAS システムオプション BUFSIZE=の現在の設定を出力データセットに使用します。
いずれも使用しない	入力データセット用エンジンと出力データセット用エンジンによって使用されるアクセス方法の種類(順次または無作為)を指定します。両方のエンジンで同じ種類のアクセスが使用されている場合、COPY ステートメントは入力データセットの BUFSIZE=値を出力データセットに使用します。同じ種類のアクセスが使用されていない場合、COPY ステートメントは SAS システムオプション BUFSIZE=の設定を出力データセットに使用します。

次の表に、COMPRESS=属性に関する COPY ステートメントの動作を要約します。

表 16.3 *CLONE と圧縮属性*

オプション	COPY ステートメント
CLONE	入力データセットの値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで COMPRESS=値を指定すると、コピーで指定エンコードが使用されることになります。
NOCLONE	動作環境の圧縮を含むコピー、または指定されている場合は、ライブラリに対する LIBNAME ステートメントの COMPRESS=オプションの値となります。
いずれも使用しない	デフォルトは CLONE となります。

次の表に、REUSE=属性に関する COPY ステートメントの動作を要約します。

表 16.4 *CLONE と空きスペースの再利用属性*

オプション	COPY ステートメント
CLONE	入力データセットの値を出力データセットに使用します。入力データセット用エンジンで空きスペースの再利用属性がサポートされていない場合、COPY ステートメントは対応する SAS システムオプションの現在の設定を使用します。ただし、OVERRIDE=オプションリストで REUSE=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	SAS システムオプション COMPRESS=と REUSE=の現在の設定を出力データセットに使用します。
いずれも使用しない	デフォルトは CLONE となります。

次の表に、OUTREP=属性に関する COPY ステートメントの動作を要約します。

表 16.5 *CLONE とデータ表現属性*

オプション	COPY ステートメント
CLONE	入力データセットのデータ表現を含むコピーとなります。ただし、OVERRIDE=オプションリストで OUTREP=値を指定すると、コピーで指定データ表現が使用されることになります。
NOCLONE	動作環境のデータ表現を含むコピー、または指定されている場合は OUT=ライブラリの LIBNAME ステートメントの OUTREP=オプションの値となります。
いずれも使用しない	デフォルトは CLONE となります。

データ表現とは、特定の動作環境でデータを保存する形式です。さまざまな動作環境では、次のような目的のさまざまな規格または規則が使用されます。

- 浮動小数点数を保存 (IEEE、IBM 390 など)
- 文字エンコーディング(ASCII または EBCDIC)
- メモリのバイトオーダリング (ビッグエンディアンまたはリトルエンディアン)
- ワード配置(4 バイト境界または 8 バイト境界)
- データ型の長さ(16 ビット、32 ビット、または 64 ビット)

ネイティブなデータ表現とは、ファイルのデータ表現が CPU 動作環境と同じ場合です。たとえば、Windows データ表現のファイルは、Windows 動作環境に対してネイティブとなります。

次の表に、ENCODING=属性に関する COPY ステートメントの動作を要約します。

表 16.6 CLONE とエンコーディング属性

オプション	COPY ステートメント
CLONE	入力データセットのエンコーディングを使用するコピー、または指定されている場合は、入力ライブラリの LIBNAME ステートメントの INENCODING=オプションの値となります。ただし、OVERRIDE=オプションリストで ENCODING=値を指定すると、コピーで指定エンコードが使用されることになります。
NOCLONE	現在のセッションエンコーディングのエンコーディングを使用するコピー、または指定されている場合は、出力ライブラリの LIBNAME ステートメントの OUTENCODING=オプションの値となります。
いずれも使用しない	デフォルトは CLONE となります。

コンピュータによって保存、送信、処理されるデータはすべてエンコーディングされます。エンコーディングは各文字を一意の数値表現にマップします。エンコーディングは、文字セットとエンコーディング方法の組み合わせです。文字セットは言語または言語グループによって使用される文字と記号のレパートリーです。エンコーディング方法は、数字をエンコーディングで使用される文字セットに割り当てるために使用される一連のルールです。

次の表に、POINTOBS=属性に関する COPY ステートメントの動作を要約します。POINTOBS=を使用するには、出力データセットを圧縮する必要があります。

表 16.7 CLONE と POINTOBS=属性

オプション	COPY ステートメント
CLONE	入力データセットの POINTOBS=値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで POINTOBS=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	LIBNAME ステートメントは、出力データセットが圧縮され、POINTOBS=オプションが指定されて出力エンジンでサポートされている場合に使用します。LIBNAME ステートメントが指定されず、データセットが圧縮されている場合、出力エンジンでサポートされているときデフォルトは POINTOBS=YES となります。
いずれも使用しない	デフォルトは CLONE となります。

CONSTRAINT=YES | NO
データセットのコピー時にすべての一貫性制約をコピーするかどうかを指定します。

デフォルト NO

ヒント 外部キーを持つ一貫性制約を含むデータセットの場合、COPY ステートメントは CONSTRAINT=YES が指定され、ライブラリ全体がコピーされる場合に一般制約と参照制約をコピーします。SELECT ステートメントまたは EXCLUDE ステートメントを使用してデータセットをコピーする場合、参照一

貫性制約はコピーされません。詳細については、“Understanding Integrity Constraints” (*SAS Language Reference: Concepts*)を参照してください。

DATECOPY
SAS ファイルが作成された SAS 内部日時と、ファイルの結果コピーに最後に変更された日時をコピーします。動作環境の日時は保持されません。

制限事項 DATECOPY は、暗号化されたファイルまたはカタログと使用できません。

DATECOPY は、結果の SAS ファイルで V8 エンジンまたは V9 エンジンが使用される場合にのみ使用できます。

ヒント MODIFY ステートメントの DTC=オプションを使用して、ファイル作成日時を変更できます。MODIFY ステートメント (508 ページ)を参照してください。

コピー中のファイルに追加処理が必要な属性が含まれている場合、最終変更日は現在の日付に変更されます。たとえば、インデックスを含むデータセットをコピーし、インデックスを再度作成する必要がある場合、最終変更日は現在の日付に変更されます。追加処理が必要で、最終変更日に影響する可能性のあるその他の属性には、一貫性制約とソートインジケータが含まれます。

ENCRYPTKEY= ***key-value***
IN=ライブラリ内で AES 暗号化を含むデータセットのコピーに必要なキー値を指定します。

注 出力ライブラリで AES 暗号化がサポートされておらず、入力データセットが AES 暗号化されている場合、COPY プロセスによってエラーが生成されます。

参照項目 “参照用一貫性制約を含む AES 暗号化データファイルのコピー” (491 ページ)

FORCE
MOVE オプションを監査証跡が存在する SAS データセットに使用できます。

注 AUDIT ファイルは、監査データセットと移動されません。

IN=*libref-2*
コピーする SAS ファイルを含む SAS ライブラリを指定します。

別名 INLIB=と INDD=

デフォルト プロシジャ入力ライブラリのライブラリ参照名

操作 選択したメンバのみコピーするには、SELECT ステートメントまたは EXCLUDE ステートメントを使用します。

INDEX=YES|NO
データセットを SAS ライブラリ間でコピーする場合にデータセットのすべてのインデックスをコピーするかどうかを指定します。

デフォルト YES

MEMTYPE=(*member-type(s)*)
処理を 1 つ以上のメンバの種類に制限します。

別名 MT=、MTYPE=

デフォルト PROC DATASETS ステートメントの MEMTYPE=を省略すると、デフォルトは MEMTYPE=ALL となります。

注 PROC COPY がテープ上の SAS ライブラリを処理し、MEMTYPE=オプションが指定されていない場合、ファイルの最後までエントリの順次ライブラリ全体がスキャンされます。順次ライブラリがマルチボリュームテープである場合、すべてのテープボリュームはマウントされます。この動作は、単一のボリュームテープライブラリの場合も同様です。

参照項目 “SAS ファイルのコピー/移動時のメンバの種類の指定” (487 ページ) および“メンバの種類” (443 ページ)

例 “例 2: SAS ファイルの操作” (548 ページ)

MOVE
SAS ファイルを入力データライブラリ (IN=オプションによって指定)から出力データライブラリ (OUT=オプションによって指定)に移動します。そして、元のファイルを入力データライブラリから削除します。

ALTER=*alter-password*
ライブラリ間で移動中の変更保護されている SAS ファイルに対する変更パスワードを与えます。MOVE オプションは元のデータライブラリから SAS ファイルを削除するため、SAS ファイルの移動には変更アクセス権限が必要となります。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

制限事項 MOVE オプションは、IN=エンジンでテーブルの削除がサポートされている場合にのみ、SAS ライブラリのメンバの削除に使用できます。テープ形式エンジンでは、テーブルの削除はサポートされていません。テープ形式エンジンを使用する場合、MOVE 操作は実行されず、警告が発行されます。

例 “例 2: SAS ファイルの操作” (548 ページ)

OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2> ...*)
入力データセットからコピーされた指定出力データセットオプションを上書きします。COPY の出力データセットコンテキストでは、一部のデータセットオプションは適切でない場合があります。

制限事項 NOCLONE オプションが指定される場合、OVERRIDE オプションは無視されます。ただし、NOCLONE オプションによって調整される属性以外のデータセット属性の変更には使用できます。

ヒント 別のホストデータ表現またはエンコードに保存されたデータセットをコピーする場合、COPY のデフォルト(CLONE)動作は、データセットの新しいコピーに他のホストデータ表現またはエンコードを保存するためのものです。OVERRIDE=(OUTREP=SESSION ENCODING=SESSION)を COPY ステートメントで指定すると、データセットの新しいコピーは、COPY を実行する SAS セッションのホストデータ表現およびエンコードでデータセットの新しいコピーが作成されます。

NOCLONE
"CLONE | NOCLONE" (480 ページ)の説明を参照してください。

詳細

ブロック I/O メソッドを使用してコピー

ブロック I/O メソッドは、一度に 1 つのオブザベーションをコピーするのではなく、データブロックをコピーするために使用されます。大きなデータセットをコピーする場合、このメソッドによりパフォーマンスが向上します。このメソッドを使用するかどうかが決定されます。すべてのデータセットで ブロック I/O メソッドが使用できるわけではありません。COPY ステートメントと Base SAS エンジンによって設定された制約があります。

使用されているコピーメソッドに関する情報を SAS ログに表示するには、MSGLEVEL=システムオプションを次のように指定できます。

```
options msglevel=i;
```

ブロック I/O メソッドが使用されていない場合、SAS ログに次のメッセージが出力されます。

```
INFO: Data set block I/O cannot be used because:
```

COPY ステートメントがブロック I/O を使用しないと判断した場合、次の説明のうちの 1 つが SAS ログに出力されます。

INFO: - データセットは異なるエンジンを使用し、異なる変数または異なる可能性がある属性を持ちます。

```
INFO: - There is no member level locking.
```

```
INFO: - The OBS option is active.
```

```
INFO: - The FIRSTOBS option is active.
```

Base SAS engine がブロック I/O メソッドを使用しないと判断した場合、次の説明のうちの 1 つが SAS ログに出力されます。

INFO: - 参照一貫性制約が存在します。

INFO: - クロス環境データアクセスが使用されています。

INFO: - ファイルが圧縮されます。

INFO: - ファイルに中断されない監査ファイルが含まれています。

パフォーマンスに問題があり、テスト用に大きなデータセットのサブセットを作成する場合、OBS=0 オプションを使用できます。この場合、ブロック I/O メソッドを無効化して、システムリソースの使用を減らします。

次の例では、OBS=0 オプションを使用して、システムリソースの使用を減らします。

```
options obs=0 msglevel=i;
proc copy in=old out=lib;
select a;
run;
```

SET ステートメントを使用した場合も、同じ結果になります。

```
data lib.new;
if 0 then set old.a;
stop;
run;
```

ライブラリ全体のコピー

SAS ライブラリ全体をコピーするには、COPY ステートメントに続いて入力ライブラリと出力ライブラリを指定します。たとえば、次のステートメントは SOURCE データライブラリのすべての SAS ファイルを DEST データライブラリにコピーします。

```
proc datasets library=source;
copy out=dest;
run;
```

選択した SAS ファイルのコピー

選択した SAS ファイルをコピするには、SELECT または EXCLUDE ステートメントを使用します。SELECT または EXCLUDE ステートメントと COPY ステートメントの併用についての詳細は、“SAS ファイルのコピー/移動時のメンバの種類の指定” (487 ページ) を参照してください。例は SAS ファイルの操作 (548 ページ) にあります。また、EXCLUDE ステートメント (497 ページ) と SELECT ステートメント (520 ページ)も参照してください。

省略したメンバリストを選択または除外することもできます。たとえば、次のステートメントはメンバ Test1、Test2、および Test3 を選択します。

```
select tabs test1-test3;
```

名前が同じ文字で始まるメンバのグループは、共通の文字に続きコロン(:)を入力すると、選択できます。たとえば、次のステートメントを指定すれば、直前の例の4メンバとそれ以外の T で始まる名前のすべてのメンバを選択できます。

```
select t:;
```

選択と同じように除外するメンバも指定できます。つまり、個々のメンバ名をリストすることもできますし、省略したリストも使用でき、共通の文字とコロン(:)の指定もできるということです。たとえば、次のステートメントは、メンバ Stats、Teams1、Teams2、Teams3、Teams4、および RBI という文字で始まるすべてのメンバを、コピー操作から除外します。

```
exclude stats teams1-teams4 rbi:;
```

MEMTYPE=オプションは、どの種類のメンバが選択または除外可能になるかどうかに影響を与えます。

SELECT ステートメントまたは EXCLUDE ステートメントが CONSTRAINT=YES と使用される場合、データセットに関する一般一貫性制約だけがコピーされます。参照一貫性制約はコピーされません。詳細については、*SAS 言語リファレンス: 解説編*の “Understanding Integrity Constraints”を参照してください。

SAS ファイルのコピー/移動時のメンバの種類の指定

COPY ステートメントの MEMTYPE= オプションは、プロシジャのその他のステートメントの MEMTYPE=オプションといくつかの点が異なります。

- スラッシュはオプションの前に指定できません。
- MEMTYPE= オプションをかっこで囲んで、その影響をその直前のメンバに制限することはできません。
- SELECT ステートメント、EXCLUDE ステートメントおよび IN=オプション(COPY ステートメントの)は、次のルールに従って、COPY ステートメントの MEMTYPE=オプションの動作に影響します。

  1. SELECT ステートメントまたは EXCLUDE ステートメントの MEMTYPE=は、COPY ステートメントの MEMTYPE=オプションに優先します。次のステートメントは、VISION.CATALOG と NUTR.DATA だけをデフォルトのデータライブラリ

から DEST データライブラリにコピーします。最初の SELECT ステートメントの MEMTYPE=値は COPY ステートメントの MEMTYPE=値に優先します。

```
proc datasets;
copy out=dest memtype=data;
select vision(memtype=catalog) nutr;
run;
```

2. IN=オプションを使用しない、またはそれをプロシジャ入力ライブラリとなるライブラリを指定するために使用する場合、PROC DATASETS ステートメントの MEMTYPE=オプションの値は、処理可能な SAS ファイルの種類を制限します。プロシジャは、ルール 1 で説明されている優先順位を使用し、コピー可能な種類をさらにサブセット化します。次のステートメントはメンバをデフォルトのデータライブラリから DEST データライブラリにコピーしません。かわりに、SELECT ステートメントで指定した MEMTYPE=値が PROC DATASETS ステートメントの MEMTYPE=オプションの値でないため、プロシジャはエラーメッセージを発行します。

```
 /* This step fails! */
proc datasets memtype=(data program);
copy out=dest;
select apples / memtype=catalog;
run;
```

3. IN=オプションでプロシジャ入力ライブラリ以外の入力データライブラリを指定する場合、PROC DATASETS ステートメントの MEMTYPE=オプションはコピー操作に影響しません。サブセット化が発生していないため、プロシジャはルール 1 で説明されている優先順序を使用して、コピー可能な種類をサブセット化します。次のステートメントは、BODYFAT.DATA を DEST データライブラリに正常にコピーします。COPY ステートメントの IN=オプションで指定した SOURCE ライブラリが PROC DATASETS ステートメントの MEMTYPE=オプションによって影響を受けないためです。

```
proc datasets library=work memtype=catalog;
copy in=source out=dest;
select bodyfat / memtype=data;
run;
```

COPY ステートメントを使用する際、ライブラリのメモリ内ディレクトリが取得されます。ライブラリ内にメンバが何千もあるにもかかわらず、わずかなメンバしかコピーされていない場合、それはパフォーマンスの問題の可能性があります。こうしたパフォーマンスの問題を解決するには、COPY ステートメントの MEMTYPE=オプションと SELECT ステートメントの組み合わせを使用します。次にこのプロセスの例を示します。

```
proc datasets lib=work;
copy out=mylib memtype=(data catalog);
select mydata x1-x10 data2;
run;
```

注: MEMTYPE=ALL またはワイルドカード指定(":")が使用される場合、パフォーマンスコードは使用できません。

ビューのコピー

NOCLONE が指定されている COPY ステートメントでは、SQL ビュー、DATA ステップビューおよび一部の SAS/ACCESS ビュー(Oracle 、Sybase)に対し、OUTREP=オプションと ENCODING= LIBNAME オプションをサポートしています。COPY ステートメントを SAS/SHARE、SAS/CONNECT などのリモートライブラリサービス(RLS)と使用する

際は、COPY ステートメントは出力ライブラリのエンコーディングおよびデータ表現をデフォルトとして使用します。

注意:

DATA ステップビューの作成時に DATA ステートメントの SOURCE=NOSAVE オプションを使用する場合、ビューを SAS のバージョン間でコピーすることはできません。

パスワード保護されている SAS ファイルのコピー

パスワード保護されている SAS ファイルをパスワードを指定せずにコピーできます。また、パスワードは続行して SAS ファイルに対応しているため、コピー後に SAS ファイルにアクセスし操作するためにパスワードを知っている必要があります。

長い変数名を持つデータセットのコピー

VALIDVARNAME=V6 システムオプションが設定されていて、かつデータセットが長い変数名を持つ場合、その長い変数名は切り捨てられ、ユニークな変数名が生成され、コピーが続行します。インデックス名についても同様です。
VALIDVARNAME=ANY で OUT=エンジンが長い変数名をサポートしていない場合、コピーは失敗します。

変数名が切り捨てられる場合、変数名は 8 バイトに短くされます。もしもこの名前がデータセットですでに定義されているものであった場合、名前は切り捨てられ、それに数字の 2 から始まる数字が追加されます。切り捨てと数字の追加は、名前が重複しないものになるまで続きます。たとえば、LONGVARNAME という変数名は、データセットで重複する名前が存在しないとすると、LONGVARN となります。すでにその名前が存在する場合は、LONGVAR2 となります。

注意:

切り捨てられた変数名は入力データセット内で定義されている名前と競合することがあります。この現象は、すでに定義されている変数名が 8 バイトの長さで最後が数字で終わっている場合に起こります。次の例では、出力データセットで切り捨てられた名前が定義されており、入力データセットからの名前が変更されています。

```
options validvarname=any;
data test;
longvar10='aLongVariableName';
retain longvar1-longvar5 0;
run;

options validvarname=v6;
proc copy in=work out=sasuser;
select test;
run;
```

この例では、LONGVAR10 は LONGVAR1 に切り捨てられて、出力データセットにおかれます。次に、元の LONGVAR1 がコピーされます。その名前はもう一意ではありません。そのため、LONGVAR2 という名前に変更されます。入力データセットの他の変数も、名称変更のアルゴリズムに従って名前が変更されます。次の例は、SAS ログからのものです。

```
1 options validvarname=any;
2 data test;
3 longvar10='aLongVariableName';
4 retain longvar1-longvar5 0;
5 run;

NOTE: データセット WORK.TEST は 1 オブザベーション、6 変数です。
NOTE: DATA ステートメント処理(合計処理時間):
```

```
処理時間 2.60 秒
CPU 時間 0.07 秒


6
7 options validvarname=v6;
8 proc copy in=work out=sasuser;
9 select test;
10 run;

NOTE: WORK.TEST を SASUSER.TEST (memtype=DATA) へコピーします。
NOTE: 変数名 longvar10 を longvar1 に切り捨てました。
NOTE: 変数 longvar1 のラベルを longvar10 に設定しました。
NOTE: 変数 LONGVAR1 はファイル SASUSER.TEST に既に存在します。代わりに LONGVAR2
を使用します。
NOTE: 変数 LONGVAR2 のラベルを LONGVAR1 に設定しました。
NOTE: 変数 LONGVAR2 はファイル SASUSER.TEST に既に存在します。代わりに LONGVAR3
を使用します。
NOTE: 変数 LONGVAR3 のラベルを LONGVAR2 に設定しました。
NOTE: 変数 LONGVAR3 はファイル SASUSER.TEST に既に存在します。代わりに LONGVAR4
を使用します。
NOTE: 変数 LONGVAR4 のラベルを LONGVAR3 に設定しました。
NOTE: 変数 LONGVAR4 はファイル SASUSER.TEST に既に存在します。代わりに LONGVAR5
を使用します。
NOTE: 変数 LONGVAR5 のラベルを LONGVAR4 に設定しました。
NOTE: 変数 LONGVAR5 はファイル SASUSER.TEST に既に存在します。代わりに LONGVAR6
を使用します。
NOTE: 変数 LONGVAR6 のラベルを LONGVAR5 に設定しました。
NOTE: データセット WORK.TEST から 1 オブザベーションを読み込みました。
NOTE: データセット SASUSER.TEST は 1 オブザベーションと 6 変数です。
NOTE: PROCEDURE COPY 処理(合計処理時間):
処理時間 13.18 秒
CPU 時間 0.31 秒


11
12 proc print data=test;
13 run;

ERROR: 値 LONGVAR10 は無効な SAS 名です。
NOTE: エラーが発生したため、このステップの処理を中止しました。
NOTE: PROCEDURE PRINT 処理(合計処理時間):
処理時間 0.15 秒
CPU 時間 0.01 秒
```

AES 暗号化データファイルのコピー

AES 暗号化でデータファイルをコピーするには、次の 2 つの方法のいずれかで ENCRYPTKEY=*key-value* を指定する必要があります。

- ライブラリ内のすべての AES 暗号化データセットに同じ ENCRYPTKEY=*key-value* が含まれる場合、次の例を使用してください。

```
proc copy in=lib1 out=lib2 encryptkey=key-value;
run;
```

- ENCRYPTKEY=*key-value* が各データセットで異なる場合、次の例に示すように SELECT ステートメントを使用してください。

```
proc copy in=lib1 out=lib2;
select mydata1 (encryptkey=key-value1) mydata2 (encryptkey=key-value2);
run;
```

AES 暗号化に関する詳細は、“AES Encryption” (*SAS Language Reference: Concepts*) を参照してください。ENCRYPTKEY=データセットオプションに関する詳細は、“ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)を参照してください。

参照用一貫性制約を含む AES 暗号化データファイルのコピー

参照用一貫性制約を含む AES 暗号化でデータファイルをコピーするには、次を実行する必要があります。

- IN=ライブラリのコンテンツ全体をコピーする
- CONSTRAINT=YES ステートメントオプションを指定する
- ENCRYPTKEY=*key-value* オプションを指定する

```
proc copy in=Lib1 out=Lib2 constraint=yes encryptkey=key-value;
run;
```

ただし、IN=LIB 内に複数のデータプライマリキーと参照用データセットのペアがあり、各ペアに異なる ENCYPTKEY=値が含まれる場合、上記のコードは機能しなくなります。たとえば、2 つのペア(

```
primarydset1/foreigndset1 having ENCRYPTKEY=secret1
primarydset2/foreigndset2 having ENCRYPTKEY=secret2
```

)がある場合、1 つのペアに対してのみ上記のスキームは機能します。すべてのペアには同じキー値が含まれる必要があります。

詳細については、“ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)を参照してください。

世代グループのコピー

COPY ステートメントを使用して、世代グループ全体をコピーできます。ただし、世代グループの特定のバージョンをコピーすることはできません。

ホスト間での SAS データセットの移送

XPORT エンジンまたは REMOTE エンジンと一緒に COPY プロシジャを使うと、ホスト間で SAS データセットを移動することができます。詳細は、*SAS ファイルの移動とアクセス*にある“SAS ファイルの移動とアクセスの方法”を参照してください。

COPY ステートメントの代わりに COPY プロシジャを使用

大まかに言えば、COPY プロシジャは DETASETS プロシジャの COPY ステートメントと同様に機能します。機能の差異を次にリストします。

- PROC COPY には IN=引数が必須です。COPY ステートメントでは IN=はオプションです。省略された場合、デフォルト値はプロシジャの入力ライブラリのライブラリ参照名です。
- PROC DATASETS は、順次データアクセスのみを許可するライブラリには使用できません。

- COPY ステートメントは NOWARN オプションを順守しますが、PROC COPY は違います。

DELETE ステートメント

SAS ファイルを SAS ライブラリから削除します。

例: “例 2: SAS ファイルの操作” (548 ページ)

構文

DELETE *SAS-file(s)*
</ <ALTER=*alter-password*>
<ENCRYPTKEY=*key-value*>
<GENNUM=ALL | HIST | REVERT | *integer*>
<MEMTYPE=*member-type*>>;

必須引数

SAS-file(s)
削除する SAS ファイルを 1 つ以上指定します。SAS ファイルに ALTER=パスワードが割り当てられている場合、そのパスワードを SAS ファイルを削除するために指定する必要があります。番号範囲リストまたはコロンリストを使用することもできます。詳細については、*SAS 言語リファレンス: 解説編*の“Data Set Lists”を参照してください。

オプション引数

ALTER=*alter-password*
削除する変更保護された SAS ファイルに対する変更パスワードを与えます。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。このオプションが必要になるのは、ABSOLUTE GENNUM または GENNUM=ALL が指定されていない場合です。詳細については、“ライブラリコンテンツと AES 暗号化” (477 ページ)を参照してください。

GENNUM=ALL|HIST|REVERT|*integer*
世代データセットの処理を制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。有効な値のリストは次のとおりです。

ALL
世代グループの基本バージョンとすべての履歴バージョンを参照します。

HIST
世代グループの基本バージョンを除く、すべての履歴バージョンを参照します。

REVERT|0
基本バージョンを削除し、存在する場合は最新履歴バージョンを基本バージョンに変更します。

integer
世代グループの特定バージョンを参照する数値です。正の数の指定は、データセット名に追加される特定の世代番号の絶対参照です(つまり、**gennum=2** は MYDATA#002 を指定します)。負の数の指定は、最新から最古への基本バージョンに関連する履歴バージョンの関連参照です(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。

参照項目 "Understanding Generation Data Sets" (*SAS Language Reference: Concepts*) および "世代データセットの処理制限" (444 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

別名 MT=, MTYPE=

デフォルト DATA

参照項目 "処理するメンバの種類の制限" (442 ページ)

例 "例 2: SAS ファイルの操作" (548 ページ)

詳細

基本
RUN グループの実行時に、SAS ファイルが即時に削除されます。削除操作は、開始前に確認できません。

SAS ファイルに ALTER=パスワードが割り当てられている場合、そのパスワードを SAS ファイルを削除するために指定する必要があります。

プロシジャ入力ライブラリに存在しない SAS ファイルを削除しようとすると、PROC DATASETS はメッセージを発行し、処理は続行されます。NOWARN が使用されている場合、メッセージは発行されません。

DELETE ステートメントを使用して、インデックスが関連付けられているデータセットを削除する場合、ステートメントはそのインデックスも削除します。

DELETE ステートメントを使用して、外部キー一貫性制約または外部キー参照を含むプライマリキーを持つデータファイルを削除することはできません。外部キーを持つデータファイルについては、データファイルを削除する前に外部キーを削除する必要があります。外部キー参照を含むプライマリキーを持つデータファイルについては、データファイルを削除する前にそのプライマリキーを参照する外部キーを削除する必要があります。

世代グループの使用

世代グループの使用の概要
世代グループを使用している場合、DELETE ステートメントを使用して、次のバージョンを削除できます。

- 基本バージョンとすべての履歴バージョンを削除します
- 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更します
- 絶対バージョンを削除します
- 関連バージョンを削除します

- すべての履歴バージョンを削除し、基本バージョンを残します

基本バージョンとすべての履歴バージョンを削除
次のステートメントは、基本バージョンとすべての履歴バージョンを削除します。データセット名は A です。

```
proc datasets;
delete A(gennum=all);

proc datasets;
delete A / gennum=all;

proc datasets gennum=all;
delete A;
```

次のステートメントは、基本バージョンとすべての履歴バージョンを削除します。データセット名は文字 A で始まります。

```
proc datasets;
delete A:(gennum=all);

proc datasets;
delete A: / gennum=all;

proc datasets gennum=all;
delete A:;
```

基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更
次のステートメントは、基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更します。データセット名は A です。

```
proc datasets;
delete A(gennum=revert);

proc datasets;
delete A / gennum=revert;

proc datasets gennum=revert;
delete A;
```

次のステートメントは、基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更します。データセット名は文字 A で始まります。

```
proc datasets;
delete A:(gennum=revert);

proc datasets;
delete A: / gennum=revert;

proc datasets gennum=revert;
delete A:;
```

絶対数によってバージョンを削除
次のステートメントは絶対数を使用して、最初の履歴バージョンを削除します。

```
proc datasets;
delete A(gennum=1);

proc datasets;
```

```
delete A / gennum=1;

proc datasets gennum=1;
delete A;
```

次のステートメントは、特定の履歴バージョンを削除します。データセット名は文字 A で始まります。

```
proc datasets;
delete A:(gennum=1);

proc datasets;
delete A: / gennum=1;

proc datasets gennum=1;
delete A:;
```

相対値によってバージョンを削除
次のステートメントは、相対値を使用して、最新履歴バージョンを削除します。データセット名は A です。

```
proc datasets;
delete A(gennum=-1);

proc datasets;
delete A / gennum=-1;

proc datasets gennum=-1;
delete A;
```

次のステートメントは、相対値を使用して、最新履歴バージョンを削除します。データセット名は A で始まります。

```
proc datasets;
delete A:(gennum=-1);

proc datasets;
delete A: / gennum=-1;

proc datasets gennum=-1;
delete A:;
```

すべての履歴バージョンを削除し、基本バージョンを残す
次のステートメントは、すべての履歴バージョンを削除し、基本バージョンを残します。データセット名は A です。

```
proc datasets;
delete A(gennum=hist);

proc datasets;
delete A / gennum=hist;

proc datasets gennum=hist;
delete A;
```

次のステートメントは、すべての履歴バージョンを削除し、基本バージョンを残します。データセット名は文字 A で始まります。

```
proc datasets;
delete A:(gennum=hist);
```

```
proc datasets;
delete A: / gennum=hist;

proc datasets gennum=hist;
delete A:;
```

EXCHANGE ステートメント

SAS ライブラリ内の 2 つの SAS ファイルの名前を入れ換えます。

例: “例 2: SAS ファイルの操作” (548 ページ)

構文

EXCHANGE *name-1=other-name-1* <*name-2=other-name-2* …>
</ <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

必須引数

name=other-name
プロシジャ入力ライブラリ内の SAS ファイルの名前を入れ換えます。*name* と *other-name* がプロシジャ入力ライブラリに存在する必要があります。

オプション引数

ALTER=*alter-password*
入れ換える名前の変更保護されている SAS ファイルに対する変更パスワードを与えます。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。同じ種類の SAS ファイルの名前のみ入れ換えることができます。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

デフォルト PROC DATASETS ステートメントで MEMTYPE=を指定しない場合、デフォルトは ALL となります。

参照項目 “処理するメンバの種類の制限” (442 ページ)

詳細

1 つの EXCHANGE ステートメントで複数の名前ペアを入れ換える場合、PROC DATASETS は EXCHANGE ステートメントで入れ換えをリストする順序ではなく、SAS ファイルの名前がディレクトリリストで発生する順序で入れ換えを実行します。

name の SAS ファイルが SAS ライブラリに存在しない場合、PROC DATASETS は EXCHANGE ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを指定します。

また、EXCHANGE ステートメントは 関連付けられているインデックスを新しい名前と対応するように入れ換えます。

EXCHANGE ステートメントにより、2 つの既存する世代グループの名前の入れ換えが可能になります。特定の世代番号を既存する基本バージョンまたは別の世代番号と入れ換えることはできません。

EXCLUDE ステートメント

SAS ファイルをコピーから除外します。

制限事項: EXCLUDE ステートメントは、COPY ステートメントの後に指定する必要があります

EXCLUDE ステートメントは、SELECT ステートメントで同じ COPY ステップに記述できません

例: “例 2: SAS ファイルの操作” (548 ページ)

構文

EXCLUDE *SAS-file(s)* </ MEMTYPE=*member-type*>;

必須引数

SAS-file(s)

コピー操作から除外する SAS ファイルを 1 つ以上指定します。EXCLUDE ステートメントで指定するすべての SAS ファイルが、COPY ステートメントの IN=オプションで指定されるライブラリ内にある必要があります。SAS ファイルが世代グループである場合、EXCLUDE ステートメントにより基本バージョンの選択のみ可能になります。

次のショートカットを使用して、EXCLUDE ステートメントに複数の SAS ファイルをリストできます。

表記	意味
x1–xn	ファイル X1 から Xn を指定します。番号は連続している必要があります。
x:	文字 X で始まるすべてのファイルを指定します。

オプション引数

MEMTYPE=*member-type*

処理を 1 つのメンバの種類に制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

別名	MTYPE=、MT=
デフォルト	PROC DATASETS ステートメント、COPY ステートメント、または EXCLUDE ステートメントで MEMTYPE=を指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目 “SAS ファイルのコピー/移動時のメンバの種類の指定” (487 ページ)および “処理するメンバの種類の制限” (442 ページ)

詳細

複数の同じ名前のファイルを除外

EXCLUDE ステートメントで複数の SAS ファイルをリストするためのショートカットを使用できます。

FORMAT ステートメント

MODIFY ステートメントで常に指定される SAS データセットの変数の出力形式を割り当て、変更、削除します。

制限事項: FORMAT ステートメントは、MODIFY RUN グループで指定する必要があります

例: “例 4: SAS データセットの変更” (555 ページ)

構文

FORMAT *variable-1* <*format-1*>
<*variable-2* <*format-2*> …>;

必須引数

variable
割り当て、変更、削除する出力形式の変数を 1 つ以上指定します。出力形式と変数の関連付けを解除する場合、リストの最後に変数をリストします。出力形式はその後に指定しません。

```
format x1-x3 4.1 time hhmm2.2 age;
```

オプション引数

format
その前にリストされている変数に適用する出力形式を指定します。出力形式を指定しない場合、FORMAT ステートメントは *variable-list* の変数と関連付けられている出力形式を削除します。

ヒント すべての出力形式をデータセットから削除するには、ATTRIB ステートメント (467 ページ) と_ALL_キーワードを使用します。

IC CREATE ステートメント

一貫性制約を作成します。

制限事項: IC CREATE ステートメントは、MODIFY RUN グループ内で指定する必要があります

注: 作成する参照制約について、外部キーでプライマリキーと同じ変数の数を同じ順序で指定する必要があります。変数は同じ種類(文字/数値)、同じ長さである必要があります。

参照項目: “Understanding Integrity Constraints” (*SAS Language Reference: Concepts*)

構文

IC CREATE <*constraint-name*=> *constraint* <MESSAGE='*message-string*' <MSGTYPE=USER>>;

必須引数

constraint
制約の種類です。有効な値のリストは次のとおりです。

NOT NULL (*variable*)
variable に特殊欠損値を含む SAS 欠損値が含まれないように指定します。

UNIQUE (*variables*)
variables の値が一意であるように指定します。この制約は DISTINCT と同じです。

DISTINCT (*variables*)
variables の値が一意であるように指定します。この制約は UNIQUE と同じです。

CHECK (WHERE-*expression*)
変数のデータ値を特定のセット、範囲、値リストに制限します。この動作は、WHERE 式によって実行されます。

PRIMARY KEY (*variables*)
プライマリキー、つまり欠損値が含まれず、その値が一意である一連の変数を指定します。

要件 重複するプライマリキー制約と外部キー制約を定義するとき、つまりデータファイルの変数がプライマリキー定義と外部キー定義の一部であるとき、まったく同じ変数を使用している場合、変数を異なる順序で定義する必要があります。

操作 プライマリキーは外部キーによって参照されるまで個別のデータファイルの値に影響します。

注 ヌル以外の制約が、新しいプライマリキー制約の定義に使用されている変数に存在する場合、そのプライマリキー制約によって既存のヌル以外の制約は置換されます。

ヌル以外の制約が、プライマリキー制約にすでに含まれる変数に定義されていない場合、新しいヌル以外の制約定義は失敗します。

FOREIGN KEY (*variables*) REFERENCES *table-name* <ON DELETE *referential-action*> <ON UPDATE *referential-action*>
外部キー、つまりその値が別のデータファイルのプライマリキー変数の値にリンクされている一連の変数を指定します。参照アクションは、外部キーによって参照されるプライマリキー変数の値が更新されるときに強制されます。

参照アクションには、RESTRICT、SET NULL、CASCADE の 3 つの種類があります。

次の操作は、RESTRICT 参照アクションによって実行できます。

a delete operation
外部キー値が削除された値に一致しない場合にのみプライマリキー行を削除します。

an update operation
外部キー値が更新対象となる現在の値と一致しない場合にのみプライマリキー値を更新します。

次の操作は、SET NULL 参照アクションによって実行できます。

a delete operation
プライマリキー行を削除し、対応する外部キー値を NULL に設定します。

an update operation
プライマリキー値を変更し、一致するすべての外部キー値を NULL に設定します。

次の操作は、CASCADE 参照アクションによって実行できます。

an update operation
プライマリキー値を変更し、さらに一致する外部キー値を同じ値に変更します。CASCADE は、削除操作にはサポートされていません。

デフォルト RESTRICT は、参照アクションが指定されていない場合、デフォルトのアクションです。

要件 重複するプライマリキー制約と外部キー制約を定義するとき、つまりデータファイルの変数がプライマリキー定義と外部キー定義の一部であるとき、次のアクションが必要です。

まったく同じ変数を使用している場合、変数を異なる順序で定義する必要があります。

外部キーの更新および削除参照アクションは、ともに RESTRICT である必要があります。

操作 SET NULL 参照アクションまたは CASCADE 参照アクションを強制する前に、プライマリキーを参照し、意図した操作に RESTRICT を指定するその他の外部キーがあるかどうかを確認します。RESTRICT が指定されている、または制約がデフォルト値に戻る場合、外部キー値が更新対象または削除対象となる値に一致しない場合を除き、RESTRICT はすべての外部キーに対して強制されます。

オプション引数

<constraint-name=>
制約のオプションの名前です。名前は有効な SAS 名である必要があります。制約名を入力しない場合、デフォルトの名前が生成されます。このデフォルトの制約名には、次の形式があります。

デフォルト名	制約の種類
NM*xxxx*	Not Null
UN*xxxx*	Unique
CK*xxxx*	Check
PK*xxxx*	Primary key

デフォルト名	制約の種類
FK*xxxx*	Foreign key

xxxx は、0001 から始まるカウンタです。

注: 名前 PRIMARY、FOREIGN、MESSAGE、UNIQUE、DISTINCT、CHECK および NOT は、*constraint-name* の値として使用することはできません。

<MESSAGE='*message-string*' <MSGTYPE=USER>>
message-string は、データが制約に違反した場合にログに書き込まれるエラーメッセージのテキストです。

```
ic create not null(socsec)
message='Invalid Social Security number';
```

<MSGTYPE=USER> は、一貫性制約エラーメッセージの形式を制御します。デフォルトで MESSAGE=オプションが指定されると、定義したメッセージが制約に関する SAS エラーメッセージにスペースで区切られて挿入されます。
MSGTYPE=USER は、メッセージの SAS 部分を非表示にします。

長さ メッセージの最大長は 250 文字です。

例 次の例に、一貫性制約の作成方法を示します。

```
ic create a = not null(x);
ic create Unique_D = unique(d);
ic create Distinct_DE = distinct(d e);
ic create E_less_D = check(where=(e < d or d = 99));
ic create primkey = primary key(a b);
ic create forkey = foreign key (a b) references table-name
on update cascade on delete set null;
ic create not null (x);
```

IC DELETE ステートメント

一貫性制約を削除します。

制限事項: IC DELETE は、MODIFY RUN グループ内にある必要があります

参照項目: “Understanding Integrity Constraints” (*SAS Language Reference: Concepts*)

構文

IC DELETE *constraint-name(s)* | _ALL_;

必須引数

constraint-name(s)
削除する制約を 1 つ以上指定します。たとえば、制約 Unique_D と Unique_E を削除するには、次のステートメントを使用します。**`ic delete Unique_D Unique_E;`**

ALL
前の MODIFY ステートメントで指定した SAS データファイルに対するすべての制約を削除します。

IC REACTIVATE ステートメント

非アクティブな外部キー一貫性制約を再アクティブ化します。

制限事項: IC REACTIVATE は、MODIFY RUN グループ内にある必要があります

参照項目: “Understanding Integrity Constraints” (*SAS Language Reference: Concepts*)

構文

IC REACTIVATE *foreign-key-name* REFERENCES *libref*;

必須引数

foreign-key-name
再アクティブ化する外部キーの名前です。

libref
外部キーによって参照されるプライマリキーを含むデータセットを含む SAS ライブラリを参照します。

例

外部キー FKEY をデータセット MYLIB.MYOWN で定義し、FKEY がデータセット MAINLIB.MAIN のプライマリキーにリンクされているとします。一貫性制約がコピー操作または移動操作によって非アクティブ化される場合、次のコードを使用して一貫性制約を再アクティブ化することができます。

```
proc datasets library=mylib;
modify myown;
ic reactivate fkey references mainlib;
run;
```

INDEX CENTILES ステートメント

インデックス付き変数のパーセント点統計量を更新します。

制限事項: INDEX CENTILES は、MODIFY RUN グループ内にある必要があります

参照項目: “Understanding SAS Indexes” (*SAS Language Reference: Concepts*)

構文

INDEX CENTILES *index(s)*
</ <REFRESH> <UPDATECENTILES= ALWAYS | NEVER | *integer*>>;

必須引数

index(s)
インデックスを 1 つ以上指定します。

オプション引数

REFRESH
UPDATECENTILES の値に関係なく、パーセント点を即時更新します。

UPDATECENTILES=ALWAYS | NEVER | *integer*
パーセント点がいつ更新されるかを指定します。データセット更新のたびにパーセント点を更新することは実用的ではありません。そのため、インデックス付き変数のパーセント点が更新される前に変更可能なデータ値のパーセントを UPDATECENTILES の値として指定できます。

有効な値のリストは次のとおりです。

ALWAYS | 0
データセットインデックスが変更された場合、データセットのクローズ時にパーセント点を更新します。ALWAYS または 0 を指定しても、結果は同じです。

NEVER | 101
パーセント点を更新しません。NEVER または 101 を指定しても、結果は同じです。

integer
パーセント点が更新される前に更新可能なインデックス付き変数の値のパーセントです。

別名	UPDCEN
デフォルト	5(パーセント)

INDEX CREATE ステートメント

MODIFY ステートメントで指定される SAS データセットの単一インデックスまたは複合インデックスを作成します。

制限事項: INDEX CREATE は、MODIFY RUN グループ内にある必要があります

参照項目: “Understanding SAS Indexes” (*SAS Language Reference: Concepts*)

例: “例 4: SAS データセットの変更” (555 ページ)

構文

INDEX CREATE *index-specification(s)*
</ <NOMISS> <UNIQUE> <UPDATECENTILES= ALWAYS | NEVER | *integer*>>;

必須引数

index-specification(s)
次の形式のうちいずれか、またはその両方になります。

variable
指定した変数上に単一インデックスを作成します。

index=(variables)
複合インデックスを作成します。*index* に指定する名前は、複合インデックスの名前です。有効な SAS 名である必要があります。いずれの変数名、またはその他の複合インデックス名と同じにはできません。変数を少なくとも 2 つ指定する必要があります。

注 インデックス名は、_N_などの自動変数の予約名、_ALL_などの特殊変数リスト名の使用を避けるなど、SAS 変数名と同じルールに従う必要があります。詳細については、*SAS 言語リファレンス: 解説編*の、“SAS 言語での単語および名前のルール”を参照してください。

オプション引数

NOMISS
すべてのインデックス変数に対する欠損値を含むすべてのオブザベーションをインデックスから除外します。

NOMISS オプションによってインデックスを作成する際、インデックスは WHERE 処理にのみ、および欠損値が WHERE 式を満たしていない場合にのみ使用されます。たとえば、次の WHERE ステートメントを使用すると、欠損値が WHERE 式を満たしているためインデックスは使用されません。

```
where dept ne '01';
```

詳細については、*SAS 言語リファレンス: 解説編*を参照してください。

BY グループ処理は、NOMISS オプションによって作成されるインデックスを無視します。

例 “例 4: SAS データセットの変更” (555 ページ)

UNIQUE
インデックス変数の値の組み合わせが一意になるように指定します。UNIQUE を指定し、複数のオブザベーションのインデックス変数の値が同じ場合、インデックスは作成されません。

例 “例 4: SAS データセットの変更” (555 ページ)

UPDATECENTILES=ALWAYS | NEVER | *integer*
パーセント点がいつ更新されるかを指定します。データセット更新のたびにパーセント点を更新することは実用的ではありません。そのため、インデックス付き変数に対するパーセント点が更新される前に変更可能なデータ値のパーセントを指定できます。有効な値のリストは次のとおりです。

ALWAYS | 0
データセットインデックスが変更された場合、データセットのクローズ時にパーセント点を更新します。ALWAYS または 0 を指定しても、結果は同じです。

NEVER | 101
パーセント点を更新しません。NEVER または 101 を指定しても、結果は同じです。

integer
パーセント点が更新される前に更新可能なインデックス付き変数の値のパーセントを指定します。

別名 UPDCEN

デフォルト 5(パーセント)

INDEX DELETE ステートメント

MODIFY ステートメントで指定した SAS データセットと関連付けられている 1 つ以上のインデックスを削除します。

制限事項: INDEX DELETE は、MODIFY RUN グループ内で指定する必要があります

注: CONTENTS ステートメントを使用すると、データセットのすべてのインデックスリストを生成できます。

構文

INDEX DELETE *index(s)* | _ALL_;

必須引数

index(s)
削除するインデックスを 1 つ以上指定します。インデックスは、前の MODIFY ステートメントで指定される SAS データセットの変数に対応している必要があります。単一インデックスと複合インデックスの両方を削除できます。

ALL
一貫性制約によって所有されるインデックスを除く、すべてのインデックスを削除します。インデックスは作成時にユーザー、一貫性制約、またはその両方によって所有されていることがマーク付けされます。インデックスがユーザーと一貫性制約の両方によって所有されている場合、インデックスは IC DELETE ステートメントと INDEX DELETE ステートメントが処理されるまで削除されません。

INFORMAT ステートメント

MODIFY ステートメントで常に指定されるデータセットの変数の入力形式を割り当て、変更、削除します。

制限事項: INFORMAT は、MODIFY RUN グループで指定する必要があります

例: “例 4: SAS データセットの変更” (555 ページ)

構文

INFORMAT *variable-1* <*informat-1*>
<*variable-2* <*informat-2*> …>;

必須引数

variable
割り当て、変更、削除する入力形式の変数を 1 つ以上指定します。入力形式と変数の関連付けを解除する場合、リストの最後に変数をリストします。入力形式はその後に指定しません。

```
informat a b 2. x1-x3 4.1 c;
```

オプション引数

informat
変数の入力形式をステートメントの入力形式の直前に指定します。入力形式を指定しない場合、INFORMAT ステートメントは *variable-list* の変数の既存する入力形式を削除します。

ヒント すべての入力形式をデータセットから削除するには、ATTRIB ステートメント (467 ページ) と_ALL_キーワードを使用します。

INITIATE ステートメント

SAS データファイルと同じ名前で、データセットの種類が AUDIT の監査ファイルを作成します。

制限事項: INITIATE ステートメントは、AUDIT RUN グループ内で指定する必要があります。

INITIATE ステートメントは監査ファイルごとに一度実行可能な必須のステートメントで、そのファイルに対する最初の AUDIT 実行グループの AUDIT ステートメントの直後におく必要があります。

例: “例 11: 監査ファイルの初期化” (578 ページ)

構文

INITIATE <AUDIT_ALL=NO | YES>;

オプション引数

AUDIT_ALL=NO|YES
記録が中断可能かどうか、監査設定が変更可能かどうかを指定します。AUDIT_ALL=YES は、すべてのイメージが記録され、中断できないように指定します。つまり、LOG ステートメントを使用して、特定のイメージの記録を無効化できません。また、SUSPEND ステートメントを使用してイベントの記録を中断できません。記録を無効化するには、イベントの記録を終了し、監査ファイルを削除する TERMINATE ステートメントを使用する必要があります。

デフォルト NO

例 “例 11: 監査ファイルの初期化” (578 ページ)

詳細

監査ファイルは、SAS データファイルへの追加、削除、更新を記録します。監査証跡は、中断、再開、終了できるようにするには、開始する必要があります。INITIATE ステートメント直前の AUDIT ステートメントは GENNUM=オプションを指定できませんが、指定したファイルが世代データセットグループを識別する場合、INITIATE ステートメントによって作成された監査ファイルは、世代グループで最も新しく作成された世代に追加されます。

次の例では、監査ファイル MYLIB.MYFILE.AUDIT を作成し、更新をデータファイル MYLIB.MYFILE.DATA に書き込み、すべての利用可能なレコードイメージを保存します。

```
proc datasets library=mylib;
audit myfile (alter=password);
initiate;
```

```
run;
```

次の例では、AUDIT_ALL=YES を使用して監査ファイルを開始します。

```
proc datasets lib=mylib; /* all audit image types will be logged
and the file cannot be suspended */
audit myfile (alter=password);
initiate audit_all=yes;
quit;
```

LABEL ステートメント

MODIFY ステートメントで指定した SAS データセットの変数ラベルを割り当て、変更、削除します。

制限事項: LABEL ステートメントは、MODIFY RUN グループで指定する必要があります

例: “例 4: SAS データセットの変更” (555 ページ)

構文

LABEL *variable-1=<'label-1' | ' '>*
<variable-2=<'label-2' | ' '> …>;

必須引数

variable=<'label'>
最大 256 文字のテキスト文字列を指定します。ラベルテキストに一重引用符が含まれている場合、ラベルの前後に二重引用符を使用するか、ラベルテキストで一重引用符を 2 つ使用して文字列を一重引用符で囲みます。ラベルをデータセットから削除するには、引用符に囲まれているブランクに相当するラベルを割り当てます。

範囲 1 - 256 文字

ヒント データセットのすべての変数ラベルを削除するには、ATTRIB ステートメント (467 ページ) と_ALL_キーワードを使用します。

LOG ステートメント

監査ファイル設定を指定します。

制限事項: LOG ステートメントは、AUDIT RUN グループ内の INITIATE ステートメントの直後に指定する必要があります。

例: “例 11: 監査ファイルの初期化” (578 ページ)

構文

LOG <ADMIN_IMAGE=YES | NO>
<BEFORE_IMAGE=YES | NO>
<DATA_IMAGE=YES | NO>
<ERROR_IMAGE=YES | NO>;

オプション引数

ADMIN_IMAGE=YES|NO
管理イベントが監査ファイルに書き込まれるかどうか(SUSPEND アクションと RESUME アクション)を指定します。

デフォルト YES

ヒント 特定のイメージを書き込まない場合、そのイメージの種類に対して NO を指定します。たとえば、次のコードはエラーイメージの書き込みをオフにしますが、管理イベント、更新前のレコードイメージ、データイメージの書き込みは継続されます。`log error_image=no;`

BEFORE_IMAGE=YES|NO
更新前のレコードイメージが監査ファイルに書き込まれるかどうかを指定します。

デフォルト YES

DATA_IMAGE=YES|NO
追加、削除、更新された後のレコードイメージが監査ファイルに書き込まれるかどうかを指定します。

デフォルト YES

ERROR_IMAGE=YES|NO
更新後のレコードイメージが監査ファイルに書き込まれるかどうかを指定します。

デフォルト YES

詳細

次の例では、同じ監査ファイルを作成しますが、エラーレコードイメージのみを保存します。

```
proc datasets library=mylib;
audit myfile (alter=password);
initiate;
log admin_image=no
before_image=no
data_image=no;
run;
```

MODIFY ステートメント

SAS ファイルの属性、および下位ステートメントの使用により SAS ファイルの変数の属性を変更します。

制限事項: ATTRIB ステートメントで LENGTH=オプションを使用して、変数の長さを変更できません。

例: “例 4: SAS データセットの変更” (555 ページ)

構文

MODIFY *SAS-file* <(*option(s)*)>
</ <CORRECTENCODING=*encoding-value*> <DTC=*SAS-date-time*>
<GENNUM=*integer*> <MEMTYPE=*member-type*>>;

オプション引数の要約

MEMTYPE=*member-type*
処理を特定の種類の SAS ファイルに制限します。

AES encryption

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

Modify generation groups

GENMAX=*number-of-generations*
世代グループのバージョンの最大数を変更します。

GENNUM=*integer*
履歴バージョンを変更します。

Modify passwords

ALTER=*password-modification*
変更パスワードを変更します。

PW=*password-modification*
読み取り、書き込み、変更のパスワードを変更します。

READ=*password-modification*
読み取りパスワードを変更します。

WRITE=*password-modification*
書き込みパスワードを変更します。

Specify data set attributes

CORRECTENCODING=*encoding-value*
文字セットのエンコーディングを変更します。

DTC=*SAS-date-time*
作成日時を指定します。

LABEL=*'data-set-label'* | ''
データセットラベルを割り当て、または変更します。

SORTEDBY=*sort-information*
データの現在の並べ替え方法を指定します。

TYPE=*special-type*
特殊データセットの種類を割り当て、変更します。

必須引数

SAS-file
プロシジャ入力ライブラリに存在する SAS ファイルを指定します。

オプション引数

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

要件 データファイルに AES 暗号化が含まれる場合、ENCRYPTKEY=データセットオプションが必要です。

ALTER=*password-modification*
MODIFY ステートメントで指定される SAS ファイルに対する変更パスワードを割り当て、変更、削除します。*password-modification* は、次のうちいずれかになります。

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

参照項目 “パスワードの操作” (513 ページ)

CORRECTENCODING=*encoding-value*
ファイルのデータの実際のエンコーディングに一致させるため、ファイルのディスクリプタ情報に記録されるエンコーディング指標の変更を可能にします。

参照項目 *SAS 各国語サポート(NLS): リファレンスガイド*の “CORRECTENCODING= Option”

DTC=*SAS-date-time*
作成時に SAS ファイルに挿入される日時スタンプの代わりとなる日時を指定します。このオプションは、各 SAS ファイル名の後にかっこで囲んで使用することはできません。スラッシュの後に DTC=を指定する必要があります。

```
modify mydata / dtc='03MAR00:12:01:00'dt;
```

制限事項 SAS ファイルの作成日時は、ファイルが実際に作成された日時よりも後に設定できません。

DTC=は、暗号化されたファイルまたは順編成ファイルと使用できません。

DTC=は、結果の SAS ファイルで V8 エンジンまたは V9 エンジンが使用される場合にのみ使用できます。

ヒント COPY プロシジャ、CPORT プロシジャ、SORT プロシジャで DATECOPY オプションを、DATASETS プロシジャで COPY ステートメントを使用する前に、DTC=を使用して SAS ファイルの作成日時を変更します。

GENMAX=*number-of-generations*
バージョンの最大数を指定します。このオプションは SAS ファイル名の後にかっこで囲んで使用します。

デフォルト 0

範囲 0-1,000

GENNUM=*integer*
世代データセットの処理を制限します。GENNUM=は、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に指定します。有効な値は *integer* です。これは、世代グループの特定バージョンを参照する数字です。正の数の指定は、データセット名に追加される特定の世代番号の絶対参照です(つまり、**gennum=2** は MYDATA#002 を指定します)。負の数の指定は、最新から最古への基本バージョンに関連する履歴バージョンの関連参照です(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。デフォルトの 0 を指定すると、基本バージョンが参照されます。

参照項目 *SAS 言語リファレンス: 解説編*の“Understanding Generation Data Sets”

LABEL=*'data-set-label'* | ' '
MODIFY ステートメントで指定される SAS データセットのデータセットラベルを割り当て、変更、削除します。一重引用符がラベルで使用されている場合、2 つの一重引用符として記述します。LABEL=または LABEL=' 'が現在のラベルを削除します。

範囲 1 - 256 文字

制限事項 ビューラベルは、ラベルの作成後は更新できません。

ヒント データセットのすべての変数ラベルを削除するには、ATTRIB ステートメント (467 ページ)を使用します。

参照項目 “例 4: SAS データセットの変更” (555 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。MEMTYPE=は、各 SAS ファイル名の後にかっこで囲んで指定できません。スラッシュの後に指定する必要があります。

別名 MTYPE=と MT=

デフォルト PROC DATASETS ステートメントまたは MODIFY ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=DATA となります。

PW=*password-modification*
MODIFY ステートメントで指定される SAS ファイルに対する読み出し、書き込み、変更のパスワードを割り当て、変更、削除します。*password-modification* は、次のうちいずれかになります。

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

参照項目 “パスワードの操作” (513 ページ)

READ=*password-modification*
MODIFY ステートメントで指定される SAS ファイルに対する読み取りパスワードを割り当て、変更、削除します。*password-modification* は、次のうちいずれかになります。

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

参照項目 “パスワードの操作” (513 ページ)

例 “例 4: SAS データセットの変更” (555 ページ)

SORTEDBY=*sort-information*
データの現在の並べ替え方法を指定します。ファイルとともに並べ替え情報が保存されますが、データが指定した方法で並べ替えられていることは確認されません。*sort-information* は、次のうちいずれかになります。

***by-clause* </ *collate-name*>**
データの現在の並べ替え方法を示します。*by-clause* の値は、PROC SORT ステップの BY ステートメントで使用できる変数とオプションです。*collate-name* は、並べ替えに使用される照合順序を指定します。デフォルトで、照合順序はホスト動作環境の照合順序です。

NULL
既存するソートインジケータを削除します。

制限事項 データは指定した順序で並べ替える必要があります。データが指定した順序になっていない場合、並べ替えは実行されません。

ヒント MODIFY SORTEDBY オプションの使用時は、番号範囲リストまたはコロンリストを使用することもできます。詳細については、“Data Set Lists” (*SAS Language Reference: Concepts*)を参照してください。

例 “例 4: SAS データセットの変更” (555 ページ)

TYPE=*special-type*
SAS データセットの特殊データセットの種類を割り当て、変更します。SAS では、次の点について確認しません。

- TYPE=オプションで指定する SAS データセットの種類(長さが 8 文字以下であるかどうかのチェックは除く)
- SAS データセットの構造が指定した種類に適している

注 TYPE=オプションと MEMTYPE=オプションを混同しないようにしてください。TYPE=オプションは、特殊な SAS データセットの種類を指定します。MEMTYPE= オプションは、SAS ライブラリ内の 1 つ以上の SAS ファイルの種類を指定します。

ヒント ほとんどの SAS データセットには、特殊な種類はありません。ただし、特定の SAS プロシジャは、CORR プロシジャのように、多くの特殊な SAS データセットを作成できます。また、SAS/STAT ソフトウェアと SAS/EIS ソフトウェアでは、特殊データセットの種類がサポートされています。

WRITE=*password-modification*
MODIFY ステートメントで指定される SAS ファイルに対する書き込みパスワードを割り当て、変更、削除します。*password-modification* は、次のうちいずれかになります。

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

参照項目 “パスワードの操作” (513 ページ)

詳細

データセットラベルと変数ラベルの変更

LABEL オプションは、データセット内のデータセットラベルまたは変数ラベルのいずれかを変更できます。データセットラベルを変更するには、次の構文を使用します。

```
modify datasetname(label='Label for Data Set');
run;
```

データセット内の 1 つ以上の変数ラベルを変更できます。データセット内の変数ラベルを変更するには、次の構文を使用します。

```
modify datasetname;
label variablename='Label for Variable';
run;
```

同じ PROC DATASETS でデータセットラベルと変数ラベルの両方を変更する例については、“例 4: SAS データセットの変更” (555 ページ)を参照してください。

パスワードの操作

パスワードを割り当て、変更、削除するため、ファイル上に現在存在する最上位レベルの保護に対しパスワードを指定する必要があります。

パスワードの割り当て

```
/* assigns a password to an unprotected file */
modify colors (pw=green);

/* assigns an Alter password to an already read-protected SAS data set */
modify colors (read=green alter=red);
```

パスワードの変更

```
/* changes the Write password from YELLOW to BROWN */
modify cars (write=yellow/brown);

/* uses Alter access to change unknown Read password to BLUE */
modify colors (read=/blue alter=red);
```

パスワードの削除

```
/* removes the Alter password RED from STATES */
modify states (alter=red/);

/* uses Alter access to remove the Read password */
modify zoology (read=green/ alter=red);

/* uses PW= as an alias for either WRITE= or ALTER= to remove unknown
Read password */
modify biology (read=/ pw=red);
```

世代グループの使用

世代数の変更

```
/* changes the number of generations on data set A to 99 */
modify A (genmax=99);
```

パスワードの削除

```
/* removes the Alter password RED from STATES#002 */
```

```
modify states (alter=red/) / gennum=2;
```

REBUILD ステートメント

無効化されたインデックスと一貫性制約を元に戻す、または削除するかどうかを指定します。

デフォルト: インデックスと一貫性制約の再作成

制限事項: REBUILD ステートメントが適用されるのは、バージョン 7 以降で作成されたデータセットのみです

構文

REBUILD *SAS-file* </ <ENCRYPTKEY=*key-value*>
<ALTER=*password*> <GENNUM=*n*> <MEMTYPE=*member-type*> <NOINDEX>>;

必須引数

SAS-file
無効化されたインデックスと一貫性制約を含む SAS データファイルを指定します。番号範囲リストまたはコロンリストを使用することもできます。

参照項目 “Data Set Lists” (*SAS Language Reference: Concepts*).

オプション引数

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

要件 データファイルに AES 暗号化が含まれる場合、ENCRYPTKEY=データセットオプションが必要です。

ALTER=*alter-password*
REBUILD ステートメントで指定される変更保護された SAS ファイルに対する変更パスワードを与えます。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

GENNUM=*integer*
世代データセットの処理を制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。有効な値は *integer* です。これは、世代グループの特定バージョンを参照する数字です。正の数の指定は、データセット名に追加される特定の世代番号の絶対参照です(つまり、`gennum=2` は MYDATA#002 を指定します)。負の数の指定は、最新から最古への基本バージョンに関連する履歴バージョンの関連参照です(つまり、`gennum=-1` は最新の履歴バージョンを参照します)。デフォルトの 0 を指定すると、基本バージョンが参照されます。

参照項目 “Understanding Generation Data Sets” (*SAS Language Reference: Concepts*)

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。

別名 MT=、MTYPE=

デフォルト PROC DATASETS ステートメントまたは REBUILD ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

NOINDEX
無効化されたインデックスと一貫性制約の削除を指定します。

制限事項 NOINDEX オプションは、参照一貫性制約を 1 つ以上含むデータファイルに使用できません。

詳細

DLDMGACTION=NOINDEX データセットまたはシステムオプションが指定され、損傷したデータファイルが見つかると、次の項目が実行されます。

- インデックスと一貫性制約を含まないデータファイルの修復
- インデックスファイルの削除
- 無効化されたインデックスと一貫性制約を反映するためのデータファイルの更新
- INPUT モードでのみ開くデータファイルの制限
- 次の警告の SAS ログへの書き込み

```
WARNING: SAS data file MYLIB.MYFILE.DATA was damaged and
has been partially repaired. To complete the repair,
execute the DATASETS procedure REBUILD statement.
```

REBUILD ステートメントは、すべてのデータファイルの無効化されたインデックスと一貫性制約を再作成または削除することにより、損傷した SAS データファイルの修復を完了します。REBUILD ステートメントは、新しいインデックスファイルを処理し、インデックスと一貫性制約を復元するために、メンバレベルロックを作成し、使用します。

インデックスファイルを再作成し、インデックスと一貫性制約を復元するには、次のコードを使用します。

```
proc datasets library=mylib
rebuild myfile
/alter=password
gennum=n
memtype=mytype;
```

無効化されたインデックスと一貫性制約を削除するには、次のコードを使用します。

```
proc datasets library=mylib
rebuild myfile /noindex;
```

REBUILD ステートメントを実行すると、データファイルは INPUT モードに制限されなくなります。

REBUILD ステートメントのデフォルトは、インデックス、一貫性制約、インデックスファイルを再作成することです。

データファイルに 1 つ以上の参照一貫性制約が含まれ、NOINDEX オプションを REBUILD ステートメントで使用する場合、次のエラーメッセージが SAS ログに書き込まれます。

```
Error: Unable to rebuild data file MYLIB.MYFILE.DATA using the
NOINDEX option because the data file contains referential
constraints. Resubmit the REBUILD statement without the
NOINDEX option to restore the data file.
```

RENAME ステートメント

MODIFY ステートメントで指定した SAS データセット内の変数の名前を変更します。

制限事項: RENAME ステートメントは、MODIFY RUN グループで指定する必要があります

例: “例 4: SAS データセットの変更” (555 ページ)

構文

RENAME *old-name-1=new-name-1*
<*old-name-2=new-name-2* …>;

必須引数

old-name=new-name
MODIFY ステートメントで指定したデータセット内の変数の名前を変更します。*old-name* は、データセット内にすでに存在する変数である必要があります。*new-name* は、データセットにすでに存在する変数の名前、またはインデックス名にはできません。新しい名前は、有効な SAS 名である必要があります。

参照項目 “Rules for Words and Names in the SAS Language” (*SAS Language Reference: Concepts*)

詳細

old-name が SAS データセット内に存在しない、または *new-name* がすでに存在する場合、PROC DATASETS は RENAME ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。

RENAME ステートメントを使用して単一インデックスがある変数の名前を変更する場合、ステートメントはインデックスの名前も変更します。

名前を変更している変数が複合インデックスで使用されると、その複合インデックスは自動的に新しい変数名を参照します。ただし、変数の名前をすでに複合インデックスに使用されている名前に変更しようとすると、エラーメッセージが表示されます。

REPAIR ステートメント

破損した SAS データセットまたはカタログを使用に適した状態に復元します。

構文

REPAIR *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*>
<ALTER=*alter-password*> <GENNUM=*integer*> <MEMTYPE=*member-type*>>;

必須引数

SAS-file(s)
プロシジャ入力ライブラリ内の 1 つ以上の SAS データセットまたはカタログを指定します。番号範囲リストまたはコロンリストを使用することもできます。

参照項目 “Data Set Lists” (*SAS Language Reference: Concepts*)

オプション引数

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

要件 データファイルに AES 暗号化が含まれる場合、ENCRYPTKEY=データセットオプションが必要です。

ALTER=*alter-password*
REPAIR ステートメントで指定される変更保護された SAS ファイルに対する変更パスワードを与えます。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

GENNUM=*integer*
世代データセットの処理を制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。有効な値は *integer* です。これは、世代グループの特定バージョンを参照する数字です。正の数の指定は、データセット名に追加される特定の世代番号の絶対参照です(つまり、`gennum=2` は MYDATA#002 を指定します)。負の数の指定は、最新から最古への基本バージョンに関連する履歴バージョンの関連参照です(つまり、`gennum=-1` は最新の履歴バージョンを参照します)。デフォルトの 0 を指定すると、基本バージョンが参照されます。

参照項目 “世代データセットの処理制限” (444 ページ) および“Understanding Generation Data Sets” (*SAS Language Reference: Concepts*)

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。

別名 MT=、MTYPE=

デフォルト PROC DATASETS ステートメントまたは REPAIR ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目 “処理するメンバの種類の制限” (442 ページ)

詳細

REPAIR ステートメントを必要とする最も一般的な状況は、次のとおりです。

- SAS データセットまたはカタログの更新中にシステムエラーが発生した場合。
- SAS データセットまたは関連するインデックス上のデバイスに破損がある場合。この場合、損傷したデータセットまたはインデックスをバックアップデバイスから復元できますが、復元後のデータセットとインデックスは対ではありません。

- SAS データセットやカタログを保存するディスクが、ファイルの書き込みが完了する前にいっぱいになった場合。ディスク容量の一部を解放する必要があることがあります。PROC DATASETS は、インデックスを含む SAS データセットの修復時と SAS カタログの修復時に空きスペースを必要とします。
- SAS データセットまたはカタログエントリの書き込み中に I/O エラーが発生した場合。

REPAIR ステートメントを SAS データセットに使用する場合、データセットのすべてのインデックスが再作成されます。また、データセットが使用に適した状態に復元されますが、復元されたデータセットにシステム障害が発生する前に行った最終更新が含まれていない場合があります。REPAIR ステートメントを使用して、PROC SORT ステップで FORCE オプションを使用することにより破壊されたインデックスを再作成することはできません。

REPAIR ステートメントをカタログに使用する場合、REPAIR ステートメントがエントリを復元したかどうかというメッセージが表示されます。カタログ全体が壊されている可能性がある場合、REPAIR ステートメントはカタログのすべてのエントリを復元しようとします。単一のエントリだけが破壊されている可能性のある場合、たとえば単一のエントリが更新中で、ディスクフルになると、ほとんどのシステムでは問題の発生時に開いていたエントリだけが破壊されている可能性があります。この場合、REPAIR ステートメントはそのエントリだけを修復しようとします。復元されたカタログ内のエントリには、システム障害または I/O エラーの前に行われた最終更新が含まれていないことがあります。REPAIR ステートメントは、切り捨てられたデータを含むエントリに対し警告メッセージを発行します。

損傷したカタログを修復するには、使用する SAS バージョンでカタログの更新が可能である必要があります。SAS バージョンでカタログの更新(または読み取り)が可能かどうかは、カタログを作成した SAS バージョンによって決定されます。

- 破損したバージョン 6 カタログは、バージョン 6 でのみ修復可能です。
- 破損したバージョン 8 カタログは、バージョン 8 または SAS®9 のいずれかで修復可能ですが、バージョン 6 では修復できません。
- 破損した SAS 9 カタログは、SAS 9 でのみ修復可能です。

REPAIR 操作に失敗した場合、SAS データセットまたはカタログをシステムのバックアップファイルから復元します。

指定したライブラリに存在しない SAS ファイルに対し REPAIR ステートメントを発行すると、PROC DATASETS は REPAIR ステートメントを含む実行グループの処理を停止し、エラーメッセージを発行します。この動作を無効にし、処理を続行するには、PROC DATASETS ステートメントで NOWARN オプションを使用します。

クロス環境データアクセス(CEDA)を使用して、損傷した外部 SAS データセットを処理する場合、CEDA では修復できません。CEDA では、損傷したデータセットの修復に必要な更新処理がサポートされていません。外部ファイルを修復するには、そのファイルをネイティブな環境に戻す必要があります。修復プロセス中にオブザベーションが失われることがあります。CEDA の詳細については、“Definition of Cross-Environment Data Access (CEDA)” (*SAS Language Reference: Concepts*)を参照してください。

RESUME ステートメント

監査ファイルへのイベントの記録を、中断されている場合は再開します。

制限事項: RESUME ステートメントは、AUDIT RUN グループ内で指定する必要があります。

例: “例 11: 監査ファイルの初期化” (578 ページ)

構文

RESUME;

詳細

RESUME、SUSPEND、TERMINIATE、USER_VAR、LOG などのその他の監査関連ステートメントは、INITIATE ステートメントが送信されるまで監査ファイルに有効になりません。詳細については、“INITIATE ステートメント” (506 ページ)を参照してください。

RESUME ステートメントには、LOG ステートメントの ADMIN_IMAGE=YES オプションが必要です。このオプションは、管理イベントが監査ファイルに書き込まれるかどうか(SUSPEND アクションと RESUME アクション)を指定します。詳細については、“LOG ステートメント” (507 ページ)を参照してください。

SAVE ステートメント

ライブラリ内の、SAVE ステートメントにリストされている以外のすべての SAS ファイルを削除します。

例: “例 3: SAS ファイルを削除せずに保存する” (552 ページ)

構文

SAVE *SAS-file(s)* </ MEMTYPE=*member-type*>;

必須引数

SAS-file(s)
SAS ライブラリから削除しない SAS ファイルを 1 つ以上指定します。

注: SAS ファイルに ALTER=パスワードが割り当てられている場合、そのパスワードを SAS ファイルを削除するために指定する必要があります。

オプション引数

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

別名	MTYPE=と MT=
デフォルト	PROC DATASETS ステートメントまたは SAVE ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。
参照項目	“処理するメンバの種類の制限” (442 ページ)
例	“例 3: SAS ファイルを削除せずに保存する” (552 ページ)

詳細

SAS-file の SAS ファイルがプロシジャ入力ライブラリに存在しない場合、PROC DATASETS は SAVE ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを指定します。

SAVE ステートメントは SAS データセットを削除する際に、これらのデータセットと関連付けられているインデックスも削除します。(削除対象の SAS データセットに ALTER=パスワードが割り当てられている場合、その ALTER=パスワードを SAS データセットを削除するために指定する必要があります。)

注意:

RUN グループのサブミット時に、ライブラリとライブラリメンバは即座に削除されます。削除操作は、開始前に確認を求められません。SAVE ステートメントは、1 つの操作で多数の SAS ファイルを削除します。どの種類の SAS ファイルが保存され、どの種類が削除されるかについて、MEMTYPE=オプションはどのように影響を与えるかを理解していることを確認してください。

SAVE ステートメントを世代グループと使用する場合、SAVE ステートメントは基本バージョンとすべての履歴バージョンを単位として扱います。特定のバージョンを保存することはできません。

SELECT ステートメント

コピーする SAS ファイルを選択します。

制限事項: SELECT ステートメントは、COPY ステートメントの後に指定する必要があります

SELECT ステートメントは、EXCLUDE ステートメントで同じ COPY ステップに記述できません

例: “例 2: SAS ファイルの操作” (548 ページ)

構文

SELECT *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*> <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

必須引数

SAS-file(s)
コピーする SAS ファイルを 1 つ以上指定します。指定するすべての SAS ファイルが、COPY ステートメントの IN=オプションで指定されるライブラリ参照名によって参照されるデータライブラリ内にある必要があります。SAS ファイルに世代グループが含まれる場合、SELECT ステートメントでは特定のバージョンを選択できないため、すべての世代がコピーされます。

オプション引数

ALTER=*alter-password*
ライブラリ間で移動中の変更保護されている SAS ファイルに対する変更パスワードを与えます。SAS ファイルを SAS ライブラリから移動し削除しているため、変更アクセス権限が必要になります。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。

参照項目 “AES 暗号化データセットの追加” (463 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバの種類に制限します。オプションは、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に使用します。

別名	MTYPE=と MT=
デフォルト	PROC DATASETS ステートメント、COPY ステートメント、または SELECT ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。
参照項目	“SAS ファイルのコピー/移動時のメンバの種類の指定” (487 ページ) および“処理するメンバの種類の制限” (442 ページ)
例	“例 2: SAS ファイルの操作” (548 ページ)

詳細

複数の同じ名前のファイルの選択

SELECT ステートメントで複数の SAS ファイルをリストするためのショートカットを使用できます。

表記	意味
x1–xn	ファイル X1 から Xn を指定します。番号は連続している必要があります。
x:	文字 X で始まるすべてのファイルを指定します。

SUSPEND ステートメント

監査ファイルへのイベントの記録を中断しますが、監査ファイルは削除しません。

制限事項: SUSPEND ステートメントは、AUDIT RUN グループ内で指定する必要があります。

例: “例 11: 監査ファイルの初期化” (578 ページ)

構文

SUSPEND;

詳細

SUSPEND、RESUME、TERMINIATE、USER_VAR、LOG などのその他の監査関連ステートメントは、INITIATE ステートメントが送信されるまで監査ファイルに有効になりません。詳細については、“INITIATE ステートメント” (506 ページ)を参照してください。

SUSPEND ステートメントには、LOG ステートメントの ADMIN_IMAGE=YES オプションが必要です。このオプションは、管理イベントが監査ファイルに書き込まれるかどうか(SUSPEND アクションと RESUME アクション)を指定します。詳細については、“LOG ステートメント” (507 ページ)を参照してください。

TERMINATE ステートメント

イベントの記録を終了し、監査ファイルを削除します。

制限事項: TERMINATE ステートメントは、AUDIT RUN グループ内で指定する必要があります。

例: “例 11: 監査ファイルの初期化” (578 ページ)

構文

TERMINATE;

詳細

TERMINIATE、SUSPEND、RESUME、USER_VAR、LOG などのその他の監査関連ステートメントは、INITIATE ステートメントが送信されるまで監査ファイルに有効になりません。詳細については、“INITIATE ステートメント” (506 ページ)を参照してください。

USER_VAR ステートメント

オブザベーションへの更新のたびに任意変数が監査ファイルに書き込まれるように定義します。USER_VAR を使用する場合、INITIATE ステートメントの後に指定してください。

制限事項: USER_VAR ステートメントは、AUDIT RUN グループ内で指定する必要があります。

USER_VAR ステートメントはオプションです。USER_VAR ステートメントを指定する場合は、適切な監査ファイルのために INITIATE ステートメントの直後に指定する必要があります。

例: “例 11: 監査ファイルの初期化” (578 ページ)

構文

USER_VAR *variable-name-1* <$> <*length*> <LABEL='*variable-label*' > <*variable-name-2* <$> <*length*> <LABEL='*variable-label*' > …>;

必須引数

variable-name
変数の名前です。

オプション引数

$
変数が文字変数であることを示します。

length
変数の長さを指定します。

デフォルト 8

LABEL='*variable-label*'
変数のラベルを指定します。

XATTR ADD ステートメント

拡張属性を変数またはデータセットに追加します。

制限事項: XATTR ADD ステートメントは、MODIFY RUN グループ内で指定する必要があります
世代データセットは拡張属性をサポートしません。

サポート: 基本エンジンのみ

注: 拡張属性には、数値または文字値を指定できます。
文字値の中に空白スペースがある場合は、欠損値とみなされます。欠損数値も使用できます。

例: “例 12: 拡張属性” (584 ページ)

構文

XATTR ADD DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 …>*;

または

XATTR ADD VAR *variable-name-1* (*attribute-name-1=attribute-value-1 < attribute-name-2=attribute-value-2 …>*) *<variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 …>*) >;

必須引数

文字値の場合、*attribute-value* を引用符で囲む必要があります(*"attribute-value"*)。

XATTR ADD DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*
拡張属性をデータセットに追加します。拡張属性がすでに存在する場合、エラーが返されます。

XATTR ADD VAR *variable-name-1* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*) *<variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*)>
拡張属性を変数に追加します。拡張属性がすでに存在する場合、エラーが返されます。

詳細

拡張属性は(名前, 値)ペアから構成されます。新しい属性を追加し、その属性がすでに存在する場合、エラーが SAS ログに書き込まれます。

XATTR DELETE ステートメント

すべての拡張属性を SAS ファイルから削除します。

制限事項: XATTR DELETE ステートメントは、MODIFY RUN グループ内で指定する必要があります

サポート: 基本エンジンのみ

例: “例 12: 拡張属性” (584 ページ)

構文

XATTR DELETE DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 …>*;

または

XATTR DELETE VAR *variable-name-1 (attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 …>) … <variable-name-2 (attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 …>)>*;

必須引数

文字値の場合、*attribute-value* を引用符で囲む必要があります(*"attribute-value"*)。

XATTR DELETE DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2> ...*
すべての拡張属性をデータセットから削除します。

XATTR DELETE VAR *variable-name-1* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*) <*variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*)>
すべての拡張属性を変数から削除します。

詳細

XATTR DELETE ステートメントを使用すると、すべてのデータセットおよび変数の拡張属性をファイルから削除できます。このコマンドを使用した後は、どの拡張属性も存在しなくなります。次の例では、すべての拡張属性をデータセットから削除します。

```
proc datasets lib=library_name nolist;
modify dataset_name;
xattr delete;
run;
quit;
```

XATTR OPTIONS ステートメント

拡張属性に必要なオプションを指定します。現在、SEGLEN=のみが有効なオプションです。

制限事項: XATTR OPTIONS ステートメントは、MODIFY RUN グループ内で指定する必要があります

サポート: 基本エンジンのみ

例: “例 12: 拡張属性” (584 ページ)

構文

XATTR OPTIONS <SEGLEN=*number-of-bytes*>;

必須引数

XATTR OPTIONS SEGLEN=*number-of-bytes*
文字属性値を保持するストレージ要素長を指定します。値は 1～32,760 バイトまでです。

デフォルト 256

XATTR REMOVE ステートメント

変数またはデータセットから拡張属性を削除します。

制限事項: XATTR REMOVE ステートメントは、MODIFY RUN グループ内で指定する必要があります

サポート: 基本エンジンのみ

例: “例 12: 拡張属性” (584 ページ)

構文

XATTR REMOVE DS *attribute-name(s)* ;

または

XATTR REMOVE VAR *variable-name-1* (*attribute-name(s)*) <*variable-name-2* (*attribute-name(s)* …)>;

必須引数

XATTR REMOVE DS *attribute-name(s)*
拡張属性をデータセットから削除します。

XATTR REMOVE VAR *variable-name-1* (*attribute-name(s)*) <*variable-name-2* (*attribute-name(s)*)>
拡張属性を変数から削除します。

詳細

作成した拡張属性をもう必要がなくなった場合、XATTR REMOVE ステートメントを使用するとその属性を変数またはデータセットから削除できます。XATTR REMOVE ステートメントによって削除されるのは、指定する拡張属性に限られます。

XATTR SET ステートメント

拡張属性を変数またはデータセットに更新または追加します。

制限事項: XATTR SET ステートメントは、MODIFY RUN グループ内で指定する必要があります
世代データセットは拡張属性をサポートしません。

サポート: 基本エンジンのみ

注: 文字値の中に空白スペースがある場合は、欠損値とみなされます。欠損数値も使用できます。

例: “例 12: 拡張属性” (584 ページ)

構文

XATTR SET DS *attribute-name-1=attribute-value-1* <*attribute-name-2="attribute-value-2"* …>;

または

XATTR SET VAR *variable-name-1* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 … >*) *<variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2> …*)>;

必須引数

文字値の場合、*attribute-value* を引用符で囲む必要があります(*"attribute-value"*)。

XATTR SET DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*
拡張属性をデータセットに更新または追加します。データセットに拡張属性が存在しない場合、拡張属性が追加されます。拡張属性が存在する場合、指定値によって更新されます。

XATTR SET VAR *variable-name-1* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*) *<variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*)>
拡張属性を変数に更新または追加します。変数と拡張属性の組み合わせが存在する場合、拡張属性が追加されます。組み合わせが存在する場合、拡張属性は指定値によって更新されます。

詳細

拡張属性が存在するかどうかわからない場合は、XATTR SET ステートメントを使用してください。拡張属性が存在する場合、その属性は更新されます。拡張属性が存在しない場合、拡張属性が追加されます。XATTR SET ステートメントは、変数またはデータセットに拡張属性が存在しない場合でも、その属性を定義します。XATTR ADD を使用するとき、その値を使用する既存の拡張属性がある場合、エラーが発生します。まだ存在しない拡張属性に XATTR UPDATE を使おうとしても、エラーが発生します。XATTR SET を使用すると、変数またはデータセットの拡張属性を定義できます。拡張属性が存在していなかった場合、これからは存在します。拡張属性が存在していた場合、新しい値が指定されます。

XATTR UPDATE ステートメント

拡張属性を変数またはデータセットに更新します。

制限事項: XATTR UPDATE ステートメントは、MODIFY RUN グループ内で指定する必要があります

サポート: 基本エンジンのみ

注: 文字値の中に空白スペースがある場合、欠損値とみなされます。欠損数値も使用できます。

例: “例 12: 拡張属性” (584 ページ)

構文

XATTR UPDATE DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 …>*;

または

XATTR UPDATE VAR *variable-name-1* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 …>*) *<variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 …>*)*>*;

必須引数

文字値の場合、*attribute-value* を引用符で囲む必要があります(*"attribute-value"*)。

XATTR UPDATE DS ***attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>***
データセット内の拡張属性を更新します。拡張属性が存在しない場合、エラーが SAS ログに書き込まれます。

XATTR UPDATE VAR ***variable-name-1 (attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>) <variable-name-2 (attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>)>***
変数の拡張属性を更新します。変数と拡張属性の組み合わせが見つからない場合、エラーが SAS ログに書き込まれます。

詳細

既存の拡張属性に変更を加えるには、XATTR UPDATE ステートメントを使用します。存在しない拡張属性を更新しようと試みると、エラーが SAS ログに書き込まれます。

結果: DATASETS プロシジャ

SAS ログとしてのディレクトリリスト

PROC DATASETS ステートメントは、NOLIST オプションが選択されない限り プロシジャ入力ライブラリの SAS ファイルをリストします。NOLIST オプションは、ログに移動するプロシジャ結果を作成しません。MEMTYPE= オプションを指定すると、指定された種類のみリストされます。DETAILS オプションを指定すると、PROC DATASETS は **`Obs, Entries or Indexes`**、**`Vars`**、および **`Label`** という追加情報列を出力します。

SAS 出力としてのディレクトリリスト

CONTENTS ステートメントは、DIRECTORY オプションを使用する場合、または DATA=_ALL_を指定する場合、プロシジャ入力ライブラリのディレクトリをリストします。

ディレクトリのみ必要な場合、NODS オプションと、DATA=オプションの_ALL_キーワードを使用します。NODS オプションは、SAS データセットを記述しません。ディレクトリのみ出力されます。

注: CONTENTS ステートメントは、ディレクトリを出力データセットに含めません。NODS オプションを使用して出力データセットを作成しようとすると、空の出力データセトを受け取ります。SQL プロシジャを使用して、SAS ライブラリに関する情報を含む SAS データセットを作成します。

注: ODS RTF 出力先を指定する場合、PROC DATASETS 出力は SAS ログと ODS 出力領域の両方に移動します。NOLIST オプションはこのいずれにも出力しませ

ん。SAS ログのみの出力を確認するには、メンバディレクトリを除外として指定し、ODS EXCLUDE ステートメントを使用します。

プロシジャの出力

CONTENTS ステートメント

プロシジャ出力を生成する PROC DATASETS の唯一のステートメントは、CONTENTS ステートメントです。このセクションでは、GROUP データセットに対する CONTENTS ステートメントからの出力を示します。出力は次のとおりです。

説明が必要な出力の項目についてのみ説明します。

データセット属性

次の出力で表示される選択フィールドの説明です。

Member Type
ライブラリメンバの種類(DATA または VIEW)です。

Protection
SAS データセットの READ、WRITE、ALTER のパスワード保護がされているかどうかを示します。

Data Set Type
特殊データセットの種類(CORR、COV、SSPC、EST、FACTOR など)を指定します(存在する場合)。

Observations
ファイル内の現在のオブザベーションの合計数です。たとえば、非常に大きなデータセットで、オブザベーションの数が倍精度浮動小数点で表現可能な最大整数値を超えている場合、カウントは欠損値として表示されます。

Deleted Observations
削除がマーク付けされているオブザベーションの数です。これらのオブザベーションは、**Observations** フィールドに表示されているオブザベーションの合計数に含まれません。非常に大きなデータセットで、削除されたオブザベーションの数が倍精度浮動整数で保存可能な数を超えている場合、カウントは欠損値として表示されます。また、COMPRESS=YES オプションを REUSE=YES オプションと POINTOBS=NO オプションのいずれか、または両方を使用する場合、**Deleted Observations** には欠損値が表示されます。

Compressed
データセットが圧縮されるかどうかを示します。データセットが圧縮される場合、出力には追加項目、**Reuse Space** が(値 YES または NO とともに)含まれます。この項目は、オブザベーションの削除時に利用可能になるスペースを再利用するかどうかを示します。

Sorted
データセットが並べ替えられるかどうかを示します。データセットを PROC SORT、PROC SQL で並べ替える場合、または並べ替え情報を SORTEDBY=データセットオプションで指定する場合、ここでは値 YES が表示され、出力に追加セクションがあります。詳細については、“並べ替え情報” (532 ページ) を参照してください。

Data Representation
データがコンピュータアーキテクチャまたは動作環境で表現される形式です。たとえば IBM PC では、文字データは ASCII エンコーディングとバイトスワップ整数によって表現されます。ネイティブなデータ表現は、データ表現がファイルにアクセスしている CPU と比較される環境を参照します。たとえば Windows データ表現のファイルは、Windows 動作環境がネイティブです。

Encoding
エンコーディング値です。エンコーディングは、コンピュータが使用可能なコードポイントと呼ばれる数値にマッピングされた文字(通常の文字、表語文字、数字、句読点、記号、コントロール文字など)のセットです。コードポイントは、エンコーディングメソッドを適用する際に文字セットの文字に割り当てられます。

アウトプット 16.2 *GROUP データセットの属性*

The DATASETS Procedure

Data Set Name	HEALTH.GROUP	Observations	148
Member Type	DATA	Variables	11
Engine	V9	Indexes	0
Created	09/03/2014 09:35:02	Observation Length	96
Last Modified	09/03/2014 09:35:02	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

エンジンおよび動作環境依存情報

CONTENTS ステートメントは、動作環境固有情報とエンジン固有情報を生成します。この情報は、動作環境によって異なります。次の出力は、Windows 動作環境のものです。

アウトプット 16.3 GROUP データセットのエンジン/ホスト依存情報

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	4
First Data Page	1
Max Obs per Page	84
Obs in First Data Page	63
Index File Page Size	4096
Number of Index File Pages	2
Number of Data Set Repairs	0
Filename	c:\procdatasets1\health\group.sas7bdat
Release Created	9.0301M0
Host Created	W32_7PRO

変数と属性のアルファベット順リスト

次の出力で選択されている列の説明です。

#
オブザベーションの各変数の論理位置です。この数字は、変数の定義時に変数に割り当てられます。

Variable
各変数の名前です。デフォルトで、変数はアルファベット順に表示されます。

注: 変数名が X1、X10、X2 の実際の照合順序ではなく、X1、X2、X10 の順序で表示されるように並べ替えられます。アンダースコアと数字を含む変数名は、非標準の並べ替え順序で表示されることがあります。たとえば、P25 と P75 は P2_5 の前に表示されます。

Type
変数の種類、文字または数値を指定します。

Len
変数の長さを指定します。これは、SAS データセット内の変数の値をそれぞれ保存するために使用されるバイト数です。

Transcode
文字変数がトランスコードされるかどうかを指定します。属性が NO の場合、トランスコードは実行されません。デフォルトで、必要な場合に文字変数はトランスコードされます。トランスコードの詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

注: SAS データセット内の変数に出力形式、入力形式、ラベルが関連付けられていない場合、またはすべての変数が TRANSCODE=YES に設定されている場合、属性の列には NOT が表示されます。

アウトプット 16.4 GROUP データセットの変数と属性のリスト

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Informat
9	BIRTH	Num	8		
4	CITY	Char	15		
3	FNAME	Char	15		
10	HIRED	Num	8	DATE7.	DATE7.
11	HPHONE	Char	12		
1	IDNUM	Char	4		
7	JOBCODE	Char	4		
2	LNAME	Char	15		
8	SALARY	Num	8	COMMA8.	
6	SEX	Char	2		
5	STATE	Char	3		

インデックスと属性のアルファベット順リスト

次の出力に表示されるセクションは、データセットとインデックスが関連付けられている場合にのみ表示されます。

#
各インデックスの数を示します。インデックスは定義されたとおりに連続して番号が付けられます。

Index
各インデックスの名前を表示します。単一インデックスの場合、インデックスの名前はデータセットの変数と同じです。

Unique Option
インデックスに一意の値を含める必要があるかどうかを示します。列に YES が含まれている場合、インデックス変数の値の組み合わせは各オブザベーションに対して一意です。

Nomiss Option
インデックスがすべてのインデックス変数に対する欠損値を除外するかどうかを示します。列に YES が含まれている場合、インデックスにはすべてのインデックス変数に対する欠損値を含むオブザベーションは含まれていません。

# of Unique Values
インデックス内の一意の値の数を示します。

Variables
複合インデックス内の変数を指定します。

アウトプット 16.5 *GROUP データセットのインデックスと属性のリスト*

Alphabetic List of Indexes and Attributes					
#	Index	Unique Option	NoMiss Option	# of Unique Values	Variables
1	vital	YES	YES	148	BIRTH SALARY

並べ替え情報

次の出力に表示されているセクションは、**Sorted** フィールドの値が YES の場合にのみ表示されます。

Sortedby
データの現在の並べ替え方法を示します。このフィールドには、PROC SORT の BY ステートメントで使用する変数とオプション、PROC SQL の列名、SORTEDBY=オプションで指定する値のいずれかが含まれます。

Validated
データが PROC SORT または SORTEDBY を使用して並べ替えられたかどうかを示します。PROC SORT または PROC SQL がデータセットを並べ替えた場合、値は YES です。SORTEDBY=データセットオプションでソートインジケータを割り当てた場合、値は NO となります。

Character Set
データの並べ替えに使用される文字セットです。このフィールドの値には、ASCII、EBCDIC、PASCII のいずれかが可能です。

Collating Sequence
データセットの並べ替えに使用される照合順序です。変換テーブル名、エンコーディング値、またはデータセットが言語ソートされる場合は LINGUISTIC が可能です。文字セットと異なる照合順序を指定しない場合、このフィールドは表示されません。

データセットが言語ソートされると、ロケール、照合形式などの追加の言語照合順序情報が **Collating Sequence** の後に表示されます。言語照合に指定可能な照合ルールのリストです。

Sort Option
データセットの並べ替え時に PROC SORT が NODUPKEY オプションを使用したかどうかを示します。このフィールドは、PROC SORT ステートメントでこのオプションを使用しなかった場合は表示されません(非表示)。

アウトプット 16.6 *GROUP データセット並べ替え情報*

Sort Information	
Sortedby	LNAME
Validated	NO
Character Set	ANSI

PROC DATASETS と ODS (Output Delivery System)

ほとんどの SAS プロシジャは、メッセージを SAS ログに送信し、プロシジャ結果を出力に送信します。PROC DATASETS は、プロシジャ結果を SAS ログとプロシジャ出力ファイルの両方に送信するため、固有のものです。ODS へのインターフェイスが作成された際に、すべてのプロシジャ結果(ログとプロシジャ出力ファイルの両方からの)が ODS で利用可能になる必要があるとされました。この機能を実装し、前のリリースとの互換性を保持するため、ODS へのインターフェイスは通常のインターフェイスと多少異なっている必要がありました。

デフォルトで、PROC DATASETS ステートメントはそれ自体で 2 つの出力オブジェクト、Members と Directory を生成します。これらのオブジェクトは、SAS ログに送られます。CONTENTS ステートメントは、デフォルトで 3 つの出力オブジェクト、Attributes、EngineHost、Variables を生成します。(さまざまなオプションの使用により、その他の出力オブジェクトが追加されます)。これらのオブジェクトは、プロシジャ出力ファイルに送られます。ODS 出力先(HTML、RTF、PRINTER など)を開いている場合、これらのオブジェクトはすべてデフォルトでその出力先に送られます。

その他のプロシジャに対して実行するのと同じように、ODS SELECT ステートメントと ODS EXCLUDE ステートメントを使用して、オブジェクトとオブジェクトの出力先を制御できます。ただし、PROC DATASETS と ODS 間の固有のインターフェイスのため、ODS SELECT ステートメントまたは ODS EXCLUDE ステートメントでキーワード LISTING を使用する際は、ログとリストの両方に影響します。

ODS テーブル名

PROC DATASETS と PROC CONTENTS は、作成するテーブルにそれぞれ名前を割り当てます。これらの名前を使用して、Output Delivery System (ODS)を使用してテーブルを選択し、出力データセットを作成する際にそのテーブルを参照できます。

PROC CONTENTS は、CONTENTS ステートメントを使用した PROC DATASETS と同じ ODS テーブルを生成します。

表 16.8 *CONTENTS ステートメントなしで DATASETS プロシジャによって生成される ODS テーブル*

ODS テーブル	説明	テーブルの生成
Directory	一般ライブラリ情報	NOLIST オプションを指定しない場合
Members	ライブラリメンバ情報	NOLIST オプションを指定しない場合

表 16.9 *CONTENTS ステートメントで PROC CONTENTS と PROC DATASETS によって生成される ODS テーブル名*

ODS テーブル	説明	テーブルの生成
属性	データセット属性	SHORT オプションを指定しない場合
Directory	一般ライブラリ情報	DATA=<*libref*>_ALL_または DIRECTORY オプションを指定する場合 *
EngineHost	エンジンおよび動作環境情報	SHORT オプションを指定しない場合

ODS テーブル	説明	テーブルの生成
IntegrityConstraints	一貫性制約の詳細リスト	データセットに一貫性制約があり、SHORT オプションを指定しない場合
IntegrityConstraintsShort	一貫性制約の簡潔リスト	データセットに一貫性制約があり、SHORT オプションを指定する場合
Indexes	インデックスの詳細リスト	データセットがインデックス化され、SHORT オプションを指定しない場合
IndexesShort	インデックスの簡潔リスト	データセットがインデックス化され、SHORT オプションを指定する場合
Members	ライブラリメンバ情報	DATA=<*libref*>_ALL_または DIRECTORY オプションを指定する場合 *
位置	データセットの論理位置別の変数の詳細リスト	VARNUM オプションを指定し、SHORT オプションを指定しない場合
PositionShort	データセットの論理位置別の変数の簡潔リスト	VARNUM オプションと SHORT オプションを指定する場合
Sortedby	詳細並べ替え情報	データセットが並べ替えられ、SHORT オプションを指定しない場合
SortedbyShort	簡潔並べ替え情報	データセットが並べ替えられ、SHORT オプションを指定する場合
Variables	アルファベット順の変数の詳細リスト	SHORT オプションを指定しない場合
VariablesShort	アルファベット順の変数の簡潔リスト	SHORT オプションを指定する場合

* PROC DATASETS の場合、NOLIST オプションと、DIRECTORY オプションまたは DATA=<*libref*>_ALL_のいずれが指定されると、NOLIST オプションは無視されます。

出力データセット

CONTENTS ステートメント

CONTENTS ステートメントは、出力データセットを生成する DATASETS プロシジャの唯一のステートメントです。

OUT= データセット

CONTENTS ステートメントの OUT=オプションは、出力データセットを作成します。各 DATA=データセットの各変数には、OUT=データセットにオブザベーションが 1 つあります。出力データセットの変数は次のとおりです。

CHARSET
: データセットの並べ替えに使用される文字セット。値は、ASCII、EBCDIC、PASCII のいずれかです。データセットにソートインジケータがない場合、ブランクが表示されます。

COLLATE
データセットの並べ替えに使用される照合順序。入力データセットのソートインジケータに照合順序が含まれていない場合、ブランクが表示されます。

COMPRESS
データセットが圧縮されるかどうかを示します。

CRDATE
データセットが作成された日付です。

DELOBS
データセットの削除がマーク付けられているオブザベーションの数です。(オブザベーションは削除のマーク付け設定が可能ですが、SAS/FSP ソフトウェアの FSEDIT プロシジャを使用するときに実際に削除されます)。

ENCRYPT
データセットが暗号化されるかどうかを示します。

ENGINE
データセット間の読み取りと書き込みに使用される方法の名前です。

FLAGS
SQL ビューの変数が保護されているかどうか(**P**)、派生変数に影響(C)するかどうかを示します。

P
変数が保護されていることを示します。変数の値は表示できますが、更新できません。

C
変数が派生変数に影響するかどうかを示します。

P または **C** が SQL ビューに適用されない場合、またはデータセットビューである場合、FLAG の値はブランクとなります。

FORMAT
変数の出力形式。出力形式と変数を関連付けない場合、FORMAT の値はブランクとなります。

FORMATD
出力形式と変数を関連付けるときに指定する小数点以下の桁数。出力形式の小数点以下桁数を指定しない場合、FORMATD の値は 0 となります。

FORMATL
出力形式の長さ。出力形式と変数を関連付ける際に出力形式の長さを指定する場合、指定する長さは FORMATL の値です。出力形式と変数を関連付ける際に出力形式の長さを指定しない場合、FMTLEN オプションを使用する場合は FORMATL の値が出力形式のデフォルト長、FMTLEN オプションを使用しない場合は 0 となります。

GENMAX
世代グループのバージョンの最大数。

GENNEXT
世代グループの次の世代番号。

GENNUM
バージョン番号。

IDXCOUNT
データセットのインデックスの数。

IDXUSAGE
インデックス内の変数の使用。可能な値は次のとおりです。

NONE
変数はインデックスの一部ではありません。

SIMPLE
変数に単一インデックスがあります。その他の変数はインデックスに含まれていません。

COMPOSITE
変数は複合インデックスの一部です。

BOTH
変数に単一インデックスがあります。変数は複合インデックスの一部です。

INFORMAT
変数の入力形式。入力形式と変数を関連付けない場合、値はブランクとなります。

INFORMD
入力形式と変数を関連付けるときに指定する小数点以下の桁数です。入力形式と変数を関連付ける際に小数点以下の桁数を指定しない場合、値は 0 です。

INFORML
入力形式の長さ。入力形式と変数を関連付ける際に入力形式の長さを指定する場合、指定する長さは INFORML の値です。入力形式と変数を関連付ける際に入力形式の長さを指定しない場合、FMTLEN オプションを使用する場合は INFORML の値が入力形式のデフォルト長、FMTLEN オプションを使用しない場合は 0 となります。

JUST
位置調整(0=左、1=右)。

LABEL
変数ラベル(指定されていない場合はブランク)。

LENGTH
変数長。

LIBNAME
データライブラリに使用されるライブラリ参照名。

MEMLABEL
この SAS データセットのラベル(ラベルがない場合はブランク)。

MEMNAME
変数を含む SAS データセット。

MEMTYPE
ライブラリメンバの種類 (DATA または VIEW)

MODATE
データセットが最後に変更された日付。

NAME
変数名。

NOBS
データセットのオブザベーションの数。

NODUPKEY
NODUPKEY オプションが PROC SORT ステートメントで入力データセットの並べ替えに使用されたかどうかを示します。

NPOS
データセットの変数の最初の文字の物理位置。

POINTOBS
データセットがオブザベーションによって処理可能かどうかを示します。

PROTECT
保護レベルの最初の文字。PROTECT の値は、次のうち 1 つ以上となります。

A
データセットが変更保護されていることを示します。

R
データセットが読み取り保護されていることを示します。

W
データセットが書き込み保護されていることを示します。

REUSE
オブザベーションが圧縮データセットから削除されたときに利用可能になるスペースを再利用する必要があるかどうかを示します。データセットが圧縮されていない場合、REUSE 変数の値は NO となります。

SORTED
値は、入力データセットの並べ替えの特性によって異なります。可能な値は次のとおりです。

. (ピリオド)
並べ替えられていない場合。

0
並べ替えられているが、認証されていない場合。

1
並べ替えられ、検証されている場合。

SORTEDBY
値は、並べ替えにおける変数の役割によって異なります。可能な値は次のとおりです。

. (ピリオド)
変数が入力データセットの並べ替えに使用されなかった場合。

n
n は、並べ替えにおける変数の位置を示す整数です。*n* の負の値は、データセットが変数の降順で並べ替えられることを示します。

TRANSCODE
変数がトランスコードされるかどうかを指定します。

TYPE
変数の種類 (1=数値、2=文字)。

TYPEMEM
特殊データセットの種類 (TYPE=値が指定されていない場合はブランク)。

VARNUM
データセットの変数番号。変数は表示される順序で番号が付けられます。

出力データセットは、変数 LIBNAME と MEMNAME によって並べ替えられます。

注: 変数名は、値 X1、X2、X10 が X1、X10、X2 の実際の照合順序ではなく、X1、X2、X10 の順序でリストされるように並べ替えられます。そのため、後続のステップで MEMNAME で BY ステートメントを使用する場合、出力データセットで最初に PROC SORT ステップを実行します。BY ステートメントでは、NOTSORTED オプションを使用することもできます。

次は、GROUP データセットから作成された出力データセットの例です。“例 5: SAS データセットの説明” (557 ページ) および “プロシジャの出力” (528 ページ)で示されます。

HEALTH.GRPOUT のサイズのため、次の出力は 5 つのセクションになります。

アウトプット 16.7 *出力データセットの例 — セクション 1*

Obs	LIBNAME	MEMNAME	MEMLABEL	TYPEMEM	NAME	TYPE	LENGTH	VARNUM
1	HEALTH	GROUP	Test Subjects		BIRTH	1	8	9
2	HEALTH	GROUP	Test Subjects		CITY	2	15	4
3	HEALTH	GROUP	Test Subjects		FNAME	2	15	3
4	HEALTH	GROUP	Test Subjects		HIRED	1	8	10
5	HEALTH	GROUP	Test Subjects		HPHONE	2	12	11
6	HEALTH	GROUP	Test Subjects		IDNUM	2	4	1
7	HEALTH	GROUP	Test Subjects		JOBCODE	2	4	7
8	HEALTH	GROUP	Test Subjects		LNAME	2	15	2
9	HEALTH	GROUP	Test Subjects		SALARY	1	8	8
10	HEALTH	GROUP	Test Subjects		SEX	2	2	6
11	HEALTH	GROUP	Test Subjects		STATE	2	3	5

アウトプット 16.8 出力データセットの例 — セクション 2

LABEL	FORMAT	FORMATL	FORMATD	INFORMAT	INFORML
	DATE	7	0	DATE	7
		0	0		0
		0	0		0
	DATE	7	0	DATE	7
		0	0		0
		0	0		0
		0	0		0
		0	0		0
current salary excluding bonus	COMMA	8	0		0
		0	0		0
		0	0		0

アウトプット 16.9 *出力データセットの例 — セクション 3*

INFORMD	JUST	NPOS	NOBS	ENGINE	CRDATE	MODATE	DELOBS
0	1	8	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	0	58	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	0	43	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	1	16	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	0	82	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	0	24	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	0	78	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	0	28	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	1	0	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	0	76	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0
0	0	73	148	V9	12SEP07:13:57:50	12SEP07:16:01:16	0

アウトプット 16.10 出力データセットの例 — セクション 4

IDXUSAGE	MEMTYPE	IDXCOUNT	PROTECT	FLAGS	COMPRESS	REUSE	SORTED	SORTEDBY
COMPOSITE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	1
COMPOSITE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	.
NONE	DATA	1	R--	---	NO	NO	0	.

アウトプット 16.11 出力データセットの例 — セクション 5

CHARSET	COLLATE	NODUPKEY	NODUPREC	ENCRYPT	POINTOBS	GENMAX	GENNUM	GENNEXT
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.
ANSI		NO	NO	NO	YES	0	.	.

注: CONTENTS 出力を処理のため ODS データセットに取得する方法の詳細については、“例 8: ODS 出力 ” (564 ページ)を参照してください。

OUT2=データセット

CONTENTS ステートメントの OUT2=オプションは、インデックスと一貫性制約に関する情報を含む出力データセットを作成します。出力データセットの変数は次のとおりです。

IC_OWN
インデックスが一貫性制約によって所有される場合、YES を含みます。

INACTIVE
一貫性制約が非アクティブの場合、YES を含みます。

LIBNAME
データライブラリに使用されるライブラリ参照名。

MEMNAME
変数を含む SAS データセット。

MG
IC CREATE ステートメントの MESSAGE=の値(使用される場合)。

MSGTYPE
一貫性制約に違反せず、メッセージを指定しない限り、値はブランクです。

NAME
インデックスまたは一貫性制約の名前。

NOMISS
NOMISS オプションがインデックスに定義されている場合、YES を含みます。

NUMVALS
インデックス内の重複しない値の数(パーセント点に対して表示)。

NUMVARS
インデックスまたは一貫性制約に関連する変数の数。

ONDELETE
外部キー一貫性制約に対し、該当する場合(IC CREATE ステートメントの ON DELETE オプション)は RESTRICT または SET NULL を含みます。

ONUPDATE
外部キー一貫性制約に対し、該当する場合(IC CREATE ステートメントの ON UPDATE オプション)は RESTRICT または SET NULL を含みます。

RECREATE
インデックスまたは一貫性制約の再作成に必要な SAS ステートメント。

REFERENCE
外部キー一貫性制約に対し、参照されたデータセットの名前を含みます。

TYPE
種類。インデックスの場合、値は“Index”で、一貫性制約の場合、値は一貫性制約の値(Not Null、Check、Primary Key など)です。

UNIQUE
UNIQUE オプションがインデックスに定義されている場合、YES を含みます。

UPERC
最終更新から更新されたインデックスのパーセント(パーセント点に対して表示)。

UPERCMX
更新をトリガするインデックス更新のパーセント(パーセント点に対して表示)。

WHERE
一貫性制約のチェックに対し、WHERE ステートメントを含みます。

例: DATASETS プロシジャ

例 1: データセットのすべてのラベルと出力形式の削除

要素: MODIFY ステートメントオプション
ATTRIB
CONTENTS ステートメント

他の要素: PROC CONTENTS

詳細

この例では、次のアクションを行います。

- システムオプションを設定する
- ユーザー定義 FORMAT を作成する
- データセットを作成する
- データセットからラベルと形式を削除する
- PROC CONTENTS を使用して、ラベルと形式を含むまたは含まないデータを表示する

プログラム

```
options ls=79 nodate nocenter;
title;

proc format;
value clsfmt 1='Freshman' 2='Sophmore' 3='Junior' 4='Senior';
run;

data class;
format z clsfmt.;
label x='ID NUMBER'
y='AGE'
z='CLASS STATUS';
input x y z;
datalines;
1 20 4
2 18 1
;

proc contents data=class;
run;

proc datasets lib=work memtype=data;
modify class;
attrib _all_ label=' ';
attrib _all_ format=;
run;


contents data=class;
run;
quit;
```

プログラムの説明

次のシステムオプションを設定します。

```
options ls=79 nodate nocenter;
title;
```

ユーザー定義の出力形式を値は CLSFMT で作成します。

```
proc format;
value clsfmt 1='Freshman' 2='Sophmore' 3='Junior' 4='Senior';
run;
```

CLASS という名前のデータセットを作成します。 変数 Z に CLSFMT 出力形式を使用します。そして変数 X、Y と Z にラベルを作成します。

```
data class;
format z clsfmt.;
label x='ID NUMBER'
y='AGE'
z='CLASS STATUS';
input x y z;
datalines;
1 20 4
2 18 1
;
```

ラベルと出力形式を削除する前に、PROC CONTENTS を使用してデータセットの内容を確認します。

```
proc contents data=class;
run;
```

PROC DATASETS で、MODIFY ステートメントと ATTRIB オプションを使用してすべてのラベルと出力形式を削除します。

```
proc datasets lib=work memtype=data;
modify class;
attrib _all_ label=' ';
attrib _all_ format=;
run;
```

PROC DATASETS で CONTENTS ステートメントを使ってラベルや出力形式なしでデータセットの内容を確認します。

```
contents data=class;
run;
quit;
```

出力例

アウトプット 16.12 *ラベルと形式を含むクラスデータセットの CONTENTS プロシジャ*

The CONTENTS Procedure

The CONTENTS Procedure

Data Set Name	MYLIB.CLASS	Observations	2
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	11/14/2014 09:14:16	Observation Length	24
Last Modified	11/14/2014 09:14:16	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	2715
Obs in First Data Page	2
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\procdatasets\class.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
2	x	Num	8		ID NUMBER
3	y	Num	8		AGE
1	z	Num	8	CLSFMT.	CLASS STATUS

アウトプット 16.13 ラベルと形式を含まないクラスデータセットの CONTENTS ステートメント

The CONTENTS Statement

The DATASETS Procedure

Data Set Name	MYLIB.CLASS	Observations	2
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	11/14/2014 09:18:26	Observation Length	24
Last Modified	11/14/2014 09:18:26	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	2715
Obs in First Data Page	2
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\procdatasets\class.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
2	x	Num	8
3	y	Num	8
1	z	Num	8

例 2: SAS ファイルの操作

要素: PROC DATASETS ステートメントオプション
- DETAILS
- LIBRARY=

CHANGE ステートメント

COPY ステートメントオプション
- MEMTYPE
- MOVE
- OUT=

DELETE ステートメントオプション
- MEMTYPE=

EXCHANGE ステートメント

EXCLUDE ステートメント

SELECT ステートメントオプション
- MEMTYPE=

詳細

この例では、次のアクションを行います。

- SAS ファイル名の変更
- SAS ライブラリ間の SAS ファイルのコピー
- SAS ファイルの削除
- コピーする SAS ファイルの選択
- SAS ファイル名の入れ換え
- コピー操作からの SAS ファイルの除外

プログラム

```
options pagesize=60 linesize=80 nodate pageno=1 source;

LIBNAME dest1 'SAS-library-1';
LIBNAME dest2 'SAS-library-2';
LIBNAME health 'SAS-library-3';

proc datasets library=health details;

 delete tension a2(mt=catalog);
change a1=postdrug;
exchange weight=bodyfat;

 copy out=dest1 move memtype=view;

 select spdata;

 select etest1-etest5 / memtype=catalog;

 copy out=dest2;
exclude d: mlscl oxygen test2 vision weight;
quit;
```

プログラムの説明

プログラミングステートメントを SAS ログに書き出します。 SOURCE システムオプションがこれを遂行します。

```
options pagesize=60 linesize=80 nodate pageno=1 source;

LIBNAME dest1 'SAS-library-1';
LIBNAME dest2 'SAS-library-2';
LIBNAME health 'SAS-library-3';
```

プロシジャの入力ライブラリを指定して、ディレクトリに詳細を追加します。 DETAILS は次の行をディレクトリに追加出力します: **Obs**、**Entries** または **Indexes**、**Vars** および **Label**。MEMTYPE=オプションは PROC DATASETS ステートメントには表示されないので、処理にはすべてメンバの種類が使用できます。

```
proc datasets library=health details;
```

ライブラリの 2 つのファイルを削除して、SAS データセットとカタログの名前を変更します。 DELETE ステートメントは TENSION データセットとカタログ A2 を削除します。DELETE ステートメントのデフォルトのメンバの種類は DATA であるため、MT=CATALOG は A2 にのみ適用となります。CHANGE ステートメントはカタログ A1 の名前を POSTDRUG に変更します。EXCHANGE ステートメントは WEIGHT データセットと BODYFAT データセットの名前を入れ替えます。CHANGE または EXCHANGE ステートメントのデフォルトは MEMTYPE=ALL であるため、MEMTYPE=オプションは必要ありません。

```
 delete tension a2(mt=catalog);
change a1=postdrug;
exchange weight=bodyfat;
```

処理を 1 つのメンバの種類に制限し、データビューを削除し移動します。 MEMTYPE=VIEW は処理を SAS ビューに制限します。MOVE はこのステップの SELECT ステートメントに名前があるすべての SAS ビューを HEALTH データライブラリから削除し、そして Dest1 データライブラリに移動すると指定しています。

```
copy out=dest1 move memtype=view;
```

SAS ビュー SPDATA を HEALTH データライブラリから Dest1 データライブラリに移動します。

```
select spdata;
```

カタログを他のデータライブラリに移動します。 SELECT ステートメントでは Etest1 から Etest5 までのカタログを Health データライブラリから Dest1 データライブラリに移動すると指定しています。MEMTYPE=CATALOG は COPY ステートメントの MEMTYPE=VIEW オプションを無効にします。

```
select etest1-etest5 / memtype=catalog;
```

指定の条件に該当するすべてのファイルを処理から除外します。 EXCLUDE ステートメントは文字 D で始まるすべての SAS ファイルと他にリストされているファイルをコピー操作から除外します。HEALTH データライブラリにある残りの SAS ファイルはすべて Dest2 データライブラリにコピーされます。

```
 copy out=dest2;
exclude d: mlscl oxygen test2 vision weight;
quit;
```

ログの例

ログ 16.2 *Dest1 の SAS ログ*

```
117 options pagesize=60 linesize=80 nodate pageno=1 source;
118 LIBNAME dest1 'SAS-library-1';
NOTE: Libref DEST1 was successfully assigned as follows:
Engine: V9
Physical Name: SAS-library-1\dest1
119 LIBNAME dest2 'SAS-library-2';
NOTE: Libref DEST2 was successfully assigned as follows:
Engine: V9
Physical Name: SAS-library-2\dest2
120 LIBNAME health 'SAS-library-3';
NOTE: Libref HEALTH was successfully assigned as follows:
Engine: V9
Physical Name: SAS-library-3\health


121 proc datasets library=health details;
Directory

Libref HEALTH
Engine V9
Physical Name \myfiles\health
Filename \myfiles\health



Member Obs, Entries
# Name Type or Indexes Vars Label

1 A1 CATALOG 23
2 ALL DATA 23 17
3 BODYFAT DATA 1 2
4 CONFOUND DATA 8 4
5 CORONARY DATA 39 4
6 DRUG1 DATA 6 2 JAN2005 DATA
7 DRUG2 DATA 13 2 MAY2005 DATA
8 DRUG3 DATA 11 2 JUL2005 DATA
9 DRUG4 DATA 7 2 JAN2002 DATA
10 DRUG5 DATA 1 2 JUL2002 DATA
11 ETEST1 CATALOG 1
12 ETEST2 CATALOG 1
13 ETEST3 CATALOG 1
14 ETEST4 CATALOG 1
15 ETEST5 CATALOG 1
16 ETESTS CATALOG 1
17 FORMATS CATALOG 6
18 GROUP DATA 148 11
19 GRPOUT DATA 11 40
20 INFANT DATA 149 6
21 MLSCL DATA 32 4 Multiple Sclerosis Data
22 NAMES DATA 7 4
23 OXYGEN DATA 31 7
24 PERSONL DATA 148 11
25 PHARM DATA 6 3 Sugar Study
26 POINTS DATA 6 6
27 RESULTS DATA 10 5
28 SLEEP DATA 108 6
29 SPDATA VIEW . 2
30 TEST2 DATA 15 5
31 TRAIN DATA 7 2
32 VISION DATA 16 3
33 WEIGHT DATA 83 13 California Results
34 WGHT DATA 83 13
```

```
File
# Size Last Modified

1 62464 07Mar05:14:36:20
2 13312 12Sep07:13:57:48
3 5120 12Sep07:13:57:48
4 5120 12Sep07:13:57:48
5 5120 12Sep07:13:57:48
6 5120 12Sep07:13:57:49
7 5120 12Sep07:13:57:49
8 5120 12Sep07:13:57:49
9 5120 12Sep07:13:57:49
10 5120 12Sep07:13:57:49
11 17408 04Jan02:14:20:16
12 17408 04Jan02:14:20:16
13 17408 04Jan02:14:20:16
14 17408 04Jan02:14:20:16
15 17408 04Jan02:14:20:16
16 17408 24Mar05:16:12:20
17 17408 24Mar05:16:12:20
18 25600 12Sep07:13:57:50
19 17408 24Mar05:15:33:31
20 17408 12Sep07:13:57:51
21 5120 12Sep07:13:57:50
22 5120 12Sep07:13:57:50
23 9216 12Sep07:13:57:50
24 25600 12Sep07:13:57:51
25 5120 12Sep07:13:57:51
26 5120 12Sep07:13:57:51
27 5120 12Sep07:13:57:52
28 9216 12Sep07:13:57:52
29 5120 24Mar05:16:12:21
30 5120 12Sep07:13:57:52
31 5120 12Sep07:13:57:53
32 5120 12Sep07:13:57:53
33 13312 12Sep07:13:57:53
34 13312 12Sep07:13:57:53122 delete tension
a2(mt=catalog);
123 change a1=postdrug;
124 exchange weight=bodyfat;
NOTE: Changing the name HEALTH.A1 to HEALTH.POSTDRUG (memtype=CATALOG).
NOTE: Exchanging the names HEALTH.WEIGHT and HEALTH.BODYFAT (memtype=DATA).
125 copy out=dest1 move memtype=view;
126 select spdata;
127
128 select etest1-etest5 / memtype=catalog;
NOTE: Moving HEALTH.SPDATA to DEST1.SPDATA (memtype=VIEW).
NOTE: Moving HEALTH.ETEST1 to DEST1.ETEST1 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST2 to DEST1.ETEST2 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST3 to DEST1.ETEST3 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST4 to DEST1.ETEST4 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST5 to DEST1.ETEST5 (memtype=CATALOG).
```

```
129 copy out=dest2;
130 exclude d: mlscl oxygen test2 vision weight;
131 quit;

NOTE: Copying HEALTH.ALL to DEST2.ALL (memtype=DATA).
NOTE: There were 23 observations read from the data set HEALTH.ALL.
NOTE: The data set DEST2.ALL has 23 observations and 17 variables.
NOTE: Copying HEALTH.BODYFAT to DEST2.BODYFAT (memtype=DATA).
NOTE: There were 83 observations read from the data set HEALTH.BODYFAT.
NOTE: The data set DEST2.BODYFAT has 83 observations and 13 variables.
NOTE: Copying HEALTH.CONFOUND to DEST2.CONFOUND (memtype=DATA).
NOTE: There were 8 observations read from the data set HEALTH.CONFOUND.
NOTE: The data set DEST2.CONFOUND has 8 observations and 4 variables.
NOTE: Copying HEALTH.CORONARY to DEST2.CORONARY (memtype=DATA).
NOTE: There were 39 observations read from the data set HEALTH.CORONARY.
NOTE: The data set DEST2.CORONARY has 39 observations and 4 variables.
NOTE: Copying HEALTH.ETESTS to DEST2.ETESTS (memtype=CATALOG).
NOTE: Copying HEALTH.FORMATS to DEST2.FORMATS (memtype=CATALOG).
NOTE: Copying HEALTH.GROUP to DEST2.GROUP (memtype=DATA).
NOTE: There were 148 observations read from the data set HEALTH.GROUP.
NOTE: The data set DEST2.GROUP has 148 observations and 11 variables.
NOTE: Copying HEALTH.GRPOUT to DEST2.GRPOUT (memtype=DATA).
NOTE: There were 11 observations read from the data set HEALTH.GRPOUT.
NOTE: The data set DEST2.GRPOUT has 11 observations and 40 variables.
NOTE: Copying HEALTH.INFANT to DEST2.INFANT (memtype=DATA).
NOTE: There were 149 observations read from the data set HEALTH.INFANT.
NOTE: The data set DEST2.INFANT has 149 observations and 6 variables.
NOTE: Copying HEALTH.NAMES to DEST2.NAMES (memtype=DATA).
NOTE: There were 7 observations read from the data set HEALTH.NAMES.
NOTE: The data set DEST2.NAMES has 7 observations and 4 variables.
NOTE: Copying HEALTH.PERSONL to DEST2.PERSONL (memtype=DATA).
NOTE: There were 148 observations read from the data set HEALTH.PERSONL.
NOTE: The data set DEST2.PERSONL has 148 observations and 11 variables.
NOTE: Copying HEALTH.PHARM to DEST2.PHARM (memtype=DATA).
NOTE: There were 6 observations read from the data set HEALTH.PHARM.
NOTE: The data set DEST2.PHARM has 6 observations and 3 variables.
NOTE: Copying HEALTH.POINTS to DEST2.POINTS (memtype=DATA).
NOTE: There were 6 observations read from the data set HEALTH.POINTS.
NOTE: The data set DEST2.POINTS has 6 observations and 6 variables.
NOTE: Copying HEALTH.POSTDRUG to DEST2.POSTDRUG (memtype=CATALOG).
NOTE: Copying HEALTH.RESULTS to DEST2.RESULTS (memtype=DATA).
NOTE: There were 10 observations read from the data set HEALTH.RESULTS.
NOTE: The data set DEST2.RESULTS has 10 observations and 5 variables.
NOTE: Copying HEALTH.SLEEP to DEST2.SLEEP (memtype=DATA).
NOTE: There were 108 observations read from the data set HEALTH.SLEEP.
NOTE: The data set DEST2.SLEEP has 108 observations and 6 variables.
NOTE: Copying HEALTH.TRAIN to DEST2.TRAIN (memtype=DATA).
NOTE: There were 7 observations read from the data set HEALTH.TRAIN.
NOTE: The data set DEST2.TRAIN has 7 observations and 2 variables.
NOTE: Copying HEALTH.WGHT to DEST2.WGHT (memtype=DATA).
NOTE: There were 83 observations read from the data set HEALTH.WGHT.
NOTE: The data set DEST2.WGHT has 83 observations and 13 variables.
NOTE: PROCEDURE DATASETS used (Total process time):
real time 44.04 seconds
cpu time 0.60 seconds
```

例 3: SAS ファイルを削除せずに保存する

要素: SAVE ステートメントオプション
MEMTYPE=

詳細

この例では、SAVE ステートメントを使用して、一部の SAS ファイルを削除から保存し、その他の SAS ファイルを削除します。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1 source;

LIBNAME elder 'SAS-library';

proc datasets lib=elder;

 save chronic aging clinics / memtype=data;
run;
```

プログラムの説明

プログラミングステートメントを SAS ログに書き出します。SAS オプションの SOURCE はすべてのプログラミングステートメントをログに書き出します。

```
options pagesize=40 linesize=80 nodate pageno=1 source;

LIBNAME elder 'SAS-library';
```

処理するプロシジャ入力ライブラリを指定します。

```
proc datasets lib=elder;
```

データセット CHRONIC、AGING、と CLINICS を保存し、ELDER ライブラリのその他の SAS ファイル(すべての種類)をすべて削除します。 ELDER ライブラリには CLINICS という名前のカタログと同じ名前のデータセットが あるので、MEMTYPE=DATA が必要になります。

```
 save chronic aging clinics / memtype=data;
run;
```

ログの例

ログ 16.3 *Elder ライブラリの SAS ログ*

```
161
162 options pagesize=40 linesize=80 nodate pageno=1 source;
163 LIBNAME elder 'SAS-library';
NOTE: Libref ELDER was successfully assigned as follows:
Engine: V9
Physical Name: SAS-library\elder
164 LIBNAME green 'SAS-library';
NOTE: Libref GREEN was successfully assigned as follows:
Engine: V9
Physical Name: SAS-library\green
165 proc copy in=green out=elder;

NOTE: Copying GREEN.AGING to ELDER.AGING (memtype=DATA).
NOTE: There were 1 observations read from the data set GREEN.AGING.
NOTE: The data set ELDER.AGING has 1 observations and 2 variables.
NOTE: Copying GREEN.ALCOHOL to ELDER.ALCOHOL (memtype=DATA).
NOTE: There were 1 observations read from the data set GREEN.ALCOHOL.
NOTE: The data set ELDER.ALCOHOL has 1 observations and 2 variables.
NOTE: Copying GREEN.BACKPAIN to ELDER.BACKPAIN (memtype=DATA).
NOTE: There were 1 observations read from the data set GREEN.BACKPAIN.
NOTE: The data set ELDER.BACKPAIN has 1 observations and 2 variables.
NOTE: Copying GREEN.CHRONIC to ELDER.CHRONIC (memtype=DATA).
NOTE: There were 1 observations read from the data set GREEN.CHRONIC.
NOTE: The data set ELDER.CHRONIC has 1 observations and 2 variables.
NOTE: Copying GREEN.CLINICS to ELDER.CLINICS (memtype=CATALOG).
NOTE: Copying GREEN.CLINICS to ELDER.CLINICS (memtype=DATA).
NOTE: There were 1 observations read from the data set GREEN.CLINICS.
NOTE: The data set ELDER.CLINICS has 1 observations and 2 variables.
NOTE: Copying GREEN.DISEASE to ELDER.DISEASE (memtype=DATA).
NOTE: There were 1 observations read from the data set GREEN.DISEASE.
NOTE: The data set ELDER.DISEASE has 1 observations and 2 variables.
NOTE: Copying GREEN.GROWTH to ELDER.GROWTH (memtype=DATA).
NOTE: There were 1 observations read from the data set GREEN.GROWTH.
NOTE: The data set ELDER.GROWTH has 1 observations and 2 variables.
NOTE: Copying GREEN.HOSPITAL to ELDER.HOSPITAL (memtype=CATALOG).
NOTE: PROCEDURE COPY used (Total process time):
real time 2.42 seconds
cpu time 0.04 seconds


166 proc datasets lib=elder;
Directory

Libref ELDER
Engine V9
Physical Name \myfiles\elder
Filename \myfiles\elder


Member File
# Name Type Size Last Modified

1 AGING DATA 5120 12Sep07:15:52:52
2 ALCOHOL DATA 5120 12Sep07:15:52:52
3 BACKPAIN DATA 5120 12Sep07:15:52:53
4 CHRONIC DATA 5120 12Sep07:15:52:53
5 CLINICS CATALOG 17408 12Sep07:15:52:53
6 CLINICS DATA 5120 12Sep07:15:52:53
7 DISEASE DATA 5120 12Sep07:15:52:54
8 GROWTH DATA 5120 12Sep07:15:52:54
9 HOSPITAL CATALOG 17408 12Sep07:15:52:54
```

```
167 save chronic aging clinics / memtype=data;
168 run;

NOTE: Saving ELDER.CHRONIC (memtype=DATA).
NOTE: Saving ELDER.AGING (memtype=DATA).
NOTE: Saving ELDER.CLINICS (memtype=DATA).
NOTE: Deleting ELDER.ALCOHOL (memtype=DATA).
NOTE: Deleting ELDER.BACKPAIN (memtype=DATA).
NOTE: Deleting ELDER.CLINICS (memtype=CATALOG).
NOTE: Deleting ELDER.DISEASE (memtype=DATA).
NOTE: Deleting ELDER.GROWTH (memtype=DATA).
NOTE: Deleting ELDER.HOSPITAL (memtype=CATALOG).
```

例 4: SAS データセットの変更

要素:

PROC DATASETS ステートメントオプション
- NOLIST

FORMAT ステートメント

INDEX CREATE ステートメントオプション
- NOMISS
- UNIQUE

INFORMAT ステートメント

LABEL ステートメント

MODIFY ステートメントオプション
- LABEL=
- READ=
- SORTEDBY=

RENAME ステートメント

詳細

この例では、MODIFY ステートメントと、それに従属しているステートメントを使用して 2 つの SAS データセットを変更します。“例 5: SAS データセットの説明” (557 ページ) に GROUP データセットへの変更を示します。

この例には、次のアクションが含まれます。

- SAS ファイルの変更
- SAS データセットのラベル付け
- SAS データセットへの READ パスワードの追加
- SAS データセットの現在の並べ替え方法の表示
- SAS データセットのインデックスの作成
- SAS データセット内の変数への入力形式と出力形式の割り当て
- SAS データセット内の変数の名前変更
- SAS データセット内の変数のラベル付け

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1 source;
```

```
LIBNAME health
'SAS-library';

proc datasets library=health nolist;

 modify group (label='Test Subjects' read=green sortedby=lname);

 index create vital=(birth salary) / nomiss unique;

 informat birth date7.;
format birth date7.;

 label salary='current salary excluding bonus';

 modify oxygen;
rename oxygen=intake;
label intake='Intake Measurement';
quit;
```

プログラムの説明

プログラミングステートメントを SAS ログに書き出します。 SAS オプションの SOURCE はプログラミングステートメントをログに書き出します。

```
options pagesize=40 linesize=80 nodate pageno=1 source;

LIBNAME health
'SAS-library';
```

HEALTH を処理するプロシジャ入力ライブラリに指定します。 NOLIST は HEALTH データライブラリのディレクトリリストの表示を抑制します。

```
proc datasets library=health nolist;
```

データセットにラベルを追加し、読み取りパスワードを付与し、データの並べ替えを指定します。 LABEL=はデータセット GROUP にデータセットラベルを追加します。READ=は読み取りパスワード GREEN を付与します。パスワードは SAS ログでは XXX…と表示されます。SAS はファイルの変更保護を含まないレベルのパスワード保護が指定された場合は、警告メッセージを出します。SORTEDBY=はデータの並べ替え方法を指定します。

```
 modify group (label='Test Subjects' read=green sortedby=lname);
```

GROUP データセットの変数 BIRTH と SALARY に複合インデックス VITAL を作成します。 NOMISS は BIRTH と SALARY に欠損値があるオブザベーションをインデックスから除外します。UNIQUE は BIRTH と SALARY の値が一意の組み合わせであるオブザベーションのみでインデックスを作成すると指定します。

```
 index create vital=(birth salary) / nomiss unique;
```

変数 BIRTH に入力形式と出力形式をそれぞれ割り当てます。

```
 informat birth date7.;
format birth date7.;
```

変数 SALARY にラベルを割り当てます。

```
 label salary='current salary excluding bonus';
```

変数の名前を変更し、ラベルを割り当てます。 変数 OXYGEN を INTAKE に名称変更してデータセット OXYGEN を変更し、変数 INTAKE にラベルを割り当てます。

```
 modify oxygen;
rename oxygen=intake;
label intake='Intake Measurement';
quit;
```

ログの例

ログ 16.4 *Health ライブラリの SAS ログ*

```
169 options pagesize=40 linesize=80 nodate pageno=1 source;
170 LIBNAME health 'SAS-library';
NOTE: Libref HEALTH was successfully assigned as follows:
Engine: V9
Physical Name: SAS-library\health

NOTE: PROCEDURE DATASETS used (Total process time):
real time 8:06.11
cpu time 0.54 seconds


171 proc datasets library=health nolist;
172 modify group (label='Test Subjects' read=XXXXX sortedby=lname);
WARNING: The file HEALTH.GROUP.DATA is not ALTER protected. It could be
deleted or replaced without knowing the password.
173 index create vital=(birth salary) / nomiss unique;
NOTE: Composite index vital has been defined.
NOTE: MODIFY was successful for HEALTH.GROUP.DATA.
174 informat birth date7.;
175 format birth date7.;
176 label salary='current salary excluding bonus';
177 modify oxygen;
178 rename oxygen=intake;
NOTE: Renaming variable oxygen to intake.
179 label intake='Intake Measurement';
180 quit;

NOTE: MODIFY was successful for HEALTH.OXYGEN.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
real time 15.09 seconds
cpu time 0.06 seconds
```

例 5: SAS データセットの説明

要素: CONTENTS ステートメントオプション
DATA=

他の要素: SAS データセットオプション
READ=

詳細

この例に、GROUP データセットに対する CONTENTS ステートメントからの出力を示します。出力には、“例 4: SAS データセットの変更” (555 ページ)で GROUP データセットに行われた変更が表示されます。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME health
'SAS-library';

proc datasets library=health nolist;

 contents data=group (read=green) out=grpout;
title 'The Contents of the GROUP Data Set';
run;
quit;
```

プログラムの説明

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME health
'SAS-library';
```

Health をプロシジャの入力ライブラリに指定し、ディレクトリリストの表示を抑制します。

```
proc datasets library=health nolist;
```

データセット GROUP から出力データセット GRPOUT を作成します。 書き込むデータセットとして GROUP を指定し、データセット GROUP に読み取りアクセスを付与し、そして OUT=で指定して出力データセット GRPOUT を作成します。

```
 contents data=group (read=green) out=grpout;
title 'The Contents of the GROUP Data Set';
run;
quit;
```

出力例

アウトプット 16.14 *GROUP データセットの内容*

The Contents of the GROUP Data Set

The CONTENTS Procedure

Data Set Name	HEALTH.GROUP	Observations	148
Member Type	DATA	Variables	11
Engine	V9	Indexes	0
Created	09/03/2014 10:35:02	Observation Length	96
Last Modified	09/03/2014 10:35:02	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

アウトプット 16.15 *エンジン/ホスト依存 情報*

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	3
First Data Page	1
Max Obs per Page	84
Obs in First Data Page	63
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\\procdatasets\health\group.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

アウトプット 16.16 *変数と属性のアルファベット順リスト*

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Informat
9	BIRTH	Num	8		
4	CITY	Char	15		
3	FNAME	Char	15		
10	HIRED	Num	8	DATE7.	DATE7.
11	HPHONE	Char	12		
1	IDNUM	Char	4		
7	JOBCODE	Char	4		
2	LNAME	Char	15		
8	SALARY	Num	8	COMMA8.	
6	SEX	Char	2		
5	STATE	Char	3		

アウトプット 16.17 *データセットと変数のアルファベット順リスト*

Alphabetic List of Data Set Extended Attributes		
Extended Attribute	Numeric Value	Character Value
level	1	
region	.	NorthEast

Alphabetic List of Extended Attributes on Variables			
Extended Attribute	Attribute Variable	Numeric Value	Character Value
classification	jobcode	.	local
length	idnum	4	

例 6: 2 つの SAS データセットを連結する

要素: APPEND ステートメントオプション
BASE=
DATA=
FORCE=

詳細

この例では、次のアクションを行います。

- ライブラリの出力を抑制する
- 2 つのデータセットを追加する
- 追加前にデータセットを出力し、追加後に新しいデータセットを出力する

プログラム

```
options pagesize=40 linesize=64 nodate pageno=1;

LIBNAME exp 'SAS-library';

proc datasets library=exp nolist;

 append base=exp.results data=exp.sur force;
run;

proc print data=exp.results noobs;
title 'The EXP.RESULTS Data Set';
run;
```

プログラムの説明

この例では、あるデータセットを別のデータセットに追加します。

BASE=データセット(EXP.RESULTS)

データセット Exp.Sur には変数 WT6MOS が含まれますが、Exp.Results データセットは含まれません。

```
options pagesize=40 linesize=64 nodate pageno=1;

LIBNAME exp 'SAS-library';
```

EXP ライブラリの出力を抑制します。 LIBRARY=は EXP をプロシジャの入力ライブラリとして指定します。NOLIST は EXP ライブラリのディレクトリリストの表示を抑制します。

```
proc datasets library=exp nolist;
```

データセット EXP.SUR を EXP.RESULTS データセットに追加します。 APPEND ステートメントはデータセット EXP.SUR を EXP.RESULTS に追加します。FORCE は EXP.SUR にある変数が EXP.RESULTS にない場合でも APPEND ステートメントの追加操作を実行させます。APPEND は変数 WT6MOS を EXP.RESULTS に追加はしません。

```
 append base=exp.results data=exp.sur force;
run;
```

データセットを出力します。

```
proc print data=exp.results noobs;
title 'The EXP.RESULTS Data Set';
run;
```

出力: 2 つの SAS データセットの連結

アウトプット 16.18 *結果データセット*

The RESULTS Data Set

id	treat	initwt	wt3mos	age
1	Other	166.28	146.98	35
2	Other	214.42	210.22	54
3	Other	172.46	159.42	33
5	Other	175.41	160.66	37
6	Other	173.13	169.40	20
7	Other	181.25	170.94	30
10	Other	239.83	214.48	48
11	Other	175.32	162.66	51
12	Other	227.01	211.06	29
13	Other	274.82	251.82	31

アウトプット 16.19 *Sur データセット*

The EXP.SUR Data Set

ID	treat	initwt	wt3mos	wt6mos	age
14	surgery	203.60	169.78	143.88	38
17	surgery	171.52	150.33	123.18	42
18	surgery	207.46	155.22	.	41

アウトプット 16.20 *Results および Sur データセットの連結*

The Concatenated RESULTS Data Set

ID	TREAT	INITWT	WT3MOS	AGE
1	Other	166.28	146.98	35
2	Other	214.42	210.22	54
3	Other	172.46	159.42	33
5	Other	175.41	160.66	37
6	Other	173.13	169.40	20
7	Other	181.25	170.94	30
10	Other	239.83	214.48	48
11	Other	175.32	162.66	51
12	Other	227.01	211.06	29
13	Other	274.82	251.82	31
14	surgery	203.60	169.78	38
17	surgery	171.52	150.33	42
18	surgery	207.46	155.22	41

例 7: SAS データセットのエージング

要素: AGE ステートメント

詳細

この例では、AGE ステートメントが SAS ファイルをエージングする方法を示します。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1 source;

LIBNAME daily 'SAS-library';

proc datasets library=daily nolist;

 age today day1-day7;
run;
```

プログラムの説明

プログラミングステートメントを SAS ログに書き出します。 SAS オプションの SOURCE はプログラミングステートメントをログに書き出します。

```
options pagesize=40 linesize=80 nodate pageno=1 source;

LIBNAME daily 'SAS-library';
```

DAILY をプロシジャの入力ライブラリに指定し、ディレクトリリストの表示を抑制します。

```
proc datasets library=daily nolist;
```

ファイルを削除してエージングします。 リストの最後の SAS ファイルである Day7 を削除し、Day6 を Day7 にエージング(名称変更)します。同じように Day5 を Day6 というように、TODAY が Day1 になるまで続けます。

```
 age today day1-day7;
run;
```

ログの例

ログ 16.5 *SAS ログ*

```
6 options pagesize=40 linesize=80
nodate pageno=1 source;
7
8 proc datasets library=daily nolist;
9
10 age today day1-day7;
11 run;
NOTE: Deleting DAILY.DAY7 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY6 to DAILY.DAY7 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY5 to DAILY.DAY6 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY4 to DAILY.DAY5 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY3 to DAILY.DAY4 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY2 to DAILY.DAY3 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY1 to DAILY.DAY2 (memtype=DATA).
NOTE: Ageing the name DAILY.TODAY to DAILY.DAY1 (memtype=DATA).
```

例 8: ODS 出力

要素: CONTENTS ステートメント

詳細

例では、PROC CONTENTS 出力を ODS 出力データセットに処理のため取得する方法を示します

詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

プログラム

```
title1 "PROC CONTENTS ODS Output";

options nodate nonumber nocenter formdlim='-';

data a;
x=1;
run;
```

```
ods output attributes=atr
variables=var
enginehost=eng;

ods listing close;

proc contents data=a;
run;

ods listing;

title2 "all Attributes data";

proc print data=atr noobs;
run;

title2 "all Variables data";

proc print data=var noobs;
run;

title2 "all EngineHost data";

proc print data=eng noobs;
run;

ods output attributes=atr1(keep=member cvalue1 label1
where=(attribute in ('Data Representation','Encoding'))
rename=(label1=attribute cvalue1=value))
attributes=atr2(keep=member cvalue2 label2
where=(attribute in ('Observations', 'Variables'))
rename=(label2=attribute cvalue2=value));


ods listing close;

proc contents data=a;
run;

ods listing;

data final;
set atr1 atr2;
run;

title2 "example of post-processing of ODS output data";

proc print data=final noobs;
run;

ods listing close;
```

プログラムの説明

```
title1 "PROC CONTENTS ODS Output";
```

```
options nodate nonumber nocenter formdlim='-';

data a;
x=1;
run;
```

ODS OUTPUT ステートメントで CONTENTS がダイレクトされるデータセットを指定します。

```
ods output attributes=atr
variables=var
enginehost=eng;
```

一時的にリスト出力を抑制します。

```
ods listing close;

proc contents data=a;
run;
```

リスト表示を再開します。

```
ods listing;

title2 "all Attributes data";

proc print data=atr noobs;
run;

title2 "all Variables data";

proc print data=var noobs;
run;

title2 "all EngineHost data";

proc print data=eng noobs;
run;
```

ODS 出力から特定のデータを選択します。

```
ods output attributes=atr1(keep=member cvalue1 label1
where=(attribute in ('Data Representation','Encoding'))
rename=(label1=attribute cvalue1=value))
attributes=atr2(keep=member cvalue2 label2
where=(attribute in ('Observations', 'Variables'))
rename=(label2=attribute cvalue2=value));


ods listing close;

proc contents data=a;
run;

ods listing;

data final;
set atr1 atr2;
```

```
run;

title2 "example of post-processing of ODS output data";

proc print data=final noobs;
run;

ods listing close;
```

出力: ODS の例

アウトプット 16.21 *すべての Attributes データ*

PROC CONTENTS ODS Output
all Attributes data

Member	Label1	cValue1	nValue1	Label2	cValue2	nValue2
ODS.A	Data Set Name	ODS.A	.	Observations	1	1.000000
ODS.A	Member Type	DATA	.	Variables	1	1.000000
ODS.A	Engine	V9	.	Indexes	0	0
ODS.A	Created	04/17/2014 07:22:48	1713338569	Observation Length	8	8.000000
ODS.A	Last Modified	04/17/2014 07:22:48	1713338569	Deleted Observations	0	0
ODS.A	Protection		.	Compressed	NO	.
ODS.A	Data Set Type		.	Sorted	NO	.
ODS.A	Label		.			0
ODS.A	Data Representation	WINDOWS_64	.			0
ODS.A	Encoding	wlatin1 Western (Windows)	.			0

アウトプット 16.22 *すべての Variables データ*

PROC CONTENTS ODS Output
all Variables data

Member	Num	Variable	Type	Len	Pos
ODS.A	1	x	Num	8	0

アウトプット 16.23 すべての EngineHost データ

PROC CONTENTS ODS Output
all EngineHost data

Member	Label1	cValue1	nValue1
ODS.A	Data Set Page Size	65536	65536
ODS.A	Number of Data Set Pages	1	1.000000
ODS.A	First Data Page	1	1.000000
ODS.A	Max Obs per Page	8063	8063.000000
ODS.A	Obs in First Data Page	1	1.000000
ODS.A	Number of Data Set Repairs	0	0
ODS.A	ExtendObsCounter	YES	.
ODS.A	Filename	c:\ODS\a.sas7bdat	.
ODS.A	Release Created	9.0401M0	.
ODS.A	Host Created	X64_7PRO	.

アウトプット 16.24 すべての EngineHost データの CONTENTS プロシジャ

PROC CONTENTS ODS Output
all EngineHost data

The CONTENTS Procedure

Data Set Name	ODS.A	Observations	1
Member Type	DATA	Variables	1
Engine	V9	Indexes	0
Created	04/15/2014 09:29:13	Observation Length	8
Last Modified	04/15/2014 09:29:13	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	8063
Obs in First Data Page	1
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\ODS\a.sas7bdat
Release Created	9.0401M0
Host Created	X64_7PRO

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	x	Num	8

アウトプット 16.25 ODS 出力データの後処理

PROC CONTENTS ODS Output
example of post-processing of ODS output data

Member	attribute	value
ODS.A	Data Representation	WINDOWS_64
ODS.A	Encoding	wlatin1 Western (Windows)
ODS.A	Observations	1
ODS.A	Variables	1

例 9: ソートインジケータの情報の取得

要素: APPEND ステートメントオプション
GETSORT
SORTEDBY データセットオプション

詳細

次の例では、各アクションを説明します。

- 2 つのデータセット(オブザベーションありおよびなし)を作成する
- データセットを降順で並べ替える
- SORTEDBY データセットオプションを使用してソートインジケータを作成する
- SORT プロシジャを使用してソートインジケータを作成する

プログラム

```
data mtea;
length var1 8.;
stop;
run;

data phull;
length var1 8.;
do var1=1 to 100000;
output;
end;
run;

proc sort data=phull;
by DESCENDING var1;
run;

proc append base=mtea data=phull getsort;
run;

ods select sortedby;
```

```
proc contents data=mtea;
run;

data mysort(sortedby=var1);
length var1 8.;
do var1=1 to 10;
output;
end;
run;

ods select sortedby;

proc contents data=mysort;
run;

data mysort;
length var1 8.;
do var1=1 to 10;
output;
end;
run;

proc sort data=mysort;
by var1;
run;

ods select sortedby;

proc contents data=mysort;
run;
```

プログラムの説明

次の例では、APPEND ステートメントと GETSORT オプションを使って、ソートインジケータが継承できることを示します。

オブザベーションを含まない"シェル"データセットを作成します。

```
data mtea;
length var1 8.;
stop;
run;
```

同じ構造で多数のオブザベーションを持つ、もう1つのデータセットを作成します。 データセットを並べ替えます。

```
data phull;
length var1 8.;
do var1=1 to 100000;
output;
end;
run;

proc sort data=phull;
by DESCENDING var1;
run;
```

```
proc append base=mtea data=phull getsort;
run;

ods select sortedby;

proc contents data=mtea;
run;
```

SORTEDBY=データセットオプションを使用してソートインジケータが作成されます。

```
data mysort(sortedby=var1);
length var1 8.;
do var1=1 to 10;
output;
end;
run;

ods select sortedby;

proc contents data=mysort;
run;
```

ソートインジケータが PROC SORT で作成されます。

```
data mysort;
length var1 8.;
do var1=1 to 10;
output;
end;
run;

proc sort data=mysort;
by var1;
run;

ods select sortedby;

proc contents data=mysort;
run;
```

出力例

アウトプット 16.26 *降順ソート情報*

The CONTENTS Procedure

Sort Information	
Sortedby	DESCENDING var1
Validated	YES
Character Set	ANSI

アウトプット 16.27 *SORTEDBY=データセットオプションを使用するソートインジケータ情報*

The CONTENTS Procedure

Sort Information	
Sortedby	var1
Validated	NO
Character Set	ANSI

アウトプット 16.28 *SORT プロシジャを使用するソートインジケータ情報*

The CONTENTS Procedure

Sort Information	
Sortedby	var1
Validated	YES
Character Set	ANSI

例 10: ORDER=オプションを CONTENTS ステートメントとともに使用する

要素: CONTENTS ステートメントオプション
ORDER=
COLLATE
CASECOLLATE
IGNORECASE
VARNUM

詳細

この例は、次のアクションを示しています。

- データセットを設定する
- データセットの名前を変更する
- 複数のオプションを使用して出力を作成する

プログラム

```
options nonotes nodate nonumber nocenter formdlim ='-';

libname contents 'SAS-library';

data contents.test;
d=2;
```

```
b001 =1;
b002 =2;
b003 =3;
b001z=1;
B001a=2;
CaSeSeNsItIvE2=9;
CASESENSITIVE3=9;
D=2;
casesensitive1=9;
CaSeSeNsItIvE1a=9;
d001z=1;
CASESENSITIVE1C=9;
D001a=2;
casesensitive1b=9;
A =1;
a002 =2;
a =3;
a001z=1;
A001a=2;

run;

%let mydata=contents.test;

ods output Variables=var1(keep=Num Variable);
ods listing close;

proc contents data=&mydata;
run;

ods listing;
title "Default options";

proc print data=var1 noobs;
run;

ods output Variables=var2(keep=Num Variable);
ods listing close;

proc contents order=collate data=&mydata;
run;

ods listing;
title "order=collate option";

proc print data=var2 noobs;
run;

ods output Variables=var3(keep=Num Variable);
ods listing close;

proc contents order=casecollate data=&mydata;
run;

ods listing;
title "order=casecollate option";
```

```
proc print data=var3 noobs;
run;

ods output Variables=var4(keep=Num Variable);
ods listing close;

proc contents order=ignorecase data=&mydata;
run;

ods listing;
title "order=ignorecase option";

proc print data=var4 noobs;
run;

ods output Position=var5(keep=Num Variable);
ods listing close;

proc contents data=&mydata varnum;
run;

ods listing;
title "varnum option";

proc print data=var5 noobs;
run;
```

プログラムの説明

データセットを設定します。

```
options nonotes nodate nonumber nocenter formdlim ='-';

libname contents 'SAS-library';

data contents.test;
d=2;
b001 =1;
b002 =2;
b003 =3;
b001z=1;
B001a=2;
CaSeSeNsItIvE2=9;
CASESENSITIVE3=9;
D=2;
casesensitive1=9;
CaSeSeNsItIvE1a=9;
d001z=1;
CASESENSITIVE1C=9;
D001a=2;
casesensitive1b=9;
A =1;
a002 =2;
a =3;
a001z=1;
```

```
A001a=2;

run;
```

希望するデータセットの PROC CONTENTS 出力を作成するには、データセットの名前を MYDATA に変更します。

```
%let mydata=contents.test;

ods output Variables=var1(keep=Num Variable);
ods listing close;

proc contents data=&mydata;
run;

ods listing;
title "Default options";

proc print data=var1 noobs;
run;

ods output Variables=var2(keep=Num Variable);
ods listing close;

proc contents order=collate data=&mydata;
run;

ods listing;
title "order=collate option";

proc print data=var2 noobs;
run;

ods output Variables=var3(keep=Num Variable);
ods listing close;

proc contents order=casecollate data=&mydata;
run;

ods listing;
title "order=casecollate option";

proc print data=var3 noobs;
run;

ods output Variables=var4(keep=Num Variable);
ods listing close;

proc contents order=ignorecase data=&mydata;
run;

ods listing;
title "order=ignorecase option";

proc print data=var4 noobs;
run;
```

VARNUM オプションを使用します。 VARNUM オプションの使用時は、ODS 出力オブジェクトの名前が違います。

```
ods output Position=var5(keep=Num Variable);
ods listing close;

proc contents data=&mydata varnum;
run;

ods listing;
title "varnum option";

proc print data=var5 noobs;
run;
```

出力例

アウトプット 16.29 *COLLATE および CASECOLLATE オプションの使用*

Default options

Num	Variable
15	A
18	A001a
6	B001a
8	CASESENSITIVE3
12	CASESENSITIVE1C
7	CaSeSeNsItIvE2
10	CaSeSeNsItIvE1a
13	D001a
16	a002
17	a001z
2	b001
3	b002
4	b003
5	b001z
9	casesensitive1
14	casesensitive1b
1	d
11	d001z

order=collate option

Num	Variable
15	A
18	A001a
6	B001a
12	CASESENSITIVE1C
8	CASESENSITIVE3
10	CaSeSeNsItIvE1a
7	CaSeSeNsItIvE2
13	D001a
17	a001z
16	a002
2	b001
5	b001z
3	b002
4	b003
9	casesensitive1
14	casesensitive1b
1	d
11	d001z

order=casecollate option

Num	Variable
15	A
18	A001a
17	a001z
16	a002
2	b001
6	B001a
5	b001z
3	b002
4	b003
9	casesensitive1
10	CaSeSeNsItIvE1a
14	casesensitive1b
12	CASESENSITIVE1C
7	CaSeSeNsItIvE2
8	CASESENSITIVE3
1	d
13	D001a
11	d001z

アウトプット 16.30 *IGNORECASE および VARNUM オプションの使用*

Default options

Num	Variable
15	A
18	A001a
6	B001a
8	CASESENSITIVE3
12	CASESENSITIVE1C
7	CaSeSeNsItIvE2
10	CaSeSeNsItIvE1a
13	D001a
16	a002
17	a001z
2	b001
3	b002
4	b003
5	b001z
9	casesensitive1
14	casesensitive1b
1	d
11	d001z

order=ignorecase option

Num	Variable
15	A
16	a002
18	A001a
17	a001z
2	b001
3	b002
4	b003
6	B001a
5	b001z
9	casesensitive1
7	CaSeSeNsItIvE2
8	CASESENSITIVE3
10	CaSeSeNsItIvE1a
14	casesensitive1b
12	CASESENSITIVE1C
1	d
13	D001a
11	d001z

varnum option

Num	Variable
1	d
2	b001
3	b002
4	b003
5	b001z
6	B001a
7	CaSeSeNsItIvE2
8	CASESENSITIVE3
9	casesensitive1
10	CaSeSeNsItIvE1a
11	d001z
12	CASESENSITIVE1C
13	D001a
14	casesensitive1b
15	A
16	a002
17	a001z
18	A001a

例 11: 監査ファイルの初期化

要素: AUDIT ステートメントとオプション
- AUDIT
- INITIATE
- USER_VAR
- LOG
- SUSPEND
- RESUME
- TERMINATE

詳細

この例は、次のアクションを示しています。

- 監査ファイルを作成する

- データセットを更新する
- 監査ファイルを中断する
- 監査ファイルを終了する

プログラム

```
libname mylib "SAS-library";

data mylib.inventory;
input vendor $10. +1 item $4. +1 description $11. +1 units 4.;
datalines;
SmithFarms F001 Apples 10
Tropicana B002 OrangeJuice 45
UpperCrust C215 WheatBread 25
;
run;

proc datasets lib=mylib;
audit inventory;
initiate;
user_var reason $ 30;
quit;


proc sql;
Insert into mylib.inventory values ('Bordens','B132', 'Milk', 100,
'increase on hand');
Update mylib.inventory set units=10, reason='recounted inventory'
where item='B002';
quit;



proc datasets lib=mylib;
audit inventory;
log admin_image=no;
suspend;
quit;



proc sql;
select * from mylib.inventory(type=audit);
quit;



proc datasets lib=mylib;
audit inventory;
resume;
quit;



proc datasets lib=mylib;
audit inventory;
terminate;
```

```
quit;
```

プログラムの説明

USER_VAR を使って監査ファイルを開始します。

```
libname mylib "SAS-library";

data mylib.inventory;
input vendor $10. +1 item $4. +1 description $11. +1 units 4.;
datalines;
SmithFarms F001 Apples 10
Tropicana B002 OrangeJuice 45
UpperCrust C215 WheatBread 25
;
run;

proc datasets lib=mylib;
audit inventory;
initiate;
user_var reason $ 30;
quit;
```

データセットを更新します。

```
proc sql;
Insert into mylib.inventory values ('Bordens','B132', 'Milk', 100,
'increase on hand');
Update mylib.inventory set units=10, reason='recounted inventory'
where item='B002';
quit;
```

ADMIN イメージの記録を中止し、監査ファイルを中断します。

```
proc datasets lib=mylib;
audit inventory;
log admin_image=no;
suspend;
quit;
```

監査ファイルを表示します。

```
proc sql;
select * from mylib.inventory(type=audit);
quit;
```

監査ファイルを再開します。

```
proc datasets lib=mylib;
audit inventory;
```

```
resume;
quit;
```

監査ファイルを終了します。

```
proc datasets lib=mylib;
audit inventory;
terminate;
quit;
```

ログの例

ログ 16.6 *監査ファイルの初期化*

```
1 options nocenter;
2
3 libname mylib "SAS-library";
NOTE: Libref MYLIB was successfully assigned as follows:
Engine: V9
Physical Name: c:\mylib
4
5 data mylib.inventory;
6 input vendor $10. +1 item $4. +1 description $11. +1 units 4.;
7 datalines;

NOTE: The data set MYLIB.INVENTORY has 3 observations and 4 variables.
NOTE: DATA statement used (Total process time):
real time 3.45 seconds
cpu time 0.00 seconds


11 ;
12 run;
13
14 proc datasets lib=mylib;
NOTE: Writing HTML Body file: sashtml.htm
15 audit inventory;
16 initiate;
WARNING: The audited data file MYLIB.INVENTORY.DATA is not password protected.
Apply an Alter password to prevent accidental
deletion or replacement of it and any associated audit files.
17 user_var reason $ 30;
18 quit;

NOTE: The data set MYLIB.INVENTORY.AUDIT has 0 observations and 11 variables.
NOTE: PROCEDURE DATASETS used (Total process time):
real time 18.17 seconds
cpu time 0.79 seconds


19
20 proc sql;
21 Insert into mylib.inventory values ('Bordens','B132', 'Milk', 100,
22 'increase on hand');
NOTE: 1 row was inserted into MYLIB.INVENTORY.

23 Update mylib.inventory set units=10, reason='recounted inventory'
24 where item='B002';
NOTE: 1 row was updated in MYLIB.INVENTORY.

25 quit;
NOTE: PROCEDURE SQL used (Total process time):
real time 2.57 seconds
cpu time 0.03 seconds


26
27 proc datasets lib=mylib;
28 audit inventory;
29 log admin_image=no;
30 suspend;
31 quit;

NOTE: PROCEDURE DATASETS used (Total process time):
real time 0.01 seconds
cpu time 0.01 seconds
```

```
32
33 proc sql;
34 select * from mylib.inventory(type=audit);
35 quit;
NOTE: PROCEDURE SQL used (Total process time):
real time 0.54 seconds
cpu time 0.01 seconds


36
37 proc datasets lib=mylib;
37 ! /* resume audit file */
38 audit inventory;
39 resume;
40 quit;

NOTE: PROCEDURE DATASETS used (Total process time):
real time 0.01 seconds
cpu time 0.01 seconds


41
42 /* additional step(s) which update the inventory dataset could go here*/
43
44 proc datasets lib=mylib;
44 ! /* terminate audit file */
45 audit inventory;
46 terminate;
NOTE: Deleting MYLIB.INVENTORY (memtype=AUDIT).
47 quit;

NOTE: PROCEDURE DATASETS used (Total process time):
real time 0.01 seconds
cpu time 0.01 seconds
```

出力例

アウトプット 16.31 *MyLib ライブラリのインベントリコンテンツ*

The CONTENTS Statement

Directory	
Libref	MYLIB
Engine	V9
Physical Name	c:\mylib
Filename	c:\mylib

#	Name	Member Type	File Size	Last Modified
1	INVENTORY	DATA	128KB	11/14/2014 11:30:00
2	SALARY	DATA	128KB	04/30/2014 13:42:52

アウトプット 16.32 *MyLib ライブラリの監査ファイルリスト*

The CONTENTS Statement

Directory	
Libref	MYLIB
Engine	V9
Physical Name	c:\mylib
Filename	c:\mylib

#	Name	Member Type	File Size	Last Modified
1	INVENTORY	DATA	128KB	11/14/2014 11:30:02
	INVENTORY	AUDIT	65KB	11/14/2014 11:30:02
2	SALARY	DATA	128KB	04/30/2014 13:42:52

アウトプット 16.33 *MyLib ライブラリのインベントリデータファイルコンテンツ*

The CONTENTS Statement

vendor	item	description	units	reason	_ATDATETIME_	_ATOBSNO_	_ATRETURNCODE_	_ATUSERID_	_ATOPCODE_	_ATMESSAGE_
Bordens	B132	Milk	100	increase on hand	14NOV2014:11:26:03	4	.		DA	
Tropicana	B002	OrangeJuice	45		14NOV2014:11:26:03	2	.		DR	
Tropicana	B002	OrangeJuice	10	recounted inventory	14NOV2014:11:26:03	2	.		DW	

例 12: 拡張属性

要素: MODIFY ステートメントオプション
MODIFY
XATTR ADD DS
XATTR ADD VAR

プログラム

```
 libname mylib 'C:\mylib';

data mylib.sales;
purchase = "car";
age = 10;
income = 200000;
kids = 3;
cars = 4;
run;

proc datasets lib=mylib nolist ;
```

```
modify sales;
xattr add ds role="train" attrib="table";
xattr add var purchase ( role="target" level="nominal" )
age ( role="reject" )
income ( role="input" level="interval" );
contents data=sales;
title 'The Contents of the Sales Data Set That Contains Extended Attributes';
run;
run;
quit;
```

プログラムの説明

MyLib.Sales データセットを作成します。

```
 libname mylib 'C:\mylib';

data mylib.sales;
purchase = "car";
age = 10;
income = 200000;
kids = 3;
cars = 4;
run;
```

データセットおよび変数への拡張属性の追加。

```
proc datasets lib=mylib nolist ;
modify sales;
xattr add ds role="train" attrib="table";
xattr add var purchase ( role="target" level="nominal" )
age ( role="reject" )
income ( role="input" level="interval" );
contents data=sales;
title 'The Contents of the Sales Data Set That Contains Extended Attributes';
run;
run;
quit;
```

ログの例

ログ 16.7 拡張属性

```
355 libname mylib 'C:\mylib';
NOTE: Libref MYLIB was successfully assigned as follows:
Engine: V9
Physical Name: C:\mylib
356
357 data mylib.sales;
358 purchase = "car";
359 age = 10;
360 income = 200000;
361 kids = 3;
362 cars = 4;
363 run;

NOTE: The data set MYLIB.SALES has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):
real time 0.00 seconds
cpu time 0.00 seconds


364
365 proc datasets lib=mylib nolist ;
366 modify sales;
367 xattr add ds role= "train" attrib="table";
368 xattr add var purchase ( role="target" level="nominal" )
369 age ( role="reject" )
370 income ( role="input" level="interval" );
NOTE: MODIFY was successful for MYLIB.SALES.DATA.
371
372 contents data=sales;
373 title 'The Contents of the Sales Data Set That Contains Extended Attributes';
374
375 run;

376 quit;

NOTE: PROCEDURE DATASETS used (Total process time):
real time 1.02 seconds
cpu time 0.10 seconds
```

出力例

アウトプット 16.34 *拡張属性を持つ Sales データセットのコンテンツ*

The Contents of the Sales Data Set That Contains Extended Attributes

The DATASETS Procedure

Data Set Name	MYLIB.SALES	Observations	1
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	04/11/2014 09:10:39	Observation Length	40
Last Modified	04/11/2014 09:10:40	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label		EXTATTR Segment Length	256
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
2	age	Num	8
5	cars	Num	8
3	income	Num	8
4	kids	Num	8
1	purchase	Char	3

Alphabetic List of Data Set Extended Attributes		
Extended Attribute	Numeric Value	Character Value
attrib	.	table
role	.	train

Alphabetic List of Extended Attributes on Variables			
Extended Attribute	Attribute Variable	Numeric Value	Character Value
level	income	.	interval
level	purchase	.	nominal
role	age	.	reject
role	income	.	input
role	purchase	.	target

17 章

DATEKEYS プロシジャ

概要:DATEKEYS プロシジャ

DATEKEYS プロシジャを使用すると、日付を参照する名前を使用して、単一の日付または時系列の一連の日付を処理できます。たとえば、日付キーに関連付けられた期間の特定や、日付キーの読み取り/書き込み、日付キー定義に関する詳細の提供に使用できます。

DATEKEYS プロシジャを使用して、日付キーに関連付けられた期間を特定できます。

概念:DATEKEYS プロシジャ

SAS 日付キーは、祝日やセール期間などの特別イベントや時間計算に関連付けられた日付または時間間隔を記述します。

日付キーは、名前、その日付キーに関連付けられた単一または一連の日付、および一連の修飾子で構成されます。

DATEKEYS プロシジャは、他の SAS プロシジャで解釈できる出力データセットに結果を提供します。SAS システムオプションの EVENTDS=と組み合わせて使用すると、事前定義された SAS 日付キーとして指定できる日付キーを定義できます。事前定義された SAS 日付キーを使用する場合と同様に、定義した日付キーを日付キーワードとして使用できます。

次の例では、日付キーを作成し、MyHolidays という名前の出力データセットに日付キー定義を書き込みます。SAS システムオプションの EVENTDS=を設定することで、日付キー定義を SAS High-Performance Forecasting のプロシジャで自動的に使用できるようになります。

```
proc datekeys;
   datekeydef SuperBowl=
               '15JAN1967'd '14JAN1968'd '12JAN1969'd '11JAN1970'd
   '17JAN1971'd '16JAN1972'd '14JAN1973'd '13JAN1974'd '12JAN1975'd
   '18JAN1976'd '09JAN1977'd '15JAN1978'd '21JAN1979'd '20JAN1980'd
   '25JAN1981'd '24JAN1982'd '30JAN1983'd '22JAN1984'd '20JAN1985'd
   '26JAN1986'd '25JAN1987'd '31JAN1988'd '22JAN1989'd '28JAN1990'd
   '27JAN1991'd '26JAN1992'd '31JAN1993'd '30JAN1994'd '29JAN1995'd
   '28JAN1996'd '26JAN1997'd '25JAN1998'd '31JAN1999'd '30JAN2000'd
   '28JAN2001'd '03FEB2002'd '26JAN2003'd '01FEB2004'd '06FEB2005'd
   '05FEB2006'd '04FEB2007'd '03FEB2008'd '01FEB2009'd '07FEB2010'd
   '06FEB2011'd '05FEB2012'd '03FEB2013'd '02FEB2014'd
   / PULSE=DAY ;

   datekeydef GoodFriday=Easter / shift=-2 pulse=day;
   datekeykey EasterMonday=Easter / shift=1 pulse=day;
   datekeydata out=MyHolidays condense;
run;

options eventds=(MyHolidays);
```

```
proc hpfevents data=sashelp.citiday;
   id date interval=day start='27JAN1991'd end='01APR1991'd;
   eventkey SuperBowl;
   eventkey GoodFriday;
   eventkey EasterMonday;
   eventdata out=MyHolidayEvents condense;
   eventdummy out=MyHolidayDates;
run;
```

次の出力は結果を示しています。

図 17.1 *ユーザー定義の日付キーに基づくイベント定義データセット*

The SAS System

Obs	_NAME_	_KEYNAME_
1	SuperBowl	SUPERBOWL
2	GoodFriday	GOODFRIDAY
3	EasterMonday	EASTERMONDAY

次のステートメントは、スーパーボウル(Super Bowl)、聖金曜日(Good Friday)、復活の月曜日(Easter Monday)の各イベントの変数を示す出力データセットを表示します。最初の出力には、Month=1 の場合の結果が示されます。2 番目の出力には、Month GE 3 の場合の結果が示されます。

```
proc print data=MyHolidayDates(where=(month(date)=1));
   var date SuperBowl GoodFriday EasterMonday;
run;

proc print data=MyHolidayDates(where=(month(date) GE 3));
   var date SuperBowl GoodFriday EasterMonday;
run;
```

図 17.2 *ユーザー定義の日付キーに基づく出力データセット:Month=1*

The SAS System

Obs	DATE	SuperBowl	GoodFriday	EasterMonday
1	27JAN1991	1	0	0
2	28JAN1991	0	0	0
3	29JAN1991	0	0	0
4	30JAN1991	0	0	0
5	31JAN1991	0	0	0

図 17.3 ユーザー定義の日付キーに基づく出力データセット:Month GE 3

The SAS System

Obs	DATE	SuperBowl	GoodFriday	EasterMonday
34	01MAR1991	0	0	0
35	02MAR1991	0	0	0
36	03MAR1991	0	0	0
37	04MAR1991	0	0	0
38	05MAR1991	0	0	0
39	06MAR1991	0	0	0
40	07MAR1991	0	0	0
41	08MAR1991	0	0	0
42	09MAR1991	0	0	0
43	10MAR1991	0	0	0
44	11MAR1991	0	0	0
45	12MAR1991	0	0	0
46	13MAR1991	0	0	0
47	14MAR1991	0	0	0
48	15MAR1991	0	0	0
49	16MAR1991	0	0	0
50	17MAR1991	0	0	0
51	18MAR1991	0	0	0
52	19MAR1991	0	0	0
53	20MAR1991	0	0	0
54	21MAR1991	0	0	0
55	22MAR1991	0	0	0
56	23MAR1991	0	0	0
57	24MAR1991	0	0	0
58	25MAR1991	0	0	0
59	26MAR1991	0	0	0
60	27MAR1991	0	0	0
61	28MAR1991	0	0	0
62	29MAR1991	0	1	0
63	30MAR1991	0	0	0
64	31MAR1991	0	0	0
65	01APR1991	0	0	1

構文: DATEKEYS プロシジャ

PROC DATEKEYS<*option(s)*>;
BY *variable(s)*;
DATEKEYCALENDAR OUT=*SAS-data-set* <SUMMARY=*SAS-variable-name*>;
DATEKEYDATA IN=*SAS-data-set* | OUT=*SAS-data-set* <*option(s)*>;
DATEKEYDEF *SAS-variable-name=timing-value-list* </*qualifier-option(s)*>;
DATEKEYDSOPT LOCALE=<(ONLY)>*'POSIX locale'*;
DATEKEYKEY <*SAS-variable-name=*> *datekey-keyword* </*qualifier-option(s)*>;
DATEKEYPERIODS OUT=*SAS-data-set*;
ID *SAS-variable-name* INTERVAL=*interval* <*option(s)*>;
VAR *variable(s)*;

ステートメント	タスク	例
“PROC DATEKEYS ステートメント”	時間計算に関連付けられた日付キーを作成および管理する	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“BY ステートメント”	BY 変数によって定義されたオブザベーショングループごとに個別のカレンダ変数を取得する	
“DATEKEYCALENDAR ステートメント”	日付キーのアクティブ期間を示す変数を書き込む	Ex. 3, Ex. 4
“DATEKEYDATA ステートメント”	日付キーを入力および出力する	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DATEKEYDEF ステートメント”	他の SAS プロシジャで使用できる日付キーを定義する	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DATEKEYDSOPT ステートメント”	指定したロケールにデータセットの入出力処理を限定する	Ex. 4
“DATEKEYKEY ステートメント”	ユーザー定義または事前定義された SAS 日付キーを変更するか、別の日付キーから新しい日付キーを作成する	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DATEKEYPERIODS ステートメント”	入力時間 ID での日付キーのアクティブ期間をリストする変数を書き込む	Ex. 1
“ID ステートメント”	入力/出力データセットのオブザベーションを識別する数値変数の名前を指定する	Ex. 1, Ex. 2, Ex. 4
“VAR ステートメント”	入力変数を出力カレンダ変数データセットにコピーする	

PROC DATEKEYS ステートメント

時間計算に関連付けられた日付キーを作成および管理します。

構文

PROC DATEKEYS <*option(s)*>;

オプション引数

オプション

次のオプションを使用できます。

DATA=*SAS-data-set*

VAR、ID、および BY ステートメントで使用する変数を含む SAS データセットの名前を指定します。

ヒント DATA=オプションを指定しない場合、最後に作成された SAS データセットが使用されます。

LEAD=*number-of-periods*

カレンダ変数を入力時間 ID の範囲外に拡張するための期間数を指定します。LEAD=の値は、入力データセットの最後のオブザベーションを基準とします。BY 変数を指定した場合、LEAD=の値は各 BY グループの最後のオブザベーションを基準とします。

デフォルト 0

MAXERROR=*number*

プロシジャの実行中に生成される警告およびエラーメッセージの最大数を指定します。

デフォルト 25

ヒント このオプションは、BY グループ処理で特に役立ちます。このオプションを使用することで、繰り返されるメッセージを抑制できます。

SORTNAMES

出力データセットの日付キーと変数をアルファベット順に書き込むことを指定します。変数はグループ内で並べ替えられます。BY 変数は、他の BY 変数に対して並べ替えられます。VAR ステートメントでリストされている変数は、VAR ステートメントでリストされている他の変数に対して並べ替えられます。カレンダ変数は他のカレンダ変数に対して並べ替えられます。

BY ステートメント

BY 変数によって定義されたオブザベーショングループごとに個別のカレンダ変数を取得します。

構文

BY *variable(s)*;

必須引数

variable
BY 変数によって定義されたオブザベーショングループごとに個別のカレンダ変数を取得する際に使用する変数を指定します。

ヒント BY ステートメントが指定されていると、プロシジャでは入力データセットが BY 変数順で並べ替えられていることが前提となります。入力データセットが昇順で並べ替えられていない場合は、次の代替方法のいずれかを使用します。同様の BY ステートメントで SORT プロシジャを使用してデータを並べ替えるか、DATASETS プロシジャを使用して BY 変数のインデックスを作成します。詳細については、*Base SAS プロシジャガイド*の DATASETS プロシジャを参照してください。

DATEKEYCALENDAR ステートメント

日付キーのアクティブ期間を示す変数を書き込みます。アクティブ期間は値 1 で示され、非アクティブ期間は値 0 で示されます。

構文

DATEKEYCALENDAR OUT=*SAS-data-set* <SUMMARY=*SAS-variable-name*>;

オプション引数の要約

SUMMARY | SUM=*SAS-variable-name*

必須引数

OUT=*SAS-data-set*
ID ステートメントで指定した ID 情報に基づいて、指定した日付キーのカレンダ変数を含む出力データセットの名前を指定します。

ヒント *SAS-data-set* には、VAR、BY、および ID ステートメントで指定した変数も含まれます。

オプション引数

SUMMARY | SUM=*SAS-variable-name*
日付キーのカレンダ変数を合計し、DATEKEYCALENDAR OUT=データセットの指定した変数に結果を配置することを指定します。

SUM=変数は、時間間隔ごとのアクティブなキー日付の数として解釈できます。SUM=オプションを指定しない場合、このような変数は OUT=データセットに含まれません。

DATEKEYDATA ステートメント

日付キーデータセットからの日付キーの入力、または日付キーデータセットへの日付キーの書き込みを行います。複数の DATEKEYDATA ステートメントを指定できます。

構文

DATEKEYDATA IN=*SAS-data-set* | OUT=*SAS-data-set* <*option(s)*>;

オプション引数の要約

オプション

必須引数

IN=*SAS-data-set*
日付キー定義を含む入力データセットの名前を指定します。

OUT=*SAS-data-set*
DATEKEYDATA IN=データセットと、DATEKEYDEF および DATEKEYKEY ステートメントで指定した日付キー定義を含む出力データセットの名前を指定します。

ヒント LIST オプションを指定しない場合、他の SAS プロシジャとシステムオプションで OUT=データセットを使用して日付キーを定義できます。

オプション引数

オプション
次のオプションを使用できます。

CONDENSE
DATEKEYDATA OUT=データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

DATEKEYDATA IN=オプションは、簡略化されたデータセットと簡略化されていないデータセットの両方を読み取ります。詳細については、“DATEKEYDATA OUT=データセットの変数の識別” (614 ページ)を参照してください。

LIST
DATEKEYDATA OUT=データセットに、使用可能な日付キーのリストだけを含めることを指定します。LIST オプションを指定すると、日付キー定義に必要なパラメータは出力データセットに含まれません。

NODEFAULTS
DATEKEYDATA OUT=データセットに、事前定義された SAS 日付キーを含めないことを指定します。

DATEKEYDEF ステートメント

他の SAS プロシジャで解釈できる日付キーを定義します。これには、予測モデルに含めることができるイベントを作成する際に使用できる日付キーが含まれます。複数の DATEKEYDEF ステートメントを指定できます。

構文

DATEKEYDEF *SAS-variable-name=timing-value-list* </*qualifier-options*>;

オプション引数の要約

qualifier-options

必須引数

SAS-variable-name
DATEKEYDEF ステートメントで名前を指定します。

timing-value-list
1 つ以上の日付キー、日付、日時値、またはオブザベーション番号を指定します。また、DOLIST を指定することもできます。

オプション引数

qualifier-options
次の修飾子オプションを使用できます。

AFTER=(<DURATION=*value*>)
タイミング値の後の日付キー定義を制御するオプションを指定します。DURATION=サブオプションは、AFTER= ( )オプションのかっこ内で使用します。DURATION は、タイミング値の後の日付キーの期間を指定します。

BEFORE=(<DURATION=*value*>)
タイミング値の前の日付キー定義を制御するオプションを指定します。DURATION=サブオプションは、BEFORE= ( )オプションのかっこ内で使用します。DURATION は、タイミング値の前の日付キーの期間を指定します。

LABEL=*'SAS-label'*
日付キーに関連付けるラベルを指定します。'*SAS-label*'は、引用符で囲まれたテキスト文字列です。このテキスト文字列には最大 256 文字使用できます。

デフォルトのラベルは、'*SAS-variable-name*'です。*SAS-variable-name* は、DATEKEYDEF ステートメントで指定した名前です。ラベルは、DATEKEYDATA OUT=データセットに保存されます。

LOCALE='*POSIX locale*'
日付キーに関連付けるロケールを指定します。ロケールは、POSIX ロケール値にする必要があります。ロケール値のデフォルトはありません。

PERIOD=*interval*
日付キーの度数の間隔を指定します。たとえば、PERIOD=YEAR の場合、年単位パターンで周期的な日付キーが生成されます。

PERIOD=オプションを省略すると、日付キーは周期的にはなりません。PERIOD=オプションは、周期的ではないオブザベーション番号や、独自の周期性を持つ日付キーワードには適用されません。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

PULSE=*interval*
日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

日付キーが時間 ID 変数に対して評価される場合、デフォルトのパルスは 1 オブザベーションになります。DURATION=の値を指定せずに、PULSE=オプションを指定した場合、DURATION=の値はゼロに設定されます。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

RULE=*value*

定義されている日付キーに、少なくとも 1 つの日付キーを含む複数のタイミング値があるときに実行するアクションを指定します。

日付キーのタイミング値が、SAS 日付値、SAS 日時値、またはオブザベーション番号だけで構成されている場合、RULE=オプションは適用されません。また、タイミング値リストが単一の日付キーで構成されている場合も、RULE=オプションは適用されません。RULE=オプションは、AND 値と OR 値を受け入れます。デフォルトは RULE=OR です。

RULE=オプションの例を次に示します。

```
datekeykey JANUARY                / pulse=month;
datekeydef RainyDays='11JUL2013'd '13JUL2013'd '21JUL2013'd;
datekeydef HotDays=  '11JUL2013'd '16JUL2013'd '17JUL2013'd
                     '18JUL2013'd '19JUL2013'd;
datekeydef FridaysInJanuary=JANUARY FRIDAY / rule=and;
datekeydef JanuaryPlusFridays=JANUARY FRIDAY / rule=or;
datekeydef HotandRainyDays=RainyDays HotDays / rule=and;
```

最初のステートメントでは、タイミング値リストが単一の日付キーで構成されているため、RULE=オプションは適用されません。2 番目と 3 番目のステートメントでは、タイミング値リストが SAS 日付値だけで構成されているため、RULE=オプションは適用されません。2 つの SAS 日付値、日時値、またはオブザベーション値の間での演算は常に OR になります。4 番目のステートメントでは、RULE=AND オプションにより、月が 1 月かつ曜日が金曜日の日付だけが特定されます。5 番目のステートメントでは、RULE=OR オプションにより、1 月のすべての日付と曜日が金曜日のすべての日付が特定されます。6 番目のステートメントでは、RULE=AND オプションにより、雨天かつ暑い日が特定されます。RULE=AND は、RainyDays と HotDays の 2 つの日付キーに適用されます。

通常、2 つの別個の期間の間での AND 演算の結果は空の値になります。そのため、別個の期間の間では、常に OR 演算を使用します(例: '13JUL2013'd, '01Mar1990:15:03:00'dt, 3)。

次の表に、各オブザベーションで RULE=オプションがどのように解釈されるのかを示します。

表 17.1 *RULE=オプション値の定義*

RULE=オプション	名前	定義
AND	および	すべての日付キーおよび個々のタイミング値のグループで識別される期間を特定します。
OR	または	任意の日付キーまたは個々のタイミング値のグループで識別される期間を特定します。

SHIFT=*number*

タイミング値 δ をシフトするパルス数を指定します。デフォルトでは、タイミング値をシフトしません(δ= 0)。SHIFT=オプションを使用すると、リストのすべてのタイミング値(日付キーワードによって生成されるタイミング値を含む)がシフトされます。これによって、EASTER で SHIFT=を使用して、復活祭に基づく教会の祝

日を指定できます。たとえば、次のステートメントは、復活祭の 2 日前と定められている聖金曜日を指定しています(Montes 2001)。

```
datekeydef GoodFriday=EASTER / shift=-2 pulse=day;
```

DATEKEYDSOPT ステートメント

データセットの入出力処理を指定したロケールに限定します。

構文

DATEKEYDSOPT LOCALE=<(ONLY)> '*POSIX locale*';

オプション引数の要約

(ONLY)

必須引数

LOCALE=
入力/出力データセットのフィルタに使用するロケールを指定します。

'POSIX locale'
POSIX ロケール値を指定します。ロケール値のデフォルトはありません。

オプション引数

(ONLY)
入力データセットと出力データセットの両方で、指定したロケールだけを処理することを指定します。(ONLY)を指定しない場合、入力データセットと出力データセットの両方で、指定したロケールとデフォルト値(ロケールの指定なし)が処理されます。

DATEKEYKEY ステートメント

ユーザー定義または事前定義された SAS 日付キーを変更するか、別の日付キーから新しい日付キーを作成します。複数の DATEKEYKEY ステートメントを指定できます。

参照項目: “DATEKEYKEY ステートメントの使用” (610 ページ)

構文

DATEKEYKEY <*SAS-variable-name*=> *datekey-keyword* </*qualifier-option(s)*>;

オプション引数の要約

qualifier-option
SAS-variable-name

必須引数

datekey-keyword
日付キーに基づくイベントのデフォルトの SAS 変数名を指定します。

オプション引数

SAS-variable-name
新しい日付キーキーワードの名前を指定します。

ヒント DATEKEYCALENDAR オプションを指定すると、この名前で変数が作成され、キーワードのアクティブ期間と非アクティブ期間が示されます。

qualifier-option
次のオプションを使用できます。

AFTER=(<DURATION=*value*>)
タイミング値の後の日付キー定義を制御するオプションを指定します。DURATION=サブオプションは、AFTER= ( )オプションのかっこ内で使用します。

AFTER=オプションで使用するときには、DURATION はタイミング値の後の日付キーの期間を指定します。

BEFORE=(<DURATION=*value*>)
タイミング値の前の日付キー定義を制御するオプションを指定します。DURATION=サブオプションは、BEFORE= ( )オプションのかっこ内で使用します。

BEFORE=オプションで使用するときには、DURATION はタイミング値の前の日付キーの期間を指定します。

LABEL=*'SAS-label'*
日付キーに関連付けるラベルを指定します。'SAS-label'は、引用符で囲まれたテキスト文字列です。このテキスト文字列には最大 256 文字使用できます。デフォルトのラベルは'*SAS-variable-name*'です。*SAS-variable-name* は、DATEKEYKEY ステートメントで指定した名前です。DATEKEYKEY ステートメントで *SAS-variable-name* が指定されていない場合、事前定義された SAS 日付キーのデフォルトのラベルが使用されます。ラベルは、DATEKEYDATA OUT=データセットに保存されます。

LOCALE=*'POSIX locale'*
日付キーに関連付けるロケールを指定します。ロケールは、POSIX ロケール値にする必要があります。ロケール値のデフォルトはありません。

PERIOD=*interval*
日付キーの度数の間隔を指定します。たとえば、PERIOD=YEAR の場合、年単位パターンで周期的な日付キーが生成されます。PERIOD=オプションを省略すると、日付キーは周期的にはなりません。PERIOD=オプションは、周期的ではないオブザベーション番号や、独自の周期性を持つ日付キーワードには適用されません。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

PULSE=*interval*
日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。日付キーが時間 ID 変数に対して評価される場合、デフォルトのパルスは 1 オブザベーションになります。DURATION=の値を指定せずに、PULSE=オプションを指定した場合、DURATION=の値はゼロに設定されます。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

RULE=*value*
定義されている日付キーに、少なくとも 1 つの日付キーを含む複数のタイミング値があるときに実行するアクションを指定します。日付キーのタイミング値が、SAS 日付、SAS 日時、またはオブザベーション番号だけで構成されている場合、RULE=オプションは適用されません。また、タイミング値リストが単一の日付キーで構成されている場合も、RULE=オプションは適用されません。RULE=オプションは、AND 値と OR 値を受け入れます。デフォルトは RULE=OR です。RULE=オプションの例を次に示します。

```
datekeykey JANUARY / pulse=month;
datekeydef RainyDays='11JUL2013'd '13JUL2013'd '21JUL2013'd;
datekeydef HotDays=  '11JUL2013'd '16JUL2013'd '17JUL2013'd
                     '18JUL2013'd '19JUL2013'd;
datekeydef FridaysInJanuary=JANUARY FRIDAY / rule=and;
datekeydef JanuaryPlusFridays=JANUARY FRIDAY / rule=or;
datekeydef HotandRainyDays=RainyDays HotDays / rule=and;
```

最初のステートメントでは、タイミング値リストが単一の日付キーで構成されているため、RULE=オプションは適用されません。2 番目と 3 番目のステートメントでは、タイミング値リストが SAS 日付値だけで構成されているため、RULE=オプションは適用されません。2 つの SAS 日付値、日時値、またはオブザベーション値の間での演算は常に OR になります。4 番目のステートメントでは、RULE=AND オプションにより、月が 1 月かつ曜日が金曜日の日付だけが特定されます。5 番目のステートメントでは、RULE=OR オプションにより、1 月のすべての日付と曜日が金曜日のすべての日付が特定されます。6 番目のステートメントでは、RULE=AND オプションにより、雨天かつ暑い日が特定されます。RULE=AND は、RainyDays と HotDays の 2 つの日付キーに適用されます。

通常、2 つの別個の期間の間での AND 演算の結果は空の値になります。そのため、別個の期間の間では、常に OR 演算を使用します(例: '13JUL2013'D, '01Mar1990:15:03:00'DT, 3)。

表 17.1 (598 ページ) 各オブザベーションで RULE=オプションがどのように解釈されるのかを説明します。

SHIFT=*number*
タイミング値 δ をシフトするパルス数を指定します。デフォルトでは、タイミング値をシフトしません(δ= 0)。SHIFT=オプションを使用すると、リストのすべてのタイミング値(日付キーワードによって生成されるタイミング値を含む)がシフトされます。

DATEKEYPERIODS ステートメント

入力時間 ID での日付キーのアクティブ期間をリストする変数を書き込みます。アクティブ期間の日付、日時値、またはオブザベーション番号が、関連付けられている日付キーとともにリストされます。

構文

DATEKEYPERIODS OUT=*SAS-data-set*;

必須引数

OUT=*SAS-data-set*
ID ステートメントで指定した ID 情報に基づいて、指定した日付キーのアクティブな日付、日時値、またはオブザベーションを含む出力データセットの名前を指定しま

す。OUT=データセットには、BY ステートメントと ID ステートメントで指定した変数も含まれます。

ID ステートメント

入力/出力データセットのオブザベーションを識別する数値変数を指定します。

参照項目: "ID ステートメントの詳細" (613 ページ)

構文

ID *SAS-variable-name* INTERVAL=*interval* <*option(s)*>;

オプション引数の要約

オプション

必須引数

SAS-variable-name
入力/出力データセットのオブザベーションを識別する数値変数を指定します。*SAS-variable-name* には、SAS 日付値、SAS 時間値、SAS 日時値、またはオブザベーション番号を指定できます。

INTERVAL=*interval*
入力時間 ID の度数を指定します。たとえば、入力データセットの時間 ID が四半期ごとのオブザベーションで構成されている場合は、INTERVAL=QTR を使用します。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

オプション引数

オプション
次のオプションを使用できます。

ALIGN=*option*
出力オブザベーションの識別に使用する SAS 日付の調整を制御します。ALIGN=オプションは、BEGINNING | BEG | B、MIDDLE | MID | M、ENDING | END | E の各値を受け入れます。

デフォルト BEGINNING

END=*option*
データの終わりを表す SAS 日付値、SAS 日時値、または SAS 時間値を指定します。最後の時間 ID 変数値が END=の値よりも小さい場合は、VAR ステートメントの変数が欠損値で拡張されます。最後の時間 ID 変数値が END=の値よりも大きい場合は、変数が切り捨てられます。たとえば、END="&sysdate"d では、自動マクロ変数 SYSDATE を使用して、変数を現在の日付に拡張するか切り捨てます。このオプションと START=オプションを使用して、各 BY グループに関連付けられたデータに同じ数のオブザベーションが含まれるようにすることができます。

FORMAT=*format*
時間 ID 値の SAS フォーマットを指定します。FORMAT=オプションを指定しない場合、デフォルトのフォーマットは INTERVAL=オプションから暗黙的に取得されます。

START=*option*
データの始まりを表す SAS 日付値、SAS 日時値、または SAS 時間値を指定します。最初の時間 ID 変数値が START=の値よりも大きい場合は、VAR ステートメントの変数に欠損値の接頭辞が付けられます。最初の時間 ID 変数値が START=の値よりも小さい場合は、変数が切り捨てられます。このオプションと END=オプションを使用して、各 BY グループに関連付けられたデータに同じ数のオブザベーションが含まれるようにすることができます。

VAR ステートメント

入力変数を出力カレンダ変数データセットにコピーします。VAR ステートメントを省略すると、BY ステートメントまたは ID ステートメントに出現する変数を除く、すべての数値変数が選択されます。

構文

VAR *variable(s)*;

必須引数

variable
出力カレンダ変数データセットにコピーする数値入力変数を指定します。

DATEKEYS プロシジャの使用

DATEKEYS プロシジャの機能概要

次の表に、DATEKEYS プロシジャを制御するステートメントとオプションの概要を示します。

説明	ステートメント	オプション
ステートメント		
BY グループ処理を指定します。	BY	該当なし
カレンダ変数のデータセットを指定します。	DATEKEYCALENDAR	該当なし
日付キーデータセットを指定します。	DATEKEYDATA	該当なし
日付キー定義を指定します。	DATEKEYDEF	該当なし

説明	ステートメント	オプション
事前定義された日付キー定義を使用します。	DATEKEYKEY	該当なし
日付キー別のアクティブ期間のリストを含むデータセットを指定します。	DATEKEYPERIODS	該当なし
時間 ID 変数を指定します。	ID	該当なし
カレンダ変数データセットにコピーする変数を指定します。	VAR	該当なし
データセットオプション		
入力データセットを指定します。	PROC DATEKEYS	DATA=
日付キーのカレンダ変数を含む出力データセットを指定します。	DATEKEYCALENDAR	OUT=
日付キー入力データセットを指定します。	DATEKEYDATA	IN=
日付キー出力データセットを指定します。	DATEKEYDATA	OUT=
日付キー出力データセットを簡略化することを指定します。	DATEKEYDATA	CONDENSE
日付キー出力データセットに日付キーのリストだけを含めることを指定します。	DATEKEYDATA	LIST
日付キー出力データセットに事前定義された SAS 日付キーを含めないことを指定します。	DATEKEYDATA	NODEFAULTS
入力/出力データセットで、指定したロケールだけを処理することを指定します。	DATEKEYDSOPT	LOCALE=
日付キーのアクティブな日付を含む出力データセットを指定します。	DATEKEYPERIODS	OUT=
開始時間 ID 値を指定します。	ID	START=

説明	ステートメント	オプション
終了時間 ID 値を指定します。	ID	END=
ID 変数の出力形式を指定します。	ID	FORMAT=
カレンダ変数の出力形式オプション		
入力時間 ID の終わりを超えてカレンダ変数を拡張します。	PROC DATEKEYS	LEAD=
時間 ID 変数の度数を指定します。	ID	INTERVAL=
間隔調整を指定します。	ID	ALIGN=
その他のオプション		
出力データセットの変数を並べ替え後の順序にすることを指定します。	PROC DATEKEYS	SORTNAMES
エラーメッセージと警告メッセージを制限します。	PROC DATEKEYS	MAXERROR=

日付キー定義

日付キー定義は、参照日付キーに関連付ける期間を定義することを目的としています。これらの期間は、イベントなどの時間に依存する機能を定義する他の SAS プロシジャで解釈されます。日付キー定義の期間は、対象となる期間と比較されます。対象となる期間が日付キー定義の範囲内であれば、適切なアクションが実行されます。日付キー定義は、DATEKEYDATA ステートメントの OUT=オプションを使用することで、出力ファイルに書き込むことができます。

日付キーを定義したら、SAS 変数名を使用して参照します。DATEKEYDATA ステートメントを使用して、日付キー定義を出力ファイルに書き込むと、日付キーは SAS 変数名で識別されます。事前定義された SAS 日付キーと同様に、ユーザー定義の日付キー名を使用してイベントを指定すると、日付キー定義を使用してダミー変数が作成されます。ダミー変数名は日付キーの SAS 参照名と同じです。

各日付キーには、一意の SAS 変数名が必要です。2 つの日付キー定義が同じ名前の場合、次の規則が適用されます。

- 同じ名前を使用する 2 つの DATEKEYDEF ステートメントが存在する場合、2 番目のステートメントが使用されます。

- DATEKEYDEF ステートメントと、DATEKEYDATA ステートメントを使用して指定したデータセットの両方で日付キーが定義されている場合、DATEKEYDEF ステートメントでの定義が使用されます。
- 事前定義された SAS 日付キーではなく、DATEKEYDEF、DATEKKEYKEY、または DATEKEYDATA ステートメントを使用して定義された日付キーが使用されます。

タイミング値リストの仕様の詳細

各 DATEKEYDEF ステートメントは、1 つ以上のタイミング値を使用して定義する必要があります。タイミング値は、リストを使用して指定できます。リストの各項目として、事前定義された SAS 日付キーワード、整数、SAS 日付、SAS 日時値、または DOLIST を指定できます。たとえば、次の DATEKEYDEF ステートメントは、これらの各方法を上記の順序で使用したタイミング値を指定しています。

```
datekeydef datekey1=USINDEPENDENCE 10 '25Dec2000'd
                    '01Mar1990:15:03:00'dt
                    '01Jan2000'd to '01Mar2000'd by month;
```

これらのタイミング値は、"該当する年の 7 月 4 日、時系列またはデータセット内で 10 番目のオブザベーション、2000 年 12 月 25 日、1990 年 3 月 1 日 3:03PM、2000 年 1 月 1 日、2000 年 2 月 1 日、2000 年 3 月 1 日"と解釈されます。

次の 2 つの DATEKEYDEF ステートメントは、同一のタイミング値を指定しています。

```
datekeydef MyFirstDATEKEY='01Jan2000'd to '01Mar2000'd by month;
datekeydef MyNextDATEKEY=('01Jan2000'd, '01Feb2000'd, '01Mar2000'd );
```

タイミング値リストはかっこで囲むことができ、リストの項目はカンマで区切ることができます。数字は常にオブザベーション番号として解釈されます。DOLIST は、オブザベーション番号、SAS 日付、または SAS 日時値に基づくことができます。ただし、リストの最初と 2 番目の値は、同じ型である必要があります。SAS では、常に、2 番目の値の型が最初の値の型と同じであると想定し、その前提でステートメントの解釈を試みます。次のステートメントは、不安定な結果をもたらします。

```
datekeydef baddatekey='01Jan2000'd to '01Mar2000:00:00:00'dt by month;
```

DATEKEYS プロシジャは、予想よりもかなり長いリストを生成するか、メモリ不足で実行できなくなります。

注: DOLIST に日付型、日時型、および整数型を混在させないでください。

次の表に、タイミング値リストで使用できる祝日の日付キーワードとその定義を示します。

表 17.2 *祝日の日付キーワードと定義*

日付キーワード	定義
BOXING	12 月 26 日
CANADA	7 月 1 日
CANADAOBSERVED	7 月 1 日(7 月 1 日が日曜日の場合は 7 月 2 日)
CHRISTMAS	12 月 25 日

日付キーワード	定義
COLUMBUS	10 月の第 2 月曜日
EASTER[1]	復活の主日
FATHERS	6 月の第 3 日曜日
HALLOWEEN	10 月 31 日
LABOR	9 月の第 1 月曜日
MLK	1 月の第 3 月曜日
MEMORIAL	5 月の最後の月曜日
MOTHERS	5 月の第 2 日曜日
N<*n*>W<*w*><MON>YR	NWKDOM(*n*、*w*、*m*、*year*)で指定した日付。*m* は MON で指定される月に対応し、*year* はそのデータに関連する任意の年を表します。 例:N4W5NOVYR は THANKSGIVING と同じです。
NEWYEAR	1 月 1 日
THANKSGIVING	11 月の第 4 木曜日
THANKSGIVINGCANADA	10 月の第 2 月曜日
USINDEPENDENCE	7 月 4 日
USPRESIDENTS	2 月の第 3 月曜日(1971 年以降)
VALENTINES	2 月 14 日
VETERANS	11 月 11 日
VETERANSUSG	月曜日～金曜日のスケジュールに対応する米国政府の振替休日
VETERANSUSPS	月曜日～土曜日のスケジュール(米国郵便局)に対応する米国政府の振替休日
VICTORIA	5 月 24 日の月曜日または 5 月 24 日より前の一番近い月曜日

次の表に、タイミング値リストで使用できる季節的な日付キーワードとその定義を示します。

1 復活祭の日付は、Montes (2001)で説明されている方法を使用して計算されます。

表 17.3 季節的な日付キーワードと定義

日付キーワード	定義
SECOND_1, ...SECOND_60	指定した秒。
MINUTE_1, ...MINUTE_60	指定した分の開始。
HOUR_1, ...HOUR_24	指定した時間の開始。
SUNDAY, ...SATURDAY	時系列でのすべての日曜日など。
WEEK_1, ...WEEK_53	その年の第 *n* 週目の最初の日。PULSE=WEEK.*n* は、*n* NE 1 に対してこの日付をシフトします。
TENDAY_1, ...TENDAY_36	該当する月の 1 日、11 日、または 21 日。
SEMIMONTH_1, ...SEMIMONTH_24	該当する月の 1 日または 16 日。
JANUARY, ...DECEMBER	指定した月の 1 日。
QTR_1, QTR_2, QTR_3, QTR_4	四半期の初日。PULSE=QTR.*n* は、*n* NE 1 に対してこの日付をシフトします。
SEMIYEAR_1, SEMIYEAR_2	半期の初日。PULSE=SEMIYEAR.*n* は、*n* NE 1 に対してこの日付をシフトします。

タイミング値は、ユーザーが指定した用途および関連する時間に対して評価されます。用途と一致するタイミング値を選択します。特に、日付または時間の情報が指定されていない場合、日付と日時のタイミング値は無視され、分析に使用できるのはオブザベーション番号だけになります。

オプションを使用したタイミング値の変更の詳細

修飾子オプションは、各タイミング値で適用される機能の変更を定義します。オプションは、次の順序で適用されます。

1. SHIFT=の値が指定されている場合、SHIFT=の値と PULSE=の値(指定されている場合)を使用してリストのタイミング値がシフトされます。
2. BEFORE=オプションまたは AFTER=オプションで DURATION=の値が指定されている場合、DURATION=と PULSE=の値に基づいてシフトされたタイミング値を中心に連続する間隔が定義されます。
3. PERIOD=の値が指定されている場合、ステップ 1 と 2 の結果であるタイミング値に基づいて周期的な値が生成されます。

タイミング値とオプションに基づくアクティブな間隔の識別の詳細

日付キーが時間 ID に対して評価されると、その時間 ID の間隔に対してアクティブな間隔が識別されます。そのため、アクティブ期間は、時間 ID と日付キー定義の両方に依存します。

シフトされたタイミング値によって指定されたオブザベーション(t_i)は、**INTNX(interval,timing–value,s,'same')**によって生成された日付を含むオブザベーションです。ここで、SHIFT=*s*、PULSE=*interval* です。PULSE=の値が指定されていない場合、デフォルトは PULSE=OBS です。これは、PULSE=*interval* と同じであり、*interval* は時間 ID の間隔を表します。

表 17.4 *時系列に適用したときの日付キーの開始オブザベーションと終了オブザベーションの計算*

BEFORE=(DURATION=*value*)	**PULSE=*value***	**t_b の定義**
ALL	使用不可	t_b=1(データセットの最初のオブザベーション)、または t_b= START=で指定されたオブザベーション
m=0	指定なし	t_b=t_i(シフトされたタイミング値によって指定されたオブザベーション)
m>0	指定なし	t_b=t_i-*m*
m>=0	*interval*	t_b=日付で指定されたオブザベーション **INTNX(*interval*, *timing-value*,–*m*, 'begin')**
AFTER=(DURATION=*value*	PULSE=*value*	t_e の定義
ALL	使用不可	t_e=データセットの最後のオブザベーション、または t_e= END=で指定されたオブザベーション
n=0	指定なし	t_e=t_i(シフトされたタイミング値で指定されたオブザベーション)
n>0	指定なし	t_e=t_i+*n*
n>=0	*interval*	t_e=日付で指定されたオブザベーション **INTNX(*interval*, *timing-value*,*n*,'end')**

次の表に、日付キー定義のアクティブ期間を示します。

表 17.5 日付キー定義のアクティブ期間

BEFORE=(DURATION=*m*)	AFTER=(DURATION=*n*)	アクティブオブザベーション
m=ALL	*n*=ALL	ξ_{it}, for all t
m=finite	*n*=ALL	ξ_{it}, for all $t>=t_b$
m=ALL	*n*=finite	ξ_{it}, for all $t<=t_e$
m=finite	*n*=finite	ξ_{it}, if $t_b<=t<=t_e$

オブザベーションが日付キーのアクティブ期間内にない場合、そのオブザベーションは日付キーのアクションによって変更されません。

アクティブ期間は、常にシフトされたタイミング値で発生します。タイミング値の前に 3 つのオブザベーション、タイミング値に 1 つのオブザベーション、タイミング値の後に 4 つのオブザベーションをそれぞれ指定します。次のように、オブザベーションは合計 3+1+4=8 個になります。

```
datekeydef E1='01JAN1950'd / before=(duration=3)
                             after=(duration=4);
```

この例では、次のように、BEFORE=、AFTER=、PULSE=の各オプションの組み合わせを使用して、タイミング値の前に 3 週、タイミング値に 1 週、タイミング値の後に 4 週指定しています。

```
datekeydef E1='01JAN1950'd / before=(duration=3)
                             after=(duration=4)
                             pulse=week;
```

DURATION=ALL は、アクティブ期間を期間の始まり(BEFORE=)または終わり(AFTER=)まで拡大する必要があることを意味します。一方の DURATION=の値だけを指定した場合、もう一方の値はゼロと見なされます。どちらの DURATION=の値も指定しない場合、両方の DURATION=の値がゼロに設定されます。DURATION=ALL は、日付キー定義データセットで、"A"と表示される特殊な欠損値として表されます。詳細については、*SAS 言語リファレンス: 解説編*の Missing Values を参照してください。

DATEKEYKEY ステートメントの使用

既存の日付キー定義からの新しい日付キーの作成

DATEKEYKEY ステートメントを PROC DATEKEYS とともに使用して、既存の日付キー定義から新しい日付キーを作成し、処理に使用できます。ユーザー定義の日付キーに基づく SAS イベントは、EVENTDS=システムオプションを指定することで、PROC HPFDIAGNOSE と PROC HPFENGINE から直接使用することもできます。

ユーザー定義の日付キー変数には、ユーザーが定義したタイミング値と修飾子が含まれます。事前定義された SAS 日付キーを使用して定義された DATEKEYKEY 変数には、事前定義された日付キーキーワードに関連付けられたタイミング値と修飾子の事前定義されたセットが含まれます。ステートメントオプションを使用して、修飾子を再定義できます。オプションは DATEKEYDEF ステートメントと同じです。“概

念:DATEKEYS プロシジャ" (590 ページ)では、日付キーに基づくイベントのデフォルトの SAS 変数名は日付キーキーワードです。ただし、日付キーに別の SAS 名を指定できます。たとえば、次のステートメントを使用して、事前定義された CHRISTMAS 日付キーの名前を XMAS に変更できます。

```
datekeykey xmas=christmas;
```

事前定義された SAS 日付キーに関連付けられている修飾子を再定義し、日付キーの名前を変更しない場合、事前定義された SAS 日付キーの再定義という影響を及ぼします。この再定義が発生するのは、事前定義された SAS 定義よりもユーザー定義が優先されるためです。次の例では、ハロウィンのパルスが 1 日で、感謝祭のパルスが 1 か月である FALLHOLIDAYS という名前のイベントを生成します。

```
datekeykey thanksgiving / pulse=month;
eventcomb fallholidays=halloween thanksgiving;
```

次の表に、事前定義された SAS 日付キーキーワードを作成する方法を示します。また、それらの事前定義された日付キーのデフォルトの修飾子オプションも示します。

表 17.6 *DATEKEYKEY の事前定義されたイベントキーワードの定義*

変数名または変数名の形式	説明	修飾子オプション
AO<obs>OBS AO<date>D AO<datetime>DT	外れ値	BEFORE=(DURATION=0) AFTER=(DURATION=0)
LS<obs>OBS LS<date>D LS<datetime>DT	レベルシフト	BEFORE=(DURATION=0) AFTER=(DURATION=ALL)
TLS<obs>OBS<n> TLS<date>D<n> TLS<datetime>DT<n>	一時レベルシフト	BEFORE=(DURATION=0) AFTER=(DURATION=<n>)
NLS<obs>OBS NLS<date>D NLS<datetime>DT	負のレベルシフト	BEFORE=(DURATION=0) AFTER=(DURATION=ALL)
CBLS<obs>OBS CBLS<date>D CBLS<datetime>DT	米国国勢調査局のレベルシフト	SHIFT=–1 BEFORE=(DURATION=ALL) AFTER=(DURATION=0)
TC<obs>OBS TC<date>D TC<datetime>DT	一時変更	BEFORE=(DURATION=0) AFTER=(DURATION=ALL)

変数名または変数名の形式	説明	修飾子オプション
<date keyword>	日付パルス	BEFORE=(DURATION=0) AFTER=(DURATION=0)
LINEAR QUAD CUBIC	多項式傾向	BEFORE=(DURATION=ALL) AFTER=(DURATION=ALL) デフォルトのタイミング値は 0 オブザベーション
INVERSE LOG	傾向	BEFORE=(DURATION=0) AFTER=(DURATION=ALL) デフォルトのタイミング値は 0 オブザベーション
<seasonal keywords>	季節的	PULSE=はキーワードに依存 BEFORE=(DURATION=0) AFTER=(DURATION=0) キーワードに基づくタイミング値

ユーザー定義の日付キーの変更または複製

DATEKEYKEY ステートメントを同様の方法で使用して、ユーザー定義の日付キーを変更または複製できます。次の例では、DATEKEYDEF ステートメントを使用して、SPRING という名前の単純なイベントを定義します。DATEKEYKEY ステートメントを使用して、SPRING イベントの定義を変更します。次に、DATEKEYKEY ステートメントを使用して、前に定義した SPRING というユーザーイベントに基づく SPRINGBREAK という名前の新しいイベントを作成します。したがって、この例では、合計 2 つの日付キー(SPRING と SPRINGBREAK)を定義します。DATEKEYKEY ステートメントを使用して、修飾子を変更できます。このステートメントを使用してタイミング値を変更することはできません。

```
datekeydef spring='20mar2005'd;
datekeykey spring / pulse=day;
datekeykey SPRINGBREAK=spring / pulse=week;
```

前出の日付キーを SPRINGHOLIDAYS という名前のデータセットに保存したら、次の例の最初の DATEKEYKEY ステートメントは、SPRING を FirstDayOfSpring という名前の日付キーとして複製しています。2 番目の DATEKEYKEY ステートメントは、SPRINGBREAK の日付キー名の大文字/小文字を変更しています。

```
datekeydata in=springholidays;
datekeykey FirstDayOfSpring=spring;
datekeykey Springbreak=springbreak;
```

前に定義した日付キーを参照する日付キー名では、大文字と小文字は区別されません。ただし、新しい日付キーを作成する際に使用する日付キー名は、DATEKEYDATA OUT=データセットの_NAME_変数で大文字と小文字の区別が保持されます。

SAS 日付キーキーワード

表 17.2 (606 ページ)に記載されている日付キーワードを、事前定義された SAS 日付キーキーワードとして DATEKEYDEF ステートメントで使用できます。タイミング値は、表 17.2 (606 ページ)での定義に従います。デフォルトの修飾子は、表 17.6 (611 ページ)に示されています。表 17.3 (608 ページ)SAS 日付キーキーワードとして使用できる季節的なキーワードを示します。季節的なキーワードのデフォルトの修飾子は、表 17.6 (611 ページ)に示されています。表 17.7 (613 ページ)日付とオブザベーション番号を、AO、LS、TLS、NLS、CBLS、および TC タイプの事前定義された日付キーにどのようにエンコードするかを記述します。

事前定義された SAS 日付キーワードには、アクティブ期間だけが含まれます。

表 17.7 *AO、LS、TLS、NLS、CBLS、TC タイプの DATEKEYKEY 変数名へのデータ情報のエンコーディング*

変数名の形式	例	参照先
AO<int>OBS	AO15OBS	15 番目のオブザベーション
AO<date>D	AO01JAN2000D	'01JAN2000'D
AO<date>h<hr>m<min>s<sec>DT	AO01Jan2000h12m34s56DT	'01Jan2000:12:34:56'DT
TLS<int>OBS<n>	TLS15OBS10	15 番目のオブザベーション
TLS<date>D<n>	TLS01JAN2000D10	'01JAN2000'D
TLS<date>h<hr>m<min>s<sec>DT<n>	TLS01Jan2000h12m34s56DT10	'01Jan2000:12:34:56'DT

ID ステートメントの詳細

ID ステートメントは、SAS 変数名と間隔を受け入れます。ID 変数の値は、SAS 日付値、時間値、日時値、またはオブザベーション番号であることが前提となります。また、ID ステートメントでは、アクティブ期間と非アクティブ期間を調べるために必要な度数も指定します。指定された情報は、DATEKEYCALENDAR および DATEKEYPERIODS ステートメントを使用して出力されるすべての変数に影響します。DATEKEYCALENDAR ステートメントと DATEKEYPERIODS ステートメントのどちらも指定されていない場合、ID ステートメントは処理に影響を及ぼしません。これは、DATEKEYDEF の定義は時間 ID 値および度数とは無関係であるためです。ID ステートメントを指定する場合、INTERVAL=オプションも指定する必要があります。ID ステートメントを指定しない場合は、(BY グループに対する)オブザベーション番号が時間 ID として使用されます。この場合、オブザベーション番号に基づく日付キーのタイミング値だけが入力時間 ID に適用されて、カレンダ変数が作成されます。SAS 日付値または日時値に基づくタイミング値は無視されます。

データセットの出力

データセットの出力の作成

DATEKEYS プロシジャでは、DATEKEYCALENDAR OUT=、DATEKEYDATA OUT=、DATEKEYPERIODS OUT=の各データセットを作成できます。DATEKEYDATA OUT=データセットには、別の SAS プロシジャへの入力に使用できる日付キー定義が含まれます。DATEKEYDATA OUT=ステートメントで LIST オプションを指定すると、出力データセットに使用可能な日付キーのリストが含まれます。DATEKEYCALENDAR OUT=データセットには、入力時間 ID に関連する日付キーのアクティブ期間を示すカレンダインジケータ変数が含まれます。DATEKEYPERIODS OUT=データセットには、入力時間 ID および関連付けられた日付に対応するアクティブな日付キーに関する情報が含まれます。

DATEKEYCALENDAR OUT=データセットの変数の識別

DATEKEYCALENDAR OUT=データセットには、BY ステートメントでリストされている変数、ID 変数、VAR ステートメントで定義されている変数、プロシジャによって生成されたカレンダ変数が含まれます。カレンダインジケータ変数は、入力時間 ID に関連する日付キーのアクティブ期間を示します。DATEKEYCALENDAR ステートメントでは、SUM=オプションを指定できます。この場合、SUM=オプションで指定した変数が DATEKEYCALENDAR OUT=データセットに含まれます。この変数には、カレンダインジケータ変数の合計が含まれます。

DATEKEYDATA OUT=データセットの変数の識別

DATEKEYDATA OUT=データセットには、次の変数が含まれます。CONDENSE オプションのデフォルト値も提供されます。変数内のすべてのオブザベーションがデフォルト値と同じである場合、日付キー定義データセットからその変数を除外できます。

CLASS
すべての日付キーのクラスが DATEKEY であることを指定します。_CLASS_のデフォルトは DATEKEY です。

DATEINTRVL
日付の DOLIST の間隔を指定します。_DATEINTRVL_のデフォルトは間隔なしです。これは、"."で表されます。

DTINTRVL
日時の DOLIST の間隔を指定します。_DTINTRVL_のデフォルトは間隔なしです。これは、"."で表されます。

_DUR_AFTER_
タイミング値の後の期間数を指定します。_DUR_AFTER_のデフォルトは 0 です。

_DUR_BEFORE_
タイミング値の前の期間数を指定します。_DUR_BEFORE_のデフォルトは 0 です。

ENDDATE
DOLIST で使用する最後の日付タイミング値を指定します。_ENDDATE_のデフォルトは日付なしです。これは欠損値で表されます。

ENDDT
DOLIST で使用する最後の日時タイミング値を指定します。_ENDDT_のデフォルトは日時なしです。これは欠損値で表されます。

ENDOBS
DOLIST で使用する最後のオブザベーション番号タイミング値を指定します。_ENDOBS_のデフォルトはオブザベーション番号なしです。これは欠損値で表されます。

KEYNAME
事前定義された日付キーキーワードまたはユーザー定義の日付キーキーワードを指定します。All_KEYNAME_の値は大文字で表示されます。ただし、_KEYNAME_の値がユーザー定義のキーワードを参照している場合、実際の名前では大文字と小文字を混在させることができます。_KEYNAME_のデフォルトはキー名なしです。これは、"."で表されます。

LABEL
日付キーのラベルまたは説明を指定します。ラベルを指定しない場合、デフォルトのラベル値は"."と表示されます。詳細については、*SAS システムオプション: リファレンス*の LABEL システムオプションを参照してください。

LOCALE
日付キーのロケールを指定します。_LOCALE_値は有効な POSIX ロケール値です。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*の LOCALE システムオプションを参照してください。デフォルトはありません。

NAME
日付キーの参照名を指定します。_NAME_は、保持されている文字種(大文字/小文字)で表示されます。_NAME_は SAS 変数名であるため、任意の文字種を使用して日付キーを参照できます。_NAME_変数は必須です。デフォルトはありません。

OBSINTRVL
オブザベーション番号の DOLIST の間隔の長さを指定します。_OBSINTRVL_のデフォルトは間隔なしです。これは、"."で表されます。

PERIOD
日付キーを繰り返す必要がある度数の間隔を指定します。この値がない場合、日付キーは周期的にはなりません。_PERIOD_のデフォルトは間隔なしです。これは、"."で表されます。

PULSE
DURATION の値の単位を定義する間隔を指定します。_PULSE_のデフォルトは間隔なし(1 オブザベーション)です。これは、"."で表されます。

RULE
日付キーのタイミング値を組み合わせるときに使用する規則を指定します。日付キーの_RULE_のデフォルトは OR です。

SHIFT
タイミング値をシフトする PULSE=interval の数を指定します。シフトは、正(時間を進める)にすることも負(時間を戻す)にすることもできます。PULSE=を指定しない場合、オブザベーションでシフトが発生します。_SHIFT_のデフォルトは 0 です。

STARTDATE
日付タイミング値または DOLIST で使用する最初の日付タイミング値を指定します。_STARTDATE_のデフォルトは日付なしです。これは欠損値で表されます。

STARTDT
日時タイミング値または DOLIST で使用する最初の日時タイミング値を指定します。_STARTDT_のデフォルトは日時なしです。これは欠損値で表されます。

STARTOBS
オブザベーション番号タイミング値または DOLIST で使用する最初のオブザベーション番号タイミング値を指定します。_STARTOBS_のデフォルトはオブザベーション番号なしです。これは欠損値で表されます。

DATEKEYPERIODS OUT=データセットの変数の識別

DATEKEYPERIODS OUT=データセットには、次の変数が含まれます。

LOCALE
日付キーのロケールを指定します。_LOCALE_値は有効な POSIX ロケール値です。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*の LOCALE システムオプションを参照してください。デフォルトはありません。

NAME
日付キーの参照名を指定します。_NAME_は、保持されている文字種(大文字/小文字)で表示されます。_NAME_は SAS 変数名であるため、任意の文字種を使用して日付キーを参照できます。_NAME_変数は必須です。デフォルトはありません。

STARTDATE
日付タイミング値または DOLIST で使用する最初の日付タイミング値を指定します。_STARTDATE_のデフォルトは日付なしです。これは欠損値で表されます。

TIME ID
日付を指定します。

LIST オプションを使用した日付キーのリストの作成

DATEKEYDATA ステートメントで LIST オプションを使用して、日付キーのリストを作成できます。DATEKEYDATA OUT=ステートメントで LIST オプションを指定すると、出力データセットが変更されます。LIST オプションは、データセットで定義されている使用可能な日付キーをリストするデータセットを作成することを目的としています。LIST オプションを指定すると、_LOCALE_変数、_NAME_変数、および_LABEL_変数だけがデータセットに含まれます。データセットには、日付キーごとに 1 つのオブザベーションが含まれます。

BY ステートメント変数のデータセットの作成

DATEKEYPERIODS ステートメントを使用して、BY ステートメントのデータセットを作成できます。DATEKEYPERIODS OUT=データセットには、次の項目が含まれます。

- BY ステートメントでリストされている変数
- 入力時間 ID に関連付けられているアクティブな日付キーのリスト
- アクティブな日付キーに関連付けられている ID 変数の日付
- 時間 ID のアクティブ期間に関連付けられている開始日、日時、またはオブザベーション値

DATEKEYPERIODS OUT=データセットには、BY 変数と ID 変数に加え、次の変数も含まれます。

NAME
日付キーの参照名を指定します。_NAME_は、保持されている文字種(大文字/小文字)で表示されます。表示される日付キーは、ID 変数とオブザベーションの _STARTDATE_、_STARTDT_、または_STARTOBS_の値で指定された日付に対してアクティブです。

時間 ID 変数に基づいて、次のいずれかの変数が含まれます。

STARTDATE
時間 ID 変数の値に関連する INTERVAL=オプションで指定された期間の始まりに関連付ける日付を指定します。

STARTDT
時間 ID 変数の値に関連する INTERVAL=オプションで指定された期間の始まりに関連付ける日時を指定します。

STARTOBS
時間 ID 変数の値に関連する INTERVAL=オプションで指定された期間の始まりに関連付けるオブザベーション番号を指定します。

書き込まれる出力

DATEKEYS プロシジャには、ログに記録された警告メッセージとエラーメッセージ以外に書き込まれる出力はありません。

例: DATEKEYS プロシジャ

例 1: 日付キー定義データセットの作成方法

要素: PROC DATEKEYS ステートメントオプション
ID
DATEKEYDEF
DATEKEYKEY
DATEKEYDATA
DATEKEYPERIODS

詳細

この例では、2 つの方法で日付キー定義データセットを作成します。最初の方法では、DATA ステップを使用して、商品が販売された金曜日の日付キーデータセットを作成します。2 番目の方法では、DATEKEYS プロシジャを使用して、日付キー定義データセットを作成します。TimeSeriesDates データセットには、時系列用の時間 ID 変数が含まれます。

DATA ステップの WhiteSaleDates で定義される日付は、DATEKEYS プロシジャを使用して定義できます。アクティブな日付は、アウトプット 17.1 (619 ページ)に示すデータセット内の日付と同じです。ただし、DATEKEYS プロシジャでの定義は連続的です。

プログラム:DATA ステップの使用

```
options eventds=(nodefaults);

data TimeSeriesDates(keep=date);
   set sashelp.citiday;
   format date date.;
run;

data WhiteSaleDates(keep=_name_ _startdate_);
   set TimeSeriesDates;
   _name_='WhiteSale';
   if (month(date)=1) then do;
      if (year(date)=1991 or year(date)=1992) then do;
         if (weekday(date)=6) then do;
            _startdate_=date;
         end;
         else delete;
         end;
      else delete;
      end;
   else delete;
   format _startdate_ date.;
run;

proc print data=WhiteSaleDates;
run;
```

プログラムの説明

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。NODEFAULTS オプションは、デフォルトのイベント定義を使用しないことを指定します。event-data-set リストで指定されたイベントだけが使用されます。

```
options eventds=(nodefaults);
```

TimeSeriesDates データセットを作成します。 この例のデータセットを作成するには、DATA ステップで sashelp.citiday を使用します。

```
data TimeSeriesDates(keep=date);
   set sashelp.citiday;
   format date date.;
run;
```

EVENTDS=システムオプションで使用するために、WhiteSalesDates という日付キーデータセットを作成します。 DATA ステップを使用して日付キーデータセットを作成するときは、特定の日付キーを定義するすべてのオブザベーションが連続している必要があります。必要に応じて、SORT プロシジャを使用して、_NAME_変数でデータセットを並べ替えます。DATA ステップでは、1991 年 1 月から 1992 年 1 月までの間に販売された商品を記述しています。ユーザー定義の日付キーは、商品が販売された金曜日を識別します。

```
data WhiteSaleDates(keep=_name_ _startdate_);
   set TimeSeriesDates;
   _name_='WhiteSale';
   if (month(date)=1) then do;
      if (year(date)=1991 or year(date)=1992) then do;
```

```
         if (weekday(date)=6) then do;
            _startdate_=date;
         end;
         else delete;
         end;
      else delete;
      end;
   else delete;
   format _startdate_ date.;
run;

proc print data=WhiteSaleDates;
run;
```

出力:出力:HTML

アウトプット 17.1 *ユーザー定義の日付キーデータセット*

The SAS System

Obs	_NAME_	_STARTDATE_
1	WhiteSale	04JAN91
2	WhiteSale	11JAN91
3	WhiteSale	18JAN91
4	WhiteSale	25JAN91
5	WhiteSale	03JAN92
6	WhiteSale	10JAN92
7	WhiteSale	17JAN92
8	WhiteSale	24JAN92
9	WhiteSale	31JAN92

プログラム:DATEKEYS プロシジャの使用

```
proc datekeys data=sashelp.citiday;

   id date interval=day;

   datekeydef Years1991_1992='01JAN1991'd / pulse=year after=(duration=1);

   datekeykey January / pulse=month;

   datekeydef WhiteSale=JANUARY FRIDAY Years1991_1992 / rule=and;

   datekeydata out=WhiteSaleDefinitions condense;

   datekeyperiods out=WhiteSaleActiveDates;

run;

proc print data=WhiteSaleActiveDates(where=(_name_='WhiteSale'));
run;
```

```
proc print data=WhiteSaleDefinitions;
run;
```

プログラムの説明

DATEKEYS プロシジャを開始します。 DATA=オプションには、ID ステートメントで使用される変数が含まれます。

```
proc datekeys data=sashelp.citiday;
```

時間 ID 変数を指定します。 ID ステートメントは、データセットのオブザベーションを識別する数値変数の名前を指定します。ID ステートメントを使用する場合は、INTERVAL=オプションを使用する必要があります。この例では、INTERVAL=day です。

```
id date interval=day;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。PULSE=は、日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

```
datekeydef Years1991_1992='01JAN1991'd / pulse=year after=(duration=1);
```

日付キーを変更するか、新しい日付キーを作成します。 タイミング値リストで January を使用しているので、1 月 1 日を表します。DATEKEYKEY を PULSE=MONTH とともに使用すると、1 月全体を対象とする日付キーが作成されます。

```
datekeykey January / pulse=month;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。RULE=AND オプションにより、1 月の金曜日を特定します。

```
datekeydef WhiteSale=JANUARY FRIDAY Years1991_1992 / rule=and;
```

日付キーを出力データセットに書き込みます。 DATEKEYDATA OUT=データセットには、WhiteSale 日付キーの定義が含まれます。CONDENSE オプションは、出力データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

```
datekeydata out=WhiteSaleDefinitions condense;
```

入力時間 ID での日付キーのアクティブ期間をリストする変数を書き込みます。 アウトプット 17.2 (621 ページ)に示すように、DATEKEYPERIODS OUT=データセットには、WhiteSale 日付キーのアクティブ期間が含まれます。

```
datekeyperiods out=WhiteSaleActiveDates;
```

この例を実行します。 RUN ステートメントにより、プログラムが実行されます。

```
run;
```

アクティブな販売日データセットを書き込みます。 アウトプット 17.2 (621 ページ)に結果が示されます。

```
proc print data=WhiteSaleActiveDates(where=(_name_='WhiteSale'));
run;
```

WhiteSale 日付キーの定義を書き込みます。 アウトプット 17.3 (621 ページ)に結果が示されます。

```
proc print data=WhiteSaleDefinitions;
```

```
run;
```

HTML 出力

アウトプット 17.2 *日付キーのアクティブ期間データセットの出力*

The SAS System

Obs	_LOCALE_	DATE	_NAME_	_STARTDATE_
557	.	04JAN1991	WhiteSale	04JAN1991
558	.	11JAN1991	WhiteSale	11JAN1991
559	.	18JAN1991	WhiteSale	18JAN1991
560	.	25JAN1991	WhiteSale	25JAN1991
561	.	03JAN1992	WhiteSale	03JAN1992
562	.	10JAN1992	WhiteSale	10JAN1992
563	.	17JAN1992	WhiteSale	17JAN1992
564	.	24JAN1992	WhiteSale	24JAN1992
565	.	31JAN1992	WhiteSale	31JAN1992

次の出力は、WhiteSale 日付キーの DATEKEYDATA OUT=データセットの定義を示しています。

アウトプット 17.3 *日付キー定義データセットの出力*

The SAS System

Obs	_NAME_	_CLASS_	_KEYNAME_	_STARTDATE_	_PULSE_	_DUR_AFTER_	_RULE_
1	JANUARY	DATEKEY	JANUARY	.	MONTH	0	OR
2	Years1991_1992	DATEKEY	.	01JAN1991	YEAR	1	OR
3	WhiteSale	DATEKEY	JANUARY	.	.	0	AND
4	WhiteSale	DATEKEY	FRIDAY	.	.	0	AND
5	WhiteSale	DATEKEY	YEARS1991_1992	.	.	0	AND

例 2: 他の SAS プロシジャでのユーザー定義の SAS 日付キーキーワードの直接使用

要素: PROC DATEKEYS ステートメント
DATEKEYDEF
DATEKEYKEY
DATEKEYDATA
システムオプション

EVENTDS=

その他のプロシジャ

HPFDIAGNOSE

詳細

プロシジャが EVENT ステートメントをサポートしている場合は、PROC HPFEVENTS を使用せずに、“例 1: 日付キー定義データセットの作成方法” (617 ページ) に示すいずれかのデータセットを使用できます。PROC HPFEVENTS を使用すると、ユーザー定義の日付キーを使用できます。ユーザー定義の日付キーをイベントとして指定すると、イベントが自動的に作成されます。ただし、EVENTDS=システムオプションを使用して、ユーザー定義の日付キー定義データセットが指定されていることが前提となります。この例では、“例 1: 日付キー定義データセットの作成方法” (617 ページ)のデータセットを使用し、PROC HPFDIAGNOSE で EVENT ステートメントを使用します。

PROC HPFDIAGNOSE の出力は、WhiteSale イベントを含むモデルが選択されたことを示しています。このイベントは適していませんでしたが、REQUIRED=YES が指定されていました。REQUIRED=YES は、モデルの診断が失敗しない限り、イベントをモデルに含めることを指定します。

プログラム

```
proc datekeys;

   datekeydef Years1991_1992='01JAN1991'd / pulse=year after=(duration=1);

   datekeykey JANUARY / pulse=month;

   datekeydef WhiteSale=JANUARY FRIDAY Years1991_1992 / rule=and;

   datekeydata out=WhiteSaleDefinitions condense;

run;

options eventds=(WhiteSaleDefinitions);

proc hpfdiagnose data=sashelp.citiday
     print=all;

   id date interval=day;

   forecast snysecm;

   event WhiteSale / required=yes;

   arimax;

run;
```

プログラムの説明

DATEKEYS プロシジャを開始します。

```
proc datekeys;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。PULSE=は、日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

```
   datekeydef Years1991_1992='01JAN1991'd / pulse=year after=(duration=1);
```

日付キーを変更するか、新しい日付キーを作成します。 DATEKEYKEY ステートメントは、ユーザー定義または事前定義された SAS 日付キーを変更するか、別の日付キーから新しい日付キーを作成します。PULSE=は、日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

```
datekeykey JANUARY / pulse=month;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。RULE=AND オプションにより、1991 年と 1992 年の 1 月の金曜日を特定します。

```
datekeydef WhiteSale=JANUARY FRIDAY Years1991_1992 / rule=and;
```

日付キーを出力データセットに書き込みます。 CONDENSE オプションは、出力データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

```
datekeydata out=WhiteSaleDefinitions condense;
```

DATEKEYS プロシジャを実行します。

```
run;
```

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。

```
options eventds=(WhiteSaleDefinitions);
```

HPFDIAGNOSE プロシジャを開始します。 HPFDIAGNOSE プロシジャは、時系列の統計特性を自動的に診断し、適切なモデルを特定します。

```
proc hpfdiagnose data=sashelp.citiday
     print=all;
```

ID ステートメントを指定します。 ID ステートメントは、入力/出力データセットのオブザベーションを識別する数値変数の名前を指定します。ID 変数の値は SAS 日付値です。また、ID ステートメントでは、時系列に関連付ける必要な度数も指定します。INTERVAL=オプションは、入力時間 ID の度数を指定します。

```
id date interval=day;
```

データセットの診断する変数をリストします。 FORECAST ステートメントは、DATA=オプションで指定されたデータセットの診断対象となる変数をリストします。これらの変数は、HPFENGINE プロシジャで予測する従属変数または応答変数です。

```
forecast snysecm;
```

イベントの名前を指定します。 EVENT ステートメント名によってイベントが識別されます。REQUIRED オプションは、モデルの診断が失敗しない限り、イベントをモデルに含めることを指定します。

```
event WhiteSale / required=yes;
```

適切な ARIMAX 仕様を見つけます。 ARIMAX モデルは、イベントを含め、傾向変動要因、季節変動要因、および回帰変数をモデル化します。HPFDIAGNOSE プロシジャは、最初に断続性テストを実行します。系列が断続的でない場合、ARIMAX モデルはそのデータに適しています。

```
arimax;
```

HPFDIAGNOSE プロシジャを実行します。

```
run;
```

HTML 出力

次の出力は結果を示しています。

アウトプット 17.4 *PROC HPFDIAGNOSE での EVENT ステートメントの使用*

The SAS System

The HPFDIAGNOSE Procedure

Variable Information	
Name	SNYSECM
Label	STOCK MKT INDEX:NYSE COMPOSITE, (WSJ)
First	01JAN1988
Last	05FEB1992
Number of Observations Read	1497

Seasonal Dickey-Fuller Unit Root Test(Seasonality=7)				
Type	Rho	Pr < Rho	Tau	Pr < Tau
Zero Mean				
Single Mean				

Dickey-Fuller Unit Root Test Summary				
Variable	Seasonality	Zero Mean	Mean	Trend
SNYSECM	1	NO	NO	NO

Seasonal Dickey-Fuller Unit Root Test Summary				
Variable	Seasonality	Zero Mean	Mean	Trend
SNYSECM	7	NO	NO	

ARIMA Model Specification												
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Model Criterion	Statistic	Status
SNYSECM	NONE	YES	0	0	4	0	0	2	7	RMSE	2.7772	OK

ARIMA Event Selection				
Event Name	Selected	d	D	Status
WHITESALE	REQUIRED	0	0	Not Improved

ARIMA Outlier Selection					
Variable	Type	Obs	Time	Chi-Square	Approx Pr > ChiSq
SNYSECM	LS	677	07NOV1989	334.86	<.0001
	LS	1138	11FEB1991	183.43	<.0001

ARIMA Model Specification After Adjusting for Events and Outliers														
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Outlier	Event	Model Criterion	Statistic	Status
SNYSECM	NONE	YES	0	0	4	0	0	2	7	2	1	RMSE	2.2440	OK

例 3: DATEKEYS プロシジャを使用したカレンダ変数の取得

要素: PROC DATEKEYS ステートメント
DATEKEYDEF
DATEKEYKEY
DATEKEYDATA
ID
DATEKEYCALENDAR
システムオプション
EVENTDS=

詳細

“概念:DATEKEYS プロシジャ” (590 ページ)に示す日付キー定義について考えてみます。DATEKEYS プロシジャを使用して、MyHolidays データセットで定義された日付キー定義のアクティブ期間を示すカレンダ変数を作成できます。

プログラム

```
options eventds=(nodefaults);

data Year2010;
   do date='01JAN2010'd to '31DEC2010'd;
      output;
   end;
   format date date.;
run;

proc datekeys;

   datekeydef SuperBowl=
                '15JAN1967'd '14JAN1968'd '12JAN1969'd '11JAN1970'd
    '17JAN1971'd '16JAN1972'd '14JAN1973'd '13JAN1974'd '12JAN1975'd
    '18JAN1976'd '09JAN1977'd '15JAN1978'd '21JAN1979'd '20JAN1980'd
    '25JAN1981'd '24JAN1982'd '30JAN1983'd '22JAN1984'd '20JAN1985'd
    '26JAN1986'd '25JAN1987'd '31JAN1988'd '22JAN1989'd '28JAN1990'd
    '27JAN1991'd '26JAN1992'd '31JAN1993'd '30JAN1994'd '29JAN1995'd
    '28JAN1996'd '26JAN1997'd '25JAN1998'd '31JAN1999'd '30JAN2000'd
    '28JAN2001'd '03FEB2002'd '26JAN2003'd '01FEB2004'd '06FEB2005'd
    '05FEB2006'd '04FEB2007'd '03FEB2008'd '01FEB2009'd '07FEB2010'd
    '06FEB2011'd '05FEB2012'd '03FEB2013'd '02FEB2014'd
                         / pulse=day;

   datekeydef GoodFriday=Easter / shift=-2 pulse=day;

   datekeykey EasterMonday=Easter / shift=1 pulse=day;

   datekeydata out=MyHolidays condense;

run;

options eventds=(MyHolidays);

proc datekeys data=Year2010;

   id date interval=day;

   datekeycalendar out=MyHolidaysIn2010;
```

```
run;

proc print data=MyHolidaysIn2010(where=(month(date)=2));

run;

proc print data=MyHolidaysIn2010(where=(month(date)=4));
run;
```

プログラムの説明

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。NODEFAULTS オプションは、デフォルトのイベント定義を使用しないことを指定します。event-data-set リストで指定されたイベントだけが使用されます。

```
options eventds=(nodefaults);
```

SAS データセットを作成します。 一連の日付を含む Year2010 データセットを作成します。

```
data Year2010;
   do date='01JAN2010'd to '31DEC2010'd;
      output;
   end;
   format date date.;
run;
```

DATEKEYS プロシジャを開始します。

```
proc datekeys;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。PULSE=は、日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

```
   datekeydef SuperBowl=
                '15JAN1967'd '14JAN1968'd '12JAN1969'd '11JAN1970'd
    '17JAN1971'd '16JAN1972'd '14JAN1973'd '13JAN1974'd '12JAN1975'd
    '18JAN1976'd '09JAN1977'd '15JAN1978'd '21JAN1979'd '20JAN1980'd
    '25JAN1981'd '24JAN1982'd '30JAN1983'd '22JAN1984'd '20JAN1985'd
    '26JAN1986'd '25JAN1987'd '31JAN1988'd '22JAN1989'd '28JAN1990'd
    '27JAN1991'd '26JAN1992'd '31JAN1993'd '30JAN1994'd '29JAN1995'd
    '28JAN1996'd '26JAN1997'd '25JAN1998'd '31JAN1999'd '30JAN2000'd
    '28JAN2001'd '03FEB2002'd '26JAN2003'd '01FEB2004'd '06FEB2005'd
    '05FEB2006'd '04FEB2007'd '03FEB2008'd '01FEB2009'd '07FEB2010'd
    '06FEB2011'd '05FEB2012'd '03FEB2013'd '02FEB2014'd
                          / pulse=day;
```

日付キーを定義します。 DATEKEYDEF ステートメントは、GoodFriday という名前の日付キーを定義し、Easter という名前の日付キーのタイミング値と等しいことを指定します。PULSE=オプションを指定すると、DURATION=の値がゼロに設定されます。SHIFT=は、タイミング値をシフトするパルス数を指定します。SHIFT=オプションを使用すると、すべてのタイミング値がシフトされます。

```
   datekeydef GoodFriday=Easter / shift=-2 pulse=day;
```

日付キーを変更するか、新しい日付キーを作成します。 EasterMonday という日付キー名を指定した DATEKEYKEY ステートメントの値は、Easter という名前の日付キーです。Easter を日付キーのタイミング値として扱うときは、タイミング値の定義だけが使用さ

れ、修飾子(SHIFT や PULSE など)は定義には含まれません。Easter を日付キーとして扱うときは、選択した修飾子(SHIFT や PULSE など)が新しい定義に含まれます。PULSE=オプションを指定すると、DURATION=の値がゼロに設定されます。SHIFT=は、タイミング値をシフトするパルス数を指定します。SHIFT=オプションを使用すると、すべてのタイミング値がシフトされます。

```
datekeykey EasterMonday=Easter / shift=1 pulse=day;
```

日付キーを出力データセットに書き込みます。 DATEKEYDATA ステートメントは、MyHolidays という名前の出力データセットに日付キーを書き込みます。CONDENSE オプションは、出力データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

```
datekeydata out=MyHolidays condense;
```

DATEKEYS プロシジャを実行します。

```
run;
```

EVENTSDS=システムオプションを設定します。 EVENTSDS=システムオプションは、イベントを定義するデータセットを指定します。

```
options eventds=(MyHolidays);
```

DATEKEYS プロシジャを開始します。 DATA=オプションは、ID ステートメントで使用される変数を含むデータセットの名前を指定します。

```
proc datekeys data=Year2010;
```

時間 ID 変数を指定します。 ID ステートメントは、データセットのオブザベーションを識別する数値変数の名前を指定します。ID ステートメントを使用する場合は、INTERVAL=オプションを使用する必要があります。この例では、INTERVAL=day です。

```
id date interval=day;
```

日付キーのアクティブ期間を示す変数を含むデータセットを作成します。 DATEKEYCALENDAR OUT=ステートメントは、作成されるデータセットを指定します。このデータセットには、ID ステートメントで指定した情報に基づくカレンダ変数が含まれます。DATEKEYCALENDAR OUT=データセットには、MyHolidays データセットで定義された祝日を識別するカレンダ変数が含まれます。アウトプット 17.5 (630 ページ)には、2010 年 2 月の値が示されます。アウトプット 17.6 (631 ページ)には、2010 年 4 月の値が示されます。DATEKEYCALENDAR OUT=データセットの MyHolidaysIn2010 には、入力時間 ID に対応するアクティブ期間と非アクティブ期間だけが含まれます。この入力時間 ID は、度数が日単位であり、2010 年全体にわたるため、結果は 2010 年のカレンダ変数になります。

```
datekeycalendar out=MyHolidaysIn2010;
run;
```

---

2 月の MyHolidaysIn2010 データセットの出力を書き込みます。アウトプット 17.5 (630 ページ)に結果が示されます。

```
proc print data=MyHolidaysIn2010(where=(month(date)=2));
```

DATEKEYS プロシジャを実行します。

```
run;
```

---

4 月の MyHolidaysIn2010 データセットの出力を書き込みます。アウトプット 17.6 (631 ページ)に結果が示されます。

```
proc print data=MyHolidaysIn2010(where=(month(date)=4));
run;
```

HTML 出力

アウトプット 17.5 *2010 年の日単位の時間 ID に基づく 2 月の祝日カレンダ*

The SAS System

Obs	date	SuperBowl	GoodFriday	EasterMonday
32	01FEB2010	0	0	0
33	02FEB2010	0	0	0
34	03FEB2010	0	0	0
35	04FEB2010	0	0	0
36	05FEB2010	0	0	0
37	06FEB2010	0	0	0
38	07FEB2010	1	0	0
39	08FEB2010	0	0	0
40	09FEB2010	0	0	0
41	10FEB2010	0	0	0
42	11FEB2010	0	0	0
43	12FEB2010	0	0	0
44	13FEB2010	0	0	0
45	14FEB2010	0	0	0
46	15FEB2010	0	0	0
47	16FEB2010	0	0	0
48	17FEB2010	0	0	0
49	18FEB2010	0	0	0
50	19FEB2010	0	0	0
51	20FEB2010	0	0	0
52	21FEB2010	0	0	0
53	22FEB2010	0	0	0
54	23FEB2010	0	0	0
55	24FEB2010	0	0	0
56	25FEB2010	0	0	0
57	26FEB2010	0	0	0
58	27FEB2010	0	0	0
59	28FEB2010	0	0	0

アウトプット 17.6 2010 年の日単位の時間 ID に基づく 4 月の祝日カレンダ

The SAS System

Obs	date	SuperBowl	GoodFriday	EasterMonday
91	01APR2010	0	0	0
92	02APR2010	0	1	0
93	03APR2010	0	0	0
94	04APR2010	0	0	0
95	05APR2010	0	0	1
96	06APR2010	0	0	0
97	07APR2010	0	0	0
98	08APR2010	0	0	0
99	09APR2010	0	0	0
100	10APR2010	0	0	0
101	11APR2010	0	0	0
102	12APR2010	0	0	0
103	13APR2010	0	0	0
104	14APR2010	0	0	0
105	15APR2010	0	0	0
106	16APR2010	0	0	0
107	17APR2010	0	0	0
108	18APR2010	0	0	0
109	19APR2010	0	0	0
110	20APR2010	0	0	0
111	21APR2010	0	0	0
112	22APR2010	0	0	0
113	23APR2010	0	0	0
114	24APR2010	0	0	0
115	25APR2010	0	0	0
116	26APR2010	0	0	0
117	27APR2010	0	0	0
118	28APR2010	0	0	0
119	29APR2010	0	0	0
120	30APR2010	0	0	0

例 4: DATEKEYDSOPT ステートメントを使用したデータセットのフィルタ

要素: PROC DATEKEYS ステートメント
DATEKEYKEY
DATEKEYDEF
DATEKEYDATA
ID
DATEKEYDSOPT
DATEKEYCALENDAR

システムオプション
EVENTDS=

詳細

DATEKEYS プロシジャを使用して、入力データセットをフィルタし、指定したロケールに関連付けられた日付キーデータだけを含むデータセットを作成できます。DATEKEYDSOPT ステートメントは、DATEKEYDATA、DATEKEYPERIODS、および DATEKEYCALENDAR ステートメントで指定したデータセットに適用されます。

この例では、Canada、CanadaObserved、Boxing の各祝日の英語/カナダの日付キー定義を作成します。また、USINDEPENDENCE の英語/米国の日付キー定義も作成します。NewYearsEve の定義では、ロケールは指定しません。

プログラム

```
options eventds=(nodefaults);

data December2010;
   do date='01DEC2010'd to '31DEC2010'd;
      output;
   end;
   format date date.;
run;

proc datekeys;

   datekeykey Canada / locale=en_CA;
   datekeykey CanadaObserved / locale=en_CA;
   datekeykey Boxing / locale='en_CA';
   datekeykey USINDEPENDENCE / locale=en_US;

   datekeydef NewYearsEve=NEWYEAR / shift=-1 pulse=day;

   datekeydata out=MyHolidays condense;

run;

options eventds=(MyHolidays);

proc datekeys data=December2010;

   id date interval=day;

  datekeydsopt locale=en_CA;

   datekeycalendar out=AllCAHolidaysInDecember2010;

run;
```

```
proc datekeys data=December2010;

   id date interval=day;

   datekeydsopt locale=(ONLY)en_CA;

   datekeycalendar out=OnlyCAHolidaysInDecember2010;

run;

proc print data=MyHolidays;
run;

proc print data=AllCAHolidaysInDecember2010;
run;

proc print data=OnlyCAHolidaysInDecember2010;
run;
```

プログラムの説明

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。NODEFAULTS オプションは、デフォルトのイベント定義を使用しないことを指定します。event-data-set リストで指定されたイベントだけが使用されます。

```
options eventds=(nodefaults);
```

SAS データセットを作成します。 一連の日付を含む December2010 データセットを作成します。

```
data December2010;
   do date='01DEC2010'd to '31DEC2010'd;
      output;
   end;
   format date date.;
run;
```

DATEKEYS プロシジャを開始します。

```
proc datekeys;
```

新しい日付キーを作成します。 DATEKEYKEY ステートメントは、新しい日付キーを作成します。LOCALE=オプションは、日付キーに関連付けるロケールを指定します。ロケールは、POSIX ロケール値にする必要があります。

```
   datekeykey Canada / locale=en_CA;
   datekeykey CanadaObserved / locale=en_CA;
   datekeykey Boxing / locale='en_CA';
   datekeykey USINDEPENDENCE / locale=en_US;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを識別します。SHIFT=オプションは、タイミング値 6 をシフトするパルス数を指定します。このオプションを使用すると、リストのすべてのタイミング値(日付キーワードによって生成されるタイミング値を含む)がシフトされます。PULSE=オプションは使用する間隔を指定します。

```
   datekeydef NewYearsEve=NEWYEAR / shift=-1 pulse=day;
```

日付キーを出力データセットに書き込みます。 DATEKEYDATA ステートメントは、出力データセットを作成します。CONDENSE オプションは、出力データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

```
datekeydata out=MyHolidays condense;
```

DATEKEYS プロシジャを実行します。

```
run;
```

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。

```
options eventds=(MyHolidays);
```

DATEKEYS プロシジャを開始します。 DATEKEYS プロシジャは、時間計算に関連付けられた日付キーを作成します。DATA=オプションには、ID ステートメントで使用される変数が含まれます。

```
proc datekeys data=December2010;
```

ID ステートメントを指定します。 ID ステートメントは、入力/出力データセットのオブザベーションを識別する数値変数の名前を指定します。ID 変数の値は SAS 日付値です。また、ID ステートメントでは、時系列に関連付ける必要な度数も指定します。

```
id date interval=day;
```

特定のロケールのデータセットを作成します。 DATEKEYDSOPT ステートメントは、データセットの処理を指定したロケールに限定します。このステートメントを使用すると、ロケール値 en_CA が、指定された日付キー定義とロケールが指定されていない日付キー定義を使用して、AllCAHolidaysInDecember2010 データセットが作成されます。

```
datekeydsopt locale=en_CA;
```

日付キーのアクティブ期間を示す変数を含むデータセットを作成します。
DATEKEYCALENDAR ステートメントは、日付キーのアクティブ期間を示す変数を書き込みます。OUT=オプションは、作成されるデータセットを表します。このデータセットには、ID ステートメントで指定した情報に基づくカレンダ変数が含まれます。

```
datekeycalendar out=AllCAHolidaysInDecember2010;
```

DATEKEYS プロシジャを実行します。

```
run;
```

DATEKEYS プロシジャを開始します。 DATA=オプションには、ID ステートメントで使用される変数が含まれます。

```
proc datekeys data=December2010;
```

ID ステートメントを指定します。 ID ステートメントは、入力/出力データセットのオブザベーションを識別する数値変数の名前を指定します。ID 変数の値は SAS 日付値です。また、ID ステートメントでは、時系列に関連付ける必要な度数も指定します。

```
id date interval=day;
```

特定のロケールのデータセットを作成します。 DATEKEYDSOPT ステートメントは、指定したロケールに関連付けられた日付キーデータだけを含むデータセットを作成します。LOCALE=(ONLY)en_CA を指定すると、入力データセットと出力データセットの両方

で、指定したロケールだけが処理されます。ロケールは、POSIX ロケール値にする必要があります。

```
datekeydsopt locale=(ONLY)en_CA;
```

日付キーのアクティブ期間を示す変数を含むデータセットを作成します。
DATEKEYCALENDAR ステートメントは、日付キーのアクティブ期間を示す変数を書き込みます。OUT=オプションは、作成されるデータセットを指定します。このデータセットには、ID ステートメントで指定した情報に基づくカレンダ変数が含まれます。

```
datekeycalendar out=OnlyCAHolidaysInDecember2010;
```

DATEKEYS プロシジャを実行します。

```
run;
```

MyHolidays データセットを書き込みます。結果については、アウトプット 17.7 (635 ページ)を参照してください。

```
proc print data=MyHolidays;
run;
```

AllCAHolidaysInDecember2010 データセットを書き込みます。結果については、アウトプット 17.8 (636 ページ)を参照してください。

```
proc print data=AllCAHolidaysInDecember2010;
run;
```

OnlyCAHolidaysInDecember2010 データセットを書き込みます。結果については、アウトプット 17.9 (637 ページ)を参照してください。

```
proc print data=OnlyCAHolidaysInDecember2010;
run;
```

HTML 出力

アウトプット 17.7 *MyHolidays で指定されたすべての祝日定義*

The SAS System

Obs	_LOCALE_	_NAME_	_CLASS_	_KEYNAME_	_PULSE_	_SHIFT_
1	EN_CA	Canada	DATEKEY	CANADA	DAY	0
2	EN_CA	CanadaObserved	DATEKEY	CANADAOBSERVED	DAY	0
3	en_CA	Boxing	DATEKEY	BOXING	DAY	0
4	EN_US	USINDEPENDENCE	DATEKEY	USINDEPENDENCE	DAY	0
5	.	NewYearsEve	DATEKEY	NEWYEAR	DAY	-1

アウトプット 17.8 *2010 年 12 月のカナダのすべての祝日を示す祝日カレンダ*

The SAS System

Obs	date	Canada	CanadaObserved	Boxing	NewYearsEve
1	01DEC2010	0	0	0	0
2	02DEC2010	0	0	0	0
3	03DEC2010	0	0	0	0
4	04DEC2010	0	0	0	0
5	05DEC2010	0	0	0	0
6	06DEC2010	0	0	0	0
7	07DEC2010	0	0	0	0
8	08DEC2010	0	0	0	0
9	09DEC2010	0	0	0	0
10	10DEC2010	0	0	0	0
11	11DEC2010	0	0	0	0
12	12DEC2010	0	0	0	0
13	13DEC2010	0	0	0	0
14	14DEC2010	0	0	0	0
15	15DEC2010	0	0	0	0
16	16DEC2010	0	0	0	0
17	17DEC2010	0	0	0	0
18	18DEC2010	0	0	0	0
19	19DEC2010	0	0	0	0
20	20DEC2010	0	0	0	0
21	21DEC2010	0	0	0	0
22	22DEC2010	0	0	0	0
23	23DEC2010	0	0	0	0
24	24DEC2010	0	0	0	0
25	25DEC2010	0	0	0	0
26	26DEC2010	0	0	1	0
27	27DEC2010	0	0	0	0
28	28DEC2010	0	0	0	0
29	29DEC2010	0	0	0	0
30	30DEC2010	0	0	0	0
31	31DEC2010	0	0	0	1

アウトプット 17.9 2010 年 12 月のカナダのみの祝日を示す祝日カレンダ

The SAS System

Obs	date	Canada	CanadaObserved	Boxing
1	01DEC2010	0	0	0
2	02DEC2010	0	0	0
3	03DEC2010	0	0	0
4	04DEC2010	0	0	0
5	05DEC2010	0	0	0
6	06DEC2010	0	0	0
7	07DEC2010	0	0	0
8	08DEC2010	0	0	0
9	09DEC2010	0	0	0
10	10DEC2010	0	0	0
11	11DEC2010	0	0	0
12	12DEC2010	0	0	0
13	13DEC2010	0	0	0
14	14DEC2010	0	0	0
15	15DEC2010	0	0	0
16	16DEC2010	0	0	0
17	17DEC2010	0	0	0
18	18DEC2010	0	0	0
19	19DEC2010	0	0	0
20	20DEC2010	0	0	0
21	21DEC2010	0	0	0
22	22DEC2010	0	0	0
23	23DEC2010	0	0	0
24	24DEC2010	0	0	0
25	25DEC2010	0	0	0
26	26DEC2010	0	0	1
27	27DEC2010	0	0	0
28	28DEC2010	0	0	0
29	29DEC2010	0	0	0
30	30DEC2010	0	0	0
31	31DEC2010	0	0	0

参考文献

Montes, M. J. “Calculation of the Ecclesiastical Calendar.” 2001. URI: (http://www.smart.net/~mmontes/ec-cal.html).

Montes, M. J. “Algorithm for Calculating the Date of Easter in the Gregorian Calendar.” 2001. URI: (http://www.smart.net/~mmontes/nature1876.html).

18 章

DELETE プロシジャ

概要: DELETE プロシジャ

DELETE プロシジャは、保存先のディスクまたはテープから SAS ファイルを削除します。PROC DELETE を使用して次の作業を実行します。

- 永続 SAS ファイルまたは一時 SAS ファイルのいずれかを削除する
- 次のように名前が同じで接尾辞が数値のデータセットのリストを削除する

```
proc delete data=x1-x3;
run;
```

- メンバタイプの削除
- GENNUM=を使用して世代データセットを削除する
- GENNUM=ALL オプションと ENCRYPTKEY=オプションを使用する場合に、AES の暗号化されたデータセットを削除する

概念: DELETE プロシジャ

DATASETS プロシジャにおいて DELETE ステートメントではなく PROC DELETE を使用する利点の 1 つは、SAS データセットの削除にメモリ内ディレクトリを使用しないことです。

DELETE プロシジャはプリント出力を生成しません。その結果、DELETE プロシジャの方が処理速度が早くなります。

構文: DELETE プロシジャ

ヒント: DATASETS プロシジャの DELETE ステートメントと同様の機能を実行できます。

PROC DELETE DATA=*SAS-file(s)* <*option(s)*>;

ステートメント	タスク	例
"PROC DELETE ステートメント"	SAS ライブラリから SAS ファイルを削除する	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5

PROC DELETE ステートメント

SAS ライブラリから SAS ファイルを削除します。

構文

PROC DELETE <LIBRARY=*libref*> DATA=*SAS-file(s)*
(<GENNUM=ALL | HIST | REVERT | *integer*>
<MEMTYPE=*member-type*>
<ENCRYPTKEY=*key-value*>
<ALTER=*alter-password*>);

オプション引数の要約

ALTER=*alter-password*
変更保護されている SAS ファイルに Alter パスワードを付与します。

ENCRYPTKEY=*key-value*
ENCRYPTKEY=オプションは、AES 暗号化キー値により保護されているデータセットのロックを解除するものです。

GENNUM=ALL | HIST | REVERT | *integer*
世代データセットの処理を制限します。

LIBRARY=*libref*
SAS ライブラリの場所と関連付けられている SAS 名を指定します。

MEMTYPE=(*member-type(s)*)
削除を特定の種類の SAS ファイルに制限します。

必須引数

DATA= ***SAS-file(s)***
削除する SAS ファイルを 1 つ以上指定します。

注: 番号範囲リストを使用することもできます。詳細については、“Data Set Lists” (*SAS Language Reference: Concepts*)を参照してください。コロンリストは使用できません。

オプション引数

ALTER=*alter-password*
変更保護されている SAS ファイルに Alter パスワードを付与します。

参照項目 “DATASETS プロシジャでパスワードを使用する” (441 ページ)

ENCRYPTKEY=*key-value*
ENCRYPTKEY=オプションは、AES 暗号化キー値により保護されているデータセットのロックを解除するものです。ENCRYPTKEY=オプションは、データセットを開く必要がある場合にのみ必要です。したがってこのオプションは、GENNUM=ALL の使用時のみ必要となります。

参照項目 “例 7: ENCRYPTKEY=オプションの使用” (646 ページ)

GENNUM=ALL | HIST | REVERT | ***integer***
世代データセットの処理を制限します。各 SAS ファイルの名前の後にある丸括弧内のオプションを使用します。有効な値は次のとおりです。

ALL
世代グループの基本バージョンとすべての履歴バージョンを参照します。

HIST
世代グループの基本バージョンを除く、すべての履歴バージョンを参照します。

REVERT | 0
基本バージョンを削除し、存在する場合は最新履歴バージョンを基本バージョンに変更します。

integer
世代グループの特定バージョンを参照する数値です。正の数を指定すると、データセット名に追加される特定の世代番号の絶対参照になります(つまり、`gennum=2` は MYDATA#002 を指定することになります)。

参照項目
“Understanding Generation Data Sets” (*SAS Language Reference: Concepts*)

“世代データセットの処理制限” (444 ページ)

例
“例 1: 複数の SAS データセットの削除” (643 ページ)

“例 2: 基本バージョンとすべての履歴バージョンの削除” (644 ページ)

“例 3: 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更” (644 ページ)

“例 4: 絶対数によってバージョンを削除” (645 ページ)

“例 5: すべての履歴バージョンの削除および基本バージョンの保留” (645 ページ)

LIBRARY=*libref*
SAS ライブラリの場所と関連付けられている SAS 名を指定します。

別名 LIB=

MEMTYPE=(*member-type(s)*)
削除を 1 つ以上のメンバタイプに制限します。たとえば、MyFile という名前のカタログとデータセットが MyLib ライブラリにあり、カタログのみを削除する場合は、MEMTYPE=オプションを使用します。

```
proc delete lib=MyLib data=MyFile (memtype=catalog);
run;
```

ACCESS
アクセスディスクリプタファイル (SAS/ACCESS ソフトウェアによって作成)

ALL
すべてのメンバの種類

CATALOG
SAS カタログ

DATA
SAS データファイル

FDB
財務データベース

MDDB
多次元データベース

PROGRAM
保存されたコンパイル済み SAS プログラム

VIEW
SAS ビュー

別名 MTYPE=、MT=

デフォルト DATA

例 “例 6: MEMTYPE=オプションの使用” (646 ページ)

詳細

世代グループの使用

世代グループで作業しているとき、PROC DELETE を使用すれば次のバージョンを削除できます。

- 基本バージョンとすべての履歴バージョンを削除する。“例 2: 基本バージョンとすべての履歴バージョンの削除” (644 ページ)を参照してください。
- 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更する。“例 3: 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更” (644 ページ)を参照してください。

- 絶対バージョンを削除する。“例 4: 絶対数によってバージョンを削除” (645 ページ)を参照してください。
- すべての履歴バージョンを削除し、基本バージョンは残す。“例 5: すべての履歴バージョンの削除および基本バージョンの保留” (645 ページ)を参照してください。

特定のメンバタイプの削除

MEMTYPE=オプションを使用して、削除するメンバタイプを指定できます。MEMTYPE=オプションのデフォルトは DATA です。詳細については、“MEMTYPE=(*member-type(s)*)” (642 ページ)を参照してください。

一貫性制約での作業

DELETE プロシジャを使用しても、外部キー一貫性制約を持つ、または外部キー参照を含むプライマリキーを持つデータファイルは削除できません。外部キーを持つデータファイルについては、データファイルを削除する前に外部キーを削除する必要があります。外部キー参照を含むプライマリキーを持つデータファイルについては、データファイルを削除する前にそのプライマリキーを参照する外部キーを削除する必要があります。

例: DELETE プロシジャ

例 1: 複数の SAS データセットの削除

要素: PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例では、次のタスクについて説明します。

- ライブラリからデータセットを削除する
- 各データセットのすべての履歴バージョンを削除する

プログラム

```
proc delete data=MyLib.A MyLib.B MyLib.C (gennum=all);
run;
```

プログラムの説明

A、B および C という名前の SAS データセットを MyLib という名前の SAS ライブラリから削除します。 GENNUM=オプションは、各データセットの履歴バージョンをすべて削除します。

```
proc delete data=MyLib.A MyLib.B MyLib.C (gennum=all);
run;
```

例 2: 基本バージョンとすべての履歴バージョンの削除

要素: PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例では、次のタスクについて説明します。

- ライブラリから 1 つのデータセットを削除する
- データセットのすべての履歴バージョンを削除する

プログラム

```
proc delete data=MyLib.A (gennum=all);
run;
```

プログラムの説明

MyLib.A という名前のデータセットとすべての履歴バージョンを削除します。

```
proc delete data=MyLib.A (gennum=all);
run;
```

例 3: 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更

要素: PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例では、次のタスクについて説明します。

- ライブラリから 1 つのデータセットを削除する
- 最新履歴バージョンの名前を変更する

次のステートメントは、MyLib.A という名前のデータセットを削除し、その最新履歴バージョンの名前を変更します。

プログラム

```
proc delete data=MyLib.A(gennum=revert1);
run;
```

プログラムの説明

MyLib.A という名前のデータセットを削除し、最新履歴バージョンの名前を変更します。

```
proc delete data=MyLib.A(gennum=revert1);
run;
```

例 4: 絶対数によってバージョンを削除

要素: PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例では、データセットの最初の履歴バージョンを削除します。

プログラム

```
proc delete data=MyLib.A(gennum=1);
run;
```

プログラムの説明

GENNUM=オプションで、MyLib.A とうい名前のデータセットの最初の履歴バージョンを削除します。 GENNUM=*integer* を使用して削除する履歴バージョンを選択します。

```
proc delete data=MyLib.A(gennum=1);
run;
```

例 5: すべての履歴バージョンの削除および基本バージョンの保留

要素: PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例では、データセットの基本バージョンを除くすべての履歴バージョンを削除します。

プログラム

```
proc delete data=MyLib.A(gennum=hist);
run;
```

プログラムの説明

GENNUM=HIST オプションを使用してすべての履歴バージョンを削除し、MyLib.A データセットの基本バージョンは保留します。

```
proc delete data=MyLib.A(gennum=hist);
run;
```

例 6: MEMTYPE=オプションの使用

要素: PROC DELETE ステートメントオプション
DATA=
MEMTYPE=

詳細

この例では、特定の SAS ライブラリの CATALOG ファイルを削除します。

プログラム

```
proc delete lib=MyLib data=MyFile (memtype=catalog);
run;
```

プログラムの説明

MEMTYPE=オプションでは、SAS ライブラリで削除するファイルのタイプを指定します。MyLib ライブラリ内に MyFile という名前のその他のメンバタイプがある場合、それらは削除されません。

```
proc delete lib=MyLib data=MyFile (memtype=catalog);
run;
```

例 7: ENCRYPTKEY=オプションの使用

要素: PROC DELETE ステートメントオプション
DATA=
ENCRYPTKEY=
GENNUM=

詳細

この例では、次のタスクについて説明します。

- AES 暗号化データセットのロックを解除する
- すべての履歴バージョンと MyLib.A という名前の基本 AES データセットを削除する

プログラム

```
proc delete data=MyLib.A (gennum=ALL encryptkey=key-value);
run;
```

プログラムの説明

MyLib.A という名前の基本 AES データセットとすべての履歴バージョンを削除します。 ENCRYPTKEY=オプションは、データセットが AES 暗号化されている場合に使用する必要があります。

```
proc delete data=MyLib.A (gennum=ALL encryptkey=key-value);
run;
```

例 8: ALTER=オプションの使用

要素: PROC DELETE ステートメントオプション
ALTER=
DATA=

詳細

この例では、パスワードで保護されているデータセットを削除します。

プログラム

```
proc delete data=MyLib.A (alter=alter-password);
run;
```

プログラムの説明

MyLib.A とうい名前のパスワードで保護されているデータセットを削除します。 データセットがパスワードで保護されている場合は、パスワードを指定する必要があります。

```
proc delete data=MyLib.A (alter=alter-password);
run;
```

例 9: DATA=リスト機能の使用

要素: PROC DELETE ステートメントオプション
DATA=

詳細

この例では、リスト機能を使用して複数のデータセットを削除します。

プログラム

```
proc delete data=X1-X5;
run;
```

プログラムの説明

X1、X2、X3、X4、X5 という名前の複数のデータセットを削除します。 このリスト機能を使用するには、データセットの名前が同じで、その末尾が数値の接尾辞であることが必要です。LIBRARY=オプションが指定されていない場合、データセットは Work ライブラリから削除されます。

```
proc delete data=X1-X5;
run;
```

例 10: LIBRARY=オプションの使用

要素: PROC DELETE ステートメントオプション
DATA=
LIB=

詳細

次のステートメントでは、特定の SAS ライブラリからデータセットを削除します。

プログラム

```
proc delete lib=MyLib data=A;
run;
```

プログラムの説明

指定された MyLib という名前の SAS ライブラリにある A データセットを削除します。 LIBRARY=オプションの別名は LIB=です。

```
proc delete lib=MyLib data=A;
run;
```

例 11: LIBRARY=オプションおよびリスト機能の使用

要素: PROC DELETE ステートメントオプション
DATA=
LIB=

詳細

この例では、指定された MyLib という名前の SAS ライブラリのデータセット X1、X2、X3、X4、X5 を削除します。LIBRARY=オプションの別名は LIB=です。

注: これらデータセットの末尾は数値の接尾辞とします。

プログラム

```
proc delete lib=MyLib data=X1-X5;
run;
```

プログラムの説明

指定された MyLib という名前の SAS ライブラリからデータセット X1、X2、X3、X4、X5 を削除します。 このリスト機能を使用するには、すべてのデータセットの名前が同じで、その末尾が数値の接尾辞であることが必要です。LIBRARY=オプションの別名は LIB=です。

```
proc delete lib=MyLib data=X1-X5;
run;
```

19 章

DISPLAY プロシジャ

概要:DISPLAY プロシジャ

DISPLAY プロシジャは SAS/AF アプリケーションを実行します。これらのアプリケーションは、SAS/AF ソフトウェアの BUILD プロシジャで作成される各種エントリによって構成されています。このエントリは SAS カタログに格納されています。SAS/AF アプリケーションの作成に関する詳細なドキュメントについては、*Guide to SAS/AF Applications Development* を参照してください。

構文: DISPLAY プロシジャ

PROC DISPLAY CATALOG=*libref.catalog.entry.type* <BATCH>;

PROC DISPLAY ステートメント

SAS/AF アプリケーションを実行します。

例: “例: SAS/AF アプリケーションの実行” (652 ページ)

構文

PROC DISPLAY CATALOG=*libref.catalog.entry.type* <BATCH>;

必須引数

CATALOG=*libref.catalog.entry.type*
カタログエントリに 4 レベル名を指定します。

libref
カタログが格納されている SAS ライブラリを指定します。

catalog
カタログ名を指定します。

entry
エントリ名を指定します。

type
エントリの種類を指定します。種類は次のいずれかです。詳細に関しては、オンラインヘルプの BUILD プロシジャのカタログエントリの種類についての説明を参照してください。

- CBT
- FRAME
- HELP
- MENU
- PROGRAM
- SCL

オプション引数

BATCH
PROGRAM と SCL エントリをバッチモードで実行します。PROGRAM エントリにディスプレイが含まれる場合にそれが実行されなければ、次のエラーメッセージが表示されます。**ERROR: ウィンドウを割り当てられません。**

制限事項 PROC DISPLAY は、PROGRAM、FRAME、あるいは SCL エントリに引数を渡すことはできません。

DISPLAY プロシジャの使用

DISPLAY プロシジャを使用して、NODMS バッチモードで稼働するアプリケーションを実行できます。SCL の SUBMIT ブロックを介して DISPLAY プロシジャ SAS プログラミングステートメントを実行する場合、そのステートメントは PROC DISPLAY が実行されるまで処理されないことに留意してください。

SAS ウィンドウ環境を使用している場合は、アプリケーションの実行に AF コマンドを使用できます。AF コマンドを使うと、SUBMIT ブロックは即時に実行されます。AFA コマンドを使用して、複数のアプリケーションを同時に実行できます。

例: SAS/AF アプリケーションの実行

要素: PROC DISPLAY ステートメント
CATALOG =引数

詳細

会社が請求データベースから統計情報を収集する SAS/AF アプリケーションを開発したとします。また、このアプリケーションは INVOICES.WIDGETS という名前のカタログに、FRAME エントリとして SASUSER ライブラリに格納されています。このアプリケーションは、次の SAS コードで起動できます。

```
proc display catalog=sasuser.invoices.widgets.frame;
run;
```

20 章

DS2 プロシジャ

概要: DS2 プロシジャ

DS2 プロシジャを使用すると、Base SAS セッションから DS2 言語のステートメントをサブミットできます。このプロシジャでは、DS2 データアクセス技術によって要求が処理されます。この技術は、リレーショナルデータにアクセスして管理や共有を行うための、拡張可能でスレッド式の、高性能で規格に基づいた方法をサポートします。

このプロシジャでは、実稼働のマルチユーザーサーバーコンテキストで使用する前に、分離された開発またはテスト環境で言語コードを開発できます。また、このプロシジャによって、従来の SAS データアクセスサービスを使用してジョブが実行されるようにサブミットする、大規模で複雑なアプリケーションで DS2 言語を使用できるようになります。

DS2 は、上級データ操作に適した SAS プログラミング言語です。DS2 は Base SAS に実装され、SAS DATA ステップとコア機能を共有します。DS2 は、変数の範囲、ユーザー定義メソッド、ANSI SQL データ型、ユーザー定義パッケージを追加することで、DATA ステップを上回っています。DS2 SET ステートメントは、埋め込み型 FedSQL 構文を受け入れ、実行時生成型クエリは DS2 とサポートされるデータベース間で対話式

にデータを交換できます。これによって、入力テーブルの SQL 前処理を行えるようになるため、2 つの言語の機能を効果的に組み合わせることができます。

DS2 言語の詳細については、*SAS DS2 Language Reference* を参照してください。

概念: DS2 プロシジャ

DS2 言語の利点

DS2 プログラムは次のようなアプリケーション用に作成されています。

- 新たにサポートされるデータ型を使用した場合の結果が正確である必要がある
- 新しい表現を使用することで利益を得たり、DS2 構文で使用可能なメソッドやパッケージを作成したりする
- DS2 プログラムから SAS FedSQL 言語を実行する必要がある
- SAS セッション外から実行する(たとえば、SAS High-Performance Analytics Server または SAS Federation Server で実行)
- SAS High-Performance Analytics Server や SAS Enterprise Miner などの製品でスレッド式処理を活用する

データソースサポート

PROC DS2 は、次のデータソースにアクセスできます。

- Aster
- UNIX および PC オペレーティング環境用の DB2
- Greenplum
- Hadoop (Hive、HDMD、HAWQ、Impala)
- MySQL
- Netezza
- ODBC データベース(Microsoft SQL Server など)
- Oracle
- PostgreSQL
- SAP (読み取り専用)
- SAP HANA
- Sybase IQ
- SAS データセット
- SAS Scalable Performance Data (SPD) Engine データセット
- Teradata

構文: DS2 プロシジャ

PROC DS2 <*option(s)*>;
…*DS2 言語ステートメント*
RUN;
RUN CANCEL;
QUIT;

ステートメント	タスク	例
“PROC DS2 ステートメント”	後続の入力が DS2 言語ステートメントであるように指定します	Ex. 1, Ex. 2, Ex. 4
“RUN CANCEL ステートメント”	前の DS2 言語ステートメントをキャンセルします	Ex. 3

PROC DS2 ステートメント

後続の入力が DS2 言語ステートメントであるように指定します。

操作: デフォルトでは、既存のテーブルを上書きすることはできません。置換出力テーブルが作成される前に出力テーブルが削除されるように指定するには、OVERWRITE=YES テーブルオプションを使用します。OVERWRITE=テーブルオプションの詳細については、*SAS DS2 Language Reference* を参照してください。

DS2 プロシジャでは、RUN ステートメントから DS2 ステートメントをサブミットする必要があります。つまり、SAS は RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

デフォルトでは、このプロシジャは存在しない数値を SAS の欠落値として処理します。存在しない値が ANSI SQL null 値として処理されるように要求するには、ANSIMODE を指定します。

例: “例 1: DS2 コードの導入” (667 ページ)
“例 2: SAS データセットの作成” (668 ページ)
“例 4: 条件に基づいたテーブルの作成” (671 ページ)

構文

PROC DS2 <*option(s)*>;

オプション引数の要約

ANSIMODE
存在しない値が ANSI SQL ヌル値として処理されるように指定します。

BYPARTITION=YES | NO
DS2 プログラムの入力データがデータベース内処理でデータベースの内部で実行される場合に、自動的に再パーティション化されるかどうかを決定します。

DS2ACCEL=NO | YES
SAS データベース内コードアクセラレータを使用して、サポートされる環境で DS2 コードを並列処理できるかどうかを決定します。

ERRORSTOP | NOERRORSTOP
プロシジャにエラーが発生した場合、その実行を停止するかどうかを指定します。

EXEC | NOEXEC
構文が正確であることを確認した後、ステートメントを実行するかどうかを指定します。

LABEL | NOLABEL
列見出しとして列ラベルまたは列名を使用するかどうかを指定します。

NOLIBS CONN="*connection-string*"
デフォルトのデータソース接続を、指定されたデータソース接続ストリングで上書きします。

NUMBER
Row という名前の列(行が取得される際のデータの行(オブザベーション)番号)を含めるように指定します。

SCOND=WARNING | NONE | NOTE | ERROR
DS2 変数宣言厳密モードの SAS ログで PROC DS2 が表示するメッセージのレベルを指定します。このモードでは、すべての変数を DS2 プログラム内で宣言する必要があります。

STIMER
経過時間統計などのシステムパフォーマンス統計のサブセットが SAS ログに書き込まれるように指定します。

STMTMEMLIMIT=*n* | *n*M | *n*G
割り当てられたメモリが他の PROC DS2 処理をサポートするために使用できるように、基本クエリ(SELECT ステートメントなど)に使用されるメモリ量を制限するように指定します。

XCODE=ERROR | WARNING | IGNORE
NLS トランスコーディングが失敗した場合の SAS セッションの動作を制御します。

オプション引数

ANSIMODE
存在しない値が ANSI SQL ヌル値として処理されるように指定します。デフォルトでは、PROC DS2 は存在しない値を SAS 欠落値として処理します。動作の違いに関する詳細は、*SAS DS2 Language Reference* で、DS2 がヌル値と SAS の欠損値を処理する方法に関する情報を参照してください。

BYPARTITION=YES | NO
DS2 プログラムの入力データがデータベース内処理でデータベースの内部で実行される場合に、自動的に再パーティション化されるかどうかを決定します。

YES
入力データが最初の BY 変数によって自動的に再パーティション化されるように指定します。すべての BY グループは同じデータパーティション内にあり、同じスレッドによって処理されます。各スレッドは、データグループ全体に対して BY 処理を実行します。

NO
DS2 プログラムに BY ステートメントがある場合でも、入力データが再パーティション化されないように指定します。データの各グループは異なるデータパーティションにあり、各グループは異なる DS2 スレッドによって処理されます。各スレッドは 1 つのグループから一部のデータを取得し、各グループは複数のスレ

ッドによって処理されます。DS2 プログラムは、最終データ集計を要求する必要があります。

デフォルト YES

DS2ACCEL=NO | YES

SAS データベース内コードアクセラレータを使用して、サポートされる環境で DS2 コードを並列処理できるかどうかを決定します。SAS データベース内コードアクセラレータを使用して、DS2 スレッドプログラムをデータベースにパブリッシュし、データベース内でスレッドプログラムを並列処理できます。Hadoop または Teradata を使用している場合でも、DS2 データプログラムはデータベース内にパブリッシュされ、そこで実行されます。

NO
DS2 コードがサポートされる並列環境で実行されないようにします。DS2 コードは Base SAS セッションで実行されます。

YES
DS2 コードがサポートされる並列環境で実行されるようにします。

別名 INDB=NO | YES

デフォルト デフォルトは、DS2ACCEL=システムオプションによって決定されます。システムオプションのデフォルト値は、NONE です。DS2ACCEL=システムオプションの詳細については、*SAS DS2 Language Reference* を参照してください。

操作 INDB=は PROC DS2 DS2ACCEL=オプションのエイリアスです。PROC DS2 INDB=オプション(現在は DS2ACCEL=)のデフォルト値によって SAS データベース内コードアクセラレータがデータベース内処理を自動的にトリガできていた以前のリリースから、動作に変更が加えられました。最初の保守リリース SAS 9.4 に関して、デフォルトが変更され、SAS データベース内コードアクセラレータはサポートされる並列環境で実行されなくなりました。

PROC DS2 ステートメントで DS2ACCEL=オプションを指定すると、DS2ACCEL=システムオプションよりもそれが優先されます。

注 DS2ACCEL=システムオプションは、サイト管理者によって制限されている可能性があります。このシステムオプションが制限されている場合、PROC DS2 DS2ACCEL=オプションによってオーバーライドすることはできません。

ERRORSTOP | NOERRORSTOP

プロシジャにエラーが発生した場合、その実行を停止するかどうかを指定します。バッチまたは非対話型セッションでは、ERRORSTOP はプロシジャにステートメントの実行停止を命令しますが、エラー発生後も構文のチェックは継続するように命令します。NOERRORSTOP は、プロシジャにステートメントの実行を命令し、エラー発生後も構文のチェックを継続するように命令します。

デフォルト 対話型 SAS セッションでの NOERRORSTOP、バッチまたは非対話型セッションでの ERRORSTOP

操作 このオプションが役立つのは、EXEC オプションが有効な場合のみです。

ヒント ERRORSTOP は、SAS がバッチまたは非対話型実行モードで実行している場合にのみ有効です。

NOERRORSTOP は、エラー発生後もバッチジョブに SQL プロシジャステートメントの実行を継続させる場合に役立ちます。

EXEC | NOEXEC
構文が正確であることを確認した後、ステートメントを実行するかどうかを指定します。

デフォルト EXEC

ヒント NOEXEC は、ステートメントを実行せずに DB2 ステートメントの構文をチェックする場合に役立ちます。

参照項目 ERRORSTOP

LABEL | NOLABEL
列見出しとして列ラベルまたは列名を使用するかどうかを指定します。

デフォルト LABEL

操作 列にラベルがない場合、プロシジャでは列名が列見出しとして使用されます。

列エイリアスによって、ラベルまたは列名が列見出しとして上書きされます。

NOLIBS CONN="*connection-string*"
デフォルトのデータソース接続を、指定されたデータソース接続ストリングで上書きします。*Connection-string* には、以下の属性が必要です。

CATALOG="*catalog-name*"
SQL カタログの任意の ID を指定します。これにより、論理的に関連するスキーマがグループ化されます。カタログ名には最大で 32 文字を使用できます。

DRIVER="*driver-name*"
接続するデータソースを指定します。

SCHEMA=*value*
データを作成または読み込む箇所の SCHEMA=を指定します。Base SAS データに接続する場合、*value* は次の形式にする必要があります。

```
(NAME=value; PRIMARYPATH=value)
```

NAME=
SQL スキーマの任意の ID を指定します。どのような ID でも使用できます (name=myfiles など)。スキーマ ID は SAS ライブラリの物理ロケーションの別名であり、Base SAS libref に類似しています。スキーマ名は有効な SAS 名にする必要があり、最大で 32 文字を使用できます。スキーマ ID を指定する必要があります。

PRIMARYPATH=
SAS ライブラリの物理的な場所を指定します。大部分のオペレーション環境では、これはディレクトリパスになります。プライマリパスを指定する必要があります。

操作 NOLIBS CONN=を省略すると、PROC DS2 によって接続ストリングが作成され、現在割り当てられている libref の属性を使用して DS2 プログラムに渡されます。データソースへの接続を確立する方法としても、これをお勧めします。割り当てられた接続に問題がある場合は、NOLIBS CONN=を使用します。

例 次に、Base SAS データのサンプル接続ストリングを示します。

```
proc ds2 nolibs
   conn="driver=base;
```

```
catalog=base;
schema=(name=base;primarypath={c:\temp\base})";
```

NUMBER
Row という名前の列(行が取得される際のデータの行(オブザベーション)番号)を含めるように指定します。

デフォルト 行番号はありません。

SCOND=WARNING | NONE | NOTE | ERROR
DS2 変数宣言厳密モードの SAS ログで PROC DS2 が表示するメッセージのレベルを指定します。このモードでは、すべての変数を DS2 プログラム内で宣言する必要があります。DS2 変数宣言厳密モードの詳細については、*SAS DS2 Language Reference* を参照してください。

WARNING
出力を SAS ログに書き出します。

NONE
SAS ログにメッセージは書き込まれません。

NOTE
ノートを SAS ログに書き出します。

ERROR
エラーメッセージを SAS ログに書き出します。

デフォルト デフォルトは、DS2SCOND=システムオプションによって決定されます。DS2SCOND=のデフォルトは WARNING です。DS2SCOND=システムオプションの詳細については、*SAS DS2 Language Reference* を参照してください。

操作 PROC DS2 ステートメントで SCOND=オプションを指定すると、DS2SCOND=システムオプションよりもそれが優先されます。

STIMER
経過時間統計などのシステムパフォーマンス統計のサブセットが SAS ログに書き込まれるように指定します。STIMER が有効な場合、このプロシジャは各ステップと SAS セッション全体に使用されるコンピュータリソースのリストを SAS ログに書き込むように指定します。

デフォルト パフォーマンス統計は SAS ログに書き込まれません。

操作 SAS システムオプション FULLSTIMER が有効な場合、コンピュータリソースの詳細リストが SAS ログに書き込まれます。

STMTMEMLIMIT=*n* | *n*M | *n*G
割り当てられたメモリが他の PROC DS2 処理をサポートするために使用できるように、基本クエリ(SELECT ステートメントなど)に使用されるメモリ量を制限するように指定します。メモリ制限は 1 (バイト)、1,048,576 (MB)、または 1,073,741,824 (GB) の倍数で指定します。たとえば、`23m` という値を指定すると、24,117,248 バイトのメモリが指定されます。

ヒント 通常、DS2 でメモリ問題エラーが報告されないかぎり、メモリ制限を指定する必要はありません。

XCODE=ERROR | WARNING | IGNORE
NLS トランスコーディングが失敗した場合の SAS セッションの動作を制御します。トランスコーティングの失敗は、行の入力または出力操作中や、ストリングの割り当て中に発生する可能性があります。トランスコーディングは、あるエンコーディングから別のエンコーディングに文字データを変換するプロセスです。

ERROR
実行時エラーが発生し、行処理が停止するように指定します。エラーメッセージが SAS ログに書き込まれます。これはデフォルトの動作です。

WARNING
互換性のない文字が代替文字に設定されるように指定します。警告メッセージが SAS ログに書き込まれます。

IGNORE
互換性のない文字が代替文字に設定されるように指定します。SAS ログにメッセージは書き込まれません。

デフォルト ERROR

RUN CANCEL ステートメント

前の DS2 言語ステートメントをキャンセルします。

ヒント: RUN CANCEL ステートメントは、誤字を入力した場合に役立ちます。

例: “例 3: ラインプロンプトモードでの現在のステップの終了” (670 ページ)

構文

RUN CANCEL;

DS2 プロシジャの使用

データソース接続

PROC DS2 は、現在割り当てられている libref の属性を使用してデータソースに接続します。属性にはデータの物理的場所や、一部のデータソースについては、データサーバーへのアクセスに使用されるネットワーク情報などのアクセス情報、ユーザー ID およびパスワードが含まれます。

最初に SAS エンジンの LIBNAME ステートメントをサブミットし、次に PROC DS2 をサブミットします。サポートされるエンジンには、BASE (V9、V8、および V7)、SPD Engine、SAS/ACCESS エンジン Aster、DB2、Greenplum、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、SAP、Sybase IQ、Teradata などがあります。

この例では、以前割り当てた libref の属性を使用して PROC DS2 がデータソースにどのようにアクセスするかを示します。LIBNAME ステートメントは libref MyFiles を割り当て、BASE エンジンを指定し、SAS データセットの物理的場所を指定します。DS2 プログラムは、LIBNAME ステートメントで指定された場所で SAS データセット MyFiles.Table1 を作成します。

```
libname MyFiles base 'C:\Myfiles\Base';

proc ds2;
   data MyFiles.Table1;
      dcl double j j2;
      method run();
         do j = 1 to 1000;
            j2 = 2*j;
            output;
         end;
      end;
   enddata;
run;
quit;
```

PROC DS2 は、接続情報専用に libref 属性を使用します(物理的な場所など)。PROC DS2 では、動作を定義する libref 属性は使用されません。たとえば、BASE エンジンに以前にサブミットした LIBNAME ステートメントによって、SAS データセットが圧縮されるように指定されている場合、比較属性はプロシジャによって使用されません。

注: PROC DS2 はただちに接続されるため、LIBNAME ステートメントに DEFER=YES オプションが含まれる場合はエラーが生成されます。

z/OS 固有

SAS ライブラリの物理的な場所は、HFS パスの指定場所にする必要があります。

RUN グループ処理

PROC DS2 は RUN グループ処理をサポートしています。RUN グループ処理では、プロシジャを終了しなくても RUN グループをサブミットできます。

RUN グループ処理を使用するには、プロシジャを起動して複数の RUN グループをサブミットします。RUN グループは、1 つ以上のアクションステートメントを含み、RUN ステートメントで終わるステートメントのグループです。プロシジャを終了しないかぎり、アクティブなまま持続し、PROC ステートメントを再サブミットする必要はありません。

注: PROC DS2 を使用している場合、DS2 プログラムは RUN ステートメントによって区切られます。RUN ステートメントの後に別の DS2 コードが検出されると、このコードによって、前の RUN ステートメントよりも前に新しい別個の DS2 プログラムが DS2 プログラムから構成されます。

RUN グループ処理を終了するには、RUN CANCEL ステートメントをサブミットします。サブミットされなかったステートメントは終了されます。プロシジャを停止するには、QUIT ステートメントをサブミットします。サブミットされなかったステートメントも終了されます。

DS2 テーブルオプションの適用

PROC DS2 を使用してデータソースにアクセスする場合、後続の DS ステートメントで DS2 テーブルオプションを適用できます。テーブルオプションは、バッファページサイズの割り当てやパスワードの指定などの処理をテーブル上で実行できるようにするアクションを指定します。DS2 テーブルオプションは、Base SAS データセットオプションと同じ機能の多くを実行します。

DS2 テーブルオプションは、PROC DS2 内でデータソースにアクセスするときに、オプションを適用するために使用されます。たとえば、次のコードは新しいテーブルの恒久バッファページのサイズを指定するために、テーブルオプションを SAS データセットに適用します。

```
libname MyFiles base 'C:\Myfiles\Base';

proc ds2;
   data MyFiles.Table1 (bufsize=16k);
      dcl double j j2;
      method run();
         do j = 1 to 1000;
            j2 = 2*j;
            output;
         end;
      end;
enddata;
run;
quit;
```

使用可能なテーブルオプションのリストについては、*SAS DS2 Language Reference* を参照してください。

マクロ変数

リテラル文字列でのマクロ変数の使用

マクロ変数を指定すると、シンボルの置換によってプログラム内でテキストを動的に変更できます。プログラム内でマクロ変数を参照すると、マクロプロセッサによって参照値は指定したマクロ変数の値に置き換えられます。

PROC DS2 では、後続の DS ステートメントでマクロ変数を使用できます。ただし、二重引用符はマクロプロセッサがマクロ変数の参照を解決するために必須ですが、マクロ変数がリテラル文字列内で実行される場合、その文字列を二重引用符で囲むことはできません。DS2 ステートメントは、二重引用符で囲まれた文字列をテーブル名または列名などの区切り識別記号(大文字/小文字を区別する)とみなすため、文字列を二重引用符で囲むことはできません。

リテラル文字列内でマクロ変数を参照するには、SAS マクロ関数%TSLIT を使用します。この関数によって、リテラル文字列を二重引用符で囲む必要がなくなり、入力値は一重引用符で囲まれるようになります。たとえば、次のステートメントには &SYSHOSTNAME マクロ変数を指定するための%TSLIT 関数が含まれており、それが実行されるコンピュータのホスト名が返されます。

```
if hostname = %tslit(&syshostname) then ...
```

%TSLIT マクロ関数は、デフォルトのオートコールマクロライブラリに保存されます。詳細については、*SAS DS2 Language Reference* にある“Referencing a Macro Variable in a Delimited Identifier”を参照してください。

プロシジャによるマクロ変数セットの使用

PROC DS2 は、各ステートメントの実行後、特定の値を使用してマクロ変数を設定します。これらのマクロ変数はマクロ内部でテストし、PROC ステップの実行を継続するかどうかを決定できます。各ステートメントの実行後、次のマクロ変数がそれぞれの値を使用して更新されます。

SYSCC
: 動作環境に返される現在の SAS 条件コードが格納されています。終了時に、SAS は動作環境に対して意味のある値を持つ戻りコードにこの条件コードを変換します。SYSCC を使用して、ジョブ条件コードをリセットし、後続ステップを実行できない状態から回復できます。0 よりも大きい値が生成された場合は、警告メッセージかエラーメッセージを確認する必要があります。

SQLRC
PROC DS2 ステートメントの成功を示す次のステータス値が格納されています。

0
PROC ステートメントはエラーなしで正常に完了しました。

4
PROC ステートメントで警告が発行される状況が発生しました。ステートメントは引き続き実行されます。

8
PROC ステートメントでエラーが発生しました。ステートメントはこの時点で停止されました。

16
PROC ステートメントで実行時エラーが発生しました。このエラーコードが使用されるのは、たとえばサブクエリ(1 つの値のみを返す)が複数の行に対して評価する場合などです。このようなエラーが検出されるのは、実行時のみです。

セキュリティ

PROC DS2 は、SAS パスワードで保護されたデータセットファイルをサポートします。

Base SAS ソフトウェアを使用して、ファイルに SAS パスワードを割り当てることで、SAS データセットや SPD データセットへのアクセスを制限できます。読み取り、書き込み、変更という、3 つのレベルの保護を 指定できます。

PROC DS2 では、DS2 テーブルオプション(ALTER=、PW=、READ=、および WRITE=)を使用してデータソースのパスワードを割り当てるか、指定します。たとえば、READ、WRITE、および ALTER の各パスワードを SAS データセットに割り当てる場合、次のコードは DS2 テーブルオプション PW=を適用します。

```
libname MyFiles base 'C:\Myfiles\Base';

proc ds2;
   data MyFiles.Table1 (pw=luke);
      dcl double j j2;
      method run();
         do j = 1 to 1000;
            j2 = 2*j;
            output;
         end;
      end;
enddata;
run;
quit;
```

DS2 テーブルオプションは、Base SAS データセットオプションと同じ機能の多くを実行します。ただし、Base SAS データセットオプションは PROC FEDSQL ステートメントでサポートされません。このため、PROC DS2 でデータソースにアクセスする場合、DS2 テーブルオプションを使用してパスワードの割り当てまたは指定を行う必要があります。

SAS パスワードは、SAS システムの外部にある SAS ファイルへのアクセスを制御しません。SAS の外部にある SAS ファイルへのアクセスを制御するには、オペレーティングシステムで提供されているユーティリティやファイルシステムセキュリティを使用する必要があります。SAS パスワードの詳細については、*SAS 言語リファレンス: 解説編*を参照してください。

DS2 データ型サポート

PROC DS2 では、DS2 ステートメントをサブミットするときに、すべての DS2 言語データ型がサポートされます。DS2 データ型の詳細については、*SAS DS2 Language Reference* を参照してください。

ただし、Base SAS セッションでは、PROC DS2 をサブミットしない場合、DS2 データ型はあらかじめ定義されたレガシー SAS データ型へ変換されるか、または逆にそこから変換されます。その場合、値は SAS 数値および SAS 文字になります。たとえば、DS2 言語で作成されたテーブルで CONTENTS プロシジャをサブミットすると、DATE データ型は SAS 数値としてレポートされます。次の表には、DS2 データ型と、SAS データ型への変換方法またはその逆に SAS データ型からの変換方法が一覧表示されています。

表 20.1 *DS2 データ型の変換*

DS2 データ型	レガシー SAS データ型	説明
BIGINT	SAS 数値	SAS フォーマット 20 を適用します。 SAS 数値が DOUBLE であり、これは正確な数値データ型ではなく概算の数値データ型であるため、精度が失われる可能性があります。
BINARY(*n*)	SAS 文字	SAS フォーマット$n を適用します。
CHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。
DATE	SAS 数値	SAS フォーマット DATE9 を適用します。 有効な SAS 日付値は、1582-01-01 から 9999-12-31 までです。SAS 日付範囲外の日付はサポートされていないため、無効な日付として処理されます。
DECIMAL| NUMERIC(*p*,*s*)	SAS 数値	SAS フォーマット 11.2 を適用します。
DOUBLE	SAS 数値	
FLOAT(*p*)	SAS 数値	
INTEGER	SAS 数値	SAS フォーマット 11 を適用します。
NCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。
NVARCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。
REAL	SAS 数値	
SMALLINT	SAS 数値	SAS フォーマット 6 を適用します。
TIME(*p*)	SAS 数値	SAS フォーマット TIME8 を適用します。

DS2 データ型	レガシー SAS データ型	説明
TIMESTAMP(*p*)	SAS 数値	SAS フォーマット DATETIME19.2 を適用します。
TINYINT	SAS 数値	SAS フォーマット 4 を適用します。
VARBINARY(*n*)	SAS 文字	SAS フォーマット$n を適用します。
VARCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。

例: DS2 プロシジャ

例 1: DS2 コードの導入

要素: PROC DS2 ステートメント
QUIT ステートメント

他の要素: DS2 言語ステートメント

詳細

この例では、SAS ログ内で **`Hello World!`**を表示する簡単な DS2 プログラムが使用されます。このコードは SAS DATA ステップと似ていますが、デフォルトのシステムメソッドなど、異なる構文要素が含まれています(INIT、RUN、および TERM)。また、DS2 では DECIMAL、INTEGER、および VARCHAR などの非常に一般的な SQL データ型がサポートされるため、DBMS データに対してよりネイティブに処理が行われます。

プログラム

```
proc ds2;

data _null_;
   method init();
      dcl varchar(16) str;
      str = 'Hello World!';
      put str;
   end;
enddata;

run;

quit;
```

プログラムの説明

PROC DS2 ステートメントを実行します。 現在割り当てられた libref がない場合、PROC DS2 ステートメントは DS2 言語ステートメントをサブミットするための環境を設定します。

```
proc ds2;
```

DS2 言語ステートメントを入力します。 DS2 DATA ステートメントの_NULL_によって、自動的に出力が生成されないように指定されます。DS2 PUT ステートメントによって、SAS ログへの書き込みが行われます。

```
data _null_;
   method init();
      dcl varchar(16) str;
      str = 'Hello World!';
      put str;
   end;
enddata;
```

DS2 ステートメントをサブミットします。 RUN ステートメントによって、DS2 ステートメントがサブミットされます。RUN ステートメントは必須です。SAS は、RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

```
run;
```

プロシジャを停止します。 QUIT ステートメントは、プロシジャを終了します。

```
quit;
```

ログ 20.1 *Hello World!を表示する SAS ログ*

```
48   proc ds2; 49   data _null_; 50      method init(); 51         dcl
varchar(16) str; 52         str = 'Hello World!'; 53         put str; 54
end; 55   enddata; 56   run; Hello World!NOTE:Execution succeeded.No rows
affected.57   quit; NOTE:PROCEDURE DS2 used (Total process time): real
time           16.38 seconds cpu time            0.15 seconds
```

例 2: SAS データセットの作成

要素: PROC DS2 ステートメント
QUIT ステートメント

他の要素: LIBNAME ステートメント
DS2 言語ステートメント
PROC PRINT

詳細

この例では、DS2 プロシジャをサブミットしてから DS2 言語ステートメントをサブミットすることで、Base SAS セッション内で SAS データセットが作成されます。出力には、データセットの最初の 10 行が示されます。

プログラム

```
libname MyFiles base 'C:\Myfiles';

proc ds2;


   data MyFiles.BaseTable;
      declare double j j2;
      method run();
         do j = 1 to 1000;
            j2 = 2*j;
            output;
         end;
      end;
   enddata;

run;

quit;

proc print data=myfiles.basetable (obs=10);
run;
```

プログラムの説明

ライブラリ参照を作成する SAS データセットに割り当てます。 LIBNAME ステートメントは libref MyFiles を割り当て、BASE エンジンを指定し、SAS データセットの物理的場所を指定します。

```
libname MyFiles base 'C:\Myfiles';
```

PROC DS2 ステートメントを実行します。 PROC DS2 ステートメントは、libref 属性を使用してデータソースに接続し、DS2 言語ステートメントをサブミットするための環境を設定します。

```
proc ds2;
```

DS2 言語ステートメントを入力します。 DS2 DATA ステートメントによって、Myfiles.BaseTable という名前の出力テーブルが作成されます。DATA ステートメントの 2 レベルの名前によって、カタログ識別子 MyFiles (割り当てられた libref)が指定されます。DECLARE ステートメントによって、データ型 DOUBLE が変数 J および J2 に割り当てられます。METHOD ステートメントによって、出力の作成に使用される RUN システムメソッドが識別されます。OUTPUT ステートメントによって、DO ループを実行するたびに 1 行がテーブル MyFiles.BaseTable に書き込まれます。

```
   data MyFiles.BaseTable;
      declare double j j2;
      method run();
         do j = 1 to 1000;
            j2 = 2*j;
            output;
         end;
      end;
   enddata;
```

DS2 言語ステートメントをサブミットします。 RUN ステートメントによって、DS2 ステートメントがサブミットされます。RUN ステートメントは必須です。SAS は、RUN ステートメント

に到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

```
run;
```

プロシジャを停止します。 QUIT ステートメントは、プロシジャを終了します。

```
quit;
```

SAS データセットを出力します。 PRINT プロシジャによって、オブザベーションが SAS データセットに出力されます。OBS=データセットオプションによって、出力オブザベーション数は 10 に制限されます。

```
proc print data=myfiles.basetable (obs=10);
run;
```

出力:SAS データセットの作成

アウトプット 20.1 *MyFiles.BaseTable の PROC PRINT 出力*

The SAS System

Obs	j	j2
1	1	2
2	2	4
3	3	6
4	4	8
5	5	10
6	6	12
7	7	14
8	8	16
9	9	18
10	10	20

例 3: ラインプロンプトモードでの現在のステップの終了

要素: PROC DS2 ステートメント
RUN CANCEL ステートメント
QUIT ステートメント

他の要素: DS2 言語ステートメント

詳細

次の例は、ラインプロンプトモードセッションでの RUN CANCEL ステートメントの有効性を示しています。コード内の 6 番目のステートメントには、列(Y ではなく Z)の無効な値が含まれています。RUN CANCEL は PROC DS2 ステップを終了し、実行されないようにします。

プログラム

```
proc ds2;
data xy_data;
   declare double x y;
   method init();
      do x = 1 to 5;
         z = 2*x;
      end;
   end;
enddata;
run cancel;
quit;
```

ログ 20.2 *キャンセルした PROC DS2 ステップを示す SAS ログ*

```
17   proc ds2; NOTE:Writing HTML Body file: sashtml1.htm 18   data xy_data;
19      dcl double x y; 20      method init(); 21         do x = 1 to 5;
22            z = 2*x; 23         end; 24      end; 25   enddata; 26   run
cancel; NOTE:DS2 query cancelled per user request.27   quit; NOTE:PROCEDURE DS2
used (Total process time): real time           14.33 seconds cpu time
0.71 seconds
```

例 4: 条件に基づいたテーブルの作成

要素: PROC DS2 ステートメント
QUIT ステートメント

他の要素: DS2 言語ステートメント

詳細

この例では、条件に基づいてテーブルを作成する方法を示します。プログラム 1 および 2 によって、Dept1_Items および Dept2_Items という 2 つのテーブルが作成され、2 つの部門によって使用されるアイテムのコストが格納されます。3 番目のプログラムによって、2 つのアイテムテーブル内のアイテムのコストに基づいて、Highcosts および Lowcosts という 2 つのテーブルが作成されます。プログラム 4 および 5 によって、コストテーブルのコンテンツが出力されます。

プログラム

```
proc ds2;
/* Program 1 */
data dept1_items (overwrite=yes);
   dcl varchar(20) item;
   dcl double cost;
```

```
   method init();
      item = 'staples';   cost =  1.59; output;
      item = 'pens';      cost =  3.26; output;
      item = 'envelopes'; cost = 11.42; output;
   end;
enddata;
run;
/* Program 2 */
data dept2_items (overwrite=yes);
   dcl varchar(20) item;
   dcl double cost;
   method init();
      item = 'erasers'; cost =  5.43; output;
      item = 'paper';   cost = 26.92; output;
      item = 'toner';   cost = 62.29; output;
   end;
enddata;
run;
/* Program 3 */
data lowCosts (overwrite=yes) highCosts (overwrite=yes);
   method run();
      set dept1_items dept2_items;
      if cost <= 10.00 then
         output lowCosts;
      else
         output highCosts;
      end;
enddata;
run;
/* Program 4 */
data;
   method run();
      set lowCosts;
   end;
enddata;
run;
/* Program 5 */
data;
   method run();
       set highCosts;
   end;
enddata;
run;
quit;
```

出力:条件に基づいたテーブルの作成

アウトプット 20.2 *コストテーブルの出力*

The SAS System

item	cost
staples	1.59
pens	3.26
erasers	5.43

The SAS System

item	cost
envelopes	11.42
paper	26.92
toner	62.29

21 章

EXPORT プロシジャ

概要:EXPORT プロシジャ

EXPORT プロシジャは SAS データセットからデータを読み込み、外部データソースに書き込みます。Base SAS 9.4 では、外部データソースに区切りファイルと JMP ファイルが含まれます。

区切り文字で区切られたファイルでは、区切り文字(ブランク、カンマ、タブなど)でデータ値の列が区切られます。PC ファイルに対する SAS/ACCESS インターフェースのライセンスを持っている場合、Microsoft Access データベース、Microsoft Excel ワークブック、DBF ファイルおよび Lotus スプレッドシートなど、別のファイル形式にエクスポートすることもできます。詳細については、*SAS/ACCESS Interface to PC Files: Reference* を参照してください。

SAS 9.4 以降、SAS データセットを JMP 7 以降のファイルにエクスポートできます。また、JMP 変数には最大 255 文字を使用できます。拡張属性が自動的に使用されるようになり、META=ステートメントは JMP ファイルでサポートされなくなりました。詳細については、“JMP Files” (*SAS/ACCESS Interface to PC Files: Reference*)を参照してください。

EXPORT プロシジャは、次のいずれかの方法を使用してデータをエクスポートします。

- 生成された DATA ステップコード
- 生成された SAS/ACCESS コード
- 変換エンジン

出力データソースに特有のオプションやステートメントを指定して、結果を制御します。EXPORT プロシジャは指定された出力ファイルを生成し、エクスポートに関する情報を SAS ログに書き込みます。ログには EXPORT プロシジャによって生成される DATA ステップまたは SAS/ACCESS コードが表示されます。変換エンジンを使用する場合は、コードはサブミットされません。

エクスポートウィザードまたは **External File Interface** (EFI)を使用して、SAS データセットをエクスポートするステップを段階的に確認できます。エクスポートウィザードを使用して、EXPORT プロシジャステートメントを生成し、後で使用できるようにファイルに保存できます。詳細については、“External File Interface (EFI)” (*SAS/ACCESS Interface to PC Files: Reference*)を参照してください。

エクスポートウィザードでは EFI メソッドを使用して区切り文字で区切られたファイルの読み取りおよび書き込みを行います。これは、EXPORT プロシジャやエクスポートウィザードを使用するときの動作に影響を及ぼす可能性があります。たとえば、SAS データを区切り文字で区切られたファイルにエクスポートすると、EXPORT プロシジャによって出力行の長さを超える項目が破棄されます。詳細については、*SAS ステートメント: リファレンス*の FILE ステートメントでの DROPOVER オプションに関する情報を参照してください。

エクスポートウィザードを開くには、SAS ウィンドウ環境から**ファイル** ⇨ **データのエクスポート**を選択します。インポートウィザードの詳細については、Base SAS オンラインヘルプおよびドキュメントを参照してください。詳細と例については、“Using the SAS Import and Export Wizards” (*SAS/ACCESS Interface to PC Files: Reference*)を参照してください。

構文: EXPORT プロシジャ

制限事項: EXPORT プロシジャは次の動作環境で使用できます。

- Windows
- UNIX または Linux

ファイルのパス名には、最大で 201 文字を使用できます。

PROC EXPORT DATA=<*libref.*>*SAS data set* <(*SAS data set option(s)*)>
OUTFILE="*filename*" | OUTTABLE="*tablename*"
<DBMS=*identifier*> <REPLACE> <LABEL>;

区切り文字で区切られたファイルをエクスポートするステートメント

DELIMITER=*char* | '*nn*'x;
PUTNAMES=YES | NO;

JMP ファイルをエクスポートするステートメント

DBENCODING=*12-char SAS encoding-value*;
FMTLIB=<*libref.*>*format-catalog*;
META=*libref.member-data-set*;

ステートメント	タスク	例
“PROC EXPORT ステートメント”	SAS データセットを外部データファイルにエクスポートします。	Ex. 1, Ex. 2
“DBENCODING ステートメント”	JMP ファイルにデータを保存するために使用するエンコーディングを示します。	

ステートメント	タスク	例
“DELIMITER ステートメント”	区切り文字で区切られた出力ファイルでデータ列を区切る文字を指定します。	Ex. 1
“FMTLIB ステートメント”	出力形式カタログで定義された SAS 出力形式の値を JMP ファイルの値ラベルに書き込みます。	
“META ステートメント”	SAS メタデータ情報を JMP ファイルに書き込みます。	
“PUTNAMES ステートメント”	エクスポートするデータファイルの最初の行に列見出しとして SAS 変数名を書き込みます。	Ex. 3

PROC EXPORT ステートメント

SAS データセットを外部ファイルにエクスポートします。

構文

PROC EXPORT DATA=<*libref.>SAS data set* <(*SAS data set options*)>
OUTFILE="*filename*" | OUTTABLE="*tablename*"
<DBMS=*identifier*> <REPLACE> <LABEL>;

オプション引数の要約

(*SAS データセットオプション*)
SAS データセットオプションを指定します。

DBMS=*identifier*
エクスポートするデータの種類を指定します。

LABEL
変数ラベル名を指定します。

REPLACE
既存のファイルを上書きします。

必須引数

DATA=*<libref.>SAS data set*
入力カ SAS データセットを 1 レベルまたは 2 レベルの SAS 名(ライブラリとメンバ名)で識別します。1 レベルの名前を指定した場合、EXPORT プロシジャはデフォルトで USER ライブラリ(割り当てられている場合)または WORK ライブラリのどちらかを使用します。

EXPORT プロシジャは、データターゲットが SAS データセットの出力形式をサポートしている場合のみ、SAS データセットをエクスポートできます。データの容量もデータターゲットの制限以内でなければなりません。たとえば、データファイルの中には行または列数に最大値が設定されているものもあります。SAS のユーザー定義の出力形式と入力形式をサポートできないデータファイルもあります。エクスポートする SAS データセットがターゲットファイルの制限を超える場合は、EXPORT プロシジャによって正しくエクスポートできない場合もあります。多くの場合、プロシジャ

は可能な限り最善の方法でデータを変換します。ただし、種類によっては変換できないものもあります。

SAS 9.4 の最初の管理リリースでは、VALIDMEMNAME=EXTEND システムオプションも指定されている場合、SAS データセット名に単一引用符を含めることができます。VALIDMEMNAME=を使用すると、SAS データセット名など、特定の SAS メンバの名前に関するルールが拡張されます。詳細については、*SAS 言語リファレンス: 解説編*の"Rules for SAS Data Set Names, View Names, and Item Store Names"を参照してください。

デフォルト エクスポートする SAS データセットを指定しない場合、EXPORT プロシジャは直近に作成された SAS データセットを使用します。SAS は、システム変数 _LAST_を使用してデータセットを追跡します。EXPORT プロシジャが確実に正しいデータセットを使用するようにするには、SAS データセットを指定する必要があります。

例 "例 1: 区切り文字で区切られた外部データソースにエクスポートする" (682 ページ)

"例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする" (685 ページ)

OUTFILE="*filename*" | "*fileref*"

出力 PC ファイル、スプレッドシート、または区切り文字で区切られた外部ファイルの完全パスとファイル名またはファイル参照名を指定します。ファイル参照名は物理的なファイルの場所に関連付けられた SAS 名です。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

ファイル参照名を指定した場合、または完全パスとファイル名に特殊文字(パスのバックスラッシュなど)、小文字、またはスペースが含まれない場合は、引用符を省略できます。

別名 FILE

制限事項 EXPORT プロシジャでは、DISK を除き、FILENAME ステートメントのデバイスの種類またはアクセスメソッドはサポートされません。たとえば、EXPORT プロシジャでは一時外部ファイルを作成する TEMP デバイスの種類はサポートされていません。

参照項目 *SAS/ACCESS Interface to PC Files: Reference* PC ファイル出力形式の詳細については、

例 "例 1: 区切り文字で区切られた外部データソースにエクスポートする" (682 ページ)

"例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする" (685 ページ)

OUTTABLE="*tablename*"

出力 DBMS テーブルのテーブル名を指定します。名前に特殊文字(疑問符など)、小文字、またはスペースが含まれない場合は、引用符を省略できます。DBMS テーブル名では大文字と小文字が区別される場合もあります。

要件 DBMS テーブルをエクスポートするには SAS/ACCESS Interface to PC Files のライセンスが必要です。

DBMS テーブルをインポートするときは、DBMS オプションを指定する必要があります。

オプション引数

DBMS=*identifier*
エクスポートするデータの種類を指定します。DBMS テーブルにエクスポートする場合は、有効なデータベース識別子を使用して DBMS オプションを指定する必要があります。DBMS=DLM に設定すると、デフォルトの区切り文字はスペースになります。ただし、DELIMITER='*char*'を使用することもできます。

次の値は有効な DBMS 拡張子です。

表 21.1 *Base SAS でサポートされている DBMS 識別子*

識別子	出力データソース	拡張子
CSV	区切り文字で区切られたファイル(カンマ区切りの値)	.csv
DLM	区切り文字で区切られたファイル(デフォルトの区切り文字 はブランク)	
JMP	JMP ファイル(バージョン 7 以降のフォーマット)	.jmp
TAB	区切り文字で区切られたファイル(タブ区切りの値)	.txt

制限事項 出力外部データソースを使用できるかどうかは、次の条件によって決まります。

- 動作環境、および場合によっては前の表で指定されているプラットフォーム。
- サイトに SAS/ACCESS Interface to PC Files のライセンスがあるかどうか。ライセンスがない場合、区切り文字で区切られたファイルと JMP ファイルのみを使用できます。

参照項目 *SAS/ACCESS Interface to PC Files: Reference* には、SAS/ACCESS Interface to PC Files を使用する場合のその他の DBMS 識別子のリストがあります。

例 “例 1: 区切り文字で区切られた外部データソースにエクスポートする” (682 ページ)

LABEL
変数ラベル名を指定します。SAS はこれらを、エクスポートされたテーブルの列名として書き込みます。ラベル名が存在しない場合、SAS はエクスポートされたテーブルに書き込みます。

REPLACE
既存のファイルを上書きします。REPLACE を指定しなければ、EXPORT プロシジャは既存のファイルを上書きしません。

例 “例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする” (685 ページ)

(*SAS データセットオプション*)
SAS データセットオプションを指定します。たとえば、エクスポートするデータセットにパスワードが割り当てられている場合、ALTER=、PW=、READ=、または WRITE=データセットオプションを使用できます。指定された条件を満たすデータのサブセットをエクスポートするには、WHERE オプションを使用します。SAS データ

セットオプションの詳細については、*SAS データセットオプション: リファレンス*を参照してください。

例 “例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする” (685 ページ)

DBENCODING ステートメント

JMP ファイルにデータを保存するために使用するエンコーディングを示します。

操作: DBENCODING ステートメントは、DBMS=JMP の場合にのみ有効です。

構文

DBENCODING=*12-char SAS encoding-value*;

必須引数

12-char SAS encoding-value
JMP ファイルにデータを保存するために使用するエンコーディングを示します。エンコーディングによって、文字セットの各文字が一意の数値表現にマッピングされ、コードポイントのテーブルが構成されます。各文字は、エンコーディングごとに異なる数値表現を使用できます。この値には最大で 12 文字を使用できます。

DELIMITER ステートメント

出力ファイルでデータ列を区切る文字を指定します。

デフォルト: ブランクスペース

操作: DBMS=DLM を指定する場合は、DELIMITER ステートメントも指定する必要があります。

例: “例 1: 区切り文字で区切られた外部データソースにエクスポートする” (682 ページ)

構文

DELIMITER=*char* | '*nn*'x;

必須引数

***char* | '*nn*'x**
出力ファイルで値を区切るのために使用する区切り文字を指定します。区切り文字は、単一文字または 16 進値として指定できます。例えば、列の値をアンパサンドで区切る場合は、DELIMITER='&'を指定します。

FMTLIB ステートメント

出力形式カタログで定義された SAS 出力形式の値を JMP ファイルの値ラベルに書き込みます。

操作: FMTLIB ステートメントは、DBMS=JMP の場合のみ有効です。

構文

FMTLIB=*<libref> format-catalog*;

必須引数

<libref.>format-catalog
JMP ファイルに書き込む出力形式カタログを指定します。

META ステートメント

SAS メタデータ情報を JMP ファイルに書き込みます。(非推奨)

操作: META ステートメントは、DBMS=JMP の場合のみ有効です。

構文

META=*libref.member-data-set*;

必須引数

libref.member-data-set
JMP ファイルに書き込むメタデータ情報を含む SAS データセットを指定します。

META ステートメントはサポートされなくなったため、無視されます。代わりに、拡張属性が自動的に使用されます。SAS データセットを JMP にエクスポートすると、PROC EXPORT は SAS データセット上の拡張属性を探します。その拡張属性が存在すれば、プロシジャはその属性を使用して新しい JMP ファイルを作成します。

META ステートメントはプログラム内に残っている可能性がありますが、META が拡張属性に置換され、無視されることを示す NOTE をログに生成します。

PUTNAMES ステートメント

エクスポートするデータファイルの最初の行に列見出しとして SAS 変数名を書き込みます。

デフォルト: YES

制限事項: EXPORT プロシジャの場合にのみ有効です。

注: LABEL=オプションを指定すると、SAS 変数ラベル(変数名ではない)は列見出しとして書き込まれます。

例: “例 3: PUTNAMES=ステートメントを使用したタブで区切られたファイルへのエクスポート” (686 ページ)

構文

PUTNAMES=YES | NO;

必須引数

YES
EXPORT プロシジャで次のタスクが行われるように指定します。

- エクスポートするデータファイルの最初の行に列見出し(ヘディング)として SAS 変数名を書き込みます。
- エクスポート対象のデータファイルの 2 番目の行に、SAS データセットの最初の行を書き込みます。

NO
EXPORT プロシジャで、エクスポート対象のデータファイルの最初の行に、SAS データセット値の最初の行が書き込まれるように指定します。

例: EXPORT プロシジャ

例 1: 区切り文字で区切られた外部データソースにエクスポートする

要素: PROC EXPORT ステートメントオプション
DATA=
DBMS=
OUTFILE=
REPLACE

他の要素: DELIMITER=ステートメント

詳細

この例では、SASHelp.Class データセットが、区切り文字で区切られた外部ファイルにエクスポートされます。次の例は、エクスポートされる前の SASHelp.Class データセットを示しています。

アウトプット 21.1 *PROC PRINT of SASHelp.Class*

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

プログラム

```
proc export data=sashelp.class
   outfile="c:\myfiles\class"
   dbms=dlm replace;

   delimiter='&';
run;
```

プログラムの説明

入力データセットを指定します。 ファイル名に拡張子は含まれません。DBMS=DLM によって、出力ファイルは区切り文字で区切られたファイルであることが指定されています。

```
proc export data=sashelp.class
   outfile="c:\myfiles\class"
   dbms=dlm replace;
```

DELIMITER オプションで、出力ファイルでは&(アンパサンド)によってデータフィールドが区切られることが指定されています。

```
     delimiter='&';
run;
```

ログの例

これは SAS ログの一部ですが、生成された SAS DATA ステップを含む、正常なエクスポート処理に関する情報が表示されています。

ログ 21.1 *区切り文字で区切られたファイルを作成する場合の SAS ログ*

```
1    proc export data=sashelp.class outfile="c:\myfiles\class" dbms=dlm replace;
1  !                                                              delimiter='&' ; run; 2     /
**************************************************************************** 3     *   PRODUCT:   SAS 4
*   VERSION:   9.3 5     *   CREATOR:   External File Interface 6     *   DATE:      31JAN11 7     *
DESC:      Generated SAS Datastep Code 8     *   TEMPLATE SOURCE:(None Specified.)9
***************************************************************************/ 10       data _null_;
11       %let _EFIERR_ = 0; /* set the ERROR detection macro variable */ 12       %let _EFIREC_ =
0;     /* clear export record count macro variable */ 13       file 'c:\myfiles\class' delimiter='&'
DSD DROPOVER lrecl=32767; 14       if _n_ = 1 then        /* write column names or labels */ 15
do; 16          put 17             "Name" 18          '&' 19             "Sex" 20          '&'
21             "Age" 22          '&' 23             "Height" 24          '&' 25             "Weight"
26          ; 27        end; 28      set  SASHELP.CLASS   end=EFIEOD; 29          format Name $8.;
30          format Sex $1.; 31          format Age best12.; 32          format Height best12.;
33          format Weight best12.; 34        do; 35          EFIOUT + 1; 36          put Name $ @;
37          put Sex $ @; 38          put Age @; 39          put Height @; 40          put Weight ;
41          ; 42        end; 43       if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR
detection macro variable */ 44       if EFIEOD then call symputx('_EFIREC_',EFIOUT); 45       run;
NOTE:The file 'c:\myfiles\class' is:Filename=c:\myfiles\class, RECFM=V,LRECL=32767,File Size
(bytes)=0, Last Modified=31Jan2011:09:37:14, Create Time=31Jan2011:09:37:14
```

出力例

EXPORT プロシジャによって、この外部ファイルが作成されます。

アウトプット 21.2 *外部ファイル*

```
Name&Sex&Age&Height&Weight Alfred&M&14&69&112.5 Alice&F&13&56.5&84 Barbara&F&13&65.3&98
Carol&F&14&62.8&102.5 Henry&M&14&63.5&102.5 James&M&12&57.3&83 Jane&F&12&59.8&84.5
Janet&F&15&62.5&112.5 Jeffrey&M&13&62.5&84 John&M&12&59&99.5 Joyce&F&11&51.3&50.5 Judy&F&14&64.3&90
Louise&F&12&56.3&77 Mary&F&15&66.5&112 Philip&M&16&72&150 Robert&M&12&64.8&128 Ronald&M&15&67&133
Thomas&M&11&57.5&85 William&M&15&66.5&112
```

例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする

要素: PROC EXPORT ステートメントオプション
DATA=
DBMS=
OUTFILE=
REPLACE

詳細

この例では、SAS データセット SASHelp.Class が、区切り文字で区切られたファイルにエクスポートされます。

プログラム

```
proc export data=sashelp.class (where=(sex='F'))
     outfile="c:\myfiles\Femalelist.csv"
     dbms=csv
     replace;
run;
```

プログラムの説明

エクスポートするデータセットを指定します。 WHERE オプションはオブザベーションのサブセットを要求します。OUTFILE=オプションによって出力ファイルが指定されます。DBMS=オプションによって、出力ファイルが CSV ファイルであり、存在する場合はそれによってターゲット CSV が上書きされることが指定されます。

```
proc export data=sashelp.class (where=(sex='F'))
     outfile="c:\myfiles\Femalelist.csv"
     dbms=csv
     replace;
run;
```

ログの例

これは SAS ログの一部ですが、生成された SAS DATA ステップを含む、正常なエクスポート処理に関する情報が表示されています。

ログ 21.2 *CSV ファイルへのエクスポート*

```
579  proc export data=sashelp.class (where=(sex='F')) 580       outfile="c:
\myfiles\Femalelist.csv" 581       dbms=csv 582       replace; 583  run; 584   /
********************************************************************** 585   *
PRODUCT:   SAS 586   *   VERSION:   9.4 587   *   CREATOR:   External File
Interface 588   *   DATE:      18APR14 589   *   DESC:      Generated SAS
Datastep Code 590   *   TEMPLATE SOURCE:(None Specified.)591
**********************************************************************/
592      data _null_; 593      %let _EFIERR_ = 0; /* set the ERROR detection
macro variable */ 594      %let _EFIREC_ = 0;     /* clear export record count
macro variable */ 595      file 'c:\myfiles\Femalelist.csv' delimiter=',' DSD
DROPOVER lrecl=32767 595!; 596      if _n_ = 1 then        /* write column names
or labels */ 597       do; 598         put 599            "Name" 600         ','
601            "Sex" 602         ',' 603            "Age" 604         ','
605            "Height" 606         ',' 607            "Weight" 608         ;
609       end; 610     set  SASHELP.CLASS(where=(sex='F'))   end=EFIEOD;
611         format Name $8.; 612         format Sex $1.; 613         format Age
best12.; 614         format Height best12.; 615         format Weight best12.;
616       do; 617         EFIOUT + 1; 618         put Name $ @; 619         put
Sex $ @; 620         put Age @; 621         put Height @; 622         put
Weight ; 623         ; 624       end; 625      if _ERROR_ then call
symputx('_EFIERR_',1);  /* set ERROR detection 625! macro variable */ 626
if EFIEOD then call symputx('_EFIREC_',EFIOUT); 627      run; NOTE:The file 'c:
\myfiles\Femalelist.csv' is:Filename=c:\myfiles\Femalelist.csv,
RECFM=V,LRECL=32767,File Size (bytes)=0, Last Modified=18Apr2014:11:32:07,
Create Time=18Apr2014:11:32:07
```

出力例

EXPORT プロシジャによって、この外部 CSV ファイルが作成されます。

アウトプット 21.3 *CSV ファイル*

Name	Sex	Age	Height	Weight
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112

例 3: PUTNAMES=ステートメントを使用したタブで区切られたファイルへのエクスポート

要素: PROC EXPORT ステートメントオプション
- DATA=
- DBMS=
- OUTFILE=
- PUTNAMES=
- REPLACE

詳細

この例では、SAS データセット WORK.INVOICE をタブで区切られたファイルにエクスポートする様子が示されています。最初のプログラムは PUTNAMES=ステートメントを指定して PROC EXPORT を使用していますが、2 番目のプログラムはそうではありません。このステートメントを使用すると、タブで区切られたファイルの列見出しにどのような影響が及ぼされるかを示しています。

次に、SAS データセット WORK.INVOICE がタブ区切りのファイルにエクスポートされる前の状態を示します。

アウトプット 21.4 *WORK.INVOICE の PROC PRINT*

Obs	Invoice_ID	Billed_To	Amount_Billed_in_Local_Currency	Country	Amount_Billed_in_US_Dollars	Billed_By	Billed_On	Paid_On
1	11270	39045213	8738600640.00	Brazil	3675848666.21	239185	05OCT2004	18OCT2004
2	11271	18543489	11063836.00	USA	11063836.00	457232	05OCT2004	.
3	11273	19783482	252148.50	USA	252148.50	239185	06OCT2004	11NOV2004
4	11276	14324742	1934460.00	USA	1934460.00	135673	06OCT2004	20OCT2004
5	11278	14898029	1400825.00	USA	1400825.00	239185	06OCT2004	19OCT2004
6	11280	39045213	8738600640.00	Brazil	3675848666.21	423286	07OCT2004	20OCT2004
7	11282	19783482	252148.50	USA	252148.50	457232	07OCT2004	25OCT2004
8	11285	38763919	2234301.30	Argentina	772847.05	239185	10OCT2004	30NOV2004
9	11286	43459747	14836604.08	Australia	11366352.06	423286	10OCT2004	.
10	11287	15432147	252148.50	USA	252148.50	457232	11OCT2004	04NOV2004
11	12051	39045213	8738600640.00	Brazil	3675848666.21	457232	02NOV2004	.
12	12102	18543489	11063836.00	USA	11063836.00	239185	17NOV2004	.
13	12263	19783482	252148.50	USA	252148.50	423286	05DEC2004	.
14	12468	14898029	1400825.00	USA	1400825.00	135673	24DEC2004	02JAN2005
15	12471	39045213	8738600640.00	Brazil	3675848666.21	457232	27DEC2004	.
16	12476	38763919	2234301.30	Argentina	772847.05	135673	24DEC2004	.
17	12478	15432147	252148.50	USA	252148.50	423286	24DEC2004	02JAN2005

プログラム

```
PROC PRINT DATA=WORK.INVOICE;
RUN;

PROC EXPORT DATA=WORK.INVOICE
   OUTFILE="c:\temp\invoice_names.txt"
       DBMS=TAB REPLACE;
   PUTNAMES=YES;
RUN;
PROC PRINT;
RUN;

PROC EXPORT DATA=WORK.INVOICE
   OUTFILE="c:\temp\invoice_data_1st.txt"
      DBMS=TAB REPLACE;
   PUTNAMES=NO;
RUN;

PROC PRINT;
RUN;
```

プログラムの説明

EXPORT プロシジャで PUTNAMES=YES ステートメントを使用します。 WORK.INVOICE が出力された後、PUTNAMES=YES ステートメントを使用して、エクスポート対象の区切りファイル Invoice_names.txt の最初の行に、列名として SAS 変数名を書き込みます。データの最初の行が、区切りファイルの 2 番目の行に書き込まれます。

```
PROC PRINT DATA=WORK.INVOICE;
RUN;

PROC EXPORT DATA=WORK.INVOICE
   OUTFILE="c:\temp\invoice_names.txt"
       DBMS=TAB REPLACE;
   PUTNAMES=YES;
RUN;
PROC PRINT;
RUN;
```

PUTNAMES=NO ステートメントの影響。 このステートメントを NO に設定すると、PROC EXPORT によって、エクスポート対象の区切りファイルの最初の行に、データの最初の行が書き込まれます。このため、SAS 変数名はスキップされ、列はラベルが設定されないままになります。

```
PROC EXPORT DATA=WORK.INVOICE
   OUTFILE="c:\temp\invoice_data_1st.txt"
      DBMS=TAB REPLACE;
   PUTNAMES=NO;
RUN;

PROC PRINT;
RUN;
```

SAS ログ

この SAS ログには、生成された SAS DATA ステップを含む、正常なエクスポート処理に関する情報が表示されています。ログは複数のセクションに分かれていますが、これはドキュメント内での見やすさを考慮しただけです。

```
490  PROC EXPORT DATA= WORK.INVOICE 491              OUTFILE= "c:\temp
\invoice_names.txt" 492              DBMS=TAB REPLACE; 493       PUTNAMES=YES;
494  RUN; 495   /
********************************************************************** 496   *
PRODUCT:   SAS 497   *   VERSION:   9.4 498   *   CREATOR:   External File
Interface 499   *   DATE:      24MAY14 500   *   DESC:      Generated SAS
Datastep Code 501   *   TEMPLATE SOURCE:(None Specified.)502
**********************************************************************/
503      data _null_; 504      %let _EFIERR_ = 0; /* set the ERROR detection
macro variable */ 505      %let _EFIREC_ = 0;     /* clear export record count
macro variable */ 506      file 'c:\temp\invoice_names.txt' delimiter='09'x DSD
DROPOVER lrecl=32767; 507      if _n_ = 1 then        /* write column names or
labels */ 508       do; 509         put 510            "INVNUM" 511
'09'x 512            "BILLEDTO" 513         '09'x 514            "AMTBILL"
515         '09'x 516            "COUNTRY" 517         '09'x 518
"AMTINUS" 519         '09'x 520            "BILLEDBY" 521         '09'x
522            "BILLEDON" 523         '09'x 524            "PAIDON"
525         ; 526       end; 527     set  WORK.INVOICE   end=EFIEOD; 528
format INVNUM best12.; 529         format BILLEDTO $8.; 530         format
AMTBILL dollar18.2 ; 531         format COUNTRY $20.; 532         format AMTINUS
dollar18.2 ; 533         format BILLEDBY best12.; 534         format BILLEDON
date9.; 535         format PAIDON date9.; 536       do; 537         EFIOUT + 1;
538         put INVNUM @; 539         put BILLEDTO $ @; 540         put AMTBILL
@; 541         put COUNTRY $ @; 542         put AMTINUS @; 543         put
BILLEDBY @; 544         put BILLEDON @; 545         put PAIDON ; 546         ;
547       end; 548      if _ERROR_ then call symputx('_EFIERR_',1);  /* set
ERROR detection macro variable */ 549      if EFIEOD then call
symputx('_EFIREC_',EFIOUT); 550      run;
```

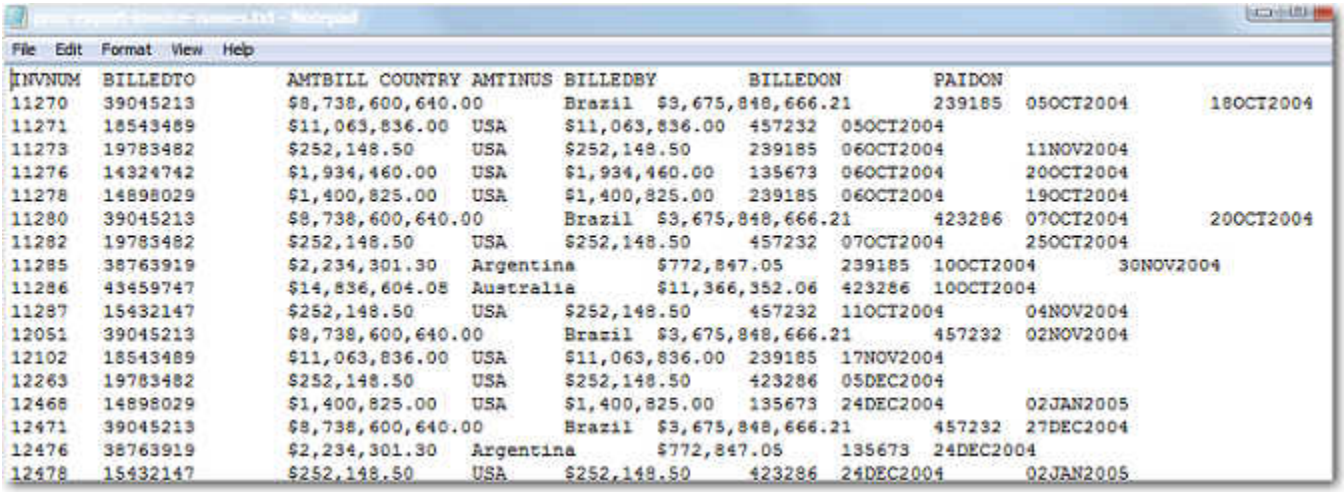
```
NOTE:The file 'c:\temp\invoice_names.txt' is:Filename=c:\temp\invoice_names.txt,
RECFM=V,LRECL=32767,File Size (bytes)=0, Last Modified=24May2014:15:46:36,
Create Time=24May2014:15:46:36 NOTE:18 records were written to the file 'c:\temp
\invoice_names.txt'.The minimum record length was 60.The maximum record length
was 84.NOTE:There were 17 observations read from the data set
WORK.INVOICE.NOTE:DATA statement used (Total process time): real time
0.03 seconds cpu time            0.03 seconds 17 records created in c:\temp
\invoice_data_1st.txt from WORK.INVOICE.NOTE:"c:\temp\invoice_data_1st.txt" file
was successfully created.NOTE:PROCEDURE EXPORT used (Total process time): real
time           0.20 seconds cpu time            0.17 seconds 551  PROC PRINT;
RUN; NOTE:No observations in data set SASUSER.SASMBC.NOTE:PROCEDURE PRINT used
(Total process time): real time           0.01 seconds cpu time            0.01
seconds 552 553 554  PROC EXPORT DATA= WORK.INVOICE 555              OUTFILE= "c:
\temp\invoice_data_1st.txt" 556              DBMS=TAB REPLACE; 557
PUTNAMES=NO; 558  RUN;
```

```
559   /************************************************************************
560   *   PRODUCT:   SAS 561   *   VERSION:   9.4 562   *   CREATOR:   External
File Interface 563   *   DATE:      24MAY14 564   *   DESC:      Generated SAS
Datastep Code 565   *   TEMPLATE SOURCE:(None Specified.)566
************************************************************************/
567      data _null_; 568      %let _EFIERR_ = 0; /* set the ERROR detection
macro variable */ 569      %let _EFIREC_ = 0;     /* clear export record count
macro variable */ 570      file 'c:\temp\invoice_data_1st.txt' delimiter='09'x
DSD DROPOVER lrecl=32767; 571     set  WORK.INVOICE   end=EFIEOD; 572
format INVNUM best12.; 573         format BILLEDTO $8.; 574         format
AMTBILL dollar18.2 ; 575         format COUNTRY $20.; 576         format AMTINUS
dollar18.2 ; 577         format BILLEDBY best12.; 578         format BILLEDON
date9.; 579         format PAIDON date9.; 580       do; 581         EFIOUT + 1;
582         put INVNUM @; 583         put BILLEDTO $ @; 584         put AMTBILL
@; 585         put COUNTRY $ @; 586         put AMTINUS @; 587         put
BILLEDBY @; 588         put BILLEDON @; 589         put PAIDON ; 590         ;
591       end; 592      if _ERROR_ then call symputx('_EFIERR_',1);  /* set
ERROR detection macro variable */ 593      if EFIEOD then call
symputx('_EFIREC_',EFIOUT); 594      run; NOTE:The file 'c:\temp
\invoice_data_1st.txt' is:Filename=c:\temp\invoice_data_1st.txt,
RECFM=V,LRECL=32767,File Size (bytes)=0, Last Modified=24May2014:15:46:36,
Create Time=24May2014:15:46:36 NOTE:17 records were written to the file 'c:\temp
\invoice_data_1st.txt'.The minimum record length was 60.The maximum record
length was 84.NOTE:There were 17 observations read from the data set
WORK.INVOICE.NOTE:DATA statement used (Total process time): real time
0.03 seconds cpu time            0.03 seconds 17 records created in c:\temp
\invoice_data_1st.txt from WORK.INVOICE.NOTE:"c:\temp\invoice_data_1st.txt" file
was successfully created.NOTE:PROCEDURE EXPORT used (Total process time): real
time           0.07 seconds cpu time            0.07 seconds 595 596  PROC
PRINT; RUN; NOTE:PROCEDURE PRINT used (Total process time): real time
0.00 seconds cpu time            0.00 seconds
```

出力例

PROC EXPORT PUTNAMES=YES ステートメントを使用して、SAS 変数名はタブで区切られたファイルの列見出しにマッピングされます。これはデフォルトの動作です。

アウトプット 21.5 *PUTNAMES=YES ステートメントを使用してタブで区切られたファイルにエクスポートされる SAS データ*



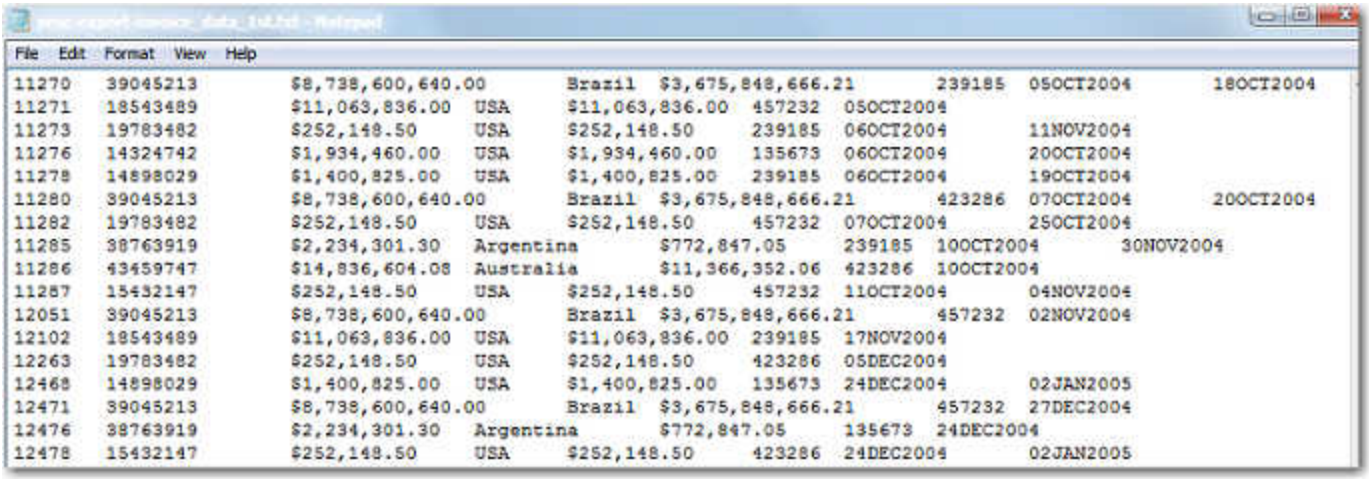
File Edit Format View Help

INVNUM	BILLEDTO	AMTBILL	COUNTRY	AMTINUS	BILLEDBY	BILLEDON	PAIDON		
11270	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	239185	05OCT2004	18OCT2004
11271	18543489	$11,063,836.00	USA		$11,063,836.00	457232	05OCT2004		
11273	19783482	$252,148.50	USA		$252,148.50	239185	06OCT2004	11NOV2004	
11276	14324742	$1,934,460.00	USA		$1,934,460.00	135673	06OCT2004	20OCT2004	
11278	14898029	$1,400,825.00	USA		$1,400,825.00	239185	06OCT2004	19OCT2004	
11280	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	423286	07OCT2004	20OCT2004
11282	19783482	$252,148.50	USA		$252,148.50	457232	07OCT2004	25OCT2004	
11285	38763919	$2,234,301.30	Argentina		$772,847.05	239185	10OCT2004	30NOV2004	
11286	43459747	$14,836,604.05	Australia		$11,366,352.06	423286	10OCT2004		
11287	15432147	$252,148.50	USA		$252,148.50	457232	11OCT2004	04NOV2004	
12051	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	457232	02NOV2004	
12102	18543489	$11,063,836.00	USA		$11,063,836.00	239185	17NOV2004		
12263	19783482	$252,148.50	USA		$252,148.50	423286	05DEC2004		
12468	14898029	$1,400,825.00	USA		$1,400,825.00	135673	24DEC2004	02JAN2005	
12471	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	457232	27DEC2004	
12476	38763919	$2,234,301.30	Argentina		$772,847.05	135673	24DEC2004		
12478	15432147	$252,148.50	USA		$252,148.50	423286	24DEC2004	02JAN2005	

出力例

PROC EXPORT PUTNAMES=NO ステートメントを使用すると、タブ区切りのファイルに名前のない列が含まれます。

アウトプット 21.6 PUTNAMES=NO ステートメントを使用してタブで区切られたファイルにエクスポートされる SAS データ



File Edit Format View Help

11270	39045213	$8,738,600,640.00	Brazil	$3,675,848,666.21	239185	05OCT2004	18OCT2004	
11271	18543489	$11,063,836.00	USA	$11,063,836.00	457232	05OCT2004		
11273	19783482	$252,148.50	USA	$252,148.50	239185	06OCT2004	11NOV2004	
11276	14324742	$1,934,460.00	USA	$1,934,460.00	135673	06OCT2004	20OCT2004	
11278	14898029	$1,400,825.00	USA	$1,400,825.00	239185	06OCT2004	19OCT2004	
11280	39045213	$8,738,600,640.00	Brazil	$3,675,848,666.21	423286	07OCT2004	20OCT2004	
11282	19783482	$252,148.50	USA	$252,148.50	457232	07OCT2004	25OCT2004	
11285	38763919	$2,234,301.30	Argentina	$772,847.05	239185	10OCT2004	30NOV2004	
11286	43459747	$14,836,604.08	Australia	$11,366,352.06	423286	10OCT2004		
11287	15432147	$252,148.50	USA	$252,148.50	457232	11OCT2004	04NOV2004	
12051	39045213	$8,738,600,640.00	Brazil	$3,675,848,666.21	457232	02NOV2004		
12102	18543489	$11,063,836.00	USA	$11,063,836.00	239185	17NOV2004		
12263	19783482	$252,148.50	USA	$252,148.50	423286	05DEC2004		
12468	14898029	$1,400,825.00	USA	$1,400,825.00	135673	24DEC2004	02JAN2005	
12471	39045213	$8,738,600,640.00	Brazil	$3,675,848,666.21	457232	27DEC2004		
12476	38763919	$2,234,301.30	Argentina	$772,847.05	135673	24DEC2004		
12478	15432147	$252,148.50	USA	$252,148.50	423286	24DEC2004	02JAN2005	

22 章

FCMP プロシジャ

概要: FCMP プロシジャ

FCMP プロシジャの動作について

SAS 関数コンパイラ(FCMP)プロシジャを使用すると、SAS 関数や CALL ルーチンやサブルーチンを、他の SAS プロシジャや DATA ステップで使う前に、作成、テスト、そして保存できます。PROC FCMP ではデータセットに保存される DATA ステップシンタックスを使って関数や CALL ルーチンやサブルーチンを作成できます。プロシジャは DATA ステップステートメントの少々のバリエーションは受け入れ、PROC FCMP で作成する関数や CALL ルーチンには SAS プログラム言語のほとんどの関数を使用できます。他の SAS 関数や CALL ルーチンやサブルーチンを呼ぶのと変わらず、DATA ステップから PROC FCMP 関数と CALL ルーチンを呼ぶことができます。これによりプログラマは独立していて再利用可能なサブルーチンで、複雑なコードも容易に読めて、書けて、メンテナンスすることができます。PROC FCMP ルーチンは、保存場所にアクセス可能であれば、どの DATA ステップまたは SAS プロシジャでも使用できます。

FCMP プロシジャでは、SAS 言語コンパイラを使用して SAS プログラムをコンパイルし実行します。コンパイラサブシステムは、SAS が動作しているコンピュータのマシン語を作成します。CMPOPT オプションで値を指定すれば、マシン語コードが最適化され実行が効率的に行われます。SAS 言語コンパイラで使用するこの種のコード作成最適化については、“CMPOPT= System Option” (*SAS System Options: Reference*)を参照してください。

PROC FCMP で作成した関数やサブルーチンは、DATA ステップや WHERE ステートメント、Output Delivery System (ODS)、および次のプロシジャで使用できます。

PROC CALIS	PROC OPTMODEL
PROC FORMAT	PROC PHREG

PROC GA	PROC QUANTREG
PROC GENMOD	PROC REPORT COMPUTE ブロック
PROC GLIMMIX	SAS リスクディメンションプロシジャ
PROC MCMC	PROC SEVERITY
PROC MODEL	PROC SIMILARITY
PROC NLIN	PROC SQL (配列引数を使う関数はサポートしていません)
PROC NLMIXED	PROC SURVEYPHREG
PROC NLP	PROC VARMAX
PROC OPTLSO	

概念: FCMP プロシジャ

関数とサブルーチンの作成

PROC FCMP では、DATA ステップシンタックスを使って関数や CALL ルーチンやサブルーチンを作成できます。PROC FCMP 関数と CALL ルーチンは、データセットに保存され、NLIN プロシジャや MODEL プロシジャや NLP プロシジャなどといった、複数の SAS/STAT プロシジャまたは SAS/ETS プロシジャまたは SAS/OR プロシジャから呼び出すことができます。単一の FCMP プロシジャステップで複数の関数と CALL ルーチンを作成できます。

関数は他のプログラミング言語で使用されるルーチンと同等です。それらは省略可能な引数(複数可)を持つ、個々に独立した計算ブロックです。サブルーチンは戻り値を持たない特別な種類の関数です。関数またはサブルーチンブロック内で作成されたすべての変数はそのサブルーチンのローカルとなります。

関数とサブルーチンの作成: 例

次の例では関数とサブルーチンを定義します。関数は FUNCTION ステートメントで始まり、サブルーチンは SUBROUTINE ステートメントで始まります。DAY_DATE 関数は、日付を数値の曜日に変換し、INVERSE サブルーチンは単一の逆関数を計算します。最後は ENDSUB ステートメントで終わります。

```
proc fcmp outlib=sasuser.MySubs.MathFncs;

   function day_date(indate, type $);
      if type="DAYS" then wkday=weekday(indate);
      if type="YEARS" then wkday=weekday(indate*365);
      return(wkday);
   endsub;
```

```
   subroutine inverse(in, inv);
      outargs inv;
      if in=0 then inv=.;
      else inv=1/in;
   endsub;
run;
```

関数とサブルーチンは、DATA ステップ構文の後に続きます。現在の FCMP プロシジャステップですでに定義されている関数とサブルーチン、および DATA ステップ関数のほとんどは、これらのルーチン内から呼び出すことができます。上記の例で、DATA ステップ関数 WEEKDAY は、DAY_DATE によって呼び出されます。

本例のルーチンは、MathFncs というパッケージ内の Sasuser.MySubs データセットに保存されます。パッケージは、ユーザーによって指定される関連ルーチンの集まりです。このように、データセット内の関連するサブルーチンと関数がグループ化されます。PROC FCMP は、PROC FCMP ステートメントの OUTLIB=オプションによりコンパイルするサブルーチンの保存場所を認識し、LIBRARY=オプションによりライブラリの読み込み場所(C または SAS)を認識します。

注: 関数名とサブルーチン名は、1 つのパッケージ内で一意である必要があります。ただし、異なるパッケージに、同じ名前のサブルーチンと関数を含めることはできます。あいまいさがある場合に特定のルーチンを選択するには、サブルーチン名の接頭辞としてパッケージ名と期間を使用します。たとえば、INVERSE の MthFncs バージョンにアクセスするには、MthFncs.inverse を使用します。

独自の関数の作成

独自の関数と CALL ルーチンの作成の利点

PROC FCMP では、DATA ステップシンタックスを使って関数や CALL ルーチンやサブルーチンを作成できます。独自の関数と CALL ルーチンを作成することの利点には、次のことがあります。

- 関数や CALL ルーチンによって、プログラムは読みやすく、書きやすく、変更しやすくなります。
- 関数や CALL ルーチンは独立しています。ルーチンを呼ぶプログラムは、ルーチンの実装には影響されません。
- 関数や CALL ルーチンは再利用できます。関数やルーチンが保存されているデータセットにアクセスできれば、どんなプログラムでもルーチンを呼び出せます。

注: 作成する PROC FCMP ルーチンは存在するビルトイン SAS 関数と同じ名前は持てません。もしも名前が同じであれば、SAS は同じ名前のビルトイン SAS 関数またはサブルーチンが存在していますというエラーメッセージを生成します。

ユーザー定義の関数の作成

次のプログラムでは、DATA ステップから PROC FCMP 関数を作成および呼ぶのに使うシンタックスを示します。この例では、治験期間の実験日数を計算します。

この例では、TRIAL という名前のパッケージの中に STUDY_DAY という名前の関数を作成します。パッケージとは、固有名を持ち、Sasuser.Funcs データセットに保存されているルーチンの集合です。STUDY_DAY は *intervention_date* と *event_date* の 2 個の数値引数を持てます。ルーチン内では、DATA ステップの構文を使って次の 2 つの日付の差を計算します。*intervention_date* より前の日数は-1 から始まって減少し、*intervention_date* 以降の日付は 1 から始まり増加します。この関数は実験日数に 0 を返すことはありません。

STUDY_DAY は DATA ステップコードから他の関数と何も変りなく呼ぶことができます。DATA ステップで STUDY_DAY への呼び出しが発生すると、この関数は関数の従来のライブラリで検出されません。SAS は STUDY_DAY を含むパッケージを探すのに、CMPLIB=システムオプションに指定されているライブラリやデータセットを検索します。この例では、STUDY_DAY は Sasuser.Funcs.Trial にあります。プログラムは関数を呼び、*start* と *today* の変数値を渡し、変数 sd で結果を返します。

```
proc fcmp outlib=sasuser.funcs.trial;
   function study_day(intervention_date, event_date);
       n=event_date-intervention_date;
       if n <= 0 then
       n=n+1;
       return(n);
   endsub;

options cmplib=sasuser.funcs;
data _null_;
   start='15Feb2006'd;
   today='27Mar2006'd;
   sd=study_day(start, today);
   put sd=;
run;
```

ログ 22.1 *ユーザー定義関数 STUDY_DAY の出力*

```
sd=41
```

ライブラリオプションの使用

PROC FCMP は OUTLIB=または INLIB=オプションと一緒に使用できます。このプロシジャのシンタックスは次の形式になります。

```
proc fcmp
outlib=libname.dataset.package
          inlib=in-libraries;
routine-declarations;
```

OUTLIB=オプションは必須で、*routine-declarations* セクションで宣言されたルーチンが保存されているパッケージを指定します。

routine-declarations セクションで宣言されたルーチンは、他のパッケージにある FCMP ルーチンを呼ぶこともできます。ルーチンを探して、そして呼び出しの妥当性を確認するために、SAS は INLIB=オプションで指定されているデータセットを検索します。INLIB=オプションの形式は次のようになります。

```
inlib=library.dataset
inlib=(library1.dataset1 library2.dataset2 ... libraryN.datasetN)
inlib=library.datasetM - library.datasetN
```

宣言されているルーチンが他のパッケージの FCMP ルーチンを呼ばない場合は、INLIB=オプションを指定する必要はありません。

関数の宣言

プログラムの *routine-declarations* セクションに、関数(複数可)と CALL ルーチンを宣言します。ルーチンには 4 つのパーツがあります。

- 名前
- パラメータ(複数可)

- コードの本体
- RETURN ステートメント

これらのキーワードを FUNCTION または SUBROUTINE キーワードと ENDSUB キーワードの間に指定します。関数のシンタックスは次の形式になります。

```
function
name(argument-1 <, argument-2, ...>);
   program-statements;
   return(expression);
endsub;
```

FUNCTION キーワードの後に、関数名と引数を指定します。関数の宣言の引数は仮引数とよばれ、関数の本体で使用できます。文字列引数を指定するには、引数名の後にドルマーク($)を付加してください。関数では、すべての引数は値で渡されます。つまり、実際の引数の値、変数または呼び出し側の環境から関数に渡される値は、関数がそれを使用する前にコピーされます。コピーすることによって、関数による仮引数の変更がオリジナルの値から変更していないと確認できます。

RETURN ステートメントは関数に値を返すのに使用されます。RETURN ステートメントはかっこで囲まれた表現を受け入れ、そして呼び出し側の環境に返す値を保持しています。関数の宣言は ENDSUB ステートメントで終わります。

CALL ルーチンの宣言

CALL ルーチンは、FUNCTION キーワードのかわりに SUBROUTINE キーワードを使って *routine-declarations* 内で宣言されます。関数と CALL ルーチンは同じ型ですが、CALL ルーチンは値を返さず、また CALL ルーチンではパラメータを変更できる点が違います。OUTARGS ステートメントで変更する引数を指定します。CALL ルーチンの宣言のシンタックスは次のようになります。

```
subroutine
name(argument-1 <, argument-2, ...>);
   outargs out-argument-1 <, out-argument-2, ...>;
   program-statements;
   return;
endsub;
```

OUTARGS ステートメントにある仮引数は、値ではなく参照として渡されます。つまり、CALL ルーチンによる仮引数の変更により、渡された元の変数が変更されるということです。また、CALL ルーチンの起動時に値がコピーされません。コピー数を減らすことで、CALL ルーチンと呼び出し環境の間で大量のデータを渡すときのパフォーマンスを向上させることができます。

RETURN ステートメントは、CALL ルーチンの定義内のオプションです。RETURN ステートメントの実行時、実行はすぐに呼び出し側に返されます。CALL ルーチン内の RETURN ステートメントは、値を返しません。

プログラムステートメントの作成

プログラムのプログラムステートメントセクションは、関数または CALL ルーチンによって実行される作業を説明する、一連の DATA ステップステートメントです。ほとんどの DATA ステップステートメントと関数は、PROC FCMP ルーチンからアクセス可能です。DATA ステップファイルとデータセット I/O ステートメント(INPUT、FILE、SET、MERGE など)は、PROC FCMP ルーチンから使用できません。ただし、PUT ステートメントの一部の機能はサポートされています。詳細については、“PROC FCMP と DATA ステップの相違点” (712 ページ)を参照してください。

出力形式の関数の使用

PROC FCMP により、関数を使用して、最初に値の関数を実行することにより値をフォーマットできます。値をフォーマットする関数を使用して、カスタマイズされた出力形式を作成できます。詳細については、27 章, “FORMAT プロシジャ,” (834 ページ)を参照してください。

PROC FCMP との DATA ステップステートメントの使用

DATA ステップステートメントを PROC FCMP と使用できます。ただし、PROC FCMP に対する構文と機能にはいくつかの相違点があります。詳細については、“PROC FCMP と DATA ステップの相違点” (712 ページ)を参照してください。

DROP、KEEP、FORMAT、LENGTH ステートメントの動作は、PROC FCMP と DATA ステップで同じです。

次の DATA ステップステートメントは、PROC FCMP でサポートされていません。

- DATA
- SET
- MERGE
- UPDATE
- MODIFY
- INPUT
- INFILE

FILE ステートメントのサポートは、PROC FCMP の LOG 出力先および PRINT 出力先に制限されています。OUTPUT ステートメントは PROC FCMP でサポートされていますが、関数またはサブルーチン内ではサポートされていません。

次のステートメントは PROC FCMP でサポートされていますが、DATA ステップではサポートされていません。

- FUNCTION
- STRUCT
- SUBROUTINE
- OUTARGS

構文: FCMP プロシジャ

PROC FCMP *option(s)*;
ABORT;
ARRAY *array-name*[*dimensions*] </NOSYMBOLS | *variable(s)* | *constant(s)* | (*initial-values)*>;
ATTRIB *variable(s)* <FORMAT=*format-name* ><LABEL='*label*'>< LENGTH=*length*>;
DELETEFUNC *function-name*;
DELETESUBR *subroutine-name*;
FUNCTION *function-name(argument(s))* <VARARGS> <$> <*length*>
<KIND | GROUP='*string*' <LABEL='*string-2*'>>;
LABEL *variable*='*label*';
LISTFUNC *function-name*;
LISTSUBR *subroutine-name*;
STRUCT *structure-name variable*;
SUBROUTINE *subroutine-name (argument(s))* <VARARGS>
<LABEL='*label*'> <KIND | GROUP='*string*'>;
OUTARGS *out-argument(s)*;

ステートメント	タスク	例
“PROC FCMP ステートメント”	その他の SAS プロシジャが使用するための SAS 関数を作成、テスト、保存します	Ex. 1, Ex. 2, Ex. 8, Ex. 7
“ABORT ステートメント”	現在の DATA ステップ、SAS ジョブまたは SAS セッションの実行を終了します	
“ARRAY ステートメント”	名前を変数および定数のリストと関連付けます	Ex. 8
“ATTRIB ステートメント”	変数の出力形式、ラベル、長さ情報を指定します	
“DELETEFUNC ステートメント”	OUTLIB オプションで指定される関数ライブラリから関数を削除します	
“DELETESUBR ステートメント”	OUTLIB オプションで指定される関数ライブラリからサブルーチンを削除します	
“FUNCTION ステートメント”	変更された変数値を返します	Ex. 1, Ex. 2, Ex. 7
“LABEL ステートメント”	変数のラベルを指定します	
“LISTFUNC ステートメント”	SAS リスティングの関数のソースコードを書き込みます	
“LISTSUBR ステートメント”	SAS リスティングのサブルーチンに対するソースコードを書き込みます	

ステートメント	タスク	例
“STATIC ステートメント”	変数が再度割り当てられるまで、前の呼び出しからの変数の値が保持されます。	
“STRUCT ステートメント”	構造の種類を宣言(作成)します	
“SUBROUTINE ステートメント”	コードの独立計算ブロックを宣言(作成)します	Ex. 8
“OUTARGS ステートメント”	(SUBROUTINE ステートメントとのみ使用します。)サブルーチンが更新する必要がある引数リストから引数を指定します	Ex. 2, Ex. 8

PROC FCMP ステートメント

SAS 関数、CALL ルーチン、サブルーチンを作成、テスト、保存します。

例: “例 1: 関数の作成と、DATA ステップからの関数の呼出し” (731 ページ)
“例 2: PROC FCMP による関数の作成および保存” (732 ページ)
“例 7: ユーザー定義関数での Graph Template Language (GTL)の使用” (736 ページ)
“例 8: データセットの各行を標準化する” (739 ページ)

構文

PROC FCMP *option(s)*;

オプション引数

ENCRYPT
HIDE
データセットのソースコードをエンコードするように指定します。

FLOW
プログラムの実行時にプログラムのステートメントごとのメッセージの印刷を指定します。このオプションにより、拡張出力が生成されます。

LIBRARY | INLIB=*library.dataset*
LIBRARY | INLIB=(*library-1.dataset library-2.dataset ... library-n.dataset)*
LIBRARY | INLIB=*library.datasetM - library.datasetN*
前にコンパイルされたライブラリがプログラムにリンクされるように指定します。これらのライブラリは、前の PROC FCMP ステップによって、または PROC PROTO (外部 C ルーチン)を使用することによって作成されます。

ヒント ライブラリは OUTLIB=オプションによって作成され、種類が CMPSUB の SAS ライブラリのメンバとして保存されます。LIBRARY=オプションの使用時は、サブルーチンと関数のみプログラムに読み込まれます。

宣言されているルーチンがその他のパッケージの PROC FCMP ルーチンを呼び出さない場合、INLIB=オプションを指定する必要はありません。
libref.dataset 出力形式を使用して、ライブラリの 2 レベルの名前を指定しま

す。名前 *libref* と *dataset* は、8 文字未満の有効な SAS 名である必要があります。

LIBRARY=オプションを使用してファイルのリストを指定できます。また、数値接尾辞を使用して、名前の範囲を指定できます。複数のファイルを指定する場合、単一の名前範囲の場合を除き、リストをかっこで囲む必要があります。構文の例は次のとおりです。

```
proc fcmp library=sasuser.exsubs;
proc fcmp library=(sasuser.exsubs work.examples);
proc fcmp library=lib1-lib10;
```

LIST
LISTSOURCE と LISTPROG オプションが両方ともに有効になるように指定します。

ヒント 割り当てが正しくコンパイルされたことを確認する 1 つの方法として、ソースコードとコンパイル済コードを印刷してから、割り当てステートメントの 2 つのリストを比較します。

LISTALL
LISTCODE、LISTPROG、LISTSOURCE オプションが有効になるように指定します。

LISTCODE
コンパイル済プログラムコードが印刷されるように指定します。LISTCODE には、コンパイラによって生成される一連の操作が表示されます。

ヒント LISTCODE 出力は読みにくいため、LISTPROG オプションを使用してより読みやすいコンパイル済プログラムコードのリストを取得します。

LISTFUNCS
表示可能なすべての FCMP 関数に対するプロトタイプ、またはサブルーチンが SAS リストに書き込まれるように指定します。

LISTPROG
コンパイル済プログラムが印刷されるように指定します。割り当てステートメントのリストが、コンパイラによって生成される一連の操作から生成されます。ソースステートメントテキストは、その他のステートメントに対して印刷されます。

ヒント LISTPROG オプションによって印刷される式は、その式が実際に計算される方法を必ずしも示してはいません。共通の部分式の中間結果が再利用可能ためです。ただし、式は LISTPROG オプションによる拡張出力形式で印刷されます。式が実際にどのように計算されるかについては、LISTCODE オプションからのリストを参照してください。

LISTSOURCE
プログラムのソースコードステートメントが印刷されるように指定します。

OUTLIB=*libname.dataset.package*
PROC FCMP ステップの終了時にコンパイル済サブルーチンと関数が書き込まれる出力データセットの 3 レベルの名前を指定します。この引数は必須です。構文の例は次のとおりです。

```
proc fcmp outlib=sasuser.fcmpsubs.pkt1;

proc fcmp outlib=sasuser.mysubs.math;
```

ヒント このオプションは、サブルーチンと関数を出力ライブラリに保存する場合に使用します。

現在の PROC FCMP ステップ内で宣言されるサブルーチンのみが出力ファイルに保存されます。LIBRARY=オプションを使用してロードされるサブルーチンは、出力ファイルに保存されません。OUTLIB=オプションを指定しない場合、現在の PROC FCMP ステップで宣言されるサブルーチンは保存されません。

PRINT
プログラムの実行時にプログラムのステートメントごとの結果の印刷を指定します。このオプションにより、拡張出力が生成されます。

TRACE
プログラムの実行時にプログラムの各ステートメントの各操作の結果の印刷を指定します。これらの結果は、FLOW オプションによって印刷される情報に加えて生成されます。TRACE オプションは、拡張出力を生成します。

ヒント TRACE を指定することは、FLOW、PRINT、PRINTALL を指定することと同じです。

ABORT ステートメント

現在の DATA ステップ、ジョブ、SAS セッションを終了します。

構文

ABORT;

引数なし

PROC FCMP の ABORT ステートメントに引数はありません。

ARRAY ステートメント

名前を変数と定数のリストと関連付けます。

例: “例 8: データセットの各行を標準化する” (739 ページ)

構文

ARRAY *array-name*[*dimensions*] </NOSYMBOLS | *variable(s)* | *constant(s)* | (*initial-values*)>;

必須引数

array-name
配列の名前を指定します。

dimensions
一次元配列の要素数、または多次元配列の各次元の要素数を数値で表したものです。

オプション引数

/NOSYMBOLS
数値または文字値の配列が関連付けられている要素変数なしで作成されるように指定します。この場合、配列添字によってのみ配列の要素にアクセスできます。

ヒント /NOSYMBOLS は、_TEMPORARY_とまったく同じように使用されます。

名前別の個別の配列要素変数にアクセスする必要がなければ、メモリを節約できます。

variable
配列の変数を指定します。

constant
固定値を示す数値、または文字列を指定します。文字定数を引用符で囲みます。

initial-values
配列の対応する要素の初期値を提供します。内部値をかっこ内に指定できます。

詳細

ARRAY ステートメントの基本

PROC FCMP の ARRAY ステートメントは、DATA ステップで使用される ARRAY ステートメントに似ています。ARRAY ステートメントは、名前を変数と定数のリストと関連付けます。サブスクリプトを含む配列名を使用して、配列のアイテムを参照します。

PROC FCMP で使用される ARRAY ステートメントは、DATA ステップの ARRAY ステートメントのすべての機能をサポートしているわけではありません。PROC FCMP にのみ適用される相違点のリストは、次のとおりです。

- すべての配列参照には、明示的な添字演算子が含まれている必要があります。
- PROC FCMP は名前の後にかっこを使用して、関数呼び出しを表します。配列を参照する場合は、角かっこ[ ]または中かっこ{ }を使用します。
- PROC FCMP の ARRAY ステートメントは、下限指定をサポートしていません。
- 配列には最大 6 次元を使用できます。

PROC FCMP で使用される ARRAY ステートメントの配列要素として、変数と定数の両方を使用できます。要素を定数配列に割り当てることはできません。次元指定と要素リストはオプションですが、これらの値のいずれかを入力する必要があります。配列に対し要素のリストを指定しない場合、または配列のサイズよりも少ない要素をリストする場合、PROC FCMP は数値接尾辞を配列の要素に追加し、要素リストを完了することにより、配列変数を作成します。

配列参照を PROC FCMP ルーチンに渡す

配列を CALL ルーチンに渡し、CALL ルーチンで配列の値を変更しようとする場合、CALL ルーチンの OUTARGS ステートメントで配列引数の名前を指定する必要があります。

例

ARRAY ステートメントの例は、次のとおりです。

- `array spot_rate[3] 1 2 3;`
- `array spot_rate[3] (1 2 3);`

- `array y[4] y1-y4;`
- `array xx[2,3] x11 x12 x13 x21 x22 x23;`
- `array pp p1-p12;`
- `array q[1000] /nosymbols;`

ATTRIB ステートメント

変数のフォーマット、ラベル、長さの情報を指定します。

構文

ATTRIB *variable(s)* <FORMAT=*format-name* LABEL='*label*' LENGTH=*length*>;

必須引数

variable
属性と関連付ける変数を指定します。

オプション引数

FORMAT=*format-name*
フォーマットを *variable* 引数と関連付けます。

LABEL='*label*'
ラベルを *variable* 引数と関連付けます。

LENGTH=*length*
variable 引数の変数の長さを指定します。

例

ATTRIB ステートメントの例は次のとおりです。

- `attrib x1 format=date7. label='variable x1' length=5;`
- ```
  attrib x1 format=date7. label='variable x1' length=5
         x2 length=5
         x3 label='var x3' format=4.
         x4 length=$2 format=$4.;
  ```

DELETEFUNC ステートメント

OUTLIB オプションで指定されている関数ライブラリから関数が削除されます。

構文

DELETEFUNC *function-name*;
  ```

必須引数

function-name
OUTLIB オプションで指定されている関数ライブラリから関数の名前が削除されるように指定します。

DELETESUBR ステートメント

OUTLIB オプションで指定されている関数ライブラリからサブルーチンが削除されます。

構文

DELETESUBR *subroutine-name*;

必須引数

subroutine-name
OUTLIB オプションで指定されている関数ライブラリからサブルーチンの名前が削除されるように指定します。

FUNCTION ステートメント

値を返すルーチンに対し、サブルーチン宣言を指定します。

例: “例 1: 関数の作成と、DATA ステップからの関数の呼出し” (731 ページ)
“例 2: PROC FCMP による関数の作成および保存” (732 ページ)
“例 3: FUNCTION ステートメントに数値データを使用する” (734 ページ)
“例 4: FUNCTION ステートメントによる文字データの使用” (734 ページ)
“例 5: 変数の引数を配列として使用する” (735 ページ)
“例 7: ユーザー定義関数での Graph Template Language (GTL)の使用” (736 ページ)

構文

FUNCTION *function-name*(*argument-1* <, *argument-2, ...*>)<VARARGS> <$> <*length*> <KIND | GROUP='*string*' <LABEL='*string-2*'>>;

... more-program-statements ...

RETURN (*expression*);

ENDSUB;

必須引数

function-name
関数の名前を指定します。

argument
関数に対し引数を指定します。引数名の後にドル記号($)を配置して、文字引数を指定します。次の例では、`function myfunct(arg1, arg2 $, arg3, arg4 $);` arg1 および arg3 は数値引数で、arg2 および arg4 は、文字引数です。

expression
関数から返される値を指定します。

オプション引数

VARARGS
関数が可変個引数をサポートするように指定します。VARARGS を指定すると、関数の最後の引数は配列である必要があります。

制限事項 VARARGS 引数を使用して数値変数を指定する必要があります。

参照項目 “例 5: 変数の引数を配列として使用する” (735 ページ)

$
関数が文字値を返すように指定します。$が指定されていない場合、関数は数値を返します。

length
文字値の長さを指定します。

デフォルト 8

KIND='*string*'
GROUP='*string*'
特定の属性を含む項目の集合を指定します。文字数は 32 文字以下です。

LABEL='*string-2*'
ブランクを含む最大 256 文字のラベルを指定します。

詳細

FUNCTION ステートメントは、値を返すサブルーチン宣言の特殊なケースです。関数の呼び出しに CALL ステートメントを使用しません。関数の定義は、FUNCTION ステートメントで始まり、ENDSUB ステートメントで終わります。

LABEL ステートメント

最大 256 文字のラベルを指定します。

構文

LABEL *variable*='*label*';

必須引数

variable
ラベル付けする変数の名前を指定します。

'*label*'
ブランクを含む最大 256 文字のラベルを指定します。

例

LABEL ステートメントの例は、次のとおりです。

```
• label date='Maturity Date';
```

- `label bignum='Very very large numeric value';`

LISTFUNC ステートメント

関数のソースコードをが SAS リストに書き込まれます。

構文

LISTFUNC *function-name*;

必須引数

function-name
ソースコードが SAS リストに書き込まれている関数の名前を指定します。

LISTSUBR ステートメント

サブルーチンのソースコードが SAS リストに書き込まれます。

構文

LISTSUBR *subroutine-name*;

必須引数

subroutine-name
ソースコードが SAS リストに書き込まれているサブルーチンの名前を指定します。

OUTARGS ステートメント

サブルーチンが更新する引数リストの引数を指定します。

制限事項: 多くの SAS 分析プロシジャが FCMP 関数で解析微分法を実行します。この方法で関数を使用する場合は、OUTARGS ステートメントを使用しないでください。ほとんどの場合、OUTARGS ステートメントを SUBROUTINE ステートメントとともに使用してください。

例: “例 2: PROC FCMP による関数の作成および保存” (732 ページ)
“例 8: データセットの各行を標準化する” (739 ページ)
“例 6: SUBROUTINE ステートメントを CALL ステートメントとともに使用する” (736 ページ)

構文

OUTARGS *out-argument-1* <, *out-argument-2*, ...>;

必須引数

out-argument
サブルーチンにより更新する引数リストから引数を指定します。

ヒント 配列がルーチン内の OUTARGS ステートメントに掲載されている場合は、配列は“参照別”に渡されます。掲載されていない場合は、"値別"に渡されます。

例 サブルーチンの OUTARGS ステートメントの使用例については、“SUBROUTINE ステートメント” (711 ページ)を参照してください。

STATIC ステートメント

変数が再度割り当てられるまで、前の呼び出しからの変数の値が保持されます。

構文

STATIC *variables*, <*initial-value(s)*>;

必須引数

variables
保持する値の変数名、変数リスト、または配列名を指定します。

オプション引数

initial-values
1 つ以上の先行する要素の初期値(数値または文字)を指定します。

詳細

STATIC ステートメントを使用して変数を初期化できます。

通常、関数またはサブルーチン内のローカル変数は関数またはサブルーチンへの呼び出し間で保持されません。コストのかかる初期化が必要となる場合は、STATIC 変数を使用して初期化できます。STATIC 変数はスタックで割り当てられないため、大規模なローカル配列に使用して、スタックで再割り当てやオーバーフローが発生しないようにすることができます。

例

例 1

次に、数値 STATIC の例を示します。

```
proc fcmp;
   function fdef1(in);
      static x1  1;
      if x1 = 1 then do;
         x1 = 2;
      return(in);
   end;
```

```
      return (in*2);
   endfunc;

   ans = fdef1( 1);
   put "Answer should be 1" ans=;
   ans = fdef1( 1);
   put "Answer should be 2" ans=;

run;
```

例 2

次に、文字 STATIC の例を示します。

```
proc fcmp;
   function char_func( in $) $;
   length c1 $ 32;
   static c1  "Elephant";
      if c1 = "Elephant" then
         do;
          c1 = in || c1;
    return (c1);
    end;
    return( in);
   endfunc;
   length ans $ 32;
   ans = char_func( "Big ");
   put "Answer should be >>Big Elephant<<" ans=;
   ans = char_func( "Big ");
   put "Answer should be >>Big<<" ans=;
run;
quit;
```

例 3

次に、配列 STATIC の例を示します。

```
proc fcmp ;
  function array_func( in ) ;
  array a[5] ;
  array foo[5];
  static a  first 1;
  put a[1]= foo[1]=;
     If first  then do;
       do i=1 to dim(a);
          a[i]=i;
          foo[i]=i;
  end;
  first =0;
  end;
  else do;
    do i=1 to dim(a);
       a[i]=a[i]+1;
   end;
   end;
   put a[5];
   return( in);
```

```
  endfunc;

  ans = array_func( 4);

/* should increase by 1 */
   ans = array_func( 4);

run;
```

STRUCT ステートメント

C 言語パッケージに定義されている構造タイプを宣言(作成)します。

構文

STRUCT *structure-name variable*;

必須引数

structure-name
C 言語パッケージで定義され PROC FCMP で宣言されている構造の名前を指定します。

variable
この構造タイプとして宣言する変数を指定します。

例

次に、STRUCT ステートメントの例を示します。

```
struct DATESTR matdate;
matdate.month=3;
matdate.day=22;
matdate.year=2009;
```

SUBROUTINE ステートメント

CALL ステートメントを用いて呼び出せるコードの独立計算ブロックを宣言(作成)します。

例: “例 8: データセットの各行を標準化する” (739 ページ)
“例 2: PROC FCMP による関数の作成および保存” (732 ページ)

構文

SUBROUTINE *subroutine-name* (*argument-1* <, *argument-2*, ...>)<VARARGS> <KIND | GROUP='*string*'>;

OUTARGS *out-argument-1* <, *out-argument-2*, ...>;

... *more-program-statements* ...

ENDSUB;

必須引数

subroutine-name
サブルーチンの名前を指定します。

argument
そのサブルーチンに対して 1 つ以上の引数を指定します。引数名の後にドル記号($)を配置して、文字引数を指定します。次の例では、
`subroutine mysub(arg1, arg2 $, arg3, arg4 $);` arg1 および arg3 は数値引数で、arg2 および arg4 は文字引数です。

OUTARGS
サブルーチンを通して更新する、引数リスト内の引数を指定します。

out-argument
サブルーチンにより更新する引数リストから引数を指定します。

オプション引数

VARARGS
サブルーチンが可変個引数をサポートするように指定します。VARARGS を指定する場合は、サブルーチンの最後の引数は配列である必要があります。

GROUP='*string*'
KIND='*string*'
特定の属性を含む項目の集合を指定します。文字数は 32 文字以下です。

詳細

SUBROUTINE ステートメントにより、CALL ステートメントで呼び出せるコードの独立計算ブロックを宣言(作成)できます。サブルーチンの定義は、SUBROUTINE ステートメントで始まり、ENDSUB ステートメントで終わります。SUBROUTINE ステートメントの OUTARGS ステートメントを使用すれば、サブルーチンにより更新する必要がある引数リストから引数を指定できます。

PROC FCMP と DATA ステップの相違点

PROC FCMP と DATA ステップの相違点の概要

PROC FCMP は本来、複数の SAS/STAT プロシジャ、SAS/ETS プロシジャ、SAS/OR プロシジャに対するプログラミング言語として開発されました。その実装は DATA ステップとまったく同じというわけではなく、これら 2 つの言語には相違点があります。次のセクションでは、PROC FCMP と DATA ステップの相違点をいくつか取り上げます。

PROC FCMP と DATA ステップの相違点

ABORT ステートメント

PROC FCMP の ABORT ステートメントは引数を受け入れません。

ABORT ステートメントは PROC FCMP の関数またはサブルーチン内では無効となります。プロシジャ本体でのみ有効です。

配列

PROC FCMP は名前の後にかっこを使用して、関数呼び出しを表します。配列を参照する場合は、角かっこ[ ]または中かっこ{ }の使用をお勧めします。ARR という名前の配列の場合、コードは **ARR[i]**または **ARR{i}**となります。PROC FCMP では、配列の次元数が 6 に制限されます。

PROC FCMP に対する ARRAY ステートメントの相違点に関する詳細については、“詳細” (704 ページ)を参照してください。

Data セットの入力および出力

PROC FCMP は、出力データセットの作成およびこれへの書き込みに関して、DATA ステートメントと OUTPUT ステートメントをサポートしていません。データセット入力は、SET ステートメント、MERGE ステートメント、UPDATE ステートメント、MODIFY ステートメントをサポートしていません。一般に、データはパラメーターを使用して PROC FCMP ルーチンに変換されます。大量のデータを変換する必要がある場合は、配列を PROC FCMP ルーチンに渡せます。

DATA ステップデバッガ

DATA ステップデバッガを使用するとき、PROC FCMP は他のルーチンと同様に機能します。つまり、デバッガ時にはその関数にはステップインできません。かわりにそのルーチン内で PUT ステートメントを使用してください。

DO ステートメント

DO ステートメントの次のタイプは、PROC FCMP にサポートされています。

```
do i=1, 2, 3;
```

PROC FCMP の DO ステートメントは文字ループ制御変数をサポートしていません。次のコードは PROC FCMP ではなく DATA ステップで実行できます。

```
do i='a', 'b', 'c';
```

DO ステートメントは文字インデックス変数をサポートしていません。したがって、次のコードは PROC FCMP ではサポートされていません。

```
do i='one', 'two', 'three';
```

ファイルの入力および出力

PROC FCMP は PUT ステートメントと FILE ステートメントをサポートしていますが、FILE ステートメントは LOG 出力先と PRINT 出力先に限られています。PROC FCMP には INFILE ステートメントも INPUT ステートメントもありません。

IF 式

IF 式により IF-THEN/ELSE 条件を式内で評価できます。IF 式は、PROC FCMP にはサポートされていますが DATA ステップにはサポートされていません。IF 式は IF-THEN/ELSE ステートメントに分割する必要がないため、一部の式を単純化できます。たとえば、次の 2 つのコードは同等ですが、IF 式(最初の例)は複素数式ではありません。

- ```
  x=if y < 100 then 1 else 0;
  ```
- ```
  if y < 100 then
     x=1;
  else
     x=0;
  ```

IF 式の代替は式です。つまり、操作のグループ化には DO/END ブロックではなく、かっこが使用されます。

PUT ステートメント

PUT ステートメントの構文は PROC FCMP および DATA ステップで似ていますが、操作は異なる可能性があります。PROC FCMP で、PUT ステートメントは通常プログラムデバッグに使用されます。DATA ステップで、PUT ステートメントはレポートとして、またはファイル作成ツール、デバッグツールとして使用されます。次のリストに、その他の相違点を示します。

- PROC FCMP の PUT ステートメントは、行ポインタ、形式修飾子、列出力、因数分解型リスト、反復係数、重ね打ち、_INFILE_オプション、または特殊文字$をサポートしていません。DLM=、DSD などの FILE ステートメントオプションによって提供される機能をサポートしていません。
- PROC FCMP の PUT ステートメントは、式の評価および結果の書き込みを式をかっこ内に置くことによりサポートしています。ただし、DATA ステップは、PUT ステートメントでの式の評価はサポートしていません。PROC FCMP の次の例では、式 **x/100** と **sqrt(y)/2** が評価され、結果が SAS ログに書き込まれます。

  ```
  put (x/100) (sqrt(y)/2);
  ```

  かっこは PROC FCMP で式評価に使用されるため、DATA ステップのように変数またはフォーマットリストに使用できません。
- PROC FCMP の PUT ステートメントは、かっこで囲まれている場合を除いて、添字配列名をサポートしていません。たとえば、ステートメント **put (A[i]);** は配列 A の i 番目の要素を書き込みますが、ステートメント **put A[i];** によりエラーメッセージが表示されます。
- 配列名は、上付きなしの PUT ステートメントで使用できます。そのため、次のステートメントが有効です。
  - **put A=;** (Aが配列であるとき)は、各値が要素変数の名前でラベル付けされている配列 A のすべての要素を書き込みます。
  - **put (A)*=;** は、**put A=;** と同じ出力を書き込みます。
  - **put A;** は、配列 A のすべての要素を書き込みます。
- PROC FCMP の PUT ステートメントがスペースを含む各項目の出力に続きます。これは、DATA ステップのリストモード出力に似ています。列および行の位置に関する詳細な制御は、DATA ステップより少ない程度にサポートされています。
- PROC FCMP の PUT ステートメントは、印刷項目_PDV_をサポートし、ルーチンのプログラムデータベクトルのすべての変数のフォーマットされたリストを印刷します。ステートメント **put _PDV_;** により印刷される変数リストは、ステートメント **put _ALL_;** により印刷されるものよりもはるかに読みやすくなります。

WHEN ステートメントと OTHERWISE ステートメント

WHEN と OTHERWISE ステートメントでは、複数のターゲットステートメントを利用できます。つまり、DO/END グループは複数の WHEN ステートメントに不要です。例は、次のとおりです。

```
SELECT;
   WHEN(expression-1)
statement-1;
   statement-2;
   WHEN (expression-2)
statement-3;
   statement-4;
END;
```

PROC FCMP の追加機能

PROC REPORT と計算ブロック

PROC REPORT は DATA ステップを使用して、計算ブロックを評価します。DATA ステップは PROC FCMP ルーチンを呼び出すことができるため、これらのルーチンを PROC REPORT 計算ブロックから呼び出すこともできます。

FCmp 関数エディタ

FCmp 関数エディタは、関数のパッケージを表示するためのアプリケーションで、SAS エクスプローラに組み込まれています。FCmp 関数エディタには、対話型 SAS セッションの**ソリューション**メニューからアクセスできます。詳細については、“FCmp 関数エディタについて” (781 ページ)を参照してください。

関数の暗黙値の計算

PROC FCMP は、SOLVE 関数を関数の暗黙値の計算に使用します。詳細については、“SOLVE 関数” (774 ページ)を参照してください。

PROC FCMP と Microsoft Excel

通常は SAS で使用できない多くの Microsoft Excel 関数が、PROC FCMP で実装されています。これらの関数は、sashelp.slkwxl データセットにあります。これらの関数は、SAS の Excel 関数で表示できます。

これらの関数は、次の SAS コードを使用することでも表示できます。

```
proc fcmp inlib=sashelp.slkwxl listall;
run;
```

次の例では、ODD_SLK 関数が使用されています。

```
options cmplib=sashelp.slkwxl;
data _null_;
   num    =4.2;
   odd_num=odd_slk(num);
   put 'Odd number nearest to' num ' is ' odd_num;
run;
```

```
Odd number nearest to 4.2  is 5
```

配列の操作

配列を渡す

デフォルトでは、PROC FCMP はルーチン間で“値別”の配列を渡します。ただし、配列がルーチン内の OUTARGS ステートメントでリストされる場合、配列は“参照別”に渡されます。

つまり、関数別の正式なパラメータへの変更により、渡される配列が変更されます。参照別の配列を渡すと、関数と呼び出し環境の間で大量データを効率的に渡すことがで

きます。データをコピーする必要がないためです。正式な配列を指定するための構文には、次の形式があります。

```
function
name(numeric-array-parameter[*],
character-array-parameter[*] $);
```

DATA ステップテンポラリ配列を PROC FCMP ルーチンに渡すことができます。

配列のサイズ変更

組み込み CALL ルーチン DYNAMIC_ARRAY を呼び出して、PROC FCMP ルーチンの配列をサイズ変更できます。この CALL ルーチンの構文には、次の形式があります。

```
call dynamic_array(array, new-dim1-size <, new-dim2-size, ...>);
```

SAS は DYNAMIC_ARRAY CALL ルーチンに、サイズ変更される配列と、配列の各次元の新しいサイズを渡します。動的配列で、ルーチンは必要なメモリ量を割り当てることができます。すべての可能なケースの処理に十分な大きさの配列を作成する必要はありません。

動的配列のサポートは、PROC FCMP ルーチンに制限されています。配列のサイズが変更されると、配列はそのサイズ変更を行ったルーチンでのみ使用できます。DATA ステップ配列をサイズ変更することも、PROC FCMP 動的配列を DATA ステップに返すこともできません。

PROC FCMP ルーチンを含むマクロの使用

%SYSFUNC と%SYSCALL マクロを使用して、PROC FCMP で作成するルーチンを呼び出すことができます。SAS CALL ルーチンはすべて、%SYSCALL でアクセス可能です。LABEL、VNAME、SYMPUT および EXECUTE は除きます。%SYSFUNC と%SYSCALL マクロは、最大 32 文字の SAS 関数名をサポートしています。

PROC FCMP ルーチンの変数のスコープ

変数のスコープの概念

ルーチンとプログラムを互いに独立させる重要な部分は、変数のスコープです。変数のスコープは、変数の値が使用可能なコードのセクションです。PROC FCMP ルーチンの場合、ルーチン外で宣言される変数は、ルーチン内ではアクセスできません。ルーチン内で宣言される変数は、ルーチン外ではアクセスできません。ルーチン内で作成される変数は、ローカル変数といいます。そのスコープがルーチンに対して“ローカル”であるためです。

関数はローカル変数を計算中時にスクラッチ変数として使用し、その変数は関数が返すときに使用できません。関数が呼び出されると、ローカル変数のスペースは、コールスタックでプッシュされます。関数が返す場合、ローカル変数によって使用されるスペースがコールスタックから削除されます。

別のルーチンと同じ名前のローカル変数がある場合

異なるルーチンでローカル変数の名前が同じ場合、変数のスコープの概念がわかりにくい可能性があります。この場合、各ローカル変数は重複しません。次の例では、DATA ステップ、CALL ルーチン subA および subB には、x という名前のローカル変数が含まれています。x はそれぞれその他の x 変数と異なります。プログラムの実行時に、DATA ステップは subA を呼び出し、subA は subB を呼び出します。それぞれの環境は x の値をログに書き込みます。ログ出力には、それぞれの x がどのようにその他と異なっているかが表示されます。

```
proc fcmp outlib=sasuser.funcs.math;
   subroutine subA();
      x=5;
      call subB();
      put 'In subA: ' x=;
   endsub;

   subroutine subB();
      x='subB';
      put 'In subB: ' x=;
   endsub;
run;

options cmplib=sasuser.funcs;
data _null_;
   x=99;
   call subA();
   put 'In DATA step: ' x=;
run;
```

ログ 22.2 *名前が同一である異なるルーチンにローカル変数がある場合*

```
In subB:x=subB In subA:x=5 In DATA step: x=99
```

再帰

PROC FCMP ルーチンは、再帰である可能性があります。再帰は、問題をより解決しやすい小さなものに単純化してから、より単純な解決の結果を統合して完全な解決を作り上げていく、問題解決方法です。再帰関数は、関数自体を直接的または間接的に呼び出す関数です。

ルーチンが呼び出されるたびに、ローカル変数のスペースは、コールスタックでプッシュされます。コールスタックのスペースにより、呼び出しごとのローカル変数の独立性が保証されます。ルーチンが返すとき、コールスタックに割り当てられているスペースが削除され、ローカル変数によって使用されるスペースが解放されます。再帰は、完全な解決に向けた進捗を保存するコールスタックに依存します。

ルーチンがそれ自体を呼び出すとき、呼び出しルーチンと、呼び出し中のルーチンには、中間結果のための独自のローカル変数が含まれている必要があります。呼び出しルーチンで呼び出し中のルーチンのローカル変数が変更可能な場合、再帰ソリューションのプログラムが難しくなります。コールスタックにより、呼び出しごとのローカル変数の独立性が保証されます。

次の例では、PROC FCMP の ALLPERMK ルーチンには *n* と *k* の 2 つの引数があり、このルーチンによりすべての **C(*n*, *k*) = *n*!/(*n* - *k*)!**順列(*n* 要素のうち *k* を含む)が書き込まれます。要素は、バイナリ値(0, 1)と表されます。関数 ALLPERMK は再帰関数 PERMK を呼び出し、全体のソリューションスペースを表示して、特定のフィルタに一致するアイテムのみ出力します。

```
proc fcmp outlib=sasuser.funcs.math;
   subroutine allpermk(n, k);
      array scratch[1] / nosymbols;
      call dynamic_array(scratch, n);
      call permk(n, k, scratch, 1,0);
   endsub;

subroutine permk(n, k, scratch[*], m, i);
   outargs scratch;
   if m-1=n then do;
      if i=k then
         put scratch[*];
      end;
   else do;
      scratch[m]=1;
      call permk(n, k, scratch, m+1, i+1);
      scratch[m]=0;
      call permk(n, k, scratch, m+1, i);
   end;
endsub;
run;
quit;

options cmplib=sasuser.funcs;
data _null_;
   call allpermk(5,3);
run;
```

ログ 22.3 再帰処理例の結果

```
1 1 1 0 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1
0 1 0 1 1 0 0 1 1 1
```

このプログラムでは ARRAY ステートメントの/NOSYMBOLS オプションを使用し、配列要素ごとに変数のない配列を作成します。/NOSYMBOLS 配列は、配列参照、**scratch[*m*]**によってのみアクセス可能で、DATA step _temporary_ array と同じです。/NOSYMBOLS 配列では通常の配列よりも少ないメモリが使用されます。スペースが変数に対して割り当てられていないためです。ALLPERMK では、PROC FCMP 動的配列も使用されます。

ディレクトリトランスバーサル

ディレクトリトランスバーサルの概要

DATA ステップまたはマクロを使用する場合、関数によるディレクトリ階層の表示を可能にする機能の実装は難しくなります。DATA ステップとマクロコードを使用した再帰

または類似再帰は、コード化が容易ではありません。このセクションでは、ディレクトリ階層のすべてのファイルの完全パス名を配列に入力する、DIR_ENTRIES という名前のルーチンの開発方法を説明します。この例では、PROC FCMP と DATA ステップ構文の間の類似性を表し、PROC FCMP ルーチンがプログラムを簡素化し、独立した再利用可能なコードを作成する方法を強調します。DIR_ENTRIES では、入力として次のパラメータが使用されます。

- 相対ディレクトリ
- パス名を入力する結果配列
- 結果配列に配置されるパス名の数である、出力パラメータ
- 結果配列の大きさが十分でなかったためにすべての結果セットが切り捨てられたかどうかを表す出力パラメータ

DIR_ENTRIES の制御のフローは、次のとおりです。

1. 相対ディレクトリを開きます。
2. ディレクトリのエントリごとに、次のタスクのうちいずれかを行います。
   - エントリがディレクトリの場合、DIR_ENTRIES を呼び出し、結果配列にサブディレクトリのパス名を入力します。
   - その他の場合、エントリはファイルで、ファイルのパスを結果配列に追加する必要があります。
3. 相対ディレクトリを閉じます。

ディレクトリトランスバーサルの例

ディレクトリの開閉

ディレクトリの開閉は、CALL ルーチン DIROPEN と DIRCLOSE によって処理されます。DIROPEN はディレクトリパスを受け入れ、次の制御フローを含みます。

1. FILENAME 関数を使用して、パスに対しファイル参照名を作成します。
2. FILENAME 関数が失敗すると、エラーメッセージがログに書き込まれ、返されます。
3. その他の場合、DOPEN 関数を使用して、ディレクトリを開き、ディレクトリ ID を取得します。
4. ディレクトリファイル参照名をクリアします。
5. ディレクトリ ID を返します。

DIRCLOSE CALL ルーチンにディレクトリ ID が渡され、DCLOSE に渡されます。ディレクトリが閉じられた後にプログラムでそのディレクトリ ID が使用される場合にエラーが発生するように、DIRCLOSE は渡されたディレクトリ ID を欠損に設定します。次のコードが、DIROPEN ルーチンと DIRCLOSE CALL ルーチンを実装します。

```
proc fcmp outlib=sasuser.funcs.dir;
   function diropen(dir $);
   length dir $ 256 fref $ 8;
   rc=filename(fref, dir);
   if rc=0 then do;
      did=dopen(fref);
      rc=filename(fref);
   end;
```

```
   else do;
      msg=sysmsg();
      put msg '(DIROPEN(' dir= ')';
      did=.;
   end;
   return(did);
endsub;

subroutine dirclose(did);
   outargs did;
   rc=dclose(did);
   did=.;
endsub;
```

ファイル名の収集

ファイルパスは、DIR_ENTRIES CALL ルーチンによって収集されます。DIR_ENTRIES では、次の引数が使用されます。

- 相対ディレクトリ
- 入力する結果配列
- 結果配列のエントリ数を入力する出力パラメータ
- 0 に設定する出力パラメータ(すべてのパス名が結果配列に収まる場合)、または 1 に設定する出力パラメータ(一部のパス名が配列に収まらない場合)

DIR_ENTRIES の本文は、DATA ステップのこの機能性の実行に使用されるコードとほぼ同じです。また DIR_ENTRIES は、いくつかのプログラムで簡単に再利用される CALL ルーチンでもあります。

DIR_ENTRIES は DIROPEN を呼び出し、ディレクトリを開いて、ディレクトリ ID を取得します。次にルーチンは DNUM を呼び出し、ディレクトリのエントリ数を取得します。ディレクトリのエントリごとに、DREAD が呼び出され、エントリ名が取得されます。エントリ名が利用可能で、ルーチンは MOPEN を呼び出し、エントリがファイルかディレクトリかどうかを決定します。

エントリがファイルの場合、MOPEN は正の値を返します。この場合、ファイルへの完全パスが結果配列に追加されます。結果配列がいっぱいになると、切り捨て出力引数が 1 に設定されます。

エントリがディレクトリの場合、MOPEN は 0 以下の値を返します。この場合、DIR_ENTRIES はサブディレクトリのエントリのパス名を収集します。パス名の収集は、DIR_ENTRIES を再帰的に呼び出し、サブディレクトリのパスを開始パスとして渡して行います。DIR_ENTRIES が返す場合、結果配列にはサブディレクトリのエントリのパスが含まれています。

```
subroutine dir_entries(dir $, files[*] $, n, trunc);
   outargs files, n, trunc;
   length dir entry $ 256;

   if trunc then return;

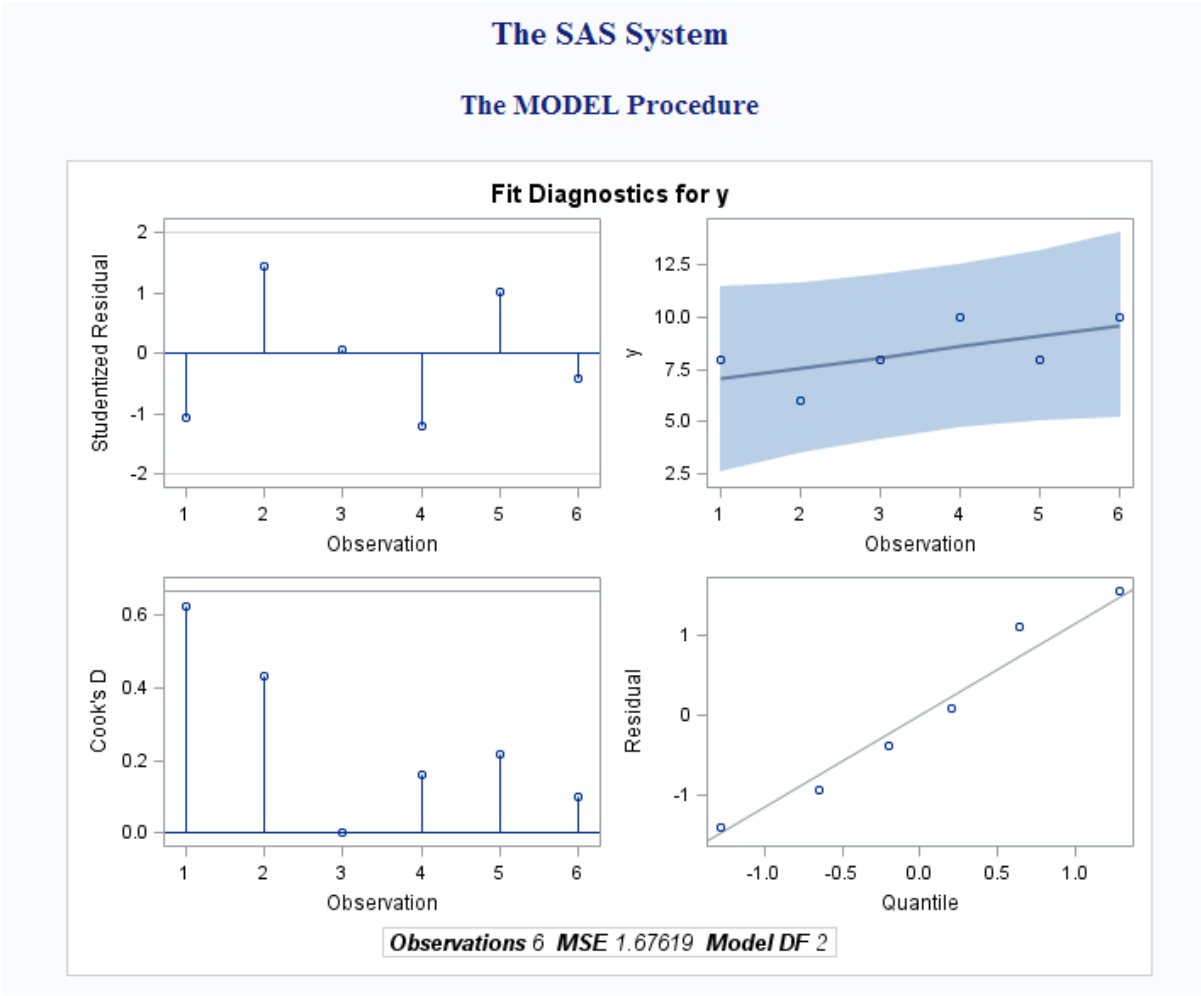
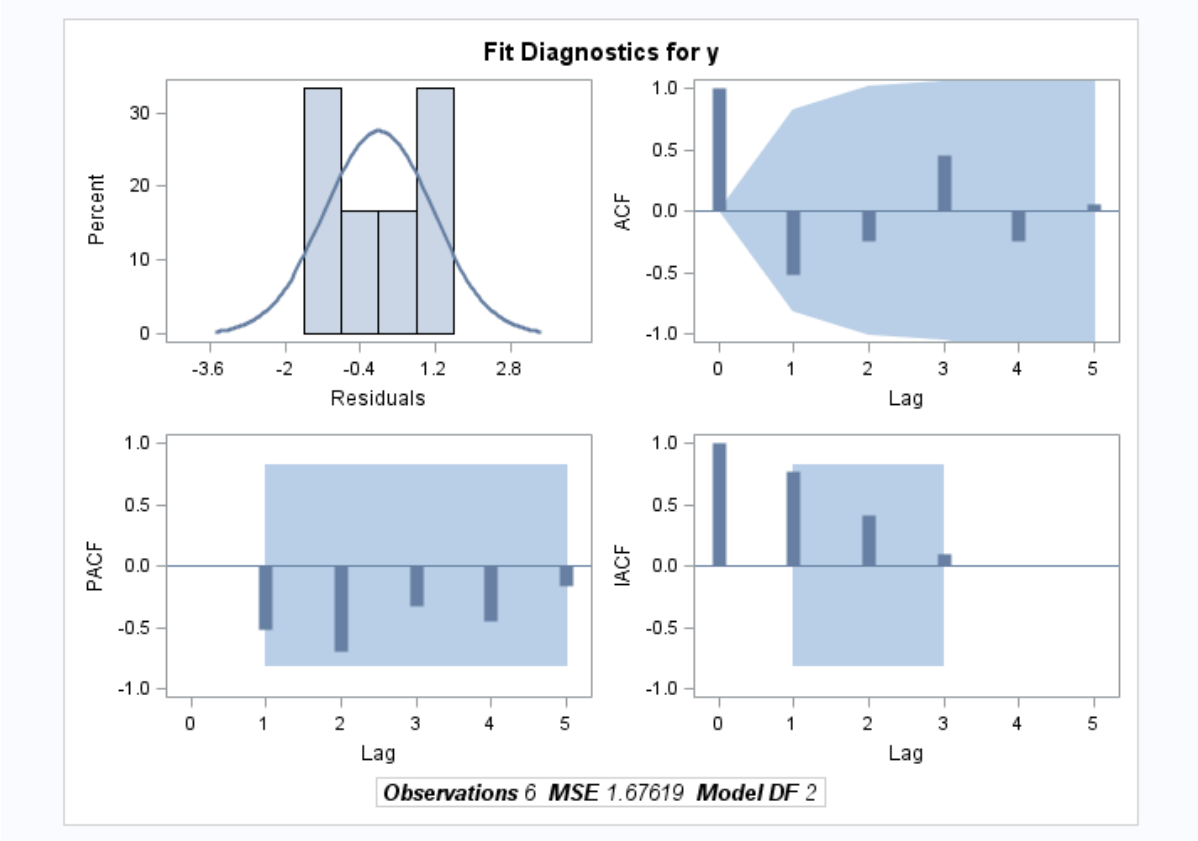
   did=diropen(dir);
   if did <= 0 then return;

   dnum=dnum(did);
   do i=1 to dnum;
      entry=dread(did, i);
         /* If this entry is a file, then add to array, */
```

```
         /* else entry is a directory, recurse.         */
      fid=mopen(did, entry);
      entry=trim(dir) || '\' || entry;
      if fid > 0 then do;
         rc=fclose(fid);
         if n < dim(files) then do;
            trunc=0;
            n=n + 1;
            files[n]=entry;
         end;
         else do;
            trunc=1;
            call dirclose(did);
            return;
         end;
      end;
      else
         call dir_entries(entry, files, n, trunc);
   end;

   call dirclose(did);
   return;
endsub;
```

DATA ステップからの DIR_ENTRIES の呼び出し

その他の DATA ステップ CALL ルーチンのように DIR_ENTRIES を起動します。検出されるすべてのファイルの保持に十分なエントリを含む配列を宣言します。次に、DIR_ENTRIES ルーチンを呼び出します。ルーチンが返すとき、結果配列はループされ、配列の各エントリは SAS ログに書き込まれます。

```
options cmplib=sasuser.funcs;
data _null_;
   array files[1000] $ 256 _temporary_;
   dnum=0;
   trunc=0;
   call dir_entries("c:\logs", files, dnum, trunc);
   if trunc then put 'ERROR: Not enough result array entries. Increase array
size.';
   do i=1 to dnum;
      put files[i];
   end;
run;
```

ログ 22.4 *DATA ステップの呼び出しの結果*

```
c:\logs\2004\qtr1.log c:\logs\2004\qtr2.log c:\logs\2004\qtr3.log c:\logs
\2004\qtr4.log c:\logs\2005\qtr1.log c:\logs\2005\qtr2.log c:\logs\2005\qtr3.log
c:\logs\2005\qtr4.log c:\logs\2006\qtr1.log c:\logs\2006\qtr2.log
```

この例に、PROC FCMP 構文と DATA ステップ間の類似点を示します。たとえば、数値式と制御フローステートメントは同じです。PROC FCMP 関数への DIROPEN の抽出により、DIR_ENTRIES が簡素化されます。作成されるすべての PROC FCMP ルーチンは、新しいコンテキストで機能するルーチンを変更することなく、その他の DATA ステップによって再利用可能です。

コンパイルされた関数とサブルーチンの場所の特定:CMPLIB=システムオプション

CMPLIB=システムオプションの概要

SAS システムオプション CMPLIB=は、前にコンパイルされた関数とサブルーチンの検索場所を指定します。FCMP 関数とサブルーチンの使用をサポートするすべてのプロシジャ(FCMP を含む)では、このシステムオプションを使用します。

関数とサブルーチンをサポートするすべてのプロシジャステートメントで LIBRARY=オプションを指定するかわりに、CMPLIB=システムオプションを使用して、すべてのプロシジャによって使用可能なライブラリを設定できます。

_DISPLAYLOC_オプションは、SAS が機能をロードする場所にあったデータセットの名前を該当するログに書き込みます。_NO_DISPLAYLOC_オプションは、データセット名がログに書き込まれないようにします。

CMPLIB=システムオプションの構文

CMPLIB=オプションの構文には、次の形式があります。

OPTIONS CMPLIB=*library*

OPTIONS CMPLIB=*(library-1 <, library-2, ...>)*

OPTIONS CMPLIB=*list-1 <, list-2, ...*

OPTIONS CMPLIB=_DISPLAYLOC_

OPTIONS CMPLIB=_NO_DISPLAYLOC_

次の説明では、前述の構文について触れます。

OPTIONS
ステートメントを OPTIONS ステートメントとして識別します。

library
前にコンパイルされたライブラリがプログラムにリンクされるように指定します。

list
ライブラリのリストを指定します。

DISPLAYLOC
PROC FCMP を使用するときは、SAS がその機能をロードした場所にあったデータセットが SAS ログに表示されるよう指定します。

要件 CMPLIB=_DISPLAYLOC_を使用するときは、PROC FCMP INLIB=オプションを使用してデータセットを指定します。

_NO_DISPLAYLOC_
PROC FCMP を使用するときは、SAS がその機能をロードした場所にあったデータセットが SAS ログに表示されないよう指定し、CMPLIB=オプション値としてのあらゆるライブラリ指定を削除します。

ヒント _DISPLAYLOC_オプションなしで CMPLIB=library-specification を指定する場合は、SAS では SAS ログのデータセット名が表示されません。

例 1:CMPLIB=システムオプションの設定

次の例に、CMPLIB=システムオプションの設定方法を示します。

- `options cmplib=sasuser.funcs;`
- `options cmplib=(sasuser.funcs work.functions mycat.funcs);`
- `options cmplib=(sasuser.func1 - sasuser.func10);`

例 2:関数のコンパイルと使用

次の例では、PROC FCMP は SIMPLE 関数をコンパイルし、それを Sasuser.Models データセットに保存します。次に、CMPLIB=システムオプションが設定され、関数が PROC MODEL によって呼び出されます。

この例からの出力は、複数のページにわたります。この出力は 5 ページに分割されます。

```
proc fcmp outlib=sasuser.models.yval;
   function simple(a, b, x);
      y=a+b*x;
      return(y);
   endsub;
run;

options cmplib=sasuser.models nodate ls=80;

data a;
   input y @@;
   x=_n_;
   datalines;
08 06 08 10 08 10
;

proc model data=a;
   y=simple(a, b, x);
   fit y / outest=est1 out=out1;
quit;
```

アウトプット 22.1 関数のコンパイルと使用:パート 1

The SAS System

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	2
Equations	1
Number of Statements	1

Model Variables	y
Parameters	a b
Equations	y

The Equation to Estimate is	
y =	F(a, b)

NOTE: At OLS Iteration 1 CONVERGE=0.001 Criteria Met.

アウトプット 22.2 関数のコンパイルと使用:パート 2

The SAS System

The MODEL Procedure
OLS Estimation Summary

Data Set Options	
DATA=	A
OUT=	OUT1
OUTEST=	EST1

Minimization Summary	
Parameters Estimated	2
Method	Gauss
Iterations	1

Final Convergence Criteria	
R	0
PPC	0
RPC(a)	64685.48
Object	0.984333
Trace(S)	1.67619
Objective Value	1.11746

Observations Processed	
Read	6
Solved	6

アウトプット 22.3 関数のコンパイルと使用:パート 3

The SAS System

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors							
Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
y	2	4	6.7048	1.6762	1.2947	0.4084	0.2605

Nonlinear OLS Parameter Estimates				
Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a	6.533333	1.2053	5.42	0.0056
b	0.514286	0.3095	1.66	0.1719

Number of Observations		Statistics for System	
Used	6	Objective	1.1175
Missing	0	Objective*N	6.7048

アウトプット 22.4 関数のコンパイルと使用:パート 4



アウトプット 22.5 関数のコンパイルと使用:パート 5



PROC MODEL の詳細については、*SAS/ETS User's Guide* を参照してください。

例 3:SAS が機能をロードした場所にあったデータセットの特定

次の例では、_DISPLAYLOC_オプションと_NO_DISPLAYLOC_オプションを使用します。_DISPLAYLOC_オプションを使用する場合、SAS では、機能をロードする場所にあったデータセットの名前が該当するログに書き込まれます。_NO_DISPLAYLOC_オプションでは、データセット名はログに書き込まれません。

```
proc fcmp outlib=work.myfuncs1.pkg;
   function myfunc();
      return(1);
   endsub;
run;

proc fcmp outlib=work.myfuncs2.pkg;
   function myfunc();
      return(2);
   endsub;
run;

proc fcmp outlib=work.myfuncs3.pkg;
   function myfunc();
      return(3);
   endsub;
run;
```

```
option CMPLIB=(myfuncs1-myfuncs3 _DISPLAYLOC_);

proc fcmp;

  a = myfunc();
  put a=;
run;

/*- turning _DISPLAYLOC_ off -*/
option CMPLIB=(myfuncs1-myfuncs3);


proc fcmp;

  a = myfunc();
  put a=;
run;

option CMPLIB=(myfuncs1 myfuncs2 _DISPLAYLOC_);

proc fcmp;

  a = myfunc();
  put a=;
run;

option CMPLIB=_DISPLAYLOC_;

proc fcmp inlib=work.myfuncs1;
  a = myfunc();
  put a=;
run;

option CMPLIB=_NO_DISPLAYLOC_;

proc fcmp inlib=work.myfuncs1;
  a = myfunc();
  put a=;
run;
```

次の結果は SAS ログの一部を示しています。_DISPLAYLOC_オプションと _NO_DISPLAYLOC オプションでは、結果が異なります。

```
116  option CMPLIB=(myfuncs1-myfuncs3 _DISPLAYLOC_); 117 118  proc fcmp; 119
120    a = myfunc(); 121    put a=; 122  run; NOTE:Function 'myfunc' loaded from
WORK.myfuncs3.PKG.NOTE:PROCEDURE FCMP used (Total process time): real
time           0.06 seconds cpu time            0.04 seconds 123 124  /*-
turning _DISPLAYLOC_ off -*/ 125  option CMPLIB=(myfuncs1-myfuncs3); 126 127
128  proc fcmp; 129 130    a = myfunc(); 131    put a=; 132  run; NOTE:PROCEDURE
FCMP used (Total process time): real time           0.06 seconds cpu
time            0.06 seconds 133 134  option CMPLIB=(myfuncs1 myfuncs2
_DISPLAYLOC_); 135 136  proc fcmp; 137 138    a = myfunc(); 139    put a=; 140
run; NOTE:Function 'myfunc' loaded from WORK.myfuncs2.PKG.NOTE:PROCEDURE FCMP
used (Total process time): real time           0.05 seconds cpu time
0.04 seconds 141 142  option CMPLIB=_DISPLAYLOC_; 143 144  proc fcmp
inlib=work.myfuncs1; 145    a = myfunc(); 146    put a=; 147  run; NOTE:Function
'myfunc' loaded from work.myfuncs1.PKG.NOTE:PROCEDURE FCMP used (Total process
time): real time           0.04 seconds cpu time            0.03 seconds
```

```
149  option CMPLIB=_NO_DISPLAYLOC_; 150 151  proc fcmp inlib=work.myfuncs1;
152    a = myfunc(); 153    put a=; 154  run; NOTE:PROCEDURE FCMP used (Total
process time): real time           0.04 seconds cpu time            0.03 seconds
```

PROC FCMP と DATA ステップの構成要素オブジェクト

SAS では、PROC FCMP、DATA ステップと使用する 2 つの事前定義されたコンポーネントオブジェクト、ハッシュオブジェクトとハッシュイテレータオブジェクトを提供しています。これらのオブジェクトを使用して、ルックアップキーに基づいてデータを迅速かつ効率的に保存、検索、フィルタ、取得できます。これらのオブジェクトは、属性、メソッド、演算子から構成されるデータエレメントです。属性は、オブジェクトと関連付けられている情報を指定するプロパティです。メソッドは、オブジェクトが実行できる操作を定義します。演算子は、特殊機能を提供します。DATA ステップオブジェクトのドット表記を使用して、コンポーネントオブジェクトの属性とメソッドにアクセスします。

SAS 9.3 以降、ハッシュ法は FCMP プロシジャを介したユーザー定義サブルーチンとして利用できます。ハッシュ法を使用すると、プログラムのスコープを拡張できるため、簡明さを損なうことなくより大きな問題を解決できます。PROC FCMP 関数にハッシュオブジェクトを埋め込むことで、既存のプログラムが簡素化され、パフォーマンスが改善されます。

ハッシュ関数を使用することで、入力文字列または数値(キー)は、整数(ハッシュ値)に変換されます。このハッシュ値が、ハッシュテーブルのインデックスとして使用されます。そのため、各キーに対して、関連するデータの保存と取得がハッシュテーブルからできるようになります。ハッシュ法は、キーを介して参照される大量の情報を検索する際に最速の方法だと考えられています。

FCMP ハッシュオブジェクトでは、次のステートメントとメソッドがサポートされます。

- ADD メソッド
- CHECK メソッド
- CLEAR メソッド
- DECLARE ステートメント
- DEFINEDATA メソッド
- DEFINEDONE メソッド
- DEFINEKEY メソッド
- DELETE メソッド
- FIND メソッド
- NUM_ITEMS メソッド
- REMOVE メソッド
- REPLACE メソッド

FCMP ハッシュイテレータオブジェクトでは、次のステートメントとメソッドがサポートされます。

- DECLARE ステートメント
- FIRST メソッド
- LAST メソッド
- NEXT メソッド
- PREV メソッド

ハッシュオブジェクトとハッシュイテレータコンポーネントオブジェクトの詳細については、*SAS コンポーネントオブジェクト: リファレンス*を参照してください。

PROC FCMP でのハッシュの使用方法と例については、PROC FCMP のハッシュ法による生産性の拡大を参照してください。

PROC FCMP とハッシュオブジェクトを使用する他の例については、PROC FCMP を使用したハッシュオブジェクトの SAS データセットへのロードを参照してください。

例: FCMP プロシジャ

例 1: 関数の作成と、DATA ステップからの関数の呼出し

要素: PROC FCMP ステートメントオプション
OUTLIB=
DATA ステップ

詳細

この例では、PROC FCMP で関数を作成してこれを DATA ステップで使用することにより、治験における研究日を計算する方法を示します。

プログラム

```
proc fcmp outlib=sasuser.funcs.trial;

   function study_day(intervention_date, event_date);

      n=event_date - intervention_date;
         if n >= 0 then
            n=n + 1;
         return(n);
   endsub;

options cmplib=sasuser.funcs;

data _null_;
   start='15Feb2010'd;
   today='27Mar2010'd;
   sd=study_day(start, today);

   put sd=;

run;
```

プログラムの説明

コンパイルされた関数と CALL ルーチンが書き込まれる出力パッケージの名前を指定します。 パッケージは、データセット Sasuser.Funcs に保存されます。

```
proc fcmp outlib=sasuser.funcs.trial;
```

STUDY_DAY という関数を作成します。 STUDY_DAY が Trial というパッケージに作成され、2 つの数値入力引数が含まれています。

```
   function study_day(intervention_date, event_date);
```

DATA ステップの IF ステートメントを使用して EVENT_DATE を計算します。 DATA ステップ構文を使用して、EVENT_DATE と INTERVENTION_DATE の間の差異を計算します。INTERVENTION_DATE より前の日は-1 で始まり、それ以下になります。

INTERVENTION_DATE 以降の日は 1 で始まり、それ以上になります(この関数は、研究日に対し 0 を返しません)。

```
      n=event_date - intervention_date;
         if n >= 0 then
            n=n + 1;
         return(n);
   endsub;
```

CMPLIB=システムオプションを使用して、プログラムコンパイラ時に含めるコンパイラサブルーチンを含む SAS データセットを指定します。

```
options cmplib=sasuser.funcs;
```

DATA ステップを作成して、関数 STUDY_DAY に対して値を生成します。 関数は、開始日と今日の日付を使用して、値を計算します。STUDY_DAY は、DATA ステップから呼び出されます。DATA ステップで STUDY_DAY への呼び出しが発生すると、この関数は関数の従来のライブラリで検出されません。STUDY_DAY を含むパッケージに対し CMPLIB システムオプションで指定される各データセットが検索されます。この場合、Sasuser.Funcs.Trial の STUDY_DAY が検出されます。

```
data _null_;
   start='15Feb2010'd;
   today='27Mar2010'd;
   sd=study_day(start, today);
```

出力を SAS ログに書き出します。

```
   put sd=;
```

SAS プログラムを実行します。

```
run;
```

ログ

ログ 22.5 *関数の作成および DATA ステップからの呼び出しの結果*

```
sd=41
```

例 2: PROC FCMP による関数の作成および保存

要素: PROC FCMP ステートメントオプション
OUTLIB=
OUTARGS ステートメント

詳細

この例では、PROC FCMP を使用して本例で使用される関数を作成し保存する方法を示します。

プログラム

```
proc fcmp outlib=sasuser.exsubs.pkt1;
```

```
   subroutine calc_years(maturity, current_date, years);
      outargs years;
      years=(maturity - current_date) / 365.25;
   endsub;

   function garkhprc (type$, buysell$, amount,
                      E, t, S, rd, rf, sig);
      if buysell="Buy" then sign=1.;
      else do;
         if buysell="Sell" then sign=-1.;
         else sign=.;
      end;

      if type="Call" then
         garkhprc=sign * amount
                        * garkhptprc (E, t, S, rd, rf, sig);
      else do;
         if type="Put" then
            garkhprc=sign * amount
                           * garkhptprc (E, t, S, rd, rf, sig);
         else garkhprc=.;
      end;

      return(garkhprc);
   endsub;

run;
```

プログラムの説明

関数パッケージ情報が保存されるエントリを指定します。 パッケージは、3 レベルの名前です。

```
proc fcmp outlib=sasuser.exsubs.pkt1;
```

満期までの年数を計算するための関数を作成します。 CALC_YEARS という汎用関数が宣言され、日数として保存される日付変数から満期までの年数が計算されます。OUTARGS ステートメントは、CALC_YEARS によって更新される変数を指定します。

```
   subroutine calc_years(maturity, current_date, years);
      outargs years;
      years=(maturity - current_date) / 365.25;
   endsub;
```

FX オプションの Garman-Kohlhagen 価格設定のための関数を作成します。 GARKHPRC という関数が宣言されます。これは FX オプションに対する Garman-Kohlhagen 価格設定を計算します。関数は、SAS 関数 GARKHCLPRC と GARKHPTPRC を使用します。

```
   function garkhprc (type$, buysell$, amount,
                      E, t, S, rd, rf, sig);
      if buysell="Buy" then sign=1.;
      else do;
         if buysell="Sell" then sign=-1.;
         else sign=.;
      end;

      if type="Call" then
```

```
            garkhprc=sign * amount
                           * garkhptprc (E, t, S, rd, rf, sig);
         else do;
            if type="Put" then
               garkhprc=sign * amount
                             * garkhptprc (E, t, S, rd, rf, sig);
            else garkhprc=.;
         end;
```

RETURN ステートメントは、GARKHPRC 関数の値を返します。

```
         return(garkhprc);
      endsub;
```

FCMP プロシジャを実行します。 RUN ステートメントは、FCMP プロシジャを実行します。

```
   run;
```

ログ

ログ 22.6 関数の保存場所

```
NOTE:Function garkhprc saved to sasuser.exsubs.pkt1.NOTE:Function calc_years
saved to sasuser.exsubs.pkt1.
```

例 3: FUNCTION ステートメントに数値データを使用する

詳細

次の例では、PROC FCMP の FUNCTION ステートメントへの入力として、数値データを使用します。

プログラム

```
proc fcmp;
   function inverse(in);
      if n=0 then inv=.;
      else inv=1/in;
      return(inv);
   endsub;
run;
```

例 4: FUNCTION ステートメントによる文字データの使用

詳細

次の例では、PROC FCMP の FUNCTION ステートメントへの入力として、文字データを使用します。FUNCTION TEST からの出力には 12 バイト長が割り当てられています。

プログラム

```
options cmplib=work.funcs;

proc fcmp outlib=work.funcs.math;
   function test(x $) $ 12;
   if x='yes' then
      return('si si si');
      else
      return('no');
   endsub;
run;

data _null_;
   spanish=test('yes');
   put spanish=;
run;
```

ログ

ログ 22.7 PROC FCMP の FUNCTION ステートメントで文字データを使用した結果

```
spanish=si si si
```

例 5: 変数の引数を配列として使用する

詳細

次の例に、変数の引数を受け入れる配列を示します。例では、summation 関数が次のように呼び出し可能であることを示します。**sum = summation(1,2,3,4,5);**

プログラム

```
options cmplib=sasuser.funcs;

proc fcmp outlib=sasuser.funcs.temp;
function summation (b[*]) varargs;
    total=0;
    do i=1 to dim(b);
        total=total + b[i];
    end;
return(total);
endsub;
sum=summation(1, 2, 3, 4, 5);
   put sum=;
run;
```

例 6: SUBROUTINE ステートメントを CALL ステートメントとともに使用する

詳細

次に、SUBROUTINE ステートメントの例を示します。SUBROUTINE ステートメントは、CALL ステートメントと使用できるコードの独立した計算ブロックを作成します。

プログラム

```
proc fcmp outlib=sasuser.funcs.temp;
subroutine inverse(in,inv) group="generic";
   outargs inv;
   if in=0 then inv=.;
   else inv=1/in;
endsub;

options cmplib=sasuser.funcs;
data _null_;
   x=5;
   call inverse(x, y);
   put x= y=;
run;
```

ログ

ログ 22.8 PROC FCMP で SUBROUTINE ステートメントを使用した結果

```
x=5 y=0.2
```

例 7: ユーザー定義関数での Graph Template Language (GTL)の使用

要素: PROC FCMP 関数
- OSCILLATE
- OSCILLATEBOUND

その他のプロシジャ
- PROC TEMPLATE
- PROC SGRENDER

詳細

次の例に、GTL EVAL 関数での関数の使用方法を示します。ここでは新しい曲線タイプ(oscillate と oscillateBound)を定義する関数の定義方法を示します。これらの関数は、GTL EVAL 関数で seriesplot と bandplot で表される新しい列の計算に使用できます。

プログラム

```
proc fcmp outlib=sasuser.funcs.curves;
```

```
   function oscillate(x,amplitude,frequency);
         if amplitude le 0 then amp=1; else amp=amplitude;
            if frequency le 0 then freq=1; else freq=frequency;
         y=sin(freq*x)*constant("e")**(-amp*x);
         return (y);
  endsub;

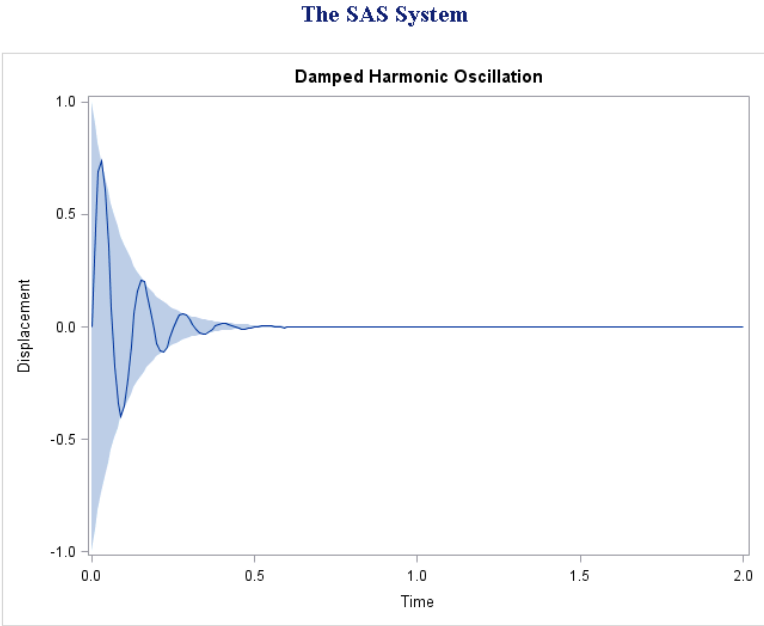
 function oscillateBound(x,amplitude);
       if amplitude le 0 then amp=1; else amp=amplitude;
       y=constant("e")**(-amp*x);
       return(y);
  endsub;
 run;


 options cmplib=sasuser.funcs;

data range;
   do time=0 to 2 by .01;
   output;
   end;
run;

proc template ;
   define statgraph damping;
   dynamic X AMP FREQ;
   begingraph;
       entrytitle "Damped Harmonic Oscillation";
       layout overlay / yaxisopts=(label="Displacement");
          if (exists(X) and exists(AMP) and exists(FREQ))
                bandplot x=X  limitlower=eval(-oscillateBound(X,AMP))
                limitupper=eval(oscillateBound(X,AMP));
                seriesplot x=X y=eval(oscillate(X,AMP,FREQ));
          endif;
        endlayout;
   endgraph;
   end;
   run;

 proc sgrender data=range template=damping;
      dynamic x="Time" amp=10 freq=50 ;
 run;
```

プログラムの説明

OSCILLATE 関数を作成します。

```
proc fcmp outlib=sasuser.funcs.curves;
   function oscillate(x,amplitude,frequency);
         if amplitude le 0 then amp=1; else amp=amplitude;
            if frequency le 0 then freq=1; else freq=frequency;
         y=sin(freq*x)*constant("e")**(-amp*x);
         return (y);
  endsub;
```

OSCILLATEBOUND 関数を作成します。

```
 function oscillateBound(x,amplitude);
```

```
      if amplitude le 0 then amp=1; else amp=amplitude;
      y=constant("e")**(-amp*x);
      return(y);
 endsub;
run;
```

PROC SGRENDER によって使用される Range と呼ばれるデータセットを作成します。

```
options cmplib=sasuser.funcs;

data range;
   do time=0 to 2 by .01;
   output;
   end;
run;
```

TEMPLATE プロシジャを使用して、SAS 出力の表示をカスタマイズできます。

```
proc template ;
   define statgraph damping;
   dynamic X AMP FREQ;
   begingraph;
       entrytitle "Damped Harmonic Oscillation";
       layout overlay / yaxisopts=(label="Displacement");
          if (exists(X) and exists(AMP) and exists(FREQ))
                bandplot x=X  limitlower=eval(-oscillateBound(X,AMP))
                limitupper=eval(oscillateBound(X,AMP));
                seriesplot x=X y=eval(oscillate(X,AMP,FREQ));
          endif;
        endlayout;
   endgraph;
   end;
   run;
```

SGRENDER プロシジャを使用して、入力変数を含むデータセットを識別し、出力用の statgraph テンプレートを割り当てます。

```
proc sgrender data=range template=damping;
     dynamic x="Time" amp=10 freq=50 ;
run;
```

出力:関数による GTL の使用

アウトプット 22.6 *ユーザー定義関数で GTL を使用した結果*



例 8: データセットの各行を標準化する

要素: PROC FCMP 関数
RUN_MACRO
RUN_SASFILE
READ_ARRAY
WRITE_ARRAY

詳細

この例では、データセットの各行の標準化の方法を示します。

プログラム

```
data numbers;
   drop i j;
   array a[5];
   do j=1 to 5;
   do i=1 to 5;
         a[i] = ranuni(12345) * (i+123.234);
      end;
      output;
      end;
   run;

%macro standardize;
%let dsname=%sysfunc(dequote(&dsname));
%let colname=%sysfunc(dequote(&colname));
proc standard data=&dsname mean=&MEAN std=&STD out=_out;
```

```
   var &colname;
run;
data &dsname;
   set_out;
run;
%mend standardize;

proc fcmp outlib=sasuser.ds.functions;
   subroutine standardize(x[*], mean, std);
      outargs x;

      rc=write_array('work._TMP_', x, 'x1');
      dsname='work._TMP_';
         colname='x1';
         rc=run_macro('standardize', dsname, colname, mean, std);
      array x2[1]_temporary_;
         rc=read_array('work._TMP_', x2);
      if dim(x2)=dim(x) then do;
         do i=1 to dim(x);
            x[i]=x2[i];
          end;
         end;
   endsub;
run;

options cmplib=(sasuser.ds);
data numbers2;
   set numbers;
   array a[5];
   array t[5]_temporary_;
   do i=1 to 5;
      t[i]=a[i];
   end;
   call standardize(t, 0, 1);
   do i=1 to 5;
      a[i]=t[i];
   end;
   output;
run;

data numbers;
       drop i j;
       array a[5];
       do j=1 to 5;
          do i=1 to 5;
             a[i]=ranuni(12345) * (i+123.234);
         end;
          output;
       end;
    run;

proc print data=work.numbers;
run;
```

プログラムの説明

5 行の乱数を含むデータセットを作成します。

```
data numbers;
   drop i j;
   array a[5];
   do j=1 to 5;
   do i=1 to 5;
         a[i] = ranuni(12345) * (i+123.234);
      end;
      output;
      end;
   run;
```

データセットを MEAN および STD の指定値で標準化するためのマクロを作成します。

```
%macro standardize;
%let dsname=%sysfunc(dequote(&dsname));
%let colname=%sysfunc(dequote(&colname));
proc standard data=&dsname mean=&MEAN std=&STD out=_out;
   var &colname;
run;
data &dsname;
   set_out;
run;
%mend standardize;
```

FCMP 関数を使用して、WRITE_ARRAY を呼び出します。これにより、データがデータセットに書き込まれます。RUN_MACRO を呼び出して、データセットのデータを標準化します。WRITE_ARRAY を呼び出して、データをデータセットに書き込みます。READ_ARRAY を呼び出して、標準化されたデータを配列に読み込みます。

```
proc fcmp outlib=sasuser.ds.functions;
   subroutine standardize(x[*], mean, std);
      outargs x;

      rc=write_array('work._TMP_', x, 'x1');
      dsname='work._TMP_';
         colname='x1';
         rc=run_macro('standardize', dsname, colname, mean, std);
      array x2[1]_temporary_;
         rc=read_array('work._TMP_', x2);
      if dim(x2)=dim(x) then do;
         do i=1 to dim(x);
            x[i]=x2[i];
          end;
         end;
   endsub;
run;
```

DATA ステップの行ごとに関数を実行します。

```
options cmplib=(sasuser.ds);
data numbers2;
   set numbers;
   array a[5];
   array t[5]_temporary_;
   do i=1 to 5;
```

```
      t[i]=a[i];
   end;
   call standardize(t, 0, 1);
   do i=1 to 5;
      a[i]=t[i];
   end;
   output;
run;

data numbers;
       drop i j;
       array a[5];
       do j=1 to 5;
          do i=1 to 5;
             a[i]=ranuni(12345) * (i+123.234);
         end;
          output;
       end;
    run;
```

出力を書き込みます。

```
proc print data=work.numbers;
run;
```

出力:データセットの行の標準化

アウトプット 22.7 *データセットの各行を標準化した結果*

The SAS System

Obs	a1	a2	a3	a4	a5
1	45.088	93.3237	104.908	35.152	23.5725
2	90.552	9.7548	92.696	89.987	97.9810
3	60.596	22.7409	19.284	50.079	58.9264
4	106.778	49.1589	22.885	20.641	30.1756
5	34.812	71.3746	44.248	101.808	79.3731

参考文献

SAS Institute Inc. 2013. *SAS Component Objects:Reference*. Cary, NC: SAS Institute Inc.

Henrick, A., D. Erdman, and S. Christian. 2013. “Hashing in PROC FCMP to Enhance Your Productivity.” *Proceedings of the SAS Global Forum 2013 Conference*, Cary, NC.SAS Institute Inc., 1–15. Available at http://support.sas.com/resources/papers/proceedings13/129-2013.pdf.

23 章

FCMP 特殊関数と CALL ルーチン

特殊関数と CALL ルーチンの概要

FCMP プロシジャは、特殊用途関数の小規模なセットを提供します。これらの関数はユーザー定義 FCMP 関数から呼び出すことができますが、DATA ステップから直接呼

び出すことはできません。これらの関数を DATA ステップで使用するには、別のユーザー定義 FCMP 関数内で特殊関数をラップする必要があります。

注: 特殊関数を直接プロシジャで呼び出すことができますが、DATA ステップで呼び出すことはできません。

カテゴリ別の関数と CALL ルーチン

配列の読み込みと、データセットへの配列の書き込み

PROC FCMP では、配列を読み込むための READ_ARRAY 関数、配列をデータセットに書き込むための WRITE_ARRAY 関数を提供しています。この機能により、PROC FCMP 配列データを SAS プログラム、マクロ、プロシジャによって処理できるようになります。

行列操作の CALL ルーチン

FCMP プロシジャでは、宣言された配列での単一の行列操作を実行するための多くの CALL ルーチンを提供しています。これらの CALL ルーチンは、自動的に FCMP プロシジャによって提供されます。ZEROMATRIX、FILLMATRIX、IDENTITY を除く、次にリストされている CALL ルーチンでは、欠損値を含む行列または配列をサポートしていません。

C Helper 関数と CALL ルーチン

複数の Helper 関数には、PROC FCMP での C 言語構築を処理するパッケージが提供されています。ほとんどの C 言語構築は、構築が参照可能になる前、または PROC FCMP によって使用可能になる前に PROC PROTO によって作成されるパッケージで定義する必要があります。ISNULL 関数、SETNULL ルーチン、および STRUCTINDEX CALL ルーチンが SAS 言語を拡張し、SAS 言語に合わない C 言語構築を処理するために追加されました。

関数から SAS コードを呼び出す関数

SAS コードを関数内から呼び出せる 2 つの関数が使用できます。RUN_MACRO 関数は、事前定義された SAS マクロを実行します。RUN_SASFILE 関数は、指定するファイル参照名から SAS コードを実行します。

特殊な目的のある関数

FCMP プロシジャでは、2 つの特殊な目的のある関数、INVCDF と LIMMOMENT が提供されます。INVCDF 関数は、累積分布関数(CDF)を定義した分布から分位点を計算します。LIMMOMENT 関数は、累積分布関数(CDF)を定義した分布の制限されたモーメントを計算します。

カテゴリ別の関数と CALL ルーチン

カテゴリ	言語要素	説明
C Helper	CALL SETNULL ルーチン (p. 756)	構造のポインタ要素を Null に設定します。
	CALL STRUCTINDEX ルーチン (p. 757)	構造の配列の各構造要素にアクセスできます。
	ISNULL 関数 (p. 763)	構造のポインタ要素が Null かどうかを決定します。
暗黙値の計算	SOLVE 関数 (p. 774)	Gauss-Newton 法を使用して、関数の暗黙値を計算します。
関数内からの SAS コードの呼び出し	RUN_MACRO 関数 (p. 769)	事前定義された SAS マクロを実行します。
	RUN_SASFILE 関数 (p. 773)	指定するファイル参照名で SAS コードを実行します。
行列操作	CALL ADDMATRIX ルーチン (p. 746)	2 つの行列、または 1 つの行列およびスカラーの elementwise 加算を実行します。
	CALL CHOL ルーチン (p. 747)	指定対称行列に対する Cholesky 分解を計算します。
	CALL DET ルーチン (p. 748)	正方である必要がある指定行列の行列式を計算します。
	CALL ELEMMULT ルーチン (p. 750)	2 つの行列の elementwise 乗算を実行します。
	CALL EXPMATRIX ルーチン (p. 751)	行列 etA を返します(入力行列 A、乗数 t の場合)。
	CALL FILLMATRIX ルーチン (p. 752)	入力行列のすべての要素値を指定値と置き換えます。
	CALL IDENTITY ルーチン (p. 753)	入力行列を単位行列に変換します。
	CALL INV ルーチン (p. 754)	正則な正方行列でなければならない、提供されている入力行列の逆行列である行列を計算します。
	CALL MULT ルーチン (p. 754)	2 つの入力行列の乗法積を計算します。
	CALL POWER ルーチン (p. 755)	正方行列を指定されているスカラー値に累乗します。
	CALL SUBTRACTMATRIX ルーチン (p. 758)	2 つの行列、または 1 つの行列およびスカラーの element-wide 減算を実行します。
	CALL TRANSPOSE ルーチン (p. 759)	行列の転置を返します。

カテゴリ	言語要素	説明
	CALL ZEROMATRIX ルーチン (p. 760)	数値入力行列のすべての要素値を 0 と置き換えます。
特殊な目的のある関数	INVCDF 関数 (p. 760)	累積分布関数(CDF)を定義した分布の分位点を計算します。
	LIMMOMENT 関数 (p. 765)	累積分布関数(CDF)を定義した分布の制限したモーメントを計算します。
配列	CALL DYNAMIC_ARRAY ルーチン (p. 749)	関数内で宣言される配列のサイズを効率的に変更できます。
	READ_ARRAY 関数 (p. 768)	データを SAS データセットから PROC FCMP 配列変数に読み込みます。
	WRITE_ARRAY (p. 778)	データを PROC FCMP 配列変数から SAS プログラム、マクロ、プロシジャによって使用可能なデータセットに書き込みます。

ディクショナリ

CALL ADDMATRIX ルーチン

2 つの行列、または 1 つの行列およびスカラーの elementwise 加算を実行します。

カテゴリ: 行列操作

要件 すべての入力行列と出力行列には、同じ次元が含まれている必要があります。

構文

CALL ADDMATRIX(*X*, *Y*, *Z*);

必須引数

X
次元 *m* x *n* (つまり X[*m*, *n*])を含む入力行列、またはスカラーを指定します。

Y
次元 *m* x *n* (つまり Y[*m*, *n*])を含む入力行列、またはスカラーを指定します。

Z
次のような次元 *m* x *n* (つまり Z[*m*, *n*])を含む出力行列を指定します。次のようになります。

$Z = X + Y$

例

次の例では ADDMATRIX CALL ルーチンを使用しています。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
   array mat2[3,2] (0.2, 0.38, -0.12, 0.98, 2, 5.2);
   array result[3,2];
   call addmatrix(mat1, mat2, result);
   call addmatrix(2, mat1, result);
   put result=;
quit;
```

アウトプット 23.1 *ADDMATRIX CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result[1, 1]=2.3 result[1, 2]=1.22
result[2, 1]=1.18 result[2, 2]=2.54 result[3, 1]=3.74 result[3, 2]=3.2
```

CALL CHOL ルーチン

指定対称行列に対する Cholesky 分解を計算します。

カテゴリ: 行列操作

別名: CHOLESKY_DECOMP

要件 入力行列と出力行列は正方で、同じ次元が含まれている必要があります。X は対称正定値で、Y は下三角行列である必要があります。

構文

CALL CHOL(*X*, *Y* <, *validate*>);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む対称正定値入力行列を指定します。

Y
次元 *m* x *m* (つまり、*m*, *m*])を含む出力行列を指定します。次のように、この変数には Cholesky 分解が含まれています。

$$Z = YY^*$$

Y は正の対角エントリを含む下三角行列で、Y*は Y の共役転置を示します。

注: X が対称正定値でない場合、Y には欠損値が入力されます。

オプション引数

validate
エラーチェックを行わないことによって処理速度を上げることができる任意の引数を指定します。引数は、次の値を受け取ることができます。

0 行列 X の対称性がチェックされます。これは、*validate* 引数が省略された場合のデフォルトです。

1 行列は、対称とみなされます。

例

次の例では、CHOL CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array xx[3,3] 2 2 3 2 4 2 3 2 6;
   array yy[3,3];
   call chol(xx, yy, 0);
   do i=1 to 3;
      put yy[i, 1] yy[i, 2] yy[i, 3];
   end;
run;
```

アウトプット 23.2 *PROC FCMP ルーチンおよび CHOL CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure 1.4142135624 0 0 1.4142135624 1.4142135624
0 2.1213203436 -0.707106781 1
```

CALL DET ルーチン

正方である必要がある指定行列の行列式を計算します。

カテゴリ: 行列操作

要件 入力行列 X は、正方にしてください。

構文

CALL DET(*X*, *a*);

必須引数

X
次元 *m* x *n* (つまり、X[*m*, *n*])を含む入力行列を指定します。

a
次のように、返された特定値を指定します。

$a = |X|$

詳細

行列式、固有値の積は、単一の数値です。行列の行列式がゼロの場合、行列は特異です(つまり、逆行列は含まれていません)。メソッドは LU 分解を実行し、対角線の積を収集します(Forsythe, Malcolm, and Moler 1967)。詳細については、*SAS/IML User's Guide* を参照してください。

例

次の例では、DET CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,3] (.03, -0.78, -0.82, 0.54, 1.74,
                    1.2, -1.3, 0.25, 1.49);
   call det(mat1, result);
   put result=;
quit;
```

アウトプット 23.3 *DET CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result=-0.052374
```

CALL DYNAMIC_ARRAY ルーチン

関数内で宣言される配列のサイズを効率的に変更できます。

カテゴリ: 配列

構文

CALL DYNAMIC_ARRAY(*array–name*, *new-dimension1–size* <, *new-dimension2–size*, ...>);

必須引数

array-name
テンポラリ配列の名前を指定します。

new-dimension-size
テンポラリ配列に新しいサイズを指定します。

詳細

関数と CALL ルーチンで宣言された配列は、/NOSYMBOLS オプションで宣言された配列と同様にサイズ変更可能です。その他の配列はサイズ変更できません。

DYNAMIC_ARRAY CALL ルーチンは配列のサイズを、提供するターゲットの次元に合わせて動的に変更しようとします。つまり、配列は動的である必要があります。配列は関数またはサブルーチンのいずれか、または/NOSYMBOLS オプションで宣言する必要があります。

DYNAMIC_ARRAY CALL ルーチンにはサイズ変更する配列と、配列の次元ごとに新しいサイズが渡されます。ALLPERMK ルーチンでは、並べ替えている要素数のサイズであるスクラッチ配列が必要になります。関数の作成時はこの値はパラメータ *n* として渡されるため、不明です。動的配列で、ルーチンは必要なメモリ量を割り当てることができます。すべての可能なケースの処理に十分な大きさの配列を作成する必要はありません。

動的配列の使用時は、サポートは PROC FCMP ルーチンに制限されます。配列のサイズが変更されると、サイズ変更された配列はサイズ変更を行ったルーチン内でのみ

使用できます。DATA ステップ配列をサイズ変更することも、PROC FCMP 動的配列を DATA ステップに返すこともできません。

例

次の例では、TEMP というテンポラリ配列を作成します。配列領域のサイズは、関数に渡されるパラメータによって異なります。

```
proc fcmp;
   function avedev_wacky(data[*]);

   length=dim(data);
   array temp[1] /nosymbols;
   call dynamic_array(temp, length);

   mean=0;
   do i=1 to length;
      mean += data[i];
      if i>1 then temp[i]=data[i-1];
      else temp[i]=0;
   end;
   mean=mean/length;

   avedev=0;
   do i=1 to length;
      avedev += abs((data[i])-temp[i] /2-mean);
   end;
   avedev=avedev/length;

   return(avedev);
endsub;

array data[10];

do i = 1 to 10;

   data[i] = i;
end;

avedev = avedev_wacky(data);
run;
```

CALL ELEMMULT ルーチン

2 つの行列の elementwise 乗算を実行します。

カテゴリ: 行列操作

要件 すべての入力行列と出力行列には、同じ次元が含まれている必要があります。

構文

CALL ELEMMULT(*X*, *Y*, *Z*);

必須引数

X
次元 *m* x *n* (つまり、X[*m*, *n*])を含む入力行列を指定します。

Y
次元 *m* x *n* (つまり、Y[*m*, *n*])を含む入力行列を指定します。

Z
次元 *m* x *n* (つまり、Z[*m*, *n*])を含む出力行列を指定します。

例

次の例では、ELEMMULT CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
   array mat2[3,2] (0.2, 0.38, -0.12, 0.98, 2, 5.2);
   array result[3,2];
   call elemmult(mat1, mat2, result);
   call elemmult(2.5, mat1, result);
   put result=;
 quit;
```

アウトプット 23.4 *ELEMMULT CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result[1, 1]=0.75 result[1, 2]=-1.95
result[2, 1]=-2.05 result[2, 2]=1.35 result[3, 1]=4.35 result[3, 2]=3
```

CALL EXPMATRIX ルーチン

行列 e^{tA} を返します(入力行列 *A*、乗数 *t* の場合)。

カテゴリ: 行列操作

要件 入力行列と出力行列は正方で、同じ次元が含まれている必要があります。*t* には、スカラー値が可能です。

構文

CALL EXPMATRIX(*X*, *t*, *Y*);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む入力行列を指定します。

t
倍精度のスカラー値を指定します。

Y
次のように、次元 *m* x *m* (つまり、Y[*m*, *m*])を含む出力行列を指定します。

$Y = \varepsilon^{tX}$

詳細

EXPMATRIX CALL では、Padé 概算アルゴリズム(Golub and van Loan (1989), p. 558 に記載)を使用します。このモジュールは、行列の各エントリを累乗しないことに注意してください。詳細については、*SAS/IML User's Guide* の EXPMATRIX ドキュメントを参照してください。

例

次の例では、EXPMATRIX CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,3] (0.3, -0.78, -0.82, 0.54, 1.74,
                    1.2, -1.3, 0.25, 1.49);
   array result[3,3];
   call expmatrix(mat1, 3, result);
   put result=;
quit;
```

アウトプット 23.5 *EXPMATRIX CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result[1, 1]=365.58043585 result[1,
2]=-589.6358476 result[1, 3]=-897.1034008 result[2, 1]=-507.0874798 result[2,
2]=838.64570481 result[2, 3]=1267.3598426 result[3, 1]=-551.588816 result[3,
2]=858.97629382 result[3, 3]=1324.8187125
```

CALL FILLMATRIX ルーチン

入力行列のすべての要素値を指定値と置き換えます。

カテゴリ: 行列操作

注: FILLMATRIX CALL ルーチンを多次元数値配列と使用できます。

構文

CALL FILLMATRIX(*X*, *Y*);

必須引数

X
入力数値行列を指定します。

Y
行列に入力する数値を指定します。

例

次の例では、FILLMATRIX CALL ルーチンを使用します。

```
options pageno=1 nodate ls=80 ps=64;
```

```
proc fcmp;
   array mat1[3, 2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
   call fillmatrix(mat1, 99);
   put mat1=;
quit;
```

アウトプット 23.6 *FILLMATRIX CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure mat1[1, 1]=99 mat1[1, 2]=99 mat1[2, 1]=99
mat1[2, 2]=99 mat1[3, 1]=99 mat1[3, 2]=99
```

CALL IDENTITY ルーチン

入力行列を単位行列に変換します。

カテゴリ: 行列操作

要件 入力行列は正方にしてください。

注: 行列の対角線要素値が 1 に設定され、その他の値が 0 に設定されます。

構文

CALL IDENTITY(*X*);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む入力行列を指定します。

例

次の例では、IDENTITY CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,3] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2,
                    -1.3, 0.25, 1.49);
   call identity(mat1);
   put mat1=;
quit;
```

アウトプット 23.7 *IDENTITY CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure mat1[1, 1]=1 mat1[1, 2]=0 mat1[1, 3]=0
mat1[2, 1]=0 mat1[2, 2]=1 mat1[2, 3]=0 mat1[3, 1]=0 mat1[3, 2]=0 mat1[3, 3]=1
```

CALL INV ルーチン

正則な正方行列でなければならない、提供されている入力行列の逆行列である行列を計算します。

カテゴリ: 行列操作

要件 入力行列と出力行列は正方で、同じ次元が含まれている必要があります。

構文

CALL INV(*X*, *Y*);

必須引数

X

次元 *m* x *m* (つまり、X[*m*, *m*])を含む入力行列を指定します。

Y

次のように、次元 *m* x *m* (つまり、Y[*m*, *m*])を含む出力行列を指定します。

$$Y[m,m] = X'[m,m]$$

'は逆行列を示し、

$$X \times Y = Y \times X = I$$

I は単位行列です。

例

次の例では、INV CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,3] (0.3, -0.78, -0.82, 0.54, 1.74,
                    1.2, -1.3, 0.25, 1.49);
   array result[3,3];
   call inv(mat1, result);
   put result=;
quit;
```

アウトプット 23.8 *Results from the INV CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result[1, 1]=4.0460407887 result[1,
2]=1.6892917399 result[1, 3]=0.8661767509 result[2, 1]=-4.173108283 result[2,
2]=-1.092427483 result[2, 3]=-1.416802558 result[3, 1]=4.230288655 result[3,
2]=1.6571719011 result[3, 3]=1.6645841716
```

CALL MULT ルーチン

2 つの入力行列の乗法積を計算します。

カテゴリ: 行列操作

要件 最初の入力行列の列数が 2 番目の行列の行数と同じである必要があります。

構文

CALL MULT(*X*, *Y*, *Z*);

必須引数

X
次元 *m* x *n* (つまり、X[*m*, *n*])を含む入力行列を指定します。

Y
次元 *n* x *p* (つまり、Y[*n*, *p*])を含む入力行列を指定します。

Z
次のように、次元 *m* x *p* (つまり、Z[*m*, *p*])を含む出力行列を指定します。

$$Z[m,p] = X[m,n] \times Y[n,p]$$

例

次の例では、MULT CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[2,3] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
   array mat2[3,2] (1, 0, 0, 1, 1, 0);
   array result[2,2];
   call mult(mat1, mat2, result);
   put result=;
quit;
```

アウトプット 23.9 *MULT CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result[1, 1]=-0.52 result[1, 2]=-0.78
result[2, 1]=1.74 result[2, 2]=1.74
```

CALL POWER ルーチン

正方行列を指定されているスカラー値に累乗します。

カテゴリ: 行列操作

制限事項: 行列乗算ルーチンの POWER CALL ルーチンの内部使用により数値精度の問題が発生することがあるため、大きなスカラー値は避ける必要があります。

要件 入力行列と出力行列は正方で、同じ次元が含まれている必要があります。

構文

CALL POWER(*X*, *a*, *Y*);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む入力行列を指定します。

a
整数のスカラー値(累乗)を指定します。

Y
次のように、次元 *m* x *m* (つまり、Y[*m*, *m*])を含む出力行列を指定します。

$Y = X^a$

詳細

スカラーは整数でない場合、整数に切り捨てられます。スカラーは 0 未満の場合、0 に変更されます。詳細については、*SAS/IML User's Guide* を参照してください。

例

次の例では、POWER CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,3] (0.3, -0.78, -0.82, 0.54, 1.74,
                    1.2, -1.3, 0.25, 1.49);
   array result[3,3];
   call power(mat1, 3, result);
   put result=;
quit;
```

アウトプット 23.10 *POWER CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result[1, 1]=2.375432 result[1,
2]=-4.299482 result[1, 3]=-6.339638 result[2, 1]=-3.031224 result[2,
2]=6.272988 result[2, 3]=8.979036 result[3, 1]=-4.33592 result[3, 2]=5.775695
result[3, 3]=9.326529
```

CALL SETNULL ルーチン

構造のポインタ要素を Null に設定します。

カテゴリ: C Helper

構文

CALL SETNULL(*pointer-element*);

必須引数

pointer-element
構造へのポインタです。

例

次の例では、PROC PROTO を使用して、“例 1: リンクされたリストの生成” (764 ページ) で説明されているものと同じ LINKLIST 構造が定義されると仮定しています。CALL SETNULL ルーチンを使用して、NEXT 要素を null に設定できます。

```
struct linklist list;
call setnull(list.next);
```

CALL STRUCTINDEX ルーチン

構造の配列の各構造要素にアクセスできます。

カテゴリ: C Helper

構文

CALL STRUCTINDEX(*structure-array*, *index*, *structure-element*);

必須引数

structure-array
配列を指定します。

index
1 から始まるインデックスです。ほとんどの SAS 配列で使用されます。

structure-element
配列の要素を示します。

例

この例の最初の部分では、次の構造と関数が PROC PROTO を使用して定義されます。

```
proc proto package=sasuser.mylib.str2;
struct point{
   short s;
   int i;
   long l;
   double d;
};

struct point_array {
   int            length;
   struct point p[2];
   char           name[32];
};
run;
```

この例の 2 番目の部分では、PROC FCMP コードセグメントに、STRUCTINDEX CALL ルーチンを使用して POINT_ARRAY 構造の P という配列の各ポイント構造要素を取得および設定するための方法が示されます。

```
options pageno=1 nodate ls=80 ps=64;

proc fcmp libname=sasuser.mylib;
```

```
   struct point_array pntarray;
   struct point pnt;

   pntarray.length=2;
   pntarray.name="My funny structure";

   /* Get each element using the STRUCTINDEX CALL routine and set values. */
do i=1 to 2;
   call structindex(pntarray.p, i, pnt);
   put "Before setting the" i "element: " pnt=;
   pnt.s=1;
   pnt.i=2;
   pnt.l=3;
   pnt.d=4.5;
   put "After setting the" i "element: " pnt=;
end;
run;
```

アウトプット 23.11 *配列のポイント構造要素の設定の結果*

```
The SAS System 1 The FCMP Procedure Before setting the 1 element:pnt {s=0, i=0,
l=0, d=0} After setting the 1 element:pnt {s=1, i=2, l=3, d=4.5} Before setting
the 2 element:pnt {s=0, i=0, l=0, d=0} After setting the 2 element:pnt {s=1,
i=2, l=3, d=4.5}
```

CALL SUBTRACTMATRIX ルーチン

2 つの行列、または 1 つの行列およびスカラーの element-wide 減算を実行します。

カテゴリ: 行列操作

要件 すべての入力行列と出力行列には、同じ次元が含まれている必要があります。

構文

CALL SUBTRACTMATRIX(*X*, *Y*, *Z*);

必須引数

X
次元 *m* x *n* (つまり X[*m*, *n*])を含む入力行列、またはスカラーを指定します。

Y
次元 *m* x *n* (つまり Y[*m*, *n*])を含む入力行列、またはスカラーを指定します。

Z
次のような次元 *m* x *n* (つまり Z[*m*, *n*])を含む出力行列を指定します。次のようになります。

$Z = X - Y$

例

次の例では、SUBTRACTMATRIX CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
   array mat2[3,2] (0.2, 0.38, -0.12, 0.98, 2, 5.2);
   array result[3,2];
   call subtractmatrix(mat1, mat2, result);
   call subtractmatrix(2, mat1, result);
   put result=;
 quit;
```

アウトプット 23.12 *SUBTRACTMATRIX CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result[1, 1]=1.7 result[1, 2]=2.78
result[2, 1]=2.82 result[2, 2]=1.46 result[3, 1]=0.26 result[3, 2]=0.8
```

CALL TRANSPOSE ルーチン

行列の転置を返します。

カテゴリ: 行列操作

構文

CALL TRANSPOSE(*X*, *Y*);

必須引数

X
次元 *m* x *n* (つまり、X[*m*, *n*])を含む入力行列を指定します。

Y
次元 *n* x *m* (つまり、Y[*n*, *m*])を含む出力行列を指定します。

詳細

$$Y = X'$$

入力行列の行数が出力行列の列数と同じで、出力行列の行数が入力行列の列数と同じである必要があります。

例

次の例では、TRANSPOSE CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
   array result[2,3];
   call transpose(mat1, result);
   put result=;
quit;
```

アウトプット 23.13 *TRANSPOSE CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure result[1, 1]=0.3 result[1, 2]=-0.82
result[1, 3]=1.74 result[2, 1]=-0.78 result[2, 2]=0.54 result[2, 3]=1.2
```

CALL ZEROMATRIX ルーチン

数値入力行列のすべての要素値を 0 と置き換えます。

カテゴリ: 行列操作

注: ZEROMATRIX CALL ルーチンを多次元数値配列と使用できます。

構文

CALL ZEROMATRIX(*X*);

必須引数

X
数値入力行列を指定します。

例

次の例では、ZEROMATRIX CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
   array mat1[3,2]  (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
   call zeromatrix(mat1);
   put mat1=;
quit;
```

アウトプット 23.14 *ZEROMATRIX CALL ルーチンの結果*

```
The SAS System 1 The FCMP Procedure mat1[1, 1]=0 mat1[1, 2]=0 mat1[2, 1]=0
mat1[2, 2]=0 mat1[3, 1]=0 mat1[3, 2]=0
```

INVCDF 関数

累積分布関数(CDF)を定義した分布の分位点を計算します。

カテゴリ: 特殊な目的のある関数

注: INVCDF は、FCMP プロシジャによって参考のために自動的に提供される特殊な目的のある関数です。

確率 *p* を CDF を含む分布(*F(x; <parameters>)*で示される)に対して指定する場合、INVCDF 関数は分位点 *q* (*F(q;<parameters>) = p* を満たす)を返します。つまり、$q=F^{-1}(p)$となります。

構文

quantile=**INVCDF**('*CDF-function-name*', *options-array*, *cumulative-probability*, *parameter-1 <, parameter-2, ...>*);

必須引数

quantile
INVCDF 関数から返される分位点を指定します。

'CDF-function-name'
CDF 関数の名前を指定します。*CDF-function-name* を引用符で囲みます。

要件 *CDF-function-name* は、FCMP プロシジャを使用して定義される関数にしてください。次のように、シグネチャが含まれている必要があります。

```
function <CDF-function-name> (x, parameter-1 <, parameter-2, …>);

endsub;
```

注 CDF が連続関数であることが推奨されます。CDF が不連続である場合、INVCDF 関数で分位点を計算できないことがあります。

options-array
INVCDF 関数と使用するオプションの配列を指定します。*Options-array* は、CDF の反転プロセスの制御とモニタに使用されます。*Options-array* には欠損値(.)が可能です。または次の順序の次の要素のうち 4 つまでを含めることができます。

initial-value
反転プロセスが開始する分位点に対し初期予測を指定します。これは、分位点に対する推定値(CDF の経験推定からなど)がある場合に有用です。

デフォルト 0.1

desired-accuracy
分位点の必要な相対的な正確性を指定します。範囲(0,0.1)の値を指定します。小さい値を指定すると、結果として分位点の推定はより正確になりますが、CDF の反転により時間がかかることがあります。

デフォルト 1.0e-8

domain-type
CDF 関数のドメインを指定します。欠損値または値 0 は、正のサポート、[0,∞)を示します。その他の値は実数直線全体にわたるサポート、(-∞,∞)を示します。

デフォルト 0

return-code
リターンステータスを指定します。*options-array* が次元 4 以上の場合、4 番目の要素にはリターンステータスが含まれています。*Return-code* の値には、次のうちいずれかが可能です。

<=0
成功を示します。負の場合、絶対値は分位点を計算するために評価された CDF 関数の回数です。絶対値が大きくなると、収束時間が長くなります。

1
分位点を計算できなかったことを示します。

cumulative-probability
分位点が必要となる累積確率値を指定します。

範囲 [0,1)

parameter
分位点が必要となる分布のパラメータを指定します。指定された CDF 関数によって要求されるものと同じパラメータ数を指定する必要があります。これらは、指定された CDF 関数によって要求されるものと同じ順序で表示する必要があります。

詳細

INVCDF 関数は、累積分布関数が *CDF-function-name* 引数によって指定される分布から、指定された累積確率に対する分位点を検索します。つまり、次の式が True の CDF 関数を反転します。

```
cumulative-probability = CDF-function-name(quantile,<parameters>)
```

εが累積確率 p に対する分位点に求められる正確性を示している場合、INVCDF は $|p - F(q)| < \varepsilon p$,のようにして分位点 q の計算を試行します。*F(x)*は、x で評価された CDF を示します。

反転プロセスをさまざまなオプションで制御できます。次に、オプション配列の例を示します。

```
array opts[4] initial epsilon support (1.5 1.0e-6 0);
```

これらの値は次の前の命令行を参照しています。

```
initial(initial-value)=1.5

epsilon(desired-accuracy)=1.0e-6

support(domain-type)=0
```

opts[4]をチェックして、関数のリターンステータスを検証できます。

比較

この関数を QUANTILE 関数の一般的な拡張とすることができます。特定の分布からのみ分位点を計算します。分布の CDF 関数がプログラムで定義可能な限り、INVCDF 関数を使用して連続分布からの分位点を計算できます。QUANTILE 関数とは異なり、この関数は DATA ステップで直接使用できません。FCMP 関数またはサブルーチンの定義内でのみ使用できます。ただし、これは制限ではありません。この関数を使用する FCMP 関数を DATA ステップから起動できるためです。次の例を参照してください。

例: 指数分布からの無作為抽出の生成

次の例に、INVCDF 関数を使用して DATA ステップでパラメトリックな分布から無作為抽出を生成する方法を示します。例では指数分布を説明に使用しますが、プログラムで CDF 関数が定義可能な分布まで拡張できます。次のステートメントでは、指数分布からの分布点の計算に INVCDF 関数と CDF 関数を使用する FCMP 関数 EXP_QUANTILE を定義します。

```
proc fcmp library=work.mycdf outlib=work.myquantile.functions;
   function exp_quantile(cdf, theta, rc);
      outargs rc;
      array opts[4] / nosym(0.1 1.0e-8 .);
      q=invcdf("exp_cdf", opts, cdf, theta);
      rc=opts[4];  /* return code */
```

```
      return(q);
   endsub;
quit;
```

前出のコードでは、次のように PROC FCMP ステップを使用して EXP_CDF 関数の定義を Work.Mycdf という FCMP ライブラリに保存したとみなします。

```
proc fcmp outlib=work.mycdf.functions;
   function exp_cdf(x, theta);
      return(1.0 - exp(-x/Theta));
   endsub;
quit;
```

EXP_QUANTILE 関数を DATA ステップから起動し、値が 50 のスケールパラメータ (theta)によって指数分布から無作為抽出を生成できるようになりました。DATA ステップを実行する前に、EXP_CDF 関数と EXP_QUANTILE 関数の場所を CMPLIB=オプションに対する適切な値で指定する必要があります。

```
options cmplib=(work.mycdf work.myquantile);

data exp_sample(keep=q);
   n=0;k=0;
   do k=1 to 500;
      if (n=100) then leave;
      rcode=.;
      q=exp_quantile(rand('UNIFORM'), 50, rcode);
      if (rcode <= 0) then do;
         n=n+1;
         output;
      end;
   end;
run;
```

ISNULL 関数

構造のポインタ要素が Null かどうかを決定します。

カテゴリ: C Helper

構文

numeric-variable = **ISNULL** (*pointer-element*);

必須引数

numeric-variable
数値を指定します。

pointer-element
別の変数のアドレスを含む変数を指定します。

例

例 1: リンクされたリストの生成

次の例で、LINKLIST 構造と GET_LIST 関数が PROC PROTO を使用して定義されます。GET_LIST 関数は、必要とされる数の要素を含むリンクされたリストを生成する外部 C ルーチンです。

```
struct linklist{
   double value;
   struct linklist * next;
};

struct linklist * get_list(int);
```

例 2: ISNULL C Helper 関数のループでの使用

次のコードセグメントに、ISNULL C Helper 関数が GET_LIST によって作成されるリンクされたリストにわたりループし、要素を書き出すことを示します。

```
proc proto package=sasuser.mylib.str2;
struct linklist{
   double value;
   struct linklist * next;
};

struct linklist * get_list(int);

externc get_list;
   struct linklist * get_list(int len){
      int i;
      struct linklist * list=0;
      list=(struct linklist*)
         malloc(len*sizeof(struct linklist));
      for (i=0;i<len-1;i++){
         list[i].value=i;
         list[i].next=&list[i+1];
      }
      list[i].value=i;
      list[i].next=0;
      return list;
   }
externcend;
run;

options pageno=1 nodate ls=80 ps=64;
proc fcmp libname=sasuser.mylib;
   struct linklist list;
   list=get_list(3);
   put list.value=;

   do while (^isnull(list.next));
      list=list.next;
      put list.value=;
   end;
run;
```

アウトプット 23.15 ISNULL C Helper 関数使用の結果

```
The SAS System 1 The FCMP Procedure list.value=0 list.value=1 list.value=2
```

LIMMOMENT 関数

累積分布関数(CDF)を定義した分布の制限したモーメントを計算します。

カテゴリ: 特殊な目的のある関数

注: LIMMOMENT は、FCMP プロシジャによって参考のために自動的に提供される特殊な目的のある関数です。

順序 *k* と上限 *u* を指定する場合、LIMMOMENT 関数は制限されたモーメントを $E[min(X,u)^k]$として計算します。この場合、*E* は、指定された CDF 関数によって定義される確率変数 X の分布を引き継いだ期待を示します。

構文

imom=**LIMMOMENT**('*CDF-function-name*', *options-array*, *order*, *limit*, *parameter-1 <, parameter-2, ...>*);

必須引数

imom
LIMMOMENT 関数から返される制限されたモーメントを指定します。

'CDF-function-name'
CDF 関数の名前を指定します。*CDF-function-name* を引用符で囲みます。

要件 *CDF-function-name* は、FCMP プロシジャを使用して定義される関数にしてください。次のように、シグネチャが含まれている必要があります。

```
function <CDF-function-name> (x, parameter-1 <, parameter-2, …>);

endsub;
```

注 CDF が連続関数であることが推奨されます。CDF が不連続である場合、LIMMOMENT 関数は制限されたモーメントを計算できないことがあります。

options-array
LIMMOMENT 関数と使用するオプションの配列を指定します。*Options-array* は、限度額の計算に使用される数値積分のプロセスの制御とモニタに使用されます。*Options-array* には欠損値(.)が可能です。または次の順序の次の要素のうち 4 つまでを含めることができます。

desired-accuracy
数値積分の必要な正確性を指定します。範囲(0,0.1)の値を指定します。小さい値を指定すると、結果としてモーメントの推定はより正確になりますが、desired-accuracy の計算にはより多くの時間がかかります。

デフォルト 1.0e-8

initial-step-size
数値積分プロセスによって最初に使用されるステップサイズを指定します。値が増えると、被積分関数が評価される回数の線が少なくなります。通常、デフォルト値 1 を使用すると、よい結果が生成されます。

デフォルト 1

maximum-iterations
必要な正確性を実現するため統合結果の絞り込みに使用される反復の最大数を指定します。この値が増えると、被積分関数が評価される回数の指数が増えます。

デフォルト 8

return-code
リターンステータスを指定します。*options-array* が次元 4 以上の場合、4 番目の要素にはリターンステータスが含まれています。*Return-code* の値には、次のうちいずれかが可能です。

<=0
成功を示します。負の場合、絶対値は制限されたモーメントを計算するために被積分関数が評価された回数です。絶対値が大きくなると、収束時間が長くなります。

1
制限されたモーメントを計算できなかったことを示します。

order
必要な制限されたモーメントの順序を指定します。

範囲 [1,10]

limit
必要な制限されたモーメントの計算に使用される上限を指定します。

要件 *limit* の値は 0 より大きいものとします。

parameter
制限されたモーメントが必要となる分布のパラメータを指定します。指定された CDF 関数によって要求されるものと同じパラメータ数を指定する必要があります。これらは、指定された CDF 関数によって要求されるものと同じ順序で表示する必要があります。

詳細

確率変数 X に確率密度関数 $f(x;\theta)$と累積分布関数 $F(x;\theta)$を含む確率分布が含まれるようにします。θ は分布のパラメータを示します。指定された上限 u に対し、この分布の k^{th} (番目)の順序の制限されたモーメントが次のように定義されます。

$$E[(X \wedge u)^k] = \int_0^u x^k f(x)dx + u^k \int_u^\infty f(x)dx = \int_0^u x^k f(x)dx + u^k(1 - F(u))$$

LIMMOMENT 関数は、次の代替式を使用します。

$$E[(X \wedge u)^k] = k \int_0^u x^{k-1}[1 - F(x)]dx$$

式には $F(x)$のみ必要なため、分布に対し CDF 関数のみ指定する必要があります。制限されたモーメントは、保険アプリケーションで保険範囲が特定の値で設定されている場合に支払われる最大金額の計算に使用されます。

数値積分プロセスをさまざまなオプションで制御できます。次に、オプション配列の例を示します。

```
array opts[4] epsilon initial maxiter (1.0e-5 1 6);
```

これらの値は前の命令行を参照します。

```
epsilon(desired-accuracy)=1.0e-5

initial(initial-step)=1

maxiter(maximum-iterations)=6
```

opts[4]をチェックして、関数のリターンステータスを検証できます。

例: 対数正規分布の制限されたモーメントの計算

この例では、LIMMOMENT 関数を使用して DATA ステップでパラメトリックな分布に対する制限されたモーメントを計算する方法を示します。例では対数正規分布を説明に使用しますが、プログラムで CDF 関数が定義可能な分布まで拡張できます。次のステートメントでは、対数正規分布からの制限されたモーメントの計算に LIMMOMENT 関数と CDF 関数を使用する FCMP 関数 LOGN_LIMMOMENT を定義します。

```
proc fcmp library=work.mycdf outlib=work.mylimmom.functions;
        function logn_limmoment(order, limit, mu, sigma, rc);
            outargs rc;
            array opts[4] / nosym (1.0e-8 . . .);

            m=limmoment("logn_cdf", opts, order, limit, mu, sigma);
            rc=opts[4]; /* return code */

            return(m);
        endsub;
    quit;
```

前出のコードでは、次のように PROC FCMP ステップを使用して LOGN_CDF 関数の定義を Work.Mycdf という FCMP ライブラリに保存したとみなします。

```
proc fcmp outlib=work.mycdf.functions;
        function logn_cdf(x, Mu, Sigma);
            if (x >= constant('MACEPS')) then do;
                z=(log(x) - Mu)/Sigma;
                return(CDF('NORMAL',z));
            end;
            return (0);
        endsub;
    quit;
```

次のように、LOGN_LIMMOMENT 関数を DATA ステップから起動できるようになりました。DATA ステップを実行する前に、LOGN_CDF 関数と LOGN_LIMMOMENT 関数の場所を CMPLIB=オプションに対し適切な値で指定する必要があります。

```
options cmplib=(work.mycdf work.mylimmom);

    data _null_;
        do order=1 to 3;
            rcode=.;
            m=logn_limmoment(order, 100, 5, 0.5, rcode);
            if (rcode > 0) then
                put "ERROR: Limited moment could not be computed.";
```

```
            else
                put 'Moment of order ' order ' with limit 100 = ' m;
        end;
    run;
```

READ_ARRAY 関数

データを SAS データセットから PROC FCMP 配列変数に読み込みます。

カテゴリ: 配列

構文

rc = **READ_ARRAY**(*data_set_name*, *array_variable* <, '*column_name_1*', '*column_name_2*' , ...>);

必須引数

rc
関数がデータセットを正常に読み込める場合は 0 です。

data_set_name
配列データの読み込み元のデータセットの名前を指定します。*data_set_name* は、読み込み元のデータセットのメンバ名(libname.memname)が含まれている文字リテラルまたは変数にしてください。

array_variable
データの読み込み先の PROC FCMP 配列変数を指定します。*array_variable* は必ず、ローカルテンポラリ配列変数にしてください。関数はデータセットのサイズにあわせてそのサイズを拡大、縮小する必要があるためです。

オプション引数

column_name
読み込まれるデータセットの特定の列に対し任意の名前を指定します。

column_name を指定した場合、引用符で囲まれたリテラル文字列にしてください。*column_name* は PROC FCMP 変数にはなりません。列名が指定されない場合、PROC FCMP はデータセットのすべての列を読み込みます。

詳細

配列とデータセット間の変換時に、配列は[row,column]のようにインデックス付けされます。

関数と CALL ルーチンで宣言された配列は、/NOSYMBOLS オプションで宣言された配列と同様にサイズ変更可能です。その他の配列はサイズ変更できません。

READ_ARRAY 関数は、入力データセットの次元に合わせて動的に配列をサイズ変更しようとします。つまり、配列は動的である必要があります。配列は関数または CALL ルーチンのいずれか、または/NOSYMBOLS オプションで宣言する必要があります。

例

この例では、SAS データセットを作成し、FCMP 配列変数に読み込みます。

```
options nodate pageno=1;
```

```
data account;
   input acct price cost;
   datalines;
1 2 3
4 5 6
;
run;

proc fcmp;
   array x[2,3] / nosymbols;
   rc=read_array('account',x);
   put x=;
run;

proc fcmp;
   array x[2,2] / nosymbols;
   rc=read_array('account', x, 'price', 'acct');
   put x=;
run;
```

アウトプット 23.16 *READ_ARRAY 関数の結果*

```
The SAS System 1 The FCMP Procedure x[1, 1]=1 x[1, 2]=2 x[1, 3]=3 x[2, 1]=4 x[2,
2]=5 x[2, 3]=6
```

```
The SAS System 2 The FCMP Procedure x[1, 1]=2 x[1, 2]=1 x[2, 1]=5 x[2, 2]=4
```

RUN_MACRO 関数

事前定義された SAS マクロを実行します。

カテゴリ: 関数内からの SAS コードの呼び出し

注: この関数の動作は、SAS での**%macro_name;**の実行に似ています。

構文

rc = **RUN_MACRO** (*'macro_name'* <, *variable_1*, *variable_2, ...*>);

必須引数

rc
関数がマクロをサブミットできる場合は 0 です。リターンコードは、マクロコールが試行されたことのみ示します。マクロが予想どおりに実行するかどうかを決定するためにマクロ自体が PROC FCMP 変数に対応する SAS マクロ変数の値を設定する必要があります。

macro_name
実行するマクロの名前を指定します。

要件 *Macro_name* は、引用符で囲まれた文字列か、実行するマクロを含む文字変数にしてください。

オプション引数

variable
同じ名前のマクロ変数によって設定される、任意の PROC FCMP 変数を指定します。これらの引数は、PROC FCMP 倍精度変数、または文字変数にしてください。

マクロの実行前に、SAS マクロ変数が PROC FCMP 変数と同じ名前と値で定義されます。マクロの実行後は、マクロ変数値は対応する PROC FCMP 変数にコピーされます。

例

例 1: PROC FCMP による事前定義されたマクロの実行

この例では、%TESTMACRO というマクロを作成してから、そのマクロを PROC FCMP 内で使用し、2 つのメンバを減算します。

```
      /* Create a macro called %TESTMACRO. */
%macro testmacro;
   %let p=%sysevalf(&a - &b);
%mend testmacro;

    /* Use %TESTMACRO within a function in PROC FCMP to subtract two numbers. */
proc fcmp outlib=sasuser.ds.functions;
   function subtract_macro(a, b);
      rc=run_macro('testmacro', a, b, p);
      if rc eq 0 then return(p);
      else return(.);
   endsub;
run;

   /* Make a call from the DATA step. */
option cmplib=(sasuser.ds);

data _null_;
   a=5.3;
   b=0.7;
   p=.;
   p=subtract_macro(a, b);
   put p=;
run;
```

ログ 23.1 *PROC FCMP での事前定義されたマクロ実行の結果*

```
p=4.6
```

例 2: RUN_MACRO 関数および DOSUBL 関数の結果の比較

この例では、%TESTMACRO を作成し、DOSUBL 関数が RUN_MACRO と同じ%TESTMACRO を呼び出す方法を比較しています。RUN_MACRO 起動時には、すべての変数が、設定できるように引数として渡されます。DOSUBL 起動時には、実行するコードにすべてのマクロ変数がインポートされ、完了時にエクスポートされます。マクロ変数&a および&b は%TESTMACRO に利用できるようになり、マクロ変数&p は DOSUBL 呼び出しに利用できるようになります。詳細については、“DOSUBL Function” (*SAS Functions and CALL Routines: Reference*)を参照してください。

```
    /* Create a macro called %TESTMACRO. */
%macro testmacro;
   %let p=%sysevalf(&a - &b);
%mend testmacro;

    /* Use %TESTMACRO within a function in PROC FCMP to subtract two numbers. */
proc fcmp outlib=sasuser.ds.functions;
   function subtract_macro(a, b);
      rc=run_macro('testmacro', a, b, p);
      if rc eq 0 then return(p);
      else return(.);
   endsub;
run;

   /* Make a call from the DATA step. */
option cmplib=(sasuser.ds);

data _null_;
   a=5.3;
   b=0.7;
   p=.;
   p=subtract_macro(a, b);
   put p= '(RUN_MACRO function and a DATA step)';
run;

%global a b p;
%put p=&p;
    /* The value should not yet be known. */
   %let a=5.3;
   %let b=0.7;
data _null_;
   rc=dosubl('%testmacro');
   run;
%put p=&p (DOSUBL function);
```

ログ 23.2 *RUN_MACRO 関数および DOSUBL 関数の結果*

```
p=4.6 (RUN_MACRO function and a DATA step) p=4.6 (DOSUBL function)
```

例 3: DATA ステップ内の DATA ステップの実行

この例には、別の DATA ステップ内から DATA ステップを実行する方法を示します。プログラムは次のセクションから構成されています。

- プログラムの最初のセクションでは、APPEND_DS というマクロを作成します。このマクロは、データセット間の書き込みと追加を行うことができます。
- プログラムの 2 番目のセクションでは、マクロを PROC FCMP の関数 writeDataset から呼び出します。
- プログラムの 3 番目のセクションでは、SALARIES データセットを作成し、変数 Department の値によってデータセットを 4 つの個別のデータセットに分割します。
- プログラムの 4 番目のセクションでは、結果を出力ウィンドウに書き込みます。

```
   /* Create a macro called APPEND_DS. */
%macro append_ds;
   /* Character values that are passed to RUN_MACRO are put           */
   /* into their corresponding macro variables inside of quotation    */
```

```
   /* marks. The quotation marks are part of the macro variable value. */
   /* The DEQUOTE function is called to remove the quotation marks.    */
   %let dsname=%sysfunc(dequote(&dsname));
   data &dsname
      %if %sysfunc(exist(&dsname)) %then %do;
         modify &dsname;
      %end;
      Name=&Name;
      WageCategory=&WageCategory;
      WageRate=&WageRate;
      output;
      stop;
   run;
%mend append_ds;

   /* Call the APPEND_DS macro from function writeDataset in PROC FCMP. */
proc fcmp outlib=sasuser.ds.functions;
   function writeDataset (DsName $, Name $, WageCategory $, WageRate);
      rc=run_macro('append_ds', dsname, DsName, Name, WageCategory, WageRate);
      return(rc);
   endsub;
run;

   /* Use the DATA step to separate the salaries data set into four separate */
   /* departmental data sets (NAD, DDG, PPD, and STD).                       */
data salaries;
   input Department $ Name $ WageCategory $ WageRate;
   datalines;
BAD Carol Salaried 20000
BAD Beth Salaried 5000
BAD Linda Salaried 7000
BAD Thomas Salaried 9000
BAD Lynne Hourly 230
DDG Jason Hourly 200
DDG Paul Salaried 4000
PPD Kevin Salaried 5500
PPD Amber Hourly 150
PPD Tina Salaried 13000
STD Helen Hourly 200
STD Jim Salaried 8000
;
run;

options cmplib=(sasuser.ds) pageno=1 nodate;
data _null_;
   set salaries;
   by Department;
   length dsName $ 64;
   retain dsName;
   if first.Department then do;
      dsName='work.' || trim(left(Department));
   end;
   rc=writeDataset(dsName, Name, WageCategory, wageRate);
run;

proc print data=work.BAD; run;
```

```
proc print data=work.DDG; run;
proc print data=work.PPD; run;
proc print data=work.STD; run;
```

アウトプット 23.17 *DATA ステップ内での DATA ステップの呼び出しの結果*

```
The SAS System 1 Wage Wage Obs Name Category Rate 1 Carol Salaried 20000 2 Beth
Salaried 5000 3 Linda Salaried 7000 4 Thomas Salaried 9000 5 Lynne Hourly 230
```

```
The SAS System 2 Wage Wage Obs Name Category Rate 1 Jason Hourly 200 2 Paul
Salaried 4000
```

```
The SAS System 3 Wage Wage Obs Name Category Rate 1 Kevin Salaried 5500 2 Amber
Hourly 150 3 Tina Salaried 13000
```

```
The SAS System 4 Wage Wage Obs Name Category Rate 1 Helen Hourly 200 2 Jim
Salaried 8000
```

RUN_SASFILE 関数

指定するファイル参照名で SAS コードを実行します。

カテゴリ: 関数内からの SAS コードの呼び出し

構文

rc = **RUN_SASFILE** (*'fileref_name'* <, *variable-1*, *variable-2*, ...>);

必須引数

rc
関数が SAS ファイルを処理するコードを実行する依頼をサブミットできる場合は 0 です。リターンコードは、呼び出しが試行されたことのみ示します。

fileref_name
SAS コードをポイントする SAS ファイル参照名の名前を指定します。

要件 *Fileref_name* は、引用符で囲まれた文字列か、SAS ファイル参照名を含む文字変数にしてください。

オプション引数

variable
同じ名前のマクロ変数によって設定される、任意の PROC FCMP 変数を指定します。これらの引数は、PROC FCMP 倍精度変数、または文字変数にしてください。

SAS ファイルを参照するコードを実行する前、SAS マクロ変数は PROC FCMP 変数と同じ名前と値で定義されます。実行後、これらのマクロ変数値は対応する PROC FCMP 変数にコピーされます。

例

次の例は、RUN_SASFILE が事前定義されているマクロではなく SAS ファイルを使用する点を除いて、RUN_MACRO の最初の例に似ています。この例では、**test.sas(a, b, c)** が現在のディレクトリに置かれているものとします。

```
      /* test.sas(a,b,c) */
data _null_;
   call symput('p', &a * &b);
run;

   /* Set a SAS fileref to point to the data set. */
filename myfileref "test.sas";

   /* Set up a function in PROC FCMP and call it from the DATA step. */
proc fcmp outlib=sasuser.ds.functions;
   function subtract_sasfile(a, b);
   rc=run_sasfile('myfileref', a, b,
p);
   if rc=0 then return(p);
      else return(.);
   endsub;
run;

options cmplib=(sasuser.ds);
data _null_;
   a=5.3;
   b=0.7;
   p=.;
   p=subtract_sasfile(a, b);
   put p=;
run;
```

SOLVE 関数

Gauss-Newton 法を使用して、関数の暗黙値を計算します。

カテゴリ: 暗黙値の計算

注: この特殊な目的のある関数は、FCMP プロシジャによって参考のために自動的に提供されます。

構文

answer = **SOLVE**('*function-name*', *options-array*, *expected-value*, *argument-1* <, *argument-2*, ...>);

必須引数

answer
SOLVE 関数から返される値を指定します。

'*function-name*'
関数の名前を指定します。*function-name* を引用符で囲みます。

options-array
SOLVE 関数と使用するオプションの配列を指定します。*Options-array* は、ルート検索プロセスの制御とモニタに使用されます。*Options-array* には欠損値(.)が可能です。または次の順序の次の要素のうち 5 つまでを含めることができます。

initial-value
暗黙値の開始値を指定します。最初の呼び出しのデフォルトは 0.001 です。同じ命令行が再度実行されると、*options-array* では前に検出された暗黙値が使用されます。

absolute-criterion
収束の値を指定します。期待値と予測値間の差異の絶対値は、収束に対する *absolute-criterion* の値未満にしてください。

デフォルト 1.0e-12

relative-criterion
収束の値を指定します。計算済みの暗黙値の変更が *relative-criterion* 値未満のときは、収束が想定されます。

デフォルト 1.0e–6

maximum-iterations
ソリューションの検索に使用する反復の最大数を指定します。

デフォルト 100

solve-status
次の値のうちいずれかになります。

0 成功。

1 エラーを減らすことができません。

2 変更ベクトルを計算できません。

3 反復の最大数を超えています。

4 最初の目的関数がありません。

expected-value
対象の関数の期待値を指定します。

argument
最小化中の関数に渡す引数を指定します。

詳細

SOLVE 関数は、次の形式の式をゼロに等しくする指定引数の値を検索します。

```
expected-value - function-name
(argument-1, argument-2, ...)
```

対象の引数を欠損値(.)で指定します。これは、上記のパラメータリストに引数のかわりに表示されます。SOLVE 関数が値を検索する場合、この関数に対して返される値が暗黙値となります。

次に、オプション配列の例を示します。

```
array opts[5] initial abconv relconv maxiter (.5 .001 1.0e-6 100);
```

これらの値は前の命令行を参照します。

- **initial** (initial-value)=.5

- **abconv** (absolute-criterion)=.001
- **relconv** (relative-criterion)=1.0e-6
- **maxiter** (maximum-iterations)=100

解決ステータスは、配列の 5 番目の要素です。この値は出力リストの opts[5]を指定して表示できます。

例

例 1: 平方根値の計算

次の SOLVE 関数の例は、方程式 y=1/sqrt(x)を満たす x の値を計算します。関数とサブルーチンは、SOLVE 関数で使用できるように、最初に定義する必要があります。この例では、関数 INVERSESQRT が最初に定義され、SOLVE 関数で使用されます。

```
options pageno=1 nodate ls=80 ps=64;

proc fcmp;
      /* define the function */
   function inversesqrt(x);
      return(1/sqrt(x));
   endsub;

   y=20;
   x=solve("inversesqrt", {.}, y, .);
   put x;
run;
```

アウトプット 23.18 平方根値の計算からの結果

```
The SAS System 1 The FCMP Procedure 0.0025
```

例 2: Garman-Kohlhagen インプライドボラティリティの計算

この例で、サブルーチン GKIMPVOL は SOLVE 関数と GARKHPRC 関数を使用して、FX オプションに対する Garman-Kohlhagen インプライドボラティリティを計算します。

この例では、次の点に注意してください。

- options_array は SOLVOPTS で、初期値が必要です。
- 期待値は FX オプションの価格です。
- サブルーチンの欠損引数はボラティリティ(sigma)です。

```
proc fcmp;
   function garkhprc(type$, buysell$, amount, E, t, S, rd, rf, sig)
      kind=pricing label='FX option pricing';

   if buysell='Buy' then sign=1.;
   else do;
      if buysell='Sell' then sign=-1.;
      else sign=.;
   end;

   if type='Call' then
```

```
      garkhprc=sign*amount*(E+t+S+rd+rf+sig);
   else do;
      if type='Put' then
         garkhprc=sign*amount*(E+t+S+rd+rf+sig);
      else garkhprc=.;
   end;
   return(garkhprc);
   endsub;

   subroutine gkimpvol(n, premium[*], typeflag[*], amt_lc[*],
                       strike[*], matdate[*], valudate, xrate,
                       rd, rf, sigma);
   outargs sigma;

   array solvopts[1] initial (0.20);
   sigma=0;
   do i=1 to n;
      maturity=(matdate[i] - valudate) / 365.25;
      stk_opt=1./strike[i];
      amt_opt=amt_lc[i] * strike[i];
      price=premium[i] * amt_lc[i];

      if typeflag[i] eq 0 then type="Call";
      if typeflag[i] eq 1 then type="Put";

         /* solve for volatility */
      sigma=sigma + solve("GARKHPRC", solvopts, price,
                          type, "Buy", amt_opt, stk_opt,
                          maturity, xrate, rd, rf, .);
   end;
   sigma=sigma / n;
endsub;
run;
```

例 3: Black-Scholes インプライドボラティリティの計算

この SOLVE 関数の例では、ビルトイン SAS 関数 BLKSHCLPRC を使用して、関数 BLKSCH を定義します。SOLVE 関数は BLKSCH 関数を使用して、オプションの Black-Scholes インプライドボラティリティを計算します。

この例では、次の点に注意してください。

- options_array は OPTS です。
- 関数の欠損引数はボラティリティ(VOLTY)です。
- PUT ステートメントは、インプライドボラティリティ(BSVOLTY)、初期値、解決ステータスの書き込みに使用されます。

```
options pageno=1 nodate ls=80 ps=64;

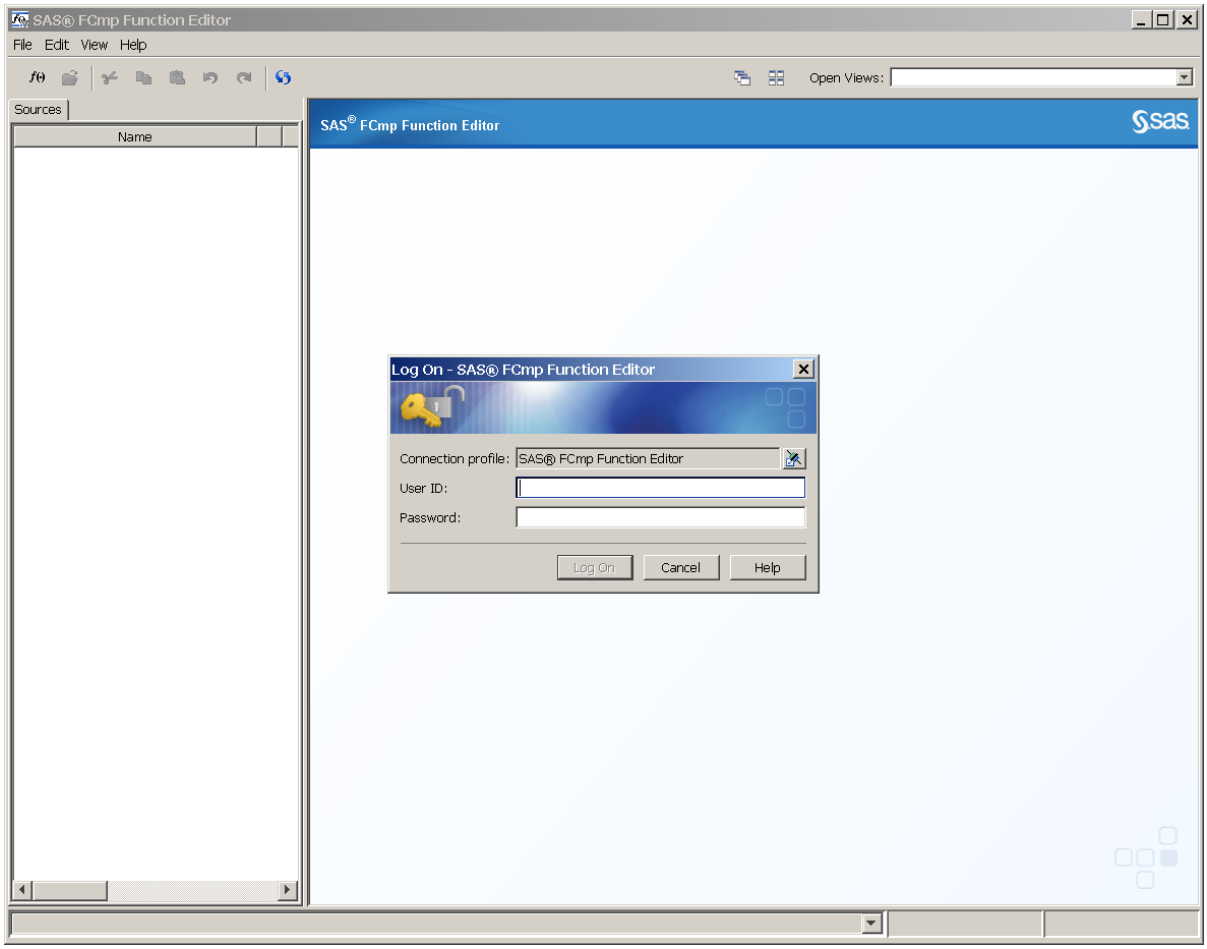
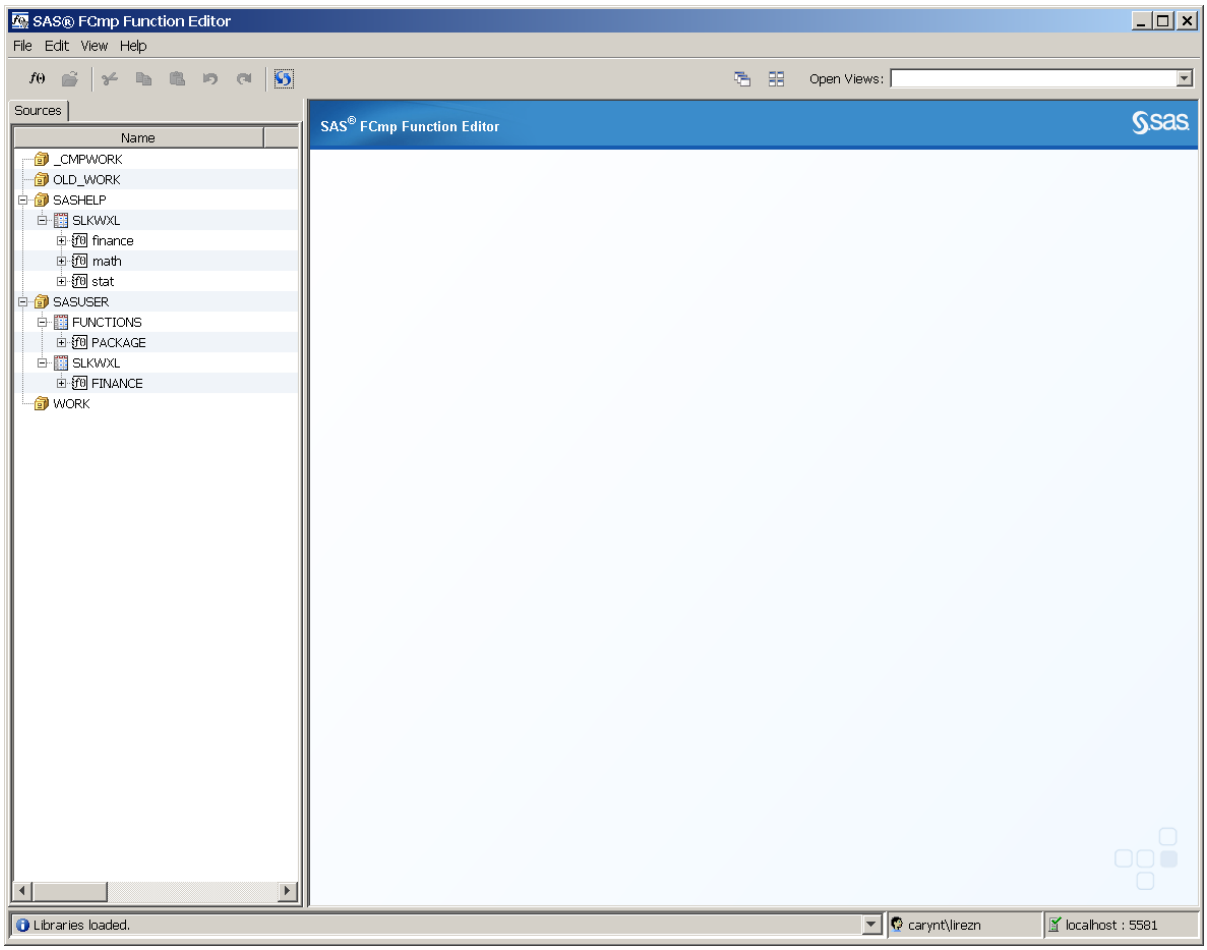
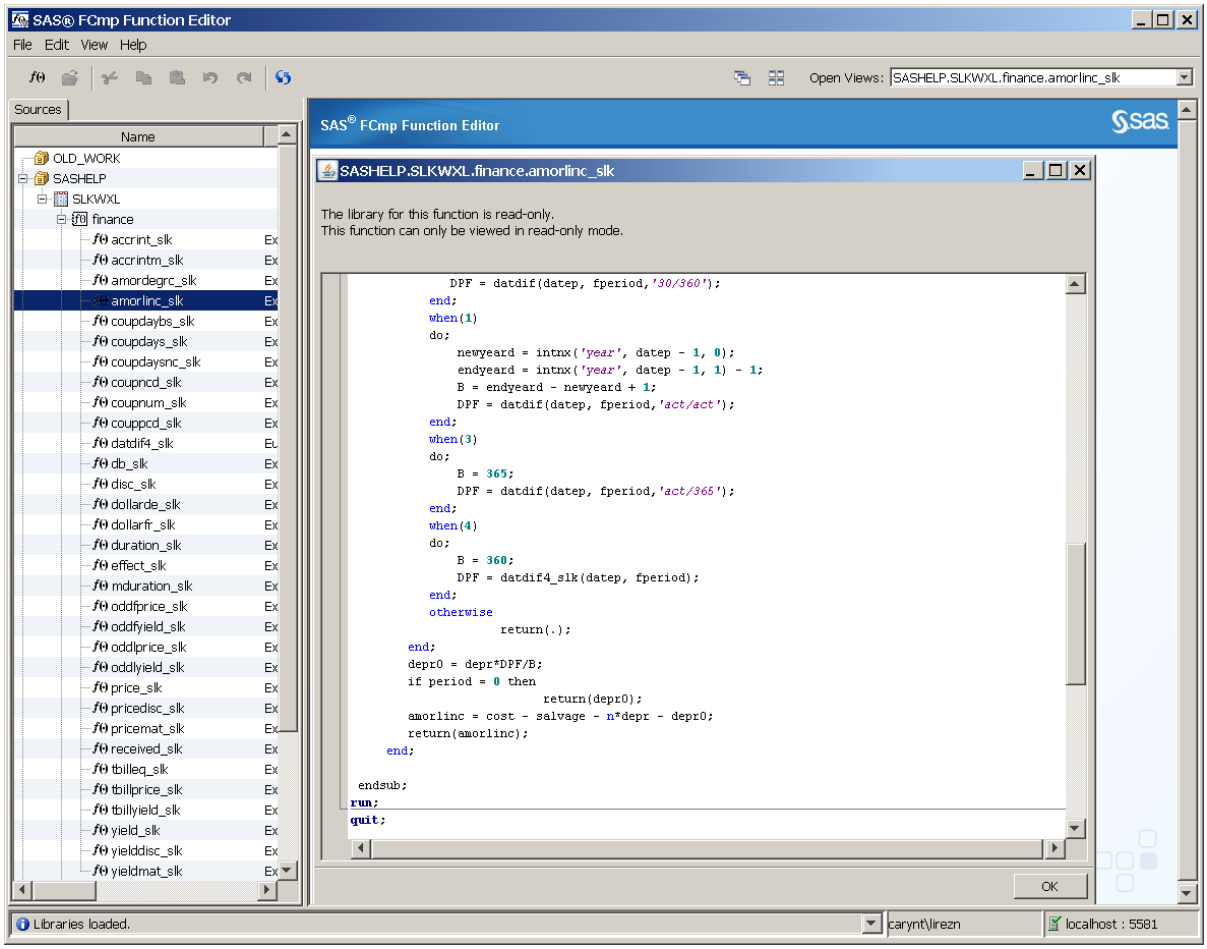
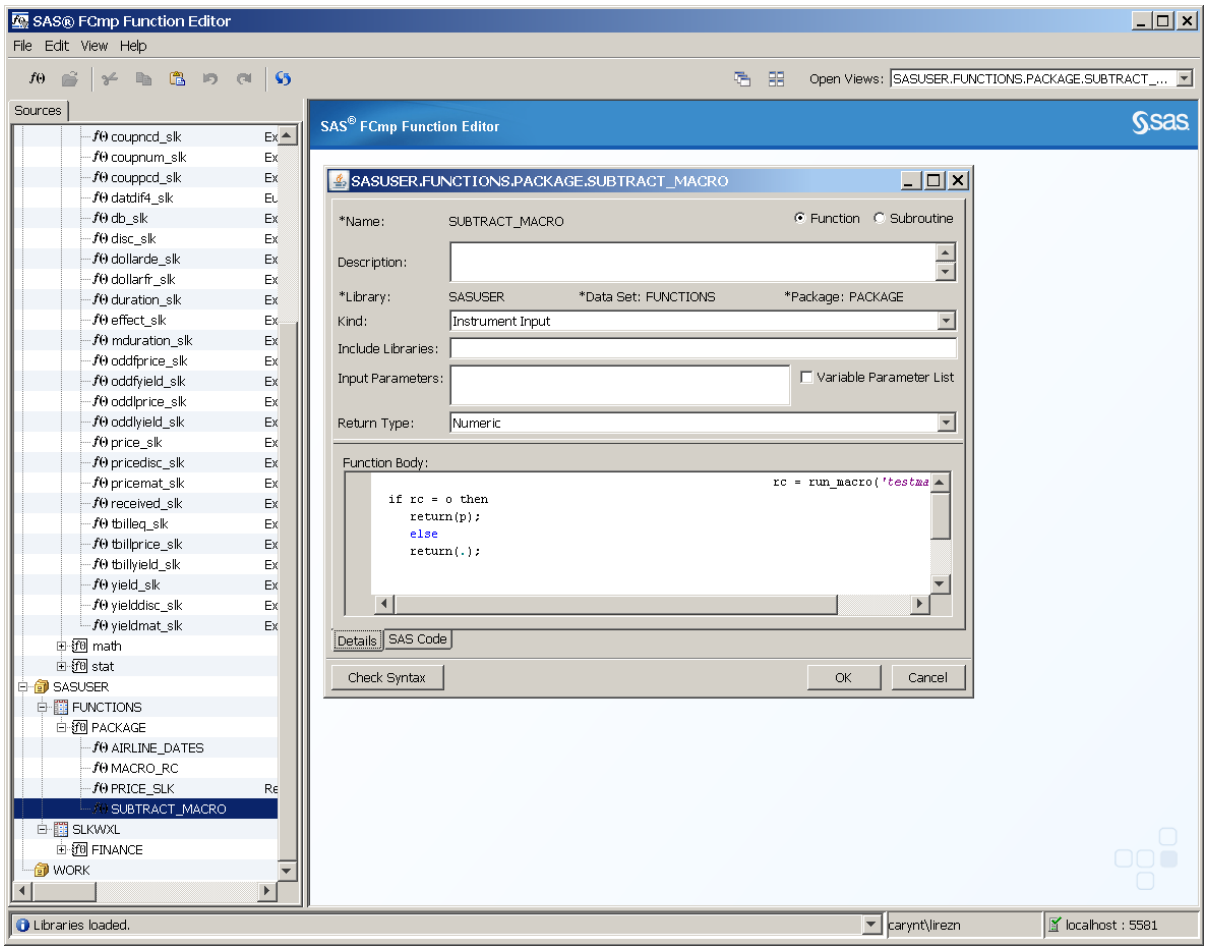
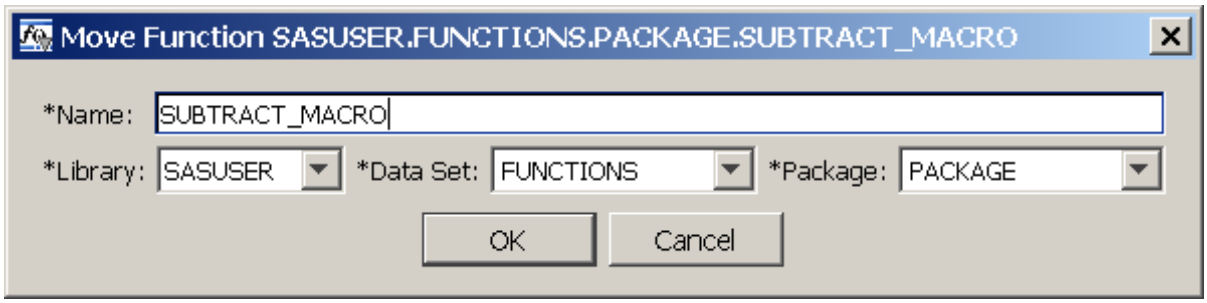
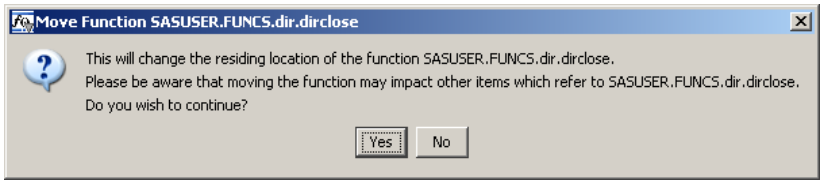
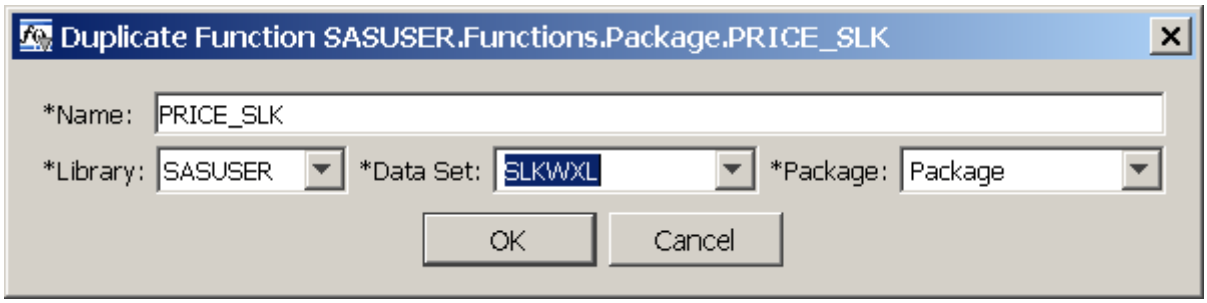
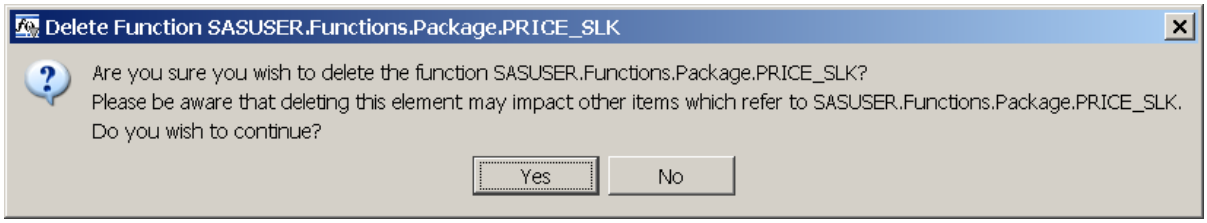
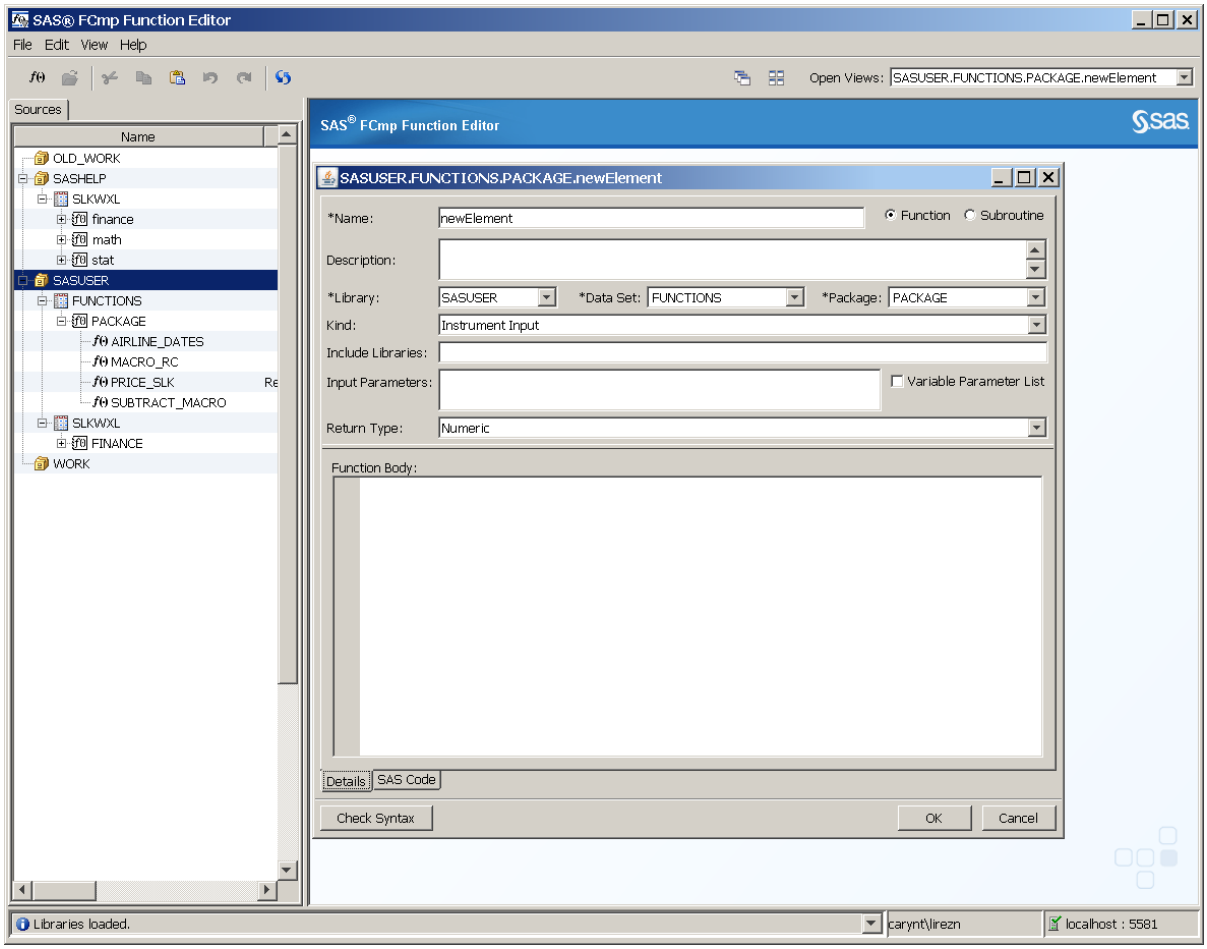
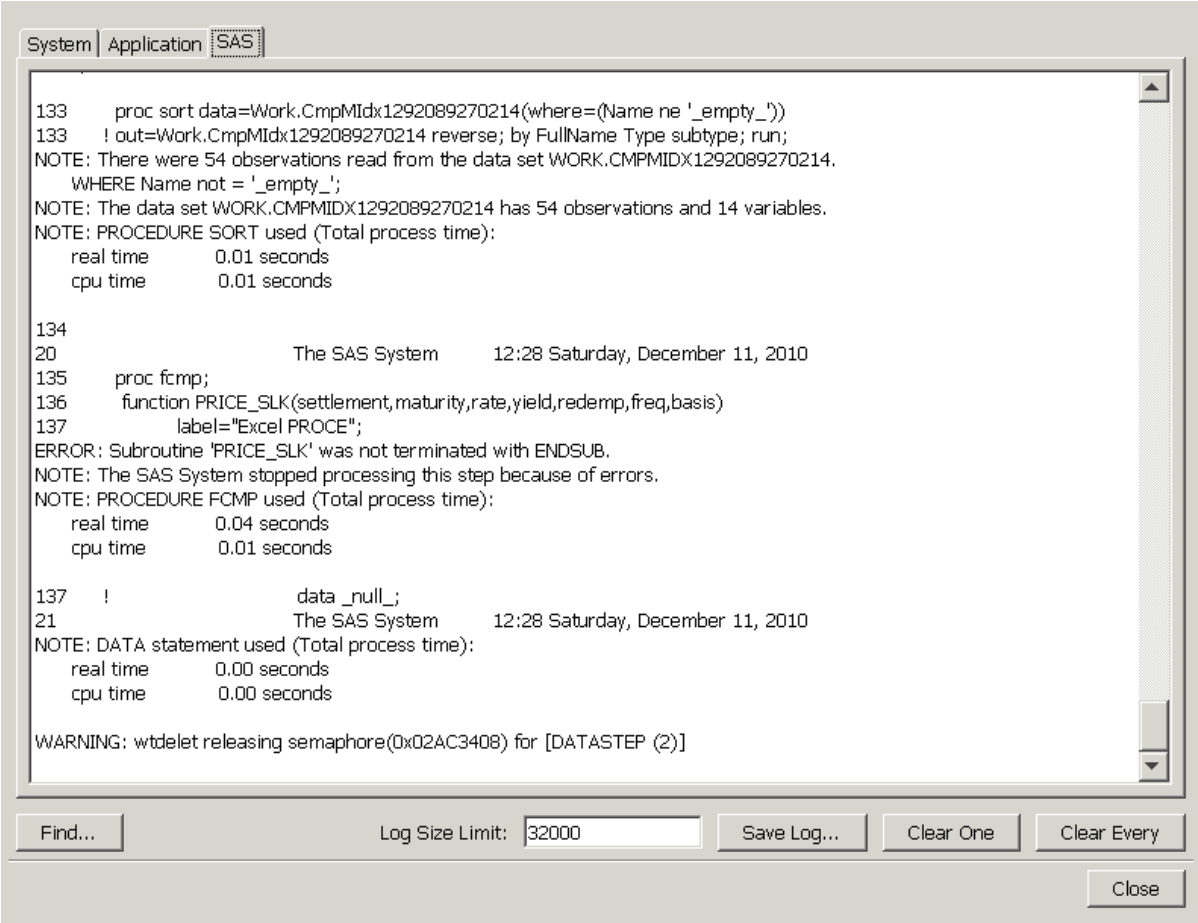
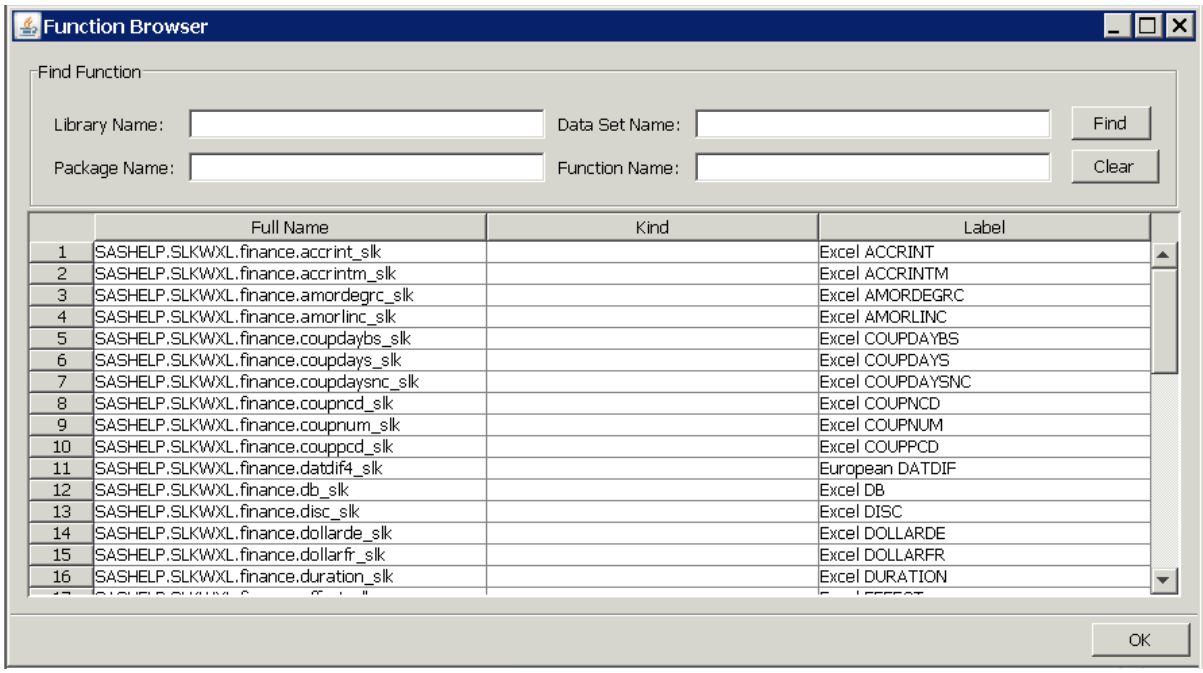
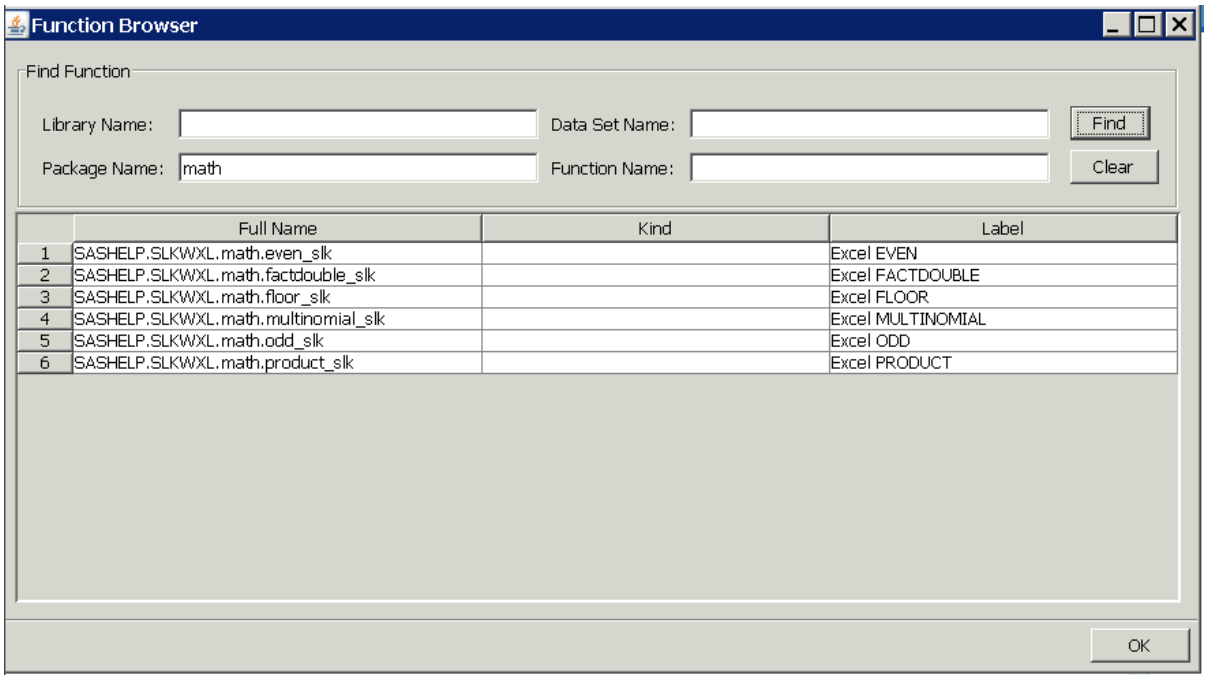
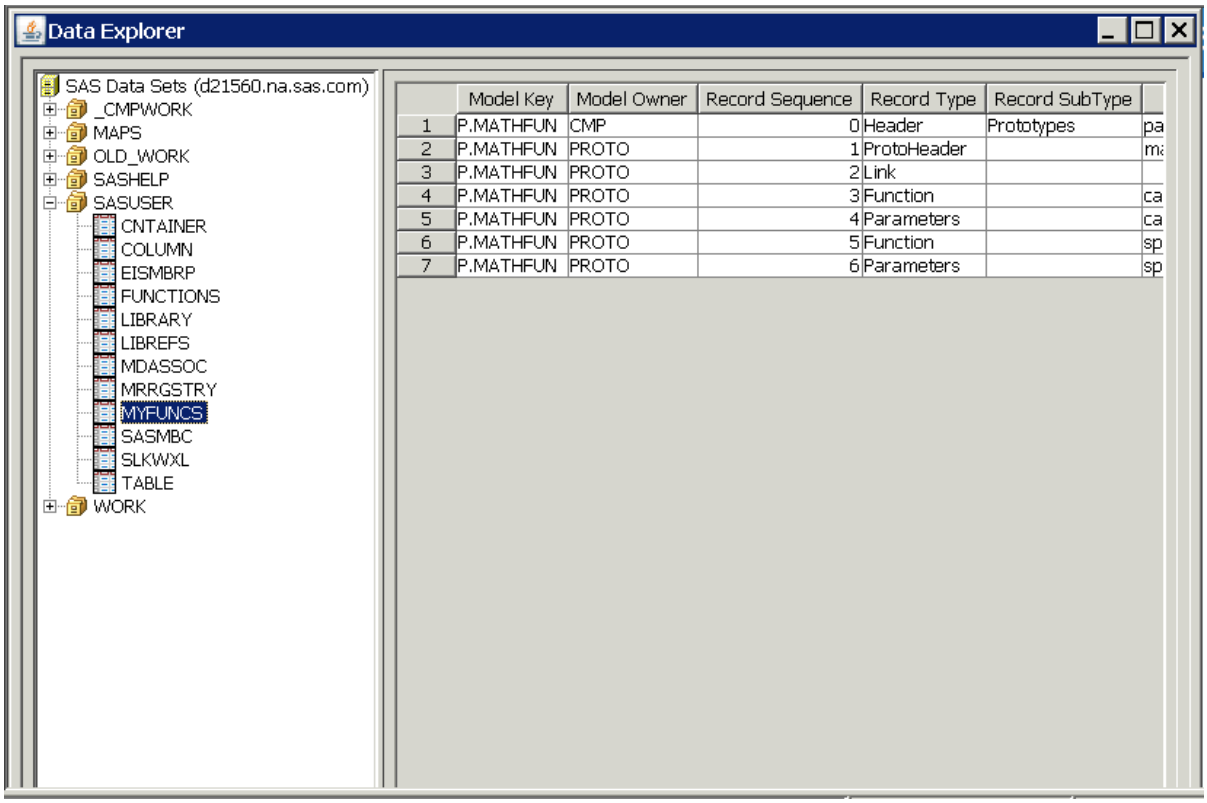
proc fcmp;
   opt_price=5;
   strike=50;
   today='20jul2010'd;
   exp='21oct2010'd;
   eq_price=50;
   intrate=.05;
   time=exp - today;
```

```
    array opts[5] initial abconv relconv maxiter status
                  (.5 .001 1.0e-6 100 -1);
    function blksch(strike, time, eq_price, intrate, volty);
       return(blkshclprc(strike, time/365.25,
                         eq_price, intrate, volty));
    endsub;
    bsvolty=solve("blksch", opts, opt_price, strike,
                              time, eq_price, intrate, .);

    put 'Option Implied Volatility:' bsvolty
        'Initial value: ' opts[1]
        'Solve status: ' opts[5];
run;
```

アウトプット 23.19 *Black-Scholes インプライドボラティリティの計算の結果*

```
                                The SAS System 1 The FCMP Procedure Option
Implied Volatility:0.4687011859 Initial value:0.5 Solve status:0
```

注: SAS 関数と外部 C 関数は SOLVE 関数で直接使用できません。これらの関数は、PROC FCMP 関数で囲む必要があります。この例で、ビルトイン SAS 関数 BLKSHCLPRC が PROC FCMP 関数 BLKSCH で囲まれると、BLKSCH が SOLVE 関数で呼び出されます。

WRITE_ARRAY

データを PROC FCMP 配列変数から SAS プログラム、マクロ、プロシジャによって使用可能なデータセットに書き込みます。

カテゴリ: 配列

注: 配列とデータセット間の変換時に、配列は[row, column]のようにインデックス付けされます。

構文

rc = **WRITE_ARRAY**(*data_set_name*, *array_variable* <, '*column_name_1*', '*column_name_2*', ...>);

必須引数

rc
関数がデータセットを正常に書き込める場合は 0 です。

data_set_name
配列データの書き込み先のデータセットの名前を指定します。*data_set_name* は、作成されるデータセットのメンバ名(libname.memname)が含まれている文字リテラルまたは変数にしてください。

array_variable
コンテンツが *data_set_name* に書き込まれる PROC FCMP 配列または行列変数を指定します。

オプション引数

column_name
作成されるデータセットの列に対して任意の名前を指定します。

column_name を指定した場合、引用符で囲まれたリテラル文字列にしてください。*column_name* は PROC FCMP 変数にはなりません。列名が指定されない場合、列名は接尾辞が数値の配列名になります。

例

例 1: WRITE_ARRAY 関数と PROC FCMP 配列変数の使用

この例では、ARRAY ステートメント、WRITE_ARRAY 関数を PROC FCMP と使用して、出力をデータセットに書き込みます。

```
options nodate pageno=1 ls=80 ps=64;

proc fcmp;
   array x[4,5] (11 12 13 14 15 21 22 23 24 25 31 32 33 34 35 41 42 43 44 45);
   rc=write_array('work.numbers', x);
run;

proc print data=work.numbers;
run;
```

アウトプット 23.20 *WRITE_ARRAY 関数を使用した結果*

```
The SAS System 1 Obs x1 x2 x3 x4 x5 1     11    12    13    14    15 2
21    22    23    24    25 3     31    32    33    34    35 4     41    42
43    44    45
```

例 2: WRITE_ARRAY 関数を使用した列名の指定

この例では、任意の *colN* 変数を使用して、列名をデータセットに書き込みます。

```
options pageno=1 nodate ps=64 ls=80;

proc fcmp;
   array x[2,3] (1 2 3 4 5 6);
   rc=write_array('numbers2', x, 'col1', 'col2', 'col3');
run;

proc print data=numbers2;
run;
```

アウトプット 23.21 *関数を使用して列名を指定した結果*

```
The SAS System 1 Obs col1 col2 col3 QNTLDEF=1 | 2 | 3 | 4 | 6
```

24 章

FCmp 関数エディタ

FCmp 関数エディタについて

PROC FCMP で作成される SAS 言語関数と CALL ルーチンは、パッケージ宣言に含まれる SAS データセットに保存されます。各パッケージ宣言には、1 つ以上の関数または CALL ルーチンが含まれています。FCmp 関数エディタには、パッケージに含まれるすべての関数と CALL ルーチンが表示されます。

FCmp 関数エディタを使用して、パッケージ宣言の関数の表示と新しい関数の作成を行うことができます。これらの新しい関数を既存のパッケージに追加したり、新しいパッケージ宣言を作成することができます。

FCmp 関数エディタを開く

Windows 動作環境を使用していて、SAS がコンピュータにローカルでインストールされている場合、Windows ではシングルサインオン機能がサポートされているため、サインオンダイアログボックスは表示されません。

Windows 動作環境を使用していない場合、または SAS がローカルでインストールされていない場合は、権限認証情報、つまりユーザー ID とパスワードを求められます。

FCmp 関数エディタを開くには、SAS セッションでメニューから**ソリューション** ⇨ **分析** ⇨ **FCmp 関数エディタ**を選択します。次のダイアログボックスが表示されます。

図 24.1 *FCmp 関数エディタの初期ダイアログボックス*



ユーザー ID とパスワードを入力し、**ログオン**をクリックすると、ポートへの接続が確立されます。ライブラリを表示するウィンドウが表示されます。

図 24.2 *ライブラリを表示する FCmp 関数エディタ*



上記のウィンドウで、左ペインに SASHELP ライブラリと SASUSER ライブラリにある関数がリストされます。WORK ライブラリは空です。WORK ライブラリには、spawning SAS セッションからは直接アクセスできません。FCmp 関数エディタは、OLD_WORK から WORK のコンテンツにアクセスできるよう、WORK ライブラリを spawning SAS セッションから OLD_WORK の場所に再マッピングします。

既存の関数の操作

関数を開く

関数を開くには、ライブラリを左ペインから選択し、展開して関数のリストが表示されるまでドリルダウンします。開く関数の名前をダブルクリックします。

読み取り専用ライブラリから関数を開く場合、次のようなウィンドウが表示されます。

図 24.3 *読み取り専用アクセス権があるライブラリの関数*



上記のウィンドウで、AMORLINC_SLK 関数が読み取り専用 SASHELP ライブラリから選択されます。スクロールバーを使用して、関数の上部にスクロールします。

書き込みアクセス権があるライブラリから関数を開く場合、次のようなウィンドウが表示されます。

図 24.4 書き込みアクセス権があるライブラリの関数



上記のウィンドウでは、SUBTRACT_MACRO が書き込み可能な SASUSER ライブラリから選択されます。

ライブラリに読み取り専用アクセス権または書き込みアクセス権のどちらがあるかによって、表示されるウィンドウが異なります。ライブラリに書き込みアクセス権がある場合、表示しているウィンドウの上部セクションに情報を入力できます。これらのフィールドは、新しい関数の作成時に使用するものと同じです。フィールドの説明については、“新しい関数の作成” (788 ページ)を参照してください。

複数の関数を開く

複数の関数を開くと、複数のウィンドウが開きます。たとえば、2 番目の関数を開くと、その関数のコードを表示する 2 番目のウィンドウが開きます。

FCmp 関数エディタの右上隅には、**ビューを開く**というフィールドが含まれています。矢印をクリックして、開いている関数をリストします。関数を選択すると、その関数のウィンドウが最前面に表示されます。

関数の表示の変更に使用できる 2 つのアイコンが、**ビューを開く**フィールドの左側にあります。

開いている関数の表示をカスケード表示します。

関数を並べて表示するように調整します。

関数の移動

関数を異なるライブラリ、データセット、パッケージに移動できます。関数を移動するには、左ペインで関数を選択します。関数を右クリックして、メニューから**移動**を選択します。次のダイアログボックスが表示されます。

図 24.5 *関数の移動ダイアログボックス*



関数の移動ダイアログボックスで、次のタスクを実行できます。

- 関数の新しい名前の入力
- 関数の移動先のライブラリの選択
- 新しいデータセット名の入力
- パッケージ名の入力

次に、**関数の移動**ダイアログボックスのフィールドの説明を示します。

名前
関数の新しい名前を指定します。

ライブラリ
移動する関数を含むライブラリを指定します。ライブラリを選択するには、**ライブラリ**フィールドのメニューを使用します。

データセット
移動する関数を含むデータセットを指定します。データセットを選択するには、データセット名を入力するか、**データセット**フィールドの下向き矢印をクリックします。データセットを選択しない場合、このフィールドの値のデフォルトは FUNCTIONS になります。

パッケージ
移動する新しい関数を含むパッケージ名を指定します。パッケージを選択するには、パッケージ名を入力するか、**パッケージ**フィールドの下向き矢印をクリックします。データセットを選択しない場合、このフィールドの値のデフォルトは PACKAGE になります。

OK をクリックすると、移動に関して警告する次のダイアログボックスが表示されます。

図 24.6 関数の移動確認ダイアログボックス



注意:
移動する関数を参照するその他の関数とマクロは、新しい関数の場所によって更新されません。この状況により、マクロなどの参照オブジェクトが同期対象外になる可能性があります。

はいまたは**いいえ**をクリックします。

関数を閉じる

左ペインで関数名を右クリックして**閉じる**を選択すると、関数を表示するウィンドウが閉じます。関数を表示するウィンドウの右下隅の **OK** をクリックして関数を閉じることもできます。

関数の複製

表示している関数を、書き込みアクセス権がある既存または新しいパッケージやライブラリに複製(コピー)できます。関数を複製するには、左ペインで関数を選択します。関数を右クリックして、メニューから**複製**を選択します。次のダイアログボックスが表示されます。

図 24.7 関数の複製ダイアログボックス



このダイアログボックスのフィールドには、複製する関数の関数名、ライブラリ、データセット、パッケージが自動的に表示されます。これらのフィールドは、関数の複製時に変更できます。

これらのフィールドの説明については、“関数の移動” (786 ページ)を参照してください。

関数の名前変更

指定パッケージ内の関数の名前を変更するには、**名前の変更**ダイアログボックスを使用します。関数を含むライブラリへの書き込みアクセス権が必要です。関数名を変更すると、新しい関数は元の関数と同じライブラリに存在します。

関数名を変更するには、左ペインで関数を選択します。関数を右クリックして、メニューから**名前の変更**を選択します。関数の新しい名前を入力して、**OK** をクリックします。

注意:

名前の変更を使用して、指定パッケージ内の関数の名前を変更できます。関数の移動と同様に、関数の名前変更によって従属のマクロとその他のエンティティは変更されません。

関数の削除

書き込みアクセス権があるライブラリから関数を削除できます。関数を削除するには、削除する関数を選択します。関数を右クリックして、メニューから**削除**を選択します。**削除**の他の項目への影響に関して警告する、次のダイアログボックスが表示されます。

図 24.8 *関数の削除確認ダイアログボックス*



はいまたは**いいえ**をクリックします。

新しい関数の作成

ライブラリ、データセット、パッケージを選択すると、新しい関数を作成できます。ライブラリで新しい関数を作成するには、新しい関数を追加するライブラリの上にカーソルを置きます。ライブラリを右クリックして、**関数の新規作成**を選択します。メニューから**ファイル** ⇨ **関数の新規作成**を選択するか、メニューバーの下の左上隅にある $f(\cdot)$ をクリックします。

次のウィンドウが表示されます。

図 24.9 *newElement ウィンドウ*



ウィンドウの右上隅には、2 つのボタン、**関数**と**サブルーチン**があります。新しい関数または新しいサブルーチンを作成するかどうかによって、いずれかのボタンをクリックします。

newElement ウィンドウには、次のフィールドが含まれています。

名前
新しい関数またはサブルーチンの名前を指定します。

説明
新しい関数またはサブルーチンの名前を説明します。

ライブラリ
新しい関数またはサブルーチンを含むライブラリを指定します。ライブラリを選択するには、ライブラリ名を入力するか、**ライブラリ**フィールドの下向き矢印をクリックします。

データセット
新しい関数またはサブルーチンを含むデータセットを指定します。データセットを選択するには、データセット名を入力するか、**データセット**フィールドの下向き矢印をクリックします。値を指定しない場合、このフィールドの値のデフォルトは FUNCTIONS になります。

パッケージ
新しい関数またはサブルーチンを含むパッケージの名前を指定します。パッケージを選択するには、パッケージ名を入力するか、**パッケージ**フィールドの下向き矢印をクリックします。**パッケージ**フィールドは必須フィールドです。値を指定しない場合、このフィールドの値のデフォルトは PACKAGE になります。

種類
指定パッケージ内の関数またはサブルーチンをグループ化できます。次の 4 つの事前定義された種類のグループが使用可能で、通常は SAS Risk Management で使用されます。

- プロジェクト
- リスクファクタ変換
- 金融商品の評価
- 金融商品の入力

これらの 4 つのグループのいずれかを使用するか、**種類**フィールドに独自の種類の値を入力できます。**種類**に値を指定した場合、左ペインの関数ツリーはパッケージの関数をその種類グループにグループ化します。

ライブラリの追加
関数またはサブルーチンに含める SAS コードを含むライブラリを指定します。

入力パラメータ
関数またはサブルーチンへの入力として使用する引数を指定します。

変数パラメータリスト
関数またはサブルーチンが可変個引数をサポートするかどうかを指定します。

戻り値の種類
関数またはサブルーチンが文字値または数値を返すかどうかを指定します。

関数本体
関数またはサブルーチンをコード化するウィンドウの領域です。

次の 3 つのボタンが **newElement** ウィンドウの左下にあります。

詳細
関数またはサブルーチンに関する説明情報(新しい関数の名前、追加ライブラリのリスト、入力パラメータなど)を書き込む領域を提供します。**関数本体**セクションで新しい関数またはサブルーチンをコード化します。**詳細**タブはデフォルトで選択されています。

SAS コード
書き込んだ関数またはサブルーチンを表示できます。**SAS コード**を選択すると、読み取り専用機能が提供されます。

構文チェック
書き込んだコードの構文をチェックできます。構文が正しい場合、構文が正しいことを示すダイアログボックスが表示されます。構文にエラーが含まれている場合、エラーを説明するダイアログボックスが表示されます。ウィンドウの左下のバーにはエラーメッセージも表示されます。構文エラーはログに書き込まれます。**表示** ⇨ **ログの表示**メニューからアクセスできます。

詳細タブおよび**関数本体**セクションの説明部分に情報を入力すると、その情報は SAS コードに変換され、**SAS コード**ボタンを選択すると表示できます。

FCmp 関数エディタでの新規ライブラリの表示

FCmp 関数エディタにログオンしている間、SAS セッションで新しいライブラリを作成できます。ただし、FCmp 関数エディタで更新ボタンをクリックしても、新規ライブラリは表示されません。新しいライブラリを表示するには、FCmp 関数エディタからログオフしてからログインしなおす必要があります。

ログウィンドウ、関数ブラウザ、データエクスプローラの表示

ログウィンドウ

ログウィンドウを表示するには、メニューから**表示** ⇨ **ログの表示**を選択します。**ログ**ウィンドウの表示時にウィンドウの左上隅にあるタブを選択して、システム、アプリケーション、プログラム結果を表示できます。

SAS ログのコンテンツを表示するには、**SAS** タブをクリックします。ログのコンテンツには、SAS サーバーからの出力が表示されます。また、SAS に送信されるコマンドも表示され、コンテキストがログ出力に追加されます。

次に、SAS ログが表示された**ログ**ウィンドウを示します。

図 24.10 *ログウィンドウ*



ログウィンドウのタブとボタン

ログウィンドウの**システム**タブには、詳細情報がシステムメッセージの形式で表示されます。メッセージが記録されていない場合、ウィンドウは空です。

システムウィンドウには、ウィンドウの右上セクションにある次の 2 つの垂直タブが含まれています。これらのタブには、関連のあるメッセージに関する情報が提供されます。

System.out
メッセージがこの場所に送られる場合、システム出力を表示します。

System.err
メッセージがこの場所に送られる場合、エラーメッセージを表示します。

ログウィンドウには、ウィンドウの右下にある次の 3 つのボタンが含まれます。

ログの保存
ログ出力を選択するファイルに保存します。

1 つクリア
アクティブなウィンドウの結果をクリアします。

すべてクリア
3 つのすべてのウィンドウの結果をクリアします。

検索ボタンは、ウィンドウの左下にあります。このボタンを選択すると、出力の検索が可能なダイアログボックスが開きます。たとえば、**SAS** タブの選択時に ERROR を検索すると、SAS ログのエラーを迅速に検索できます。

関数ブラウザ

関数ブラウザには、ウィンドウの左ペインにリストされるすべての関数が表示されます。この関数リストをフィルタして、関数のサブセットを表示できます。

関数ブラウザウィンドウを表示するには、メニューから**表示** ⇨ **関数ブラウザの表示**を選択します。次のようなウィンドウが表示されます。

図 24.11 *関数ブラウザウィンドウ*



上記で表示される一部の出力では、関数がアプリケーションツリーに表示されます。関数リストの上にある**関数ブラウザ**の各フィールドに条件を入力して、出力をフィルタし、関数のサブセットを作成できます。これらのフィールドは、**ライブラリ名**、**データセット名**、**パッケージ名**、**関数名**です。

次の表示では、**パッケージ名**フィールドがフィルタとして使用されます。ウィンドウの右下隅の **OK** ボタン、または右上隅の**検索**ボタンを押すと、次のウィンドウが表示されます。

図 24.12 *関数ブラウザからのフィルタされた出力*



リストされる数字関数は、すべての関数のサブセットです。

フィルタ条件によって、選択するフィールドに情報を入力できます。たとえば、SASHELP などの値を**ライブラリ名**フィールドに入力すると、SASHELP ライブラリ内のすべての関数が表示されます。

データエクスプローラ

データエクスプローラでは、選択するデータセットのデータを表示できます。

データエクスプローラウィンドウを表示するには、メニューから**表示** ⇨ **データエクスプローラの表示**を選択します。次のようなウィンドウが表示されます。

図 24.13 *データエクスプローラウィンドウ*



データエクスプローラウィンドウには、左ペインから選択するデータセットに基づいてデータセット情報が表示されます。

列ヘッダーをクリックし、列を移動して表示の位置を変更できます。ウィンドウの右下セクションの **OK** をクリックすると、行った変更が保存されます。

DATA ステッププログラムの関数の使用

PROC FCMP と DATA ステップ構文の操作方法の例については、“ディレクトリトランスバーサル” (718 ページ)を参照してください。

25 章

FEDSQL プロシジャ

概要: FEDSQL プロシジャ

FEDSQL プロシジャを使用すると、Base SAS セッションから FedSQL 言語のステートメントをサブミットできます。FedSQL 言語は、ANSI SQL:1999 コアスタンダードの SAS 実装です。DECIMAL、INTEGER、および VARCHAR などの新しいデータ型、その他の ANSI 1999 コアコンプライアンス機能、専用拡張子に対するサポートを提供します。FedSQL は、複数のデータソースのリレーショナルデータにアクセスし、その管理と共有を行う拡張可能なスレッド式の高性能な方法を導入するデータアクセス技術も提供します。可能な場合、FedSQL クエリは、大規模な処理を解決するためのマルチスレッド式アルゴリズムを使用して最適化されます。

アプリケーションについて、FedSQL はすべてのデータソース全体にわたって共通の SQL 構文を提供します。つまり、FedSQL は、データソースに固有の SQL Dialect でクエリを送信しなくても、各種データソースからデータにアクセスするベンダーに中立な SQL Dialect です。また、1 つの FedSQL クエリで複数のデータソース内のデータを対象にして、1 つの結果テーブルを返すことができます。

FedSQL 言語の詳細については、*SAS FedSQL Language Reference* を参照してください。

概念: FEDSQL プロシジャ

FedSQL の利点

SAS SQL プロシジャで提供される機能数よりも多い機能を必要とする SQL プログラムを作成する必要がある環境で作業している場合、FedSQL には数多くの利点があります。

- FedSQL は SQL 1999 ANSI 規格に準拠します。この規格に準拠することによって、独自の言語はもちろん、ANSI 1999 規格に準拠するその他の DBMS のネイティブ言語でもクエリを処理できます。
- FedSQL は、以前の SAS SQL 実装に比べてより多くのデータ型をサポートします。SAS/ACCESS LIBNAME エンジンを利用する従来の DBMS アクセスでは、ターゲット DBMS データ型と 2 つのレガシー SAS データ型(SAS 数値および SAS 文字)間で変換が行われます。FedSQL が DBMS に接続する場合、言語はターゲットデータソースの定義と一致するか、または必要に応じてその定義を FedSQL データ型に変換します。これによって、精度が飛躍的に高まります。
- FedSQL は連結クエリを処理し、クエリは複数のデータソースのデータにアクセスして 1 つの結果セットを返します。連結クエリは、複数のデータソースと通信し、そのデータにアクセスして、そのデータに対する処理を実行する機能です。また、FedSQL には、複数のデータベースに接続される 1 つの SQL クエリを分離し、個別のデータベースに各部分を送信する機能もあります。
- FedSQL パススルーファシリティを使用すると、データソースに接続し、そのデータソースに SQL ステートメントを直接送信して実行することができます(明示パススルー)。このファシリティでは、データソースの構文を使用することもでき、データソースによってサポートされる ANSI 以外のスタンダード SQL をサポートします。この接続では、SELECT ステートメントの FROM 句と EXECUTE ステートメントの CONNECTION TO コンポーネントを使用したネイティブ SQL 構文を使用できます。
- FedSQL は、処理のためにソース DBMS にプロセスが受け渡されるように、黙示的なパススルー(クエリコードを等価の DBMS 固有の SQL コードに変換するプロセス)もサポートします。黙示的なパススルーによって、クエリの応答時間が改善され、セキュリティが強化されます。
  - データソースでクエリを実行することで、転送中のデータのボリュームが削減されます。データソースから FedSQL へ転送される行の数が大幅に削減されるため、全体のクエリ処理時間が短縮されます。大量並列処理などのデータソース固有の機能の活用は、データソースに固有のものです。特別な機能のその他の例としては、高度な結合技術、データのパーティション化、テーブル統計、列統計があります。多くの場合、これらの機能を使用すると、データソースで FedSQL よりも SQL クエリをより迅速に実行できます。
  - セキュリティ上の利点として、実行可能なクエリの一部がデータソース側で処理されます。これによって、機密情報を含んでいる可能性がある関連テーブルをクエリ処理のために FedSQL 側に転送させる必要がなくなります。
- FedSQL 言語は、任意のサポート対象データソース内でデータを作成できます。これによって、アプリケーションの要件に最も対応するデータソースにデータを保存することができます。また、CREATE TABLE クエリ式を使用すると、既存のテーブ

ルからテーブルを作成できます。例については、“例 3: 既存のテーブルからのテーブルの作成” (812 ページ)を参照してください。

- FedSQL は、異種相関サブクエリを実行できます。相関サブクエリは、外部クエリ結果によってサブクエリの結果に影響が及ぼされる場合のクエリです。FedSQL には、サブクエリがデータソースによって実行されるよう命令し、データソースから転送される結果セットを制限する機能があります。これは、FedSQL でサブクエリ式を唯一満たす行を取得できる柔軟なクエリテキスト化技術を使用して実行されます。例については、“例 4: 相関サブクエリを使用したデータのクエリ” (814 ページ)を参照してください。
- FedSQL で異種処理を実行するための独自の方法は、結合を実行するときにサードパーティの DBMS でインデックスを活用することです。結合に含まれる 1 つのテーブルの行の値は、FedSQL ローカルスペースにコピーされます。次に、適格な値を返す特別なテキスト化技術を使用し、DBMS インデックスへの FedSQL アクセスにより結合を実行できるようにすることで、指定列値はインデックスの精査に使用されます。例については、“例 5: 結合を実行するための DBMS インデックスの作成および使用” (815 ページ)を参照してください。

データソースサポート

PROC FEDSQL は次のデータソースにアクセスできます。

- Aster
- UNIX および PC オペレーティング環境用の DB2
- Greenplum
- Hadoop (Hive、HDMD、HAWQ、Impala)
- MySQL
- Netezza
- ODBC データベース(Microsoft SQL Server など)
- Oracle
- PostgreSQL
- SAP (読み取り専用)
- SAP HANA
- Sybase IQ
- SAS データセット
- SAS Scalable Performance Data (SPD) Engine データセット
- Teradata

構文: FEDSQL プロシジャ

PROC FEDSQL <*option(s)*>;
…*FedSQL ステートメント*
QUIT;

ステートメント	タスク	例
“PROC FEDSQL ステートメント”	後続の入力が FedSQL ステートメントであることを示します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
“QUIT ステートメント”	FEDSQL プロシジャの実行を停止します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5

PROC FEDSQL ステートメント

後続の入力が FedSQL ステートメントであることを指定します

制限事項: 既存のテーブルを上書きすることはできません。DROP TABLE ステートメントを使用することでテーブルをドロップし、CREATE TABLE ステートメントでテーブルを再作成する必要があります。

操作: デフォルトでは、このプロシジャは存在しない値を SAS の欠損値として処理します。存在しない値が ANSI SQL null 値として処理されるように要求するには、ANSIMODE を指定します。

例:
“例 1: SAS データセットの作成” (807 ページ)
“例 2: 連結クエリによる複数のデータソースへのアクセス” (811 ページ)
“例 3: 既存のテーブルからのテーブルの作成” (812 ページ)
“例 4: 相関サブクエリを使用したデータのクエリ” (814 ページ)
“例 5: 結合を実行するための DBMS インデックスの作成および使用” (815 ページ)

構文

PROC FEDSQL <*option(s)*>;

オプション引数の要約

ANSIMODE
存在しない値が ANSI SQL null 値として処理されるように指定します。

AUTOCOMMIT | NOAUTOCOMMIT
デフォルトの行数が更新された後で更新内容が自動的に確定されるかどうか(つまり、テーブルに保存されるかどうか)、およびロールバックが実行可能かどうかを指定します。

ERRORSTOP | NOERRORSTOP
プロシジャにエラーが発生した場合、その実行を停止するかどうかを指定します。

EXEC | NOEXEC
構文が正確かを確認した後、ステートメントを実行するかどうかを指定します。

LABEL | NOLABEL
列見出しとして列ラベルまたは列名を使用するかどうかを指定します。

NOPRINT
結果の通常表示を抑制します。

NUMBER

Row という名前の列(行が取得される際のデータの行(オブザベーション)番号)を含めるように指定します。

STIMER
経過時間統計などのシステムパフォーマンス統計のサブセットが SAS ログに書き込まれるように指定します。

XCODE=ERROR | WARNING | IGNORE
NLS トランスコーディングが失敗した場合の SAS セッションの動作を制御します。

オプション引数

ANSIMODE
存在しない値が ANSI SQL null 値として処理されるように指定します。デフォルトでは、PROC FEDSQL プロセスは存在しない値を SAS の欠損値として処理します。動作の違いに関する詳細は、*SAS FedSQL Language Reference* で FedSQL が null 値と SAS の欠損値を処理する方法に関する情報を参照してください。

AUTOCOMMIT | NOAUTOCOMMIT
デフォルトの行数が更新された後で更新内容が自動的に確定されるかどうか(つまり、テーブルに保存されるかどうか)、およびロールバックが実行可能かどうかを指定します。

デフォルト すべての更新は各要求の送信直後に確定され、ロールバックは実行できません。

要件 トランザクションをサポートしないデータソースの場合、NOAUTOCOMMIT を指定するとエラーが返されます。トランザクションをサポートするデータソースには、Aster、DB2 (UNIX および PC ホスト用)、Greenplum、Microsoft SQL Server、MySQL、ODBC データベース、Sybase IQ が含まれます。

ERRORSTOP | NOERRORSTOP
プロシジャにエラーが発生した場合、その実行を停止するかどうかを指定します。バッチまたは非対話型セッションでは、ERRORSTOP はプロシジャにステートメントの実行停止を命令しますが、エラー発生後も構文のチェックは継続するように命令します。NOERRORSTOP は、プロシジャにステートメントの実行を命令し、エラー発生後も構文のチェックを継続するように命令します。

デフォルト 対話型 SAS セッションでの NOERRORSTOP、バッチまたは非対話型セッションでの ERRORSTOP

操作 このオプションが役立つのは、EXEC オプションが有効な場合のみです。

ヒント ERRORSTOP は、SAS がバッチまたは非対話型実行モードで実行している場合にのみ有効です。

NOERRORSTOP は、エラー発生後もバッチジョブに SQL プロシジャステートメントの実行を継続させる場合に役立ちます。

EXEC | NOEXEC
構文が正確かを確認した後、ステートメントを実行するかどうかを指定します。

デフォルト EXEC

ヒント NOEXEC は、ステートメントを実行せずに FedSQL ステートメントの構文をチェックする場合に役立ちます。

LABEL | NOLABEL
列見出しとして列ラベルまたは列名を使用するかどうかを指定します。

デフォルト LABEL

操作 列にラベルがない場合、プロシジャでは列名が列見出しとして使用されます。

列エイリアスによって、ラベルまたは列名が列見出しとして上書きされます。

NOPRINT
結果の通常表示を抑制します。

操作 NOPRINT は、ステートメントによって実行される行数が含まれる SQLOBS 自動マクロ変数の値に影響を与えます。

NUMBER
Row という名前の列(行が取得される際のデータの行(オブザベーション)番号)を含めるように指定します。

デフォルト 行番号はありません。

STIMER
経過時間統計などのシステムパフォーマンス統計のサブセットが SAS ログに書き込まれるように指定します。STIMER が有効な場合、このプロシジャは各ステップと SAS セッション全体に使用されるコンピュータリソースのリストを SAS ログに書き込むように指定します。

デフォルト パフォーマンス統計は SAS ログに書き込まれません。

操作 SAS システムオプション FULLSTIMER が有効な場合、コンピュータリソースの詳細リストが SAS ログに書き込まれます。

XCODE=ERROR | WARNING | IGNORE
NLS トランスコーディングが失敗した場合の SAS セッションの動作を制御します。トランスコーティングの失敗は、行の入力または出力操作中や、ストリングの割り当て中に発生する可能性があります。トランスコーディングは、あるエンコーディングから別のエンコーディングに文字データを変換するプロセスです。

ERROR
実行時エラーが発生し、行処理が停止するように指定します。エラーメッセージが SAS ログに書き込まれます。これはデフォルトの動作です。

WARNING
互換性のない文字が代替文字に設定されるように指定します。警告メッセージが SAS ログに書き込まれます。

IGNORE
互換性のない文字が代替文字に設定されるように指定します。SAS ログにメッセージは書き込まれません。

デフォルト ERROR

QUIT ステートメント

FEDSQL プロシジャの実行を停止します。

操作: 他の SAS プロシジャとは異なり、PROC FEDSQL ではステップの境界は認識されません。つまり、最初に QUIT ステートメントを指定せずに DATA ステップまたは他のプロシジャステップをサブミットすると、FedSQL 言語によって構文エラーが発行され、PROC FEDSQL の処理が続行されます。FEDSQL プロシジャを停止するには、QUIT ステートメントが必要です。

構文

QUIT;

詳細

FEDSQL プロシジャが QUIT ステートメントに到達すると、プロシジャによって割り当てられたすべてのリソースが解放されます。プロシジャを再度起動しなければ、FedSQL 言語ステートメントを実行することはできなくなります。ただし、データソースサーバーへの接続は LIBNAME ステートメントを介して確立されているため、その接続が失われることはありません。この結果、LIBNAME エンジンはサーバーにすでに接続されているため、同じ libref を使用するプロシジャをこれ以降に起動すると、ほぼ即時に実行されます。

FEDSQL プロシジャの使用

データソース接続

PROCFEDSQL は、現在割り当てられている libref の属性を使用してデータソースに接続します。属性にはデータの物理的場所が含まれており、一部のデータソースについては、データサーバーへのアクセスに使用されるネットワーク情報などのアクセス情報、ユーザー ID およびパスワードが含まれます。

最初に SAS エンジンの LIBNAME ステートメントを送信し、次に PROC FEDSQL を送信します。サポートされるエンジンには、BASE (V9、V8、および V7)、SPD Engine、SAS/ACCESS エンジン Aster、DB2、Greenplum、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、SAP、Sybase IQ、Teradata などがあります。

この例では、以前割り当てた libref の属性を使用して PROC FEDSQL がデータソースにどのようにアクセスするかを示します。LIBNAME ステートメントは libref MyFiles を割り当て、BASE エンジンを指定し、SAS データセットの物理的場所を指定します。FedSQL プログラムは、LIBNAME ステートメントで指定された場所で MyFiles.Table1 という名前の SAS データセットを作成します。

```
libname MyFiles base 'C:\MyFiles\Base';

proc fedsql;
   create table MyFiles.Table1 (x double);
   insert into MyFiles.Table1 values (1.0);
   insert into MyFiles.Table1 values (2.0);
```

```
   insert into MyFiles.Table1 values (3.0);
quit;
```

PROC FEDSQL は、接続情報(物理的な場所など)のみに libref 属性を使用します。PROC FEDSQL は、動作を定義する libref 属性を使用しません。たとえば、BASE エンジンに以前にサブミットした LIBNAME ステートメントによって、SAS データセットが圧縮されるように指定されている場合、比較属性はプロシジャによって使用されません。

注: PROC FEDSQL はただちに接続され、LIBNAME ステートメントに DEFER=YES オプションが含まれる場合にエラーが生成されます。

z/OS 固有
SAS ライブラリの物理的な場所は、HFS パスの指定場所にする必要があります。

FedSQL テーブルオプションの適用

PROC FEDSQL を使用してデータソースにアクセスするときに、後続の FedSQL ステートメントで FedSQL テーブルオプションを適用できます。テーブルオプションは、バッファページサイズの割り当てやパスワードの指定などの処理をテーブル上で実行できるようにするアクションを指定します。FedSQL テーブルオプションは、基本 SAS データセットオプションと同じ機能の多くを実行します。

FedSQL テーブルオプションは、PROC FEDSQL 内でデータソースにアクセスする際、オプションの適用に使用されます。たとえば、次のコードは新しいテーブルの恒久バッファページのサイズを指定するため、テーブルオプションを SAS データセットに適用します。

```
libname MyFiles base 'C:\MyFiles\Base';

proc fedsql;
   create table MyFiles.Table1 {options bufsize=16k}(x double) ;
   insert into MyFiles.Table1 values (1.0);
   insert into MyFiles.Table1 values (2.0);
   insert into MyFiles.Table1 values (3.0);
quit;
```

テーブルオプションのリストについては、*SAS FedSQL Language Reference* を参照してください。

マクロ変数

リテラル文字列でのマクロ変数の使用

マクロ変数を指定すると、シンボルの置換によってプログラム内でテキストを動的に変更できます。プログラム内でマクロ変数を参照すると、マクロプロセッサによって参照値は指定したマクロ変数の値に置き換えられます。

PROC FEDSQL では、後続の FedSQL ステートメントでマクロ変数を使用できます。ただし、二重引用符はマクロプロセッサがマクロ変数の参照を解決するために必須ですが、マクロ変数がリテラル文字列内で実行される場合、その文字列を二重引用符で囲むことはできません。FedSQL ステートメントは、二重引用符で囲まれた文字列をテーブル名または列名などの区切り識別記号(大文字/小文字を区別する)とみなすため、文字列を二重引用符で囲むことはできません。

リテラル文字列内でマクロ変数を参照するには、SAS マクロ関数%TSLIT を使用します。この関数によって、リテラル文字列を二重引用符で囲む必要がなくなり、入力値は

一重引用符で囲まれるようになります。たとえば、次のステートメントには &SYSHOSTNAME マクロ変数を指定するための%TSLIT 関数が含まれており、これによって、それが実行されるコンピュータのホスト名が返されます。

```
select %tslit(&syshostname);
```

%TSLIT マクロ関数は、デフォルトのオートコールマクロライブラリに保存されます。詳細については、*SAS FedSQL Language Reference* にある“Referencing a Macro Variable in a Delimited Identifier”を参照してください。

プロシジャによるマクロ変数セットの使用

PROC FEDSQL は、各ステートメントの実行後、特定の値を使用してマクロ変数を設定します。これらのマクロ変数はマクロ内部でテストし、PROC ステップの実行を継続するかどうかを決定できます。各ステートメントの実行後、次のマクロ変数がそれぞれの値を使用して更新されます。

SQLRC
PROC FEDSQL ステートメントの成功を示す次のステータス値が格納されます。

0
PROC ステートメントはエラーなしで正常に完了しました。

4
PROC ステートメントで警告が発行される状況が発生しました。ステートメントは引き続き実行されます。

8
PROC ステートメントでエラーが発生しました。ステートメントはこの時点で停止されました。

16
PROC ステートメントで実行時エラーが発生しました。たとえば、このエラーコードが使用されるのは、サブクエリ(1 つの値のみを返す)が複数の行に対して評価するときです。このようなエラーが検出されるのは、実行時のみです。

セキュリティ

PROC FEDSQL は、SAS パスワードで保護されたデータセットファイルをサポートします。

Base SAS ソフトウェアを使用して、ファイルに SAS パスワードを割り当てることで、SAS データセットや SPD データセットへのアクセスを制限できます。読み取り、書き込み、変更という、3 つのレベルの保護を 指定できます。

PROC FEDSQL では、FedSQL テーブルオプション(ALTER=、PW=、READ=、および WRITE=)を使用してデータソースのパスワードを割り当てるか指定します。たとえば、次のコードでは、READ、WRITE、および ALTER の各パスワードを SAS データセットに割り当てるために FedSQL テーブルオプション PW=を適用します。

```
libname MyFiles base 'C:\MyFiles\Base';

proc fedsql;
   create table MyFiles.Table1 {options pw=luke}(x double) ;
   insert into MyFiles.Table1 values (1.0);
   insert into MyFiles.Table1 values (2.0);
   insert into MyFiles.Table1 values (3.0);
quit;
```

FedSQL テーブルオプションは、基本 SAS データセットオプションと同じ機能の多くを実行します。ただし、基本 SAS データセットオプションは PROC FEDSQL ステートメントでサポートされません。そのため、PROC FEDSQL でデータソースにアクセスする際、FedSQL テーブルオプションは、パスワードの割り当てまたは指定に使用する必要があります。

SAS パスワードによって、SAS システムの外部にある SAS ファイルへのアクセスは制御されません。SAS の外部にある SAS ファイルへのアクセスを制御するには、オペレーティングシステムで提供されているユーティリティやファイルシステムセキュリティを使用する必要があります。SAS パスワードの詳細については、*SAS 言語リファレンス: 解説編*を参照してください。

FedSQL データ型サポート

PROC FEDSQL では、FedSQL ステートメントを送信する際、すべての FedSQL 言語データ型がサポートされます。FedSQL データ型に関する詳細は、*SAS FedSQL Language Reference* を参照してください。

ただし、基本 SAS セッションでは、PROC FEDSQL を送信しない場合、FedSQL データ型があらかじめ定義されたレガシー SAS データ型(SAS 数値および SAS 文字)に変換されるか、その逆の変換が行われます。たとえば、FedSQL 言語で作成されたテーブルで CONTENTS プロシジャを送信すると、DATE データ型が SAS 数値として報告されます。次の表に、FedSQL データ型と、SAS データ型への変換または SAS データ型からの変換方法を示します。

表 25.1 *FedSQL データ型の変換*

FedSQL データ型	レガシー SAS データ型	説明
BIGINT	SAS 数値	SAS フォーマット 20 を適用します。 SAS 数値が DOUBLE であり、これは正確な数値データ型ではなく概算の数値データ型であるため、精度が失われる可能性があります。
BINARY(*n*)	SAS 文字	SAS フォーマット$n を適用します。
CHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。
DATE	SAS 数値	SAS フォーマット DATE9 を適用します。 有効な SAS 日付値は、1582-01-01 から 9999-12-31 までです。SAS 日付範囲外の日付はサポートされていないため、無効な日付として処理されます。
DECIMAL|NUMERIC(*p*,**s**)	SAS 数値	SAS フォーマット 11.2 を適用します。

FedSQL データ型	レガシー SAS データ型	説明
DOUBLE	SAS 数値	
FLOAT(*p*)	SAS 数値	
INTEGER	SAS 数値	SAS フォーマット 11 を適用します。
NCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。
NVARCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。
REAL	SAS 数値	
SMALLINT	SAS 数値	SAS フォーマット 6 を適用します。
TIME(*p*)	SAS 数値	SAS フォーマット TIME8 を適用します。
TIMESTAMP(*p*)	SAS 数値	SAS フォーマット DATETIME19.2 を適用します。
TINYINT	SAS 数値	SAS フォーマット 4 を適用します。
VARBINARY(*n*)	SAS 文字	SAS フォーマット$n を適用します。
VARCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。

例: FEDSQL プロシジャ

例 1: SAS データセットの作成

要素: PROC FEDSQL ステートメント
QUIT statement

他の要素: LIBNAME statement
FedSQL 言語ステートメント
PROC CONTENTS

詳細

この例では、FEDSQL プロシジャを送信してから FedSQL ステートメントを送信して、基本 SAS セッション内で SAS データセットを作成します。PROC CONTENTS 出力によって、SAS データセットのコンテンツがリストで表示されます。

プログラム

```
libname MyFiles base 'C:\My Documents';

proc fedsql;


create table myfiles.customer
   ( id double primary key,
     name char(16),
     address char(64),
     city char(16),
     state char(2),
     country char(16),
     homephone char(16),
     workphone char(16),
     cellphone char(16),
     initorder double having format date9. label 'Initial Order'
   );

quit;

proc contents data=myfiles.customer;
run;
```

プログラムの説明

ライブラリ参照を作成する SAS データセットに割り当てます。 LIBNAME ステートメントは libref MyFiles を割り当て、BASE エンジンを指定し、SAS データセットの物理的場所を指定します。

```
libname MyFiles base 'C:\My Documents';
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントによって、libref 属性からデータソースへの接続文字列が生成され、FedSQL ステートメントを送信するための環境が設定されます。

```
proc fedsql;
```

FedSQL ステートメントを入力します。 FedSQL ステートメントによって、MyFiles.Customer という名前の SAS データセットが作成されます。FedSQL CREATE TABLE ステートメントの 2 レベルの名前によって、カタログ識別子 MyFiles (割り当てられた libref)が指定されます。

```
create table myfiles.customer
   ( id double primary key,
     name char(16),
     address char(64),
     city char(16),
     state char(2),
     country char(16),
     homephone char(16),
```

```
    workphone char(16),
    cellphone char(16),
    initorder double having format date9. label 'Initial Order'
  );
```

プロシジャを停止します。 QUIT ステートメントは、プロシジャを終了します。

```
quit;
```

SAS データセットのコンテンツがリストで表示されます。 CONTENTS プロシジャによって、SAS データセットのコンテンツが記述されます。

```
proc contents data=myfiles.customer;
run;
```

出力:SAS データセットの作成

アウトプット 25.1 *MyFiles.Customer の PROC CONTENTS 出力*

The SAS System

The CONTENTS Procedure

Data Set Name	MYFILES.CUSTOMER	Observations	0
Member Type	DATA	Variables	10
Engine	BASE	Indexes	1
Created	09/26/2012 15:02:10	Integrity Constraints	1
Last Modified	09/26/2012 15:02:10	Observation Length	184
Protection		Deleted Observations	0
Data Set Type		Compressed	NO
Label		Sorted	NO
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	16384
Number of Data Set Pages	2
First Data Page	1
Max Obs per Page	88
Obs in First Data Page	0
Index File Page Size	4096
Number of Index File Pages	2
Number of Data Set Repairs	0
Filename	C:\My Documents\customer.sas7bdat
Release Created	5.1TK
Host Created	W32_7

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
3	ADDRESS	Char	64		
9	CELLPHONE	Char	16		
4	CITY	Char	16		
6	COUNTRY	Char	16		
7	HOMEPHONE	Char	16		
1	ID	Num	8		
10	INITORDER	Num	8	DATE9.	Initial Order
2	NAME	Char	16		
5	STATE	Char	2		
8	WORKPHONE	Char	16		

Alphabetic List of Integrity Constraints

#	Integrity Constraint	Type	Variables
1	_PK0001_	Primary Key	ID

Alphabetic List of Indexes and Attributes

#	Index	Unique Option	Owned by IC	# of Unique Values
1	ID	YES	YES	0

例 2: 連結クエリによる複数のデータソースへのアクセス

要素: PROC FEDSQL ステートメント
QUIT statement

他の要素: LIBNAME statement
FedSQL 言語ステートメント

詳細

この例では、PROC FEDSQL および FedSQL SELECT ステートメントを使用することで、SAS データセットと SPD データセットの両方に同時にアクセスします。PROC FEDSQL は、2 つのデータソースへの接続に両方の libref の属性を使用します。SAS データセット MyBase.Product には、ProdId 列と Product 列が含まれます。SPD データセット MySpde.Sales には、ProdId 列、Total 列、および Country 列が含まれます。連結クエリは、複数のデータソースのデータにアクセスして 1 つの結果セットを返すクエリです。

プログラム

```
libname mybase base 'C:\Base';
libname myspde spde 'C:\Spde';

proc fedsql;

   select * from mybase.product, myspde.sales;

quit;
```

プログラムの説明

2 つのライブラリ参照を割り当てます。 1 つ目の LIBNAME ステートメントは libref MyBase を割り当て、BASE エンジンを指定し、SAS データセットの物理的な場所を指定します。2 つ目の LIBNAME ステートメントは libref MySpde を割り当て、SPD エンジンを指定し、SPD データセットの物理的な場所を指定します。

```
libname mybase base 'C:\Base';
libname myspde spde 'C:\Spde';
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントは、libref 属性からデータソースに接続します。

```
proc fedsql;
```

両方のテーブルのデータを取得します。 SELECT ステートメントは両方のテーブルから行と列を取得します。複数のテーブルが FROM 句にリストで表示される場合、それらが処理されて 1 つの結果セットが作成されます。

```
   select * from mybase.product, myspde.sales;
```

プロシジャを停止します。

```
quit;
```

出力:連結クエリによる複数のデータソースへのアクセス

アウトプット 25.2 *複数のデータソースの SELECT ステートメント出力*

The SAS System

prodid	product	prodid	total	country
1424	Rice	3234	$54,000	United States
3421	Corn	3234	$54,000	United States
3234	Wheat	3234	$54,000	United States
3485	Oat	3234	$54,000	United States
1424	Rice	1424	$75,384	Japan
3421	Corn	1424	$75,384	Japan
3234	Wheat	1424	$75,384	Japan
3485	Oat	1424	$75,384	Japan
1424	Rice	3421	$19,234	Japan
3421	Corn	3421	$19,234	Japan
3234	Wheat	3421	$19,234	Japan
3485	Oat	3421	$19,234	Japan
1424	Rice	3421	$39,483	United States
3421	Corn	3421	$39,483	United States
3234	Wheat	3421	$39,483	United States
3485	Oat	3421	$39,483	United States
1424	Rice	3975	$94,823	Argentina
3421	Corn	3975	$94,823	Argentina
3234	Wheat	3975	$94,823	Argentina
3485	Oat	3975	$94,823	Argentina

例 3: 既存のテーブルからのテーブルの作成

要素: PROC FEDSQL ステートメント
QUIT statement

他の要素: LIBNAME statement
FedSQL 言語ステートメント

詳細

この例では、AS クエリ式構文で PROC FEDSQL および CREATE TABLE ステートメントを使用して、既存のテーブルからテーブルを作成します。クエリ式は、新しいテーブルを作成するために既存のテーブルから行を選択します。

プログラム

```
libname mybase base 'C:\Base';
libname myspde spde 'C:\Spde';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;

proc fedsql;

   create table mybase.results as
      select product.prodid, product.product, customer.name,
         sales.totals, sales.country
      from myspde.product, myoracle.sales, myoracle.customer
      where product.prodid = sales.prodid and
         customer.custid = sales.custid;

   select * from mybase.results;

quit;
```

プログラムの説明

---

3 つのライブラリ参照を割り当てます。 1 つ目の LIBNAME ステートメントは libref MyBase を割り当て、BASE エンジンを指定し、作成する SAS データセットの物理的な場所を指定します。2 つ目の LIBNAME ステートメントは libref MySpde を割り当て、SPD エンジンを指定し、既存の SPD データセットの物理的な場所を指定します。3 つ目の LIBNAME ステートメントは libref MyOracle を割り当て、Oracle エンジンを指定し、既存の Oracle テーブルを含む Oracle データベースへの接続情報を指定します。

```
libname mybase base 'C:\Base';
libname myspde spde 'C:\Spde';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントは、libref 属性からデータソースに接続します。

```
proc fedsql;
```

新しいテーブルを作成します。 CREATE TABLE ステートメントは、クエリ式を使用して既存のテーブルから行を選択して、既存のテーブルから SAS データセットを作成します。SELECT ステートメントは既存のテーブルから適格な列と行を取得し、新しいテーブルを作成します。FedSQL はクエリを複数のクエリに分割し、Oracle テーブルで個別のクエリを Oracle データベースに受け渡し、結合を実行します。結果セットは FedSQL に返されます。FedSQL は Oracle 結果セットを SPD データセットに結合し、SAS データセット(Results)を作成します。

```
   create table mybase.results as
      select product.prodid, product.product, customer.name,
         sales.totals, sales.country
      from myspde.product, myoracle.sales, myoracle.customer
      where product.prodid = sales.prodid and
         customer.custid = sales.custid;
```

SAS データセット内のデータを取得します。 SELECT ステートメントは、Results という名前の SAS データセットから行と列を取得します。

```
select * from mybase.results;
```

プロシジャを停止します。

```
quit;
```

出力:既存のテーブルからのテーブルの作成

アウトプット 25.3 *結果テーブルの SELECT ステートメント*

The SAS System

PRODID	PRODUCT	NAME	TOTALS	COUNTRY
3234	Rice	Peter Frank	189400	United States
3422	Oat	Jim Stewart	2789654	United States
1424	Corn	Janet Chien	555789	Japan
3421	Wheat	Qing Ziao	781183	Japan
3975	Barley	Humberto Sertu	899453	Argentina

例 4: 相関サブクエリを使用したデータのクエリ

要素: PROC FEDSQL ステートメント
QUIT statement

他の要素: LIBNAME statement
FedSQL 言語 SELECT ステートメント

詳細

この例は、相関サブクエリを使用したデータのクエリを示しています。相関サブクエリでは、サブクエリ内の WHERE 句が外部クエリ内のテーブルの値を参照します。相関サブクエリは、外部クエリ内の各行について評価されます。相関サブクエリを使用すると、FedSQL はサブクエリと外部クエリを同時に実行します。FedSQL は、異種相関サブクエリを実行できます。FedSQL は、サブクエリがデータソースによって実行されるよう命令し、データソースから転送される結果セットを制限します。

プログラム

```
libname myspde spde 'C:\Spde';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;

proc fedsql;

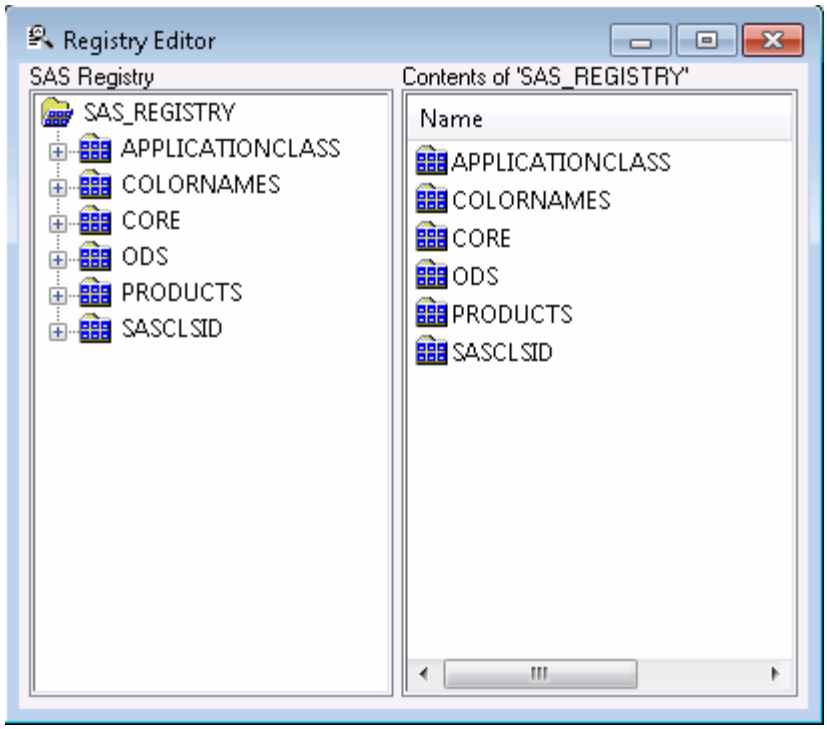
   select * from myspde.product
```

```
      where exists (select * from myoracle.sales
      where product.prodid=sales.prodid);

quit;
```

プログラムの説明

2 つのライブラリ参照を割り当てます。 1 つ目の LIBNAME ステートメントは libref MySpde を割り当て、SPD エンジンを指定し、SPD データセットの物理的な場所を指定します。2 つ目の LIBNAME ステートメントは libref MyOracle を割り当て、Oracle エンジンを指定し、Oracle データベースへの接続情報を指定します。

```
libname myspde spde 'C:\Spde';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントは、libref 属性からデータソースに接続します。

```
proc fedsql;
```

相関クエリを送信します。 FedSQL は、サブクエリ WHERE 式が Oracle データベースによって評価されるよう命令します。

```
   select * from myspde.product
      where exists (select * from myoracle.sales
      where product.prodid=sales.prodid);
```

プロシジャを停止します。

```
quit;
```

例 5: 結合を実行するための DBMS インデックスの作成および使用

要素: PROC FEDSQL ステートメント
QUIT statement

他の要素: LIBNAME statement
FedSQL 言語ステートメント

詳細

この例では、Oracle テーブルのインデックスを作成し、そのインデックスを使用して Oracle テーブルと SPD データセットの結合を実行する方法を示します。

プログラム

```
libname myspde spde 'C:\Spde';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;

proc fedsql;

   create index prodid on myoracle.sales (prodid);

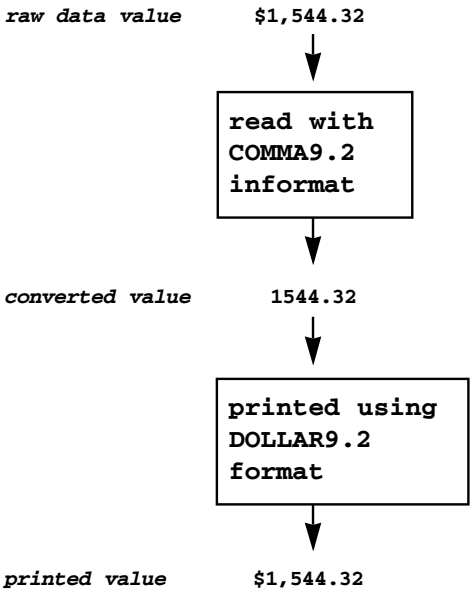
   select * from myspde.product, myoracle.sales
      where product.prodid=sales.prodid;

quit;
```

プログラムの説明

2 つのライブラリ参照を割り当てます。 1 つ目の LIBNAME ステートメントは libref MySpde を割り当て、SPD エンジンを指定し、SPD データセットの物理的な場所を指定します。2 つ目の LIBNAME ステートメントは libref MyOracle を割り当て、Oracle エンジンを指定し、Oracle データベースへの接続情報を指定します。

```
libname myspde spde 'C:\Spde';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントは、libref 属性からデータソースに接続します。

```
proc fedsql;
```

Oracle テーブルのインデックスを作成します。 CREATE INDEX ステートメントによって、Sales という名前の Oracle テーブルの ProdId 列で ProdId という名前のインデックスが作成されます。

```
create index prodid on myoracle.sales (prodid);
```

列と行を取得します。 SELECT ステートメントは、Product という名前の SPD データセットと Sales という名前の Oracle テーブルからデータを取得します。インデックスが Oracle データベース内にある場合でも、FedSQL はインデックスを活用して結合を実行できます。

```
select * from myspde.product, myoracle.sales
   where product.prodid=sales.prodid;
```

プロシジャを停止します。

```
quit;
```

26 章

FONTREG プロシジャ

概要: FONTREG プロシジャ

FONTREG プロシジャを使用して、SAS レジストリを更新し、SAS 出力で使用できるシステムフォントを含めることができます。PROC FONTREG は FreeType フォントレンダリングを使用して、さまざまな種類のフォント定義を認識し、組み込みます。SAS での組み込みと使用が可能なフォントの種類は、このドキュメントでは一括して FreeType ライブラリのフォントと呼ばれます。

注: システムフォントを SAS レジストリに含めるということは、SAS でフォントファイルの検索場所が認識されるということです。フォントファイルは、フォントが SAS プログラムで呼び出されるまで実際に使用されません。そのため、フォントを SAS レジストリに含めた後は、フォントファイルを移動、削除しないでください。

詳細については、次のソースを参照してください。

- “Specifying Fonts in SAS/GRAPH Programs” (*SAS/GRAPH: Reference*)
- “GDEVICE” (*SAS/GRAPH: Reference*)

- “FONTSLOC= System Option” (*SAS System Options: Reference*) および “SYSPRINTFONT= System Option” (*SAS System Options: Reference*)
- www.freetype.org(FreeType プロジェクトの詳細)

概念: FONTREG プロシジャ

サポート対象のフォントの種類とフォントの命名規則

フォントが SAS レジストリに追加されると、フォント名には山かっこ(< >)で囲まれた 3 文字のタグの接頭辞が付きます。この接頭辞は、フォントの種類を示します。たとえば、TrueType フォントの Arial を SAS レジストリに追加すると、レジストリでの名前は **<ttf> Arial** となります。この命名規則を使用して、同じ名前で種類が異なるフォントの追加と区別が可能です。

フォントを SAS プログラム(TEMPLATE プロシジャ、または REPORT プロシジャの STYLE=オプションなどで)で指定する場合、3 文字のタグを使用して同じ名前のフォントを区別します。

```
proc report data=sashelp.class nowd
            style(header)=[font_face='<ttf> Palatino Linotype'];
run;
```

タグをフォント指定に含めない場合、レジストリでその名前のフォントが検索されます。その名前のフォントが複数見つかった場合、次のテーブルのランクが最も高いフォントが使用されます。

表 26.1 *サポートされるフォントの種類*

ランク	Type	タグ	ファイル拡張子
1	TrueType	**<ttf>**	**.ttf**
2	Type1	**<at1>**	**.pfa** **.pfb**
3	OpenType	**<cff>**	**.otf**

注: OpenType フォントは TrueType フォントの拡張で、SAS でサポートされています。OpenType には、serif、sans-serif、monospace、symbol フォントのファミリ値が含まれます。OpenType には.otf フォントファイルが登録されます。

注: PDF と PostScript では、2 バイトの Type1 フォントをサポートしていません。

SAS では、FreeType フォントレンダリングが必要な拡張性のないフォントの種類をサポートしていません。有効なフォントとして認識されても、SAS レジストリに追加されません。

主要なベンダで生成されないフォントファイルは信頼性がない可能性があり、場合によっては SAS で使用できないことがあります。

次の SAS 出力メソッドとデバイスドライバで、FreeType フォントレンダリングを使用できます。

SAS/GRAPH GIF、GIFANIM	ユニバーサル PNG
SAS/GRAPH JPEG	ユニバーサル印刷 GIF
SAS/GRAPH PCL	ユニバーサル印刷 TIFF
SAS/GRAPH PNG	ユニバーサル印刷 PCL
SAS/GRAPH SASEMF	ユニバーサル印刷 PDF
SAS/GRAPH SASWMF	ユニバーサル PS
SAS/GRAPH TIFFP、TIFFB	ユニバーサル SVG
ユニバーサル EMF	

PROC FONTREG でフォントを登録する

PROC FONTREG は、フォントを SAS レジストリに登録するために使用されます。たとえば、Windows フォントディレクトリに Type1 フォントまたは OpenType フォントがある場合、次のコードをサブミットして、そのフォントおよび Windows のフォントディレクトリのその他すべてのフォントファイルを登録できます。

```
proc fontreg mode=add;
fontpath '!SYSTEMROOT\fonts';
run;
```

SAS レジストリからフォントを削除する

フォントを SAS レジストリから削除するには、次の方法を使用します。

- SAS レジストリエディタを使用する
- PROC REGISTRY を使用する
- REMOVE ステートメントを PROC FONTREG で使用する

SAS レジストリエディタを使用してフォントを削除するには、**ソリューション** ⇨ **アクセサリ** ⇨ **レジストリエディタ**を選択します。または、コマンドウィンドウまたは **Command ===>**プロンプトに **regedit** を入力します。

次の画面は、SAS **レジストリエディタ**ウィンドウです。

図 26.1 *SAS レジストリエディタウィンドウ*



レジストリエディタウィンドウの左ペインで、[CORE\PRINTING\FREETYPE\FONTS]キーにナビゲートします。削除するフォントを選択し、次の方法のうちいずれかを使用して削除します。

- フォント名を右クリックして、メニューから**削除**を選択します。
- **削除**ボタン  を選択します。
- **編集** ⇨ **削除** ⇨ **キー**を選択します。

PROC REGISTRY を使用してフォントを削除するには、次の例のようなプログラムをサブミットします。この例では、**<ttf> Arial** フォントを削除します。

```
   /* Write the key name for the font to an external file */
proc registry export='external-filename'
              startat='core\printing\freetype\fonts\<ttf> Arial';
run;

   /* Remove the "<ttf> Arial" font from the SAS registry */
proc registry
uninstall='external-filename' fullstatus;
run;
```

REMOVE ステートメントを PROC FONTREG で使用してフォントを削除するには、“REMOVE ステートメント” (826 ページ)を参照してください。

PROC REGISTRY の詳細については、54 章, “REGISTRY プロシジャ,” (1563 ページ)を参照してください。

PROC FONTREG でファイル参照を使用する

最初にファイル名を定義している場合は、PROC FONTREG の FONTPATH、TRUETYPE、TYPE1、OPENTYPE の各ステートメントでファイル参照を使用することができます。ファイル参照の使用例を次に示します。

```
filename fonts1 'c:\windows\fonts';
proc fontreg mode=all;
   fontpath fonts1;
run;

proc fontreg mode=all;
   truetype fonts1;
run;
```

ファイル参照を使用できれば、FILENAME ステートメントとその機能を直接使用できます。たとえば、URL を使用して使用可能なフォントを登録できます。ファイル参照のサポートによって、FILENAME ステートメントと PROC FONTREG ステップを使用できます。

フォントの別名とロケール

FONTFILE、FONTPATH、TRUETYPE、OPENTYPE ステートメントでは、別名とロケールがサポートされています。処理中のフォントに現在の SAS セッションと同じロケールのローカライズされた名前が含まれている場合、ローカライズされた名前の別名がフォントファミリを参照するために SAS レジストリに追加されます。

構文: FONTREG プロシジャ

操作: ステートメントが指定されていない場合、PROC FONTREG は FONTSLOC= SAS システムオプションで示されるディレクトリの TrueType フォントファイルを検索します。

注: 階層ファイルシステム(HFS)を使用しない z/OS サイトの場合、FONTFILE ステートメントのみサポートされます。詳細については、“FONTREG Procedure: z/OS” (*SAS Companion for z/OS*)を参照してください。

ヒント: 2 つ以上のステートメントを指定すると、それらステーメントは出現する順序で実行されます。ただし、REMOVE ステートメントは常に最初に実行されます。単一の PROC FONTREG ステップで同じステートメントを複数回使用できます。

参照項目: “FONTREG Procedure: z/OS” (*SAS Companion for z/OS*)

PROC FONTREG <*option(s)*>;
FONTFILE '*file*' <…'*file*'> || '*file-1*, *pfm-file-1*, *afm-file-1*' <...'*file-n*'>;
FONTPATH <*fileref*> '*directory*' <…'*directory*'>;
OPENTYPE <*fileref*> '*directory*' <…'*directory*'>
REMOVE '*family-name*' | '*alias*' | *family-type* | _ALL_;
TRUETYPE <*fileref*> '*directory*' <…'*directory*'>;
TYPE1 <*fileref*> '*directory*' <…'*directory*'>;

ステートメント	タスク
“PROC FONTREG ステートメント”	新しいフォントと既存のフォントの処理方法を指定します
“FONTFILE ステートメント”	処理するフォントファイルを識別します
“FONTPATH ステートメント”	処理する有効なフォントファイルを識別するディレクトリを検索します
“REMOVE ステートメント”	SAS レジストリの場所 Core\Printing\Freetype\Fonts からフォントファミリ、特定の種類のすべてのフォント、またはすべてのフォントを削除します
“TRUETYPE ステートメント”	TrueType フォントファイルを識別するディレクトリを検索します
“TYPE1 ステートメント”	有効な Type 1 フォントファイルを識別するディレクトリを検索します
“OPENTYPE ステートメント”	有効な OpenType フォントファイルを識別するディレクトリを検索します

PROC FONTREG ステートメント

SAS レジストリを更新し、SAS 出力で使用できるシステムフォントを含めることができます。

構文

PROC FONTREG <*option(s)*>;

オプション引数の要約

MODE=ADD | REPLACE | ALL
: 新しいフォントと既存のフォントの処理方法を指定します。

MSGLEVEL=VERBOSE | NORMAL | TERSE | NONE
: SAS ログに含める詳細のレベルを指定します。

NOUPDATE
: プロシジャが SAS レジストリを更新せずに実行するように指定します。

USESASHELP
: Sashelp ライブラリの SAS レジストリが更新されるように指定します。

オプション引数

MODE=ADD | REPLACE | ALL
SAS レジストリの新しいフォントと既存のフォントの処理方法を指定します。

ADD
: SAS レジストリにまだ存在していないフォントを追加するように指定します。既存のフォントは変更しません。

REPLACE
SAS レジストリにすでに存在するフォントを置き換えるように指定します。新しいフォントは追加しません。

ALL
SAS レジストリに存在していない新しいフォントを追加し、SAS レジストリにすでに存在しているフォントを置き換えるように指定します。

デフォルト ADD

例 “例 3: ディレクトリの既存の TrueType フォントファイルを置換する” (830 ページ)

MSGLEVEL=VERBOSE | NORMAL | TERSE | NONE
SAS ログに含める詳細のレベルを指定します。

VERBOSE
SAS ログメッセージには、追加されたフォント、追加されなかったフォント、認識されなかったフォントが含まれます。ログには、追加されたフォント、追加されなかったフォント、認識されなかったフォントの数を示す概要も含まれています。

NORMAL
SAS ログメッセージには追加されたフォント、および追加されたフォント、追加されなかったフォント、認識されなかったフォントの数を示す概要が含まれます。

TERSE
SAS ログメッセージには、追加されたフォント、追加されなかったフォント、認識されなかったフォントの数を示す概要のみが含まれます。

NONE
エラー以外(発生した場合)のメッセージは SAS ログに書き込まれません。

デフォルト TERSE

例 “例 2: 複数のディレクトリに含まれるすべてのフォントファイルを追加する” (829 ページ)

NOUPDATE
プロシジャが実際に SAS レジストリを更新せずに実行するように指定します。このオプションを使用して、SAS レジストリを変更する前に指定フォントでプロシジャをテストできます。

USESASHELP
Sashelp ライブラリの SAS レジストリが更新されるように指定します。このオプションを使用するには、Sashelp ライブラリへの書き込みアクセス権が必要です。USESASHELP オプションが指定されない場合、Sasuser ライブラリの SAS レジストリが更新されます。

FONTFILE ステートメント

処理するフォントファイルを指定します。

参照項目: “例 1: 単一フォントファイルの追加” (829 ページ)

構文

FONTFILE '*file*' <…'*file*'> || '*file-1*, *pfm-file-1*, *afm-file-1*' <...'*file-n*'>;

必須引数

file
フォントファイルへの完全パス名です。ファイルは、有効なフォントファイルとして認識されると処理されます。パス名はそれぞれ引用符で囲む必要があります。複数のパス名を指定する場合、パス名をスペースで区切る必要があります。

pfm-file
フォントメトリックおよび Windows フォント名の値を含む Windows 固有のファイルを指定します。

afm-file
フォントメトリックを含むファイルを指定します。

詳細

Type1 フォントを処理する

有効な Type1 フォントが TYPE1 または FONTPATH ステートメントで処理される場合、フォントファイルを含む同じディレクトリの対応する PFM または AFM フォントメトリックファイルの検索が試行されます。ファントファイル名の接頭辞が.PFM 拡張子と.AFM 拡張子と使用されて、メトリックファイル名が生成されます。これらのファイルは正常に開かれ、有効なメトリックファイルであることが決定されると、SAS レジストリへの追加時にフォントファミリのフォントと関連付けられます。

FONTFILE ステートメントで Type1 フォントを指定して、PFM ファイルまたは AFM ファイルを指定しない場合、PFM ファイルまたは AFM ファイルは検索されません。

PFM ファイルまたは AFM ファイルを指定する

フォントファイルに Type1 フォントが含まれている場合、その対応する PFM ファイルと AFM ファイルも指定できます。この例のように、ファイルに対してそれぞれ完全ホスト名(ディレクトリとファイル名)を指定する必要があり、すべてのファイルはグループ化して、引用符で囲む必要があります。

```
fontfile 'c:\winnt\fonts\alpinerg.pfb,
          c:\winnt\fonts\alpinerg.pfm,
          c:\winnt\fonts\alpinerg.afm';
```

AFM ファイルを指定して PFM ファイルを指定しない場合、この例のように、欠損している PFM ファイルに対するプレースホルダとしてカンマを使用する必要があります。

```
fontfile 'c:\winnt\fonts\alpinerg.pfb, ,
c:\winnt\fonts\alpinerg.afm';
```

PFM ファイルを指定して AFM ファイルを指定しない場合、この例のように、欠損している AFM ファイルに対するプレースホルダとしてのカンマは不要です。

```
fontfile 'c:\winnt\fonts\alpinerg.pfb,
c:\winnt\fonts\alpinerg.pfm';
```

PFM ファイルまたは AFM ファイルを指定すると、SAS ではファイルを開き、ファイルが指定されている種類のものかどうかを判別しようとします。指定されている種類のものでない場合、メッセージがログに書き込まれ、ファイルは使用されません。

PFM ファイルは Windows 固有のファイルで、フォントメトリックおよび Windows フォント名フィールドの値が含まれます。有効な PFM ファイルを指定すると、ファイルが開かれ、Windows フォント名の値が取得され、それがフォントとともに SAS レジストリに保存されます。このフィールドはファイル(EMF フォーマットファイルなど)の作成時に使用され、Windows アプリケーションにエクスポートされます。

PFM ファイルまたは AFM ファイルを指定しない

FONTFILE ステートメントで PFM ファイルまたは AFM ファイルを Type1 フォントファイルとともに指定する必要はありません。この場合、メトリックファイル情報は SAS レジストリのフォントファミリのフォントに追加されません。複数のスタイルと重みを含む既存のフォントファミリが SAS レジストリにすでに存在し、FONTFILE ステートメントがそのファミリのフォントのいずれかの置換に使用される場合、そのフォントに関するすべての情報が更新されます。置換によりホストファイル名、PFM 名、AFM 名、Windows フォント名も更新されます。

注: ファミリのフォントを置換し、そのフォントに PFM 名または AFM 名の値が含まれている場合、FONTFILE ステートメントでメトリックに対し欠損値または無効値を指定すると、対応するメトリック値がレジストリのフォントから削除されます。

注: TrueType フォントを指定した場合、PFM ファイルまたは AFM ファイルの指定を使用できません。

FONTPATH ステートメント

処理対象の有効なフォントファイルを検索するディレクトリを指定します。

参照項目: "例 2: 複数のディレクトリに含まれるすべてのフォントファイルを追加する" (829 ページ)

構文

FONTPATH *<fileref>* '*directory*' <…'*directory*'>;

必須引数

directory

検索するディレクトリを指定します。有効なフォントファイルとして認識されるすべてのファイルが処理されます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

動作環境の情報

Windows 動作環境でのみ、フォルダの場所が不明な場合は fonts フォルダを検索します。また、フォントの場所がわからなくても、システムフォントを登録できます。この情報を見つけるには、次のプログラムをサブミットします。

```
proc fontreg;
    fontpath "%sysget(systemroot)\fonts";
run;
```

%SYSGET マクロはウィンドウ環境変数 SYSTEMROOT の値を取得し、システムディレクトリの場所に解決します。fonts サブディレクトリは、システムディレクトリの 1 レベル下に置かれます。

オプション引数

fileref

FONTPATH ステートメントで使用するファイル参照を指定します。

REMOVE ステートメント

フォントファミリ、特定の種類(TrueType、Type1 など)のすべてのフォント、またはすべてのフォントを SAS レジストリの場所 Core\Printing\Freetype\Fonts から削除します。

構文

REMOVE '*family-name*' | '*alias*' | *family-type* | _ALL_;

必須引数

family-name
SAS レジストリの Core\Printing\Freetype\Fonts キーから削除するフォントのファミリ名を指定します。値にスペースが含まれている場合は、*family-name* を引用符で囲みます。

alias
family-name に対する代替名(通常は短縮形式)を指定します。値にスペースが含まれている場合は、別名を引用符で囲みます。

注 別名として指定可能な有効値は、SAS レジストリの Core\Printing\Alias\Fonts\Freetype キーにリストされます。

family-type
SAS でサポートされ、SAS レジストリから削除するフォントの種類(TrueType、Type1 など)の名前を指定します。

注: フォントの種類は存在する動作システムの場所からは削除されません。SAS レジストリからのフォントの種類の登録は、SAS でそのフォントが認識されないように削除されます。

ALL
SAS レジストリの Core\Printing\Freetype\Fonts キーのすべてのフォントファミリが削除されるように指定します。

詳細

レジストリからフォントを削除する

REMOVE ステートメントは、SAS レジストリの場所 Core\Printing\Freetype\Fonts からフォントファミリ、特定の種類のすべてのフォント、またはすべてのフォントを削除します。USESASHELP プロシジャオプションを指定すると、フォントはレジストリの Sashelp 部分から削除されます。USESASHELP を指定しない場合、フォントはレジストリの Sasuser 部分から削除されます。デフォルトでは、レジストリの Sasuser 部分から削除されます。

REMOVE ステートメントで *family-name* 引数を指定すると、ファミリ内の個別のフォントではなく、フォントファミリが削除されます。たとえば、Arial ファミリ内で複数のフォントを登録するとします。`REMOVE Arial;` ステートメントを使用すると、Arial ファミリのすべてのフォントがレジストリから削除されます。同様に、*family-type* 引数を指定し、`REMOVE Type1;` ステートメントを使用すると、Type1 フォントファミリがすべてレジストリから削除されます。

フォントの追加順序または削除順序

フォントは、その他のプロシジャステートメントを使用してレジストリで追加または置換される前に、SAS レジストリから削除されます。REMOVE ステートメントは、ステートメントが処理されるとすぐにフォントファミリをレジストリから削除します。その他のフォントステートメント(FONTFILE、FONTPATH、TRUETYPE、TYPE1)は、受け入れられる順序で処理されますが、フォント情報はすべてのステートメントが処理されるまで保存されます。次に、レジストリが更新されます。

REMOVE ステートメントで指定されるフォントを検索する

REMOVE ステートメントで指定する名前が存在しない場合、フォントタグ接頭辞(<ttf>など)が指定された名前に追加され、それが SAS レジストリに存在するかどうかが決定されます。たとえば、Arial を指定すると、<ttf>接頭辞タグが使用され、レジストリから削除できるように TrueType フォントの種類が最初に検索されます。検索が失敗した場合、<at1>接頭辞タグが使用され、レジストリから削除できるように Type1 フォントの種類が検索されます。

フォントファミリを削除できない場合

ALL、*family-type* または *family-name* 引数の情報を処理後、フォントファミリを削除できない場合、SAS レジストリの Core\Printing\Alias\Fonts\Freetype キーが検索され、指定された値が別名かどうかが決定されます。指定された値がこのキーに別名として存在する場合、その別名に対応するフォントファミリが削除され、その別名も削除されます。たとえば、Test の別名が Arial フォントファミリを参照し、**`REMOVE test;`**ステートメントを PROC FONTREG で指定すると、Test は Arial の別名であることが決定されます。Arial フォントファミリが SAS レジストリの Core\Printing\Freetype\Fonts キーから、Test 別名が SAS レジストリの Core\Printing\Alias\Fonts\Freetype キーからそれぞれ削除されます。

この時点でフォントファミリを削除できない場合、REMOVE ステートメントで指定されている値が無効であることを示すメッセージがログに書き込まれます。

TRUETYPE ステートメント

TrueType フォントファイルを検索するディレクトリを指定します。

参照項目: “例 3: ディレクトリの既存の TrueType フォントファイルを置換する” (830 ページ)

構文

TRUETYPE <*fileref*> '*directory*' <…'*directory*'>;

必須引数

directory
検索するディレクトリを指定します。有効な TrueType フォントファイルとして認識されるファイルのみが処理されます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

オプション引数

fileref
TRUETYPE ステートメントで使用するファイル参照を指定します。

TYPE1 ステートメント

有効な Type1 フォントファイルを検索するディレクトリを指定します。

構文

TYPE1 *<fileref>* '*directory*' <…'*directory*'>;

必須引数

directory
検索するディレクトリを指定します。有効な Type1 フォントファイルとして認識されるファイルのみ処理されます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

オプション引数

fileref
TYPE1 ステートメントで使用するファイル参照を指定します。

OPENTYPE ステートメント

有効な OpenType フォントファイルを検索するディレクトリを指定します。

構文

OPENTYPE *<fileref>* '*directory*' <…'*directory*'>;

必須引数

directory
検索するディレクトリを指定します。有効な OpenType フォントファイルとして認識されるファイルのみ処理されます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

オプション引数

fileref
OPENTYPE ステートメントで使用するファイル参照を指定します。

例: FONTREG プロシジャ

例 1: 単一フォントファイルの追加

要素: FONTFILE statement

詳細

この例では、単一フォントファイルを SAS レジストリに追加する方法を示します。FONTFILE ステートメントは、単一フォントファイルへの完全パスを指定します。

プログラム

```
proc fontreg;
   fontfile '<ttf> Arial';
run;
```

ログ

ログ 26.1 *単一フォントファイルの SAS レジストリへの追加*

```
SUMMARY:Files processed:1 Unusable files:0 Files identified as fonts:1 Fonts
that were processed:1 Fonts replaced in the SAS registry:0 Fonts added to the
SAS registry:1 Fonts that could not be used:0 Font Families removed from SAS
registry:0
```

例 2: 複数のディレクトリに含まれるすべてのフォントファイルを追加する

要素: MSGLEVEL=オプション
FONTPATH statement

詳細

この例では、すべての有効なフォントファイルを 2 つの異なるディレクトリから追加する方法と、詳細情報を SAS ログに書き込む方法を示します。

プログラム

```
proc fontreg msglevel=verbose;

fontpath 'your-font-directory-1' 'your-font-directory-2';
run;
```

プログラムの説明

すべての詳細を SAS ログに書き込みます。 MSGLEVEL=VERBOSE オプションは、追加されたフォント、追加されなかったフォント、および認識されなかったフォントファイルに関するすべての詳細を書き込みます。

```
proc fontreg msglevel=verbose;
```

有効なフォントを検索するディレクトリを指定します。 FONTPATH ステートメントでは複数のディレクトリを指定できます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

```
fontpath 'your-font-directory-1' 'your-font-directory-2';
run;
```

ログ

ログ 26.2 *複数のディレクトリに含まれるすべてのフォントファイルの追加機能からのメッセージ*

```
1    proc fontreg msglevel=verbose; 2    fontpath 'your-font-directory-1'
3             'your-font-directory-2'; 4    run; ERROR:FreeType base module
FT_New_Face -- unknown file format.ERROR:A problem was encountered with file
"your-font-directory-2\MODERN.FON".......more log entries .....WARNING:The
"Sasfont" font in file "your-font-directory-2\SAS1252.FON" is non-scalable.
Only scalable fonts are supported.......more log entries .....NOTE:The font
"Albertus Medium" (Style:Regular, Weight:Normal) has been added to the SAS
Registry at [CORE\PRINTING\FREETYPE\FONTS\<ttf>Albertus Medium].Because it is a
TRUETYPE font, it can be referenced as "Albertus Medium" or "<ttf>Albertus
Medium" in SAS.The font resides in file "your-font-
directory-1\albr55w.ttf".......more log entries .....WARNING:The font "Georgia"
(Style:Regular, Weight:Normal) will not be added because it already exists in
the "<ttf>Georgia" font family of the SAS Registry.......more log
entries .....SUMMARY:Files processed:138 Unusable files:3 Files identified as
fonts:135 Fonts that were processed:135 Fonts replaced in the SAS registry:0
Fonts added to the SAS registry:91 Fonts that could not be used:44 Font Families
removed from SAS registry:0 NOTE:PROCEDURE FONTREG used (Total process time):
real time           27.81 seconds cpu time            1.18 seconds
```

例 3: ディレクトリの既存の TrueType フォントファイルを置換する

要素: MODE=オプション
TRUETYPE statement

詳細

この例では、指定されているディレクトリのすべての TrueType フォントを読み込み、SAS レジストリにすでに存在するものを置き換えます。

プログラム

```
proc fontreg mode=replace;

   truetype 'your-font-directory';
run;
```

プログラムの説明

既存するフォントのみ置き換えます。 MODE=REPLACE オプションは、プロシジャのアクションを SAS レジストリですでに定義されているフォントの置換に制限します。新しいフォントは追加されません。

```
proc fontreg mode=replace;
```

TrueType フォントファイルを含むディレクトリを指定します。 TrueType フォントファイルとして認識されないディレクトリのファイルは無視されます。

```
   truetype 'your-font-directory';
run;
```

ログ

ログ 26.3 *ディレクトリの既存の TrueType フォントファイルを置換する*

```
SUMMARY:Files processed:49 Unusable files:3 Files identified as fonts:46 Fonts
that were processed:40 Fonts replaced in the SAS registry:40 Fonts added to the
SAS registry:0 Fonts that could not be used:0 Font Families removed from SAS
registry:0
```

27 章

FORMAT プロシジャ

概要: FORMAT プロシジャ

FORMAT プロシジャの動作について

FORMAT プロシジャを使用して、変数に対し独自の入力形式と出力形式を定義できます。また、入力形式または出力形式を含むカタログの一部を印刷し、入力形式または出力形式の説明を SAS データセットに保存し、SAS データセットを使用して入力形式または出力形式を作成できます。

入力形式と出力形式について

入力形式によって、生データ値の読み込み方法と保存方法が決まります。出力形式によって、変数値の印刷方法が決まります。簡単にするため、このセクションでは入力形式変換、出力形式印刷という用語を使用します。

入力形式と出力形式によって、データの種類(文字または数値)と形式(占めるバイト数、数字の小数点配置、先頭、末尾、埋め込みのブランクやゼロの処理方法など)が認識されます。SAS では、変数の読み込みと書き込みを行うための入力形式と出力形式を提供しています。SAS が提供する入力形式と出力形式の詳細な説明については、*SAS 出力形式と入力形式: リファレンス*を参照してください。

入力形式を使用して、次のことができます。

- 文字列を数字に変換(1 を **YES** に変換するなど)。
- 文字列を異なる文字列に変換(**'YES'**を**'OUI'**に変換するなど)。
- 文字列を数字に変換(**YES** を 1 に変換するなど)。
- 数字を別の数字に変換(0–9 を 1 に、10–100 を 2 に変換するなど)。

注: ユーザー定義の入力形式は文字データ読み込み専用です。文字値を実数値に変換できますが、実数を文字に変換することはできません。

出力形式を使用して、次のことができます。

- 数値を文字値として印刷(1 を **MALE** として、2 を **FEMALE** として変換するなど)。
- 文字列を異なる文字列として印刷(**YES** を **OUI** として変換するなど)。
- テンプレートを使用して数値を印刷(9458763450 を **945-876-3450** として印刷するなど)。

変数への出力形式と入力形式の関連付け法

次の図に、入力形式および出力形式と変数を関連付けた場合を要約します。COMMA*w.d* 入力形式と DOLLAR*w.d* 出力形式は、SAS によって提供されています。

図 27.1 変数への入力形式と出力形式の関連付け



図では、ドル記号とカンマを含む生データ値が読み込まれます。COMMA9.2 入力形式はドル記号とカンマを無視して、値を 1544.32 に変換します。DOLLAR9.2 出力形式は、ドル記号とカンマを含めて値を印刷します。変数への入力形式と出力形式の関連付けの詳細については、“変数に入力形式と出力形式を関連付ける ” (835 ページ)を参照してください。

概念:FORMAT プロシジャ

変数に入力形式と出力形式を関連付ける

変数への入力形式と出力形式の関連付け法

次の表に、変数に入力形式と出力形式を関連付けるためのさまざまな方法を要約します。

表 27.1 変数に入力形式と出力形式を関連付ける

ステップ	入力形式	出力形式
DATA ステップ	ATTRIB ステートメントまたは INFORMAT ステートメントを使用して、入力形式と変数を永久に関連付けます。INPUT 関数または INPUT ステートメントを使用して、DATA ステップの期間に対してのみ入力形式と変数を関連付けます。	ATTRIB ステートメントまたは FORMAT ステートメントを使用して、出力形式と変数を永久に関連付けます。PUT 関数または PUT ステートメントを使用して、DATA ステップの期間に対してのみ出力形式と変数を関連付けます。

ステップ	入力形式	出力形式
PROC ステップ	ATTRIB ステートメントと INFORMAT ステートメントは、Base SAS プロシジャで有効です。ただし Base SAS ソフトウェアでは、データがすでに SAS 変数に読み込まれているため、通常は PROC ステップで入力形式を割り当てません。	ATTRIB ステートメントまたは FORMAT ステートメントを使用して、出力形式と変数を関連付けます。出力データセットを作成するプロシジャでいずれかのステートメントを使用する場合、出力形式は入力データセットの変数と永久に関連付けられます。出力データセットを作成しない、または既存データセットを変更しないプロシジャでいずれかのステートメントを使用する場合、ステートメントは PROC ステップの期間に対してのみ出力形式と変数を関連付けます。

FORMAT ステートメントと PROC FORMAT の違い

FORMAT ステートメントと FORMAT プロシジャを混同しないようにしてください。FORMAT ステートメントと INFORMAT ステートメントは既存する出力形式と入力形式(標準 SAS またはユーザー定義)を 1 つ以上の変数と関連付けます。PROC FORMAT は、ユーザー定義の出力形式または入力形式を作成します。

変数への出力形式と入力形式の割り当て

変数への独自の出力形式または入力形式の割り当ては、次の 2 ステップのプロセスからなります。

1. 出力形式または入力形式を FORMAT プロシジャを使用して作成
2. 出力形式または入力形式を ATTRIB ステートメント、FORMAT ステートメント、INFORMAT ステートメント、または INPUT 関数、PUT 関数を使用して割り当て

ATTRIB、INFORMAT、FORMAT ステートメントに関する完全ドキュメントについては、*SAS ステートメント: リファレンス*を参照してください。INPUT および PUT 関数に関する完全なドキュメントについては、*SAS 関数と CALL ルーチン: リファレンス*を参照してください。Base SAS プロシジャでの出力形式の使用に関する詳細および例については、“フォーマットされた値” (58 ページ) を参照してください。

入力形式と出力形式の保存

出力形式カタログ

PROC FORMAT は、ユーザー定義の入力形式と出力形式を SAS カタログのエントリとして保存します。[1] の SAS ファイルに関するセクションを参照してください。LIBRARY=オプションを PROC FORMAT ステートメントで使用して、カタログを指定します。出力形式と入力形式は Work.Formats カタログに保存されます。LIBRARY=オプションを省略すると、LIBRARY=*libref* を指定し、カタログ名を指定しない場合、出力形式と入力形式は *libref* カタログに保存されます。この 1 レベル名の使用は、SAS 以外での 1 レベル名の使用とは異なります。LIBRARY=オプションでは 1 ラベル名はライブラリを示し、SAS 以外では 1 ラベル名は WORK ライブラリのファイルを示します。

カタログエントリの名前は、出力形式または入力形式の名前です。エントリの種類は、次のとおりです。

- FORMAT (数値出力形式)

1 カタログは SAS ファイルの種類で、SAS ライブラリにあります。SAS ファイルの種類または SAS ライブラリ構造が不明な場合は、*SAS 言語リファレンス: 解説編*

- FORMATC (文字出力形式)
- INFMT (数値入力形式)
- INFMTC (文字入力形式)

一時入力形式と一時出力形式

入力形式と出力形式は、WORK ライブラリのカタログに保存されているときは一時のものです。LIBRARY=オプションを省略すると、PROC FORMAT は入力形式と出力形式を一時カタログ Work.Formats に保存します。一時入力形式と一時出力形式は、それらが作成されたのと同じ SAS セッションまたはジョブでのみ取得できます。一時出力形式と一時入力形式を取得するには、適切な SAS ステートメントに出力形式または入力形式の名前を含めるだけです。SAS では自動的に Work.Formats カタログの出力形式または入力形式が検索されます。

永久入力形式と永久出力形式

ある SAS ジョブまたはセッションで作成された出力形式または入力形式を後続のジョブまたはセッションで使用する場合、その出力形式または入力形式を永久に SAS カタログに保存する必要があります。

PROC FORMAT ステートメントの LIBRARY=オプションを使用して、入力形式と出力形式を永久に保存します。LIBRARY=オプションの説明(PROC FORMAT ステートメント) (840 ページ)を参照してください。

永久入力形式と永久出力形式へのアクセス

入力形式または出力形式を永久に保存した後、それを後の SAS セッションまたはジョブで使用できます。後の SAS セッションまたはジョブで永久入力形式または永久出力形式と変数を関連付ける場合、SAS でその入力形式と出力形式にアクセスできる必要があります。そのため、LIBNAME ステートメントを使用して、その入力形式または出力形式を保存するカタログを保存するライブラリにライブラリ参照名を割り当てる必要があります。

SAS では、ユーザー定義の出力形式と入力形式の検索時に 2 つのうち 1 つの方法が使用されます。

- デフォルトで、FORMATS カタログに対し Library ライブラリ参照名によって参照されるライブラリが常に検索されます。出力形式カタログが 1 つしかない場合、次を行います。
  1. PROC FORMAT ステップを実行している SAS セッションで Library ライブラリ参照名を SAS ライブラリに割り当てます。
  2. PROC FORMAT ステートメントでオプション **`library=library`** を指定します。PROC FORMAT は、そのステップで定義される入力形式と出力形式を Library.Formats カタログに保存します。
  3. ユーザー定義の出力形式と入力形式を使用する SAS プログラムでは、LIBNAME ステートメントを含め、Library ライブラリ参照名を永久出力形式カタログを含むライブラリに割り当てます。
- 複数の出力形式カタログがある場合、または出力形式カタログの名前が Formats 以外の場合、次を行います。
  1. PROC FORMAT ステップを実行している SAS セッションで SAS ライブラリにライブラリ参照名を割り当てます。
  2. PROC FORMAT ステップで、オプション **`library=`*`libref`*** または **`library=`*`libref.catalog`*** を指定します。*libref* は、ステップ 1 で割り当てたライブラリ参照名です。

3. ユーザー定義の出力形式と入力形式を使用する SAS プログラムで、FMTSEARCH=オプションを OPTIONS ステートメントで使用し、*libref* または *libref.catalog* を出力形式カタログリストに含めます。

検索対象となる出力形式カタログリストを指定するための構文は、次のとおりです。

OPTIONS FMTSEARCH=(*catalog-specification-1*<*catalog-specification-2 …* >);

catalog-specification にはそれぞれ *libref* または *libref.catalog* が可能です。*libref* だけが指定される場合、カタログ名が Formats とみなされます。

出力形式または入力形式の検索時は、最初に Work.Formats、次に Library.Formats で検索が行われます。これは、これらのうちいずれも FMTSEARCH=リストに表示されない場合です。SAS は、出力形式または入力形式が見つかるまで FMTSEARCH=リストのカタログをリストされている順序で検索します。

詳細については、“FMTSEARCH= System Option” (*SAS System Options: Reference*)を参照してください。LIBRARY=オプションと FMTSEARCH=オプションを一緒に使用する例については、“例 13: 文字列の範囲の書き出し” (911 ページ)を参照してください。

入力形式と出力形式の欠損

SAS で見つからなかった入力形式または出力形式を参照する場合、エラーメッセージが表示され、SAS システムオプション NOFMTERR が有効でない限り処理は停止します。NOFMTERR が有効な場合、SAS では *w.*または$*w.*デフォルト出力形式を使用して、見つからない出力形式で変数の値を印刷します。たとえば、NOFMTERR を使用するには、次の OPTIONS ステートメントを使用します。

```
options nofmterr;
```

詳細については、“FMTERR System Option” (*SAS System Options: Reference*)を参照してください。

SAS でユーザー定義出力形式を使用してフォーマットする欠損変数が見つかり、MISSING=システムオプションによって欠損値に対して印刷する文字を定義する場合、欠損値は次のように決定されます。

- ユーザー定義の出力形式または入力形式に欠損値に対する value-range-set がある場合、欠損値はユーザー定義の出力形式によって定義されます。
- ユーザー定義の出力形式に欠損値に対する value-range-set が定義されていない場合、欠損値は MISSING=システムオプションによって定義されます。MISSING=システムオプションのデフォルト値は、(ピリオド)です。

入力形式と出力形式の印刷

FMTLIB オプション使用時に提供される出力は、入力形式値と出力形式値の簡単なビューを表すためのものです。

FMTLIB オプションを使用する代わりに、CNTLOUT=オプションを使用して、入力形式と出力形式に関する情報を保存する出力データセットを作成します。次に PROC PRINT または PROC REPORT を使用して、データセットを印刷します。この場合、ラベルは切り捨てられません。

注: データセットオプションを使用して、参照を保持するか、CNTLOUT=オプションを使用して追加された追加変数にドロップすることができます。

バイナリ検索による値に対するユーザー定義出力形式または入力形式の決定

SAS では、バイナリ検索を使用して、値に使用するユーザー定義出力形式または入力形式を決定します。それに対し、IF-THEN/ELSE ステートメントの使用は、基本的に値の順次検索になります。

次に示すのは、PROC FORMAT を使用して作成できるいくつかのユーザー定義出力形式です。

```
1='Yes'
2='No'
3='Possibly'
```

1 回の比較で、1 つの値がフォーマットされます。

次の IF-THEN/ELSE ステートメントを使用した場合も、1 回の比較で 1 つの値がフォーマットされます。

```
if x=1 then label='Yes';
else if x=2 then label='No';
else if x=3 then label='Possibly';
```

実行する比較の数が増える場合、バイナリ検索の方が効率が上がります。比較の数が多くなるほど、バイナリ検索の効率も上がります。

構文: FORMAT プロシジャ

制限事項: SELECT ステートメントと EXCLUDE ステートメントを同じ PROC FORMAT ステップ内で使用することはできません。

ヒント: 適切なグローバルステートメントをこのプロシジャと使用することもできます。リストについては、“グローバルステートメント” (24 ページ) を参照してください。

参照項目: “FORMAT Procedure: z/OS” (*SAS Companion for z/OS*)

PROC FORMAT <*option(s)*>;
  EXCLUDE *entry(s)*;
  INVALUE <$>*name* <(*informat-option(s)*)> <*value-range-set(s)*>;
  PICTURE *name* <(*format-option(s)*)>
    <*value-range-set-1* <(*picture-1-option(s)*)>
    <*value-range-set-2* <(*picture-2-option(s)*)>> …>;
  SELECT *entry(s)*;
  VALUE <$>*name* <(*format-option(s)*)> <*value-range-set(s)*>;

ステートメント	タスク	例
“PROC FORMAT ステートメント”	変数に対して出力形式と入力形式を定義する	Ex. 2, Ex. 10, Ex. 11, Ex. 12
“EXCLUDE ステートメント”	カタログエントリを FMTLIB オプションと CNTLOUT=オプションによる処理から除外する	

ステートメント	タスク	例
“INVALUE ステートメント”	生データ値を読み込み、変換するための入力形式を作成する	Ex. 9
“PICTURE ステートメント”	数字を印刷するためのテンプレートを作成する	Ex. 2, Ex. 4, Ex. 14
“SELECT ステートメント”	FMTLIB オプションと CNTLOUT=オプションによって処理するためのカタログエントリを選択する	Ex. 11
“VALUE ステートメント”	変数値の印刷に使用する文字列を指定する出力形式を作成する	Ex. 6, Ex. 8, Ex. 13

PROC FORMAT ステートメント

変数に対するユーザー指定の出力形式と入力形式を作成します。

ヒント: データセットオプションを CNTLIN=および CNTLOUT=データセットオプションと使用できます。リストについては、“データセットオプション” (23 ページ)を参照してください。

例: “例 2: ピクチャ形式の作成” (886 ページ)
“例 10: データセットからの出力形式の作成” (903 ページ)
“例 11: 入力形式と出力形式の説明の印刷” (907 ページ)
“例 12: (永久)出力形式の読み込み” (908 ページ)

構文

PROC FORMAT <*option(s)*>;

オプション引数の要約

CNTLIN=*input-control-SAS-data-set*
PROC FORMAT が入力形式または出力形式を作成する SAS データセットを指定します。

CNTLOUT=*output-control-SAS-data-set*
LIBRARY=オプションで指定したカタログに含まれる入力形式または出力形式に関する情報を保存する SAS データセットを作成します。

FMTLIB
LIBRARY=オプションで指定されるカタログの入力形式または出力形式に関する情報を印刷します。

LIBRARY=*libref*<*.catalog*>
PROC FORMAT ステップで作成中の入力形式または出力形式を含む SAS ライブラリまたはカタログを指定します。

LOCALE
現在の SAS ロケールに対応する出力形式カタログの作成を指定します。

MAXLABLEN=*number-of-characters*
PROC FORMAT 出力で表示される入力形式が適用された値または出力形式が適用された値の文字数を指定します。

MAXSELEN=*number-of-characters*

PROC FORMAT 出力で表示される開始値と終了値の文字数を指定します。

NOREPLACE
新しい入力形式または出力形式が同じ名前の既存する入力形式または出力形式を置き換えないようにします。

PAGE
カタログ内のそれぞれの出力形式と入力形式に関する情報を印刷します。

オプション引数

CNTLIN=*input-control-SAS-data-set*
PROC FORMAT が入力形式または出力形式を作成する SAS データセットを指定します。

CNTLIN=は、VALUE、PICTURE または INVALUE ステートメントを使用せずに出力形式と入力形式を作成します。1 レベル名を指定する場合、LIBRARY=オプションを指定するかどうかに関係なく、プロシジャはデータセットに対しデフォルトのライブラリ(WORK ライブラリまたは USER ライブラリ)だけを検索します。

注 LIBRARY=は、ライブラリまたはカタログを示します。ライブラリ参照名だけが指定されると、カタログ名 FORMATS が使用されます。

ヒント 入力制御データセットの共通ソースは、別の PROC FORMAT ステップの CNTLOUT=オプションからの出力です。

参照項目 “入力制御データセット” (881 ページ)

例 “例 10: データセットからの出力形式の作成” (903 ページ)

CNTLOUT=*output-control-SAS-data-set*
LIBRARY=オプションで指定したカタログに含まれる入力形式または出力形式に関する情報を保存する SAS データセットを作成します。CNTLOUT=オプションが記述されるのと同じステップで入力形式または出力形式を作成している場合、作成している入力形式または出力形式は CNTLOUT=データセットに含まれます。

1 レベル名を指定する場合、LIBRARY=オプションを指定するかどうかに関係なく、プロシジャはデータセットをデフォルトのライブラリ(WORK ライブラリまたは USER ライブラリ)に保存します。

注 LIBRARY=は、ライブラリまたはカタログを示します。ライブラリ参照名だけが指定されると、カタログ名 FORMATS が使用されます。

ヒント 後続の PROC FORMAT ステップで出力制御データセットを入力制御データセットとして使用できます。

参照項目 “出力制御データセット” (878 ページ)

FMTLIB
LIBRARY=オプションで指定されるカタログの入力形式または出力形式に関する情報を印刷します。特定の入力形式または出力形式に関する情報を取得するには、SELECT ステートメントまたは EXCLUDE ステートメントを使用してカタログをサブセットします。

操作 PAGE オプションにより FMTLIB が起動します。

ヒント FMTLIB からの出力が ODS LISTING 出力先で適切にフォーマットされていない場合、LINESIZE=システムオプションの値を増やします。

SELECT ステートメントまたは EXCLUDE ステートメントを使用し、FMTLIB オプションと CNTLOUT=オプションを省略する場合、プロシジャは FMTLIB オプションを起動し、FMTLIB オプション出力が行われます。

例 “例 11: 入力形式と出力形式の説明の印刷” (907 ページ)

LIBRARY=*libref*<.*catalog*>
PROC FORMAT ステップで作成中の入力形式または出力形式を含む SAS ライブラリまたはカタログを指定します。プロシジャは、これらの入力形式と出力形式を後続の SAS セッションまたはジョブで使用できるように、指定するカタログに保存します。

別名 LIB=

デフォルト LIBRARY=オプションを省略すると、出力形式と入力形式は Work.Formats カタログに保存されます。LIBRARY=オプションを指定し、*catalog* に名前を指定しない場合、出力形式と入力形式は *libref*.FORMATS カタログに保存されます。

注 LIBRARY=は、ライブラリまたはカタログを示します。ライブラリ参照名だけが指定されると、カタログ名 FORMATS が使用されます。

ヒント SAS では Library.Formats が自動的に検索されます。出力形式カタログに対し LIBRARY ライブラリ参照名を定義し使用することがあります。

FMTSEARCH=システムオプションを使用して、出力形式カタログの検索順序を制御できます。詳細については、“FMTSEARCH= System Option” (*SAS System Options: Reference*)を参照してください。

参照項目 “入力形式と出力形式の保存 ” (836 ページ)

例 “例 2: ピクチャ形式の作成” (886 ページ)

LOCALE
現在の SAS ロケールに対応する出力形式カタログの作成を指定します。SAS が作成するカタログの名前は、LIBRARY=オプションで指定された SAS ライブラリまたはカタログに、現在の SAS のロケールを表す 5 文字の POSIX ロケールが追加された名前です。

参照項目 POSIX ロケール値のリストについては、“LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

例 SAS ロケールが German_Germany の場合、POSIX ロケール値は de_DE です。次の PROC FORMAT ステートメントを使用して、SAS はカタログ mylib.formats_de_DE を作成し、このプロシジャによって作成した出力形式と入力形式を保存します。

```
proc format locale lib=mylib.formats;
```

MAXLABLEN=*number-of-characters*
CNTLOUT=データセットまたは FMTLIB オプションの出力で表示する入力形式が適用された値または出力形式が適用された値の文字数を指定します。FMTLIB オ

プションは、入力形式が適用された値または出力形式が適用された値に対し最大 40 文字を印刷します。

MAXSELEN=*number-of-characters*
CNTLOUT=データセットまたは FMTLIB オプションの出力で表示する開始値と終了値の文字数を指定します。FMTLIB オプションは、開始値と終了値に対し最大 16 文字を印刷します。

NOREPLACE
新しい入力形式または出力形式が同じ名前の既存する入力形式または出力形式を置き換えないようにします。NOREPLACE を省略すると、その入力形式または出力形式がすでに存在することがプロシジャによって警告され、それらが置き換えられます。

注 出力形式と入力形式を同じ名前にすることができます。

PAGE
カタログ内のそれぞれの出力形式と入力形式に関する情報を印刷します。

操作 PAGE オプションは、FMTLIB オプションをアクティブにします。

ヒント ODS LISTING 出力先で、それぞれの出力形式と入力形式に関する情報が**アウトプット**ウィンドウの別のページに表示されます。

EXCLUDE ステートメント

エントリを FMTLIB オプションと CNTLOUT=オプションによる処理から除外します。

制限事項: 1 つの EXCLUDE ステートメントだけを PROC FORMAT ステップで表示できます。
SELECT ステートメントと EXCLUDE ステートメントを同じ PROC FORMAT ステップ内で使用することはできません。

構文

EXCLUDE *entry(s)*;

必須引数

entry(s)
処理から除外する 1 つ以上のカタログエントリを指定します。カタログエントリの名前は、保存する入力形式または出力形式の名前と同じです。入力形式と出力形式の名前、文字と数値の入力形式と出力形式の名前をそれぞれ同じにすることができるため、EXCLUDE ステートメントで入力形式と出力形式を指定する際は特定の接頭辞を使用する必要があります。EXCLUDE ステートメントでエントリを指定する際は、次の規則に従います。

- ドル記号($)の付いた文字出力形式を含むエントリ名を前に置きます。
- アットマークとドル記号の付いた文字出力形式を含むエントリ名(@$*entry-name* など)を前に置きます。
- アットマーク(@)の付いた数値入力形式を含むエントリ名を前に置きます。
- 接頭辞の付かない数値出力形式を含むエントリ名を指定します。

詳細

名前指定のショートカット

コロン(:)とハイフン(-)ワイルドカード文字を使用して、エントリを除外できます。たとえば、次の EXCLUDE ステートメントは、文字 **a** で始まるすべての出力形式または入力形式を除外します。

```
exclude a:;
```

また、次の EXCLUDE ステートメントは、**apple** から **pear** までにアルファベット順に発生するすべての出力形式または入力形式を除外します。

```
exclude apple-pear;
```

FMTLIB 出力

PROC FORMAT ステートメントの FMTLIB または CNTLOUT なしで EXCLUDE ステートメントを使用する場合、プロシジャは FMTLIB オプションを起動し、FMTLIB オプション出力が行われます。

INVALUE ステートメント

生データ値を読み込み、変換するための入力形式を作成します。

参照項目: *SAS 出力形式と入力形式: リファレンス* で、SAS によって提供されている入力形式に関するドキュメントについて参照してください。

例: “例 9: 生の文字データを数値に変換する” (900 ページ)

構文

INVALUE <$>*name* <(*informat-option(s)*)> <*value-range-set(s)*>;

オプション引数の要約

DEFAULT=*length*
: 入力形式のデフォルト長を指定します。

FUZZ=*fuzz-factor*
: 値を範囲に一致させるための誤差定数を指定します。

JUST
: すべての入力文字列を範囲と比較する前に左寄せにします。

MAX=*length*
: 入力形式に対し最大長を指定します。

MIN=*length*
: 入力形式に対し最小長を指定します。

NOTSORTED
: 定義順で値または範囲を保存します。

UPCASE
: すべての入力文字列を範囲と比較する前に大文字にします。

Control the input template.

value-range-set(s)
: データを読み込むための変数テンプレートを指定します。

必須引数

name
作成中の入力形式に名前を付けます。

制限事項 ユーザー定義の入力形式の名前は、SAS によって提供されている入力形式の名前と同じにすることはできません。

要件 名前は有効な SAS 名である必要があります。数値入力形式名の長さは最大 31 文字が可能です。文字入力形式名の長さは最大 30 文字が可能で、数字で終わらせることはできません。文字入力形式を作成している場合、ドル記号($)を最初の文字として使用します。ドル記号を名前に追加するのは、文字入力形式が 30 文字に制限されているためです。

操作 入力形式名の最大長は、VALIDFMTNAME= システムオプションによって制御されます。詳細については、*SAS システムオプション: リファレンス*を参照してください。

ヒント 名前の後にピリオドを使用して、入力形式を後で参照します。ただし、INVALUE ステートメントでは入力形式名の後にピリオドを使用しないでください。

ユーザー作成入力形式を参照するメッセージの印刷時に、名前にはアットマーク(@)が接頭辞として付けられます。入力形式の保存時に、アットマークが入力形式に指定する名前に接頭辞として付けられます。アットマークを名前に追加するのは、名前が 31 文字または 30 文字に制限されているためです。アットマークを使用する必要があるのは、名前を EXCLUDE ステートメントまたは SELECT ステートメントで使用している場合*だけ*です。入力形式を変数と関連付ける場合は、名前に接頭辞としてアットマークを付けないでください。

オプション引数

DEFAULT=*length*
入力形式のデフォルト長を指定します。入力形式と変数を関連付ける際に特定の長さを指定しない場合、DEFAULT=の値が入力形式の長さになります。

デフォルト 文字入力形式の場合、最も長いラベルの長さ

数値入力形式の場合、等号の左側に数値データがあれば 12

引用符で囲まれた文字列の場合、最も長い文字列の長さ

範囲 1-32767

ヒント 最良の方法として、既存の入力形式を value-range set で指定している場合は、常に DEFAULT=オプションを指定してください。

FUZZ=*fuzz-factor*
値を範囲に一致させるための誤差定数を指定します。数字が範囲に完全には一致しない、または厳密には範囲内になくても *fuzz-factor* 内にあれば、入力形式はそれを一致とみなします。たとえば、次の VALUE ステートメントは誤差定数 2 を使用する LEVELS.入力形式を作成します。

```
invalue levels (fuzz=.2) 1='A'
```

```
2='B'
3='C';
```

FUZZ=.2 は、変数値が範囲のどちらかの端の値.2 内にある場合、入力形式が変数値の保存に対応するフォーマットされた値を使用することを意味します。そのため、LEVELS.入力形式は値 2.1 を B として保存します。

ヒント INVALUE ステートメントを使用して数値入力形式を作成するとき、FUZZ=0 を指定して記憶域を保存します。

ゼロ以外の誤差定数は、非常に近いが完全には一致しない数値でのみ使用します。バイナリ検索を実行するために、範囲は並べ替え順に内部保存されます(NOTSORTED オプションが使用されていない場合)。誤差定数が 1 つの範囲の終わりに追加され、次の範囲の先頭から削除れた場合に、これらの範囲が重なると、結果は予測不能なものになることがあります。値は、バイナリ検索で一致した最初の範囲に置かれます。除外演算子は、このバイナリ検索アルゴリズムを無効にするには不十分です。最良の方法として、除外演算子を使用する場合は、FUZZ=0 または NOTSORTED オプションを設定します。

数値入力形式で<除外演算子を使用する場合は、FUZZ=0 を使用するのが最もよい方法です。

JUST
すべての入力文字列を範囲と比較する前に左寄せにします。

MAX=*length*
入力形式に対し最大長を指定します。入力形式と変数を関連付ける際に、MAX=値を超える幅を指定できません。

デフォルト 40

範囲 1–32767

MIN=*length*
入力形式に対し最小長を指定します。

デフォルト 1

範囲 1–32767

NOTSORTED
定義順で値または範囲を保存します。

NOTSORTED を指定しない場合、値または範囲はデフォルトで並び替え順序で保存され、SAS では特定の値が含まれる範囲を位置付けるためのバイナリ検索アルゴリズムが使用されます。NOTSORTED を指定する場合、SAS は各範囲を一致するまで定義順に検索します。

次のうちいずれかが true の場合、NOTSORTED を使用します。

- 発生する特定範囲の尤度がわかっており、処理時間を削減するために、入力形式を使用してそれらの範囲を最初に検索する。
- FMTLIB オプションを使用して入力形式の説明を出力する際に、範囲を定義する順序を保持する。
- ORDER=DATA オプションと PRELOADFMT オプションを使用して PROC MEANS、PROC SUMMARY または PROC TABULATE のクラス変数を分析する際に、範囲を定義する順序を保持する。

値の分布が一様または不明な場合、または値の数が比較的小さい場合は、NOTSORTED を使用しないでください。NOTSORTED が指定されていないときに SAS が使用するバイナリ検索アルゴリズムにより、これらの条件下の検索のパフォーマンスが最適化されます。

CPORT プロシジャと CIMPORT プロシジャを使用して標準照合順序が異なる動作環境間で入力形式または出力形式を転送する場合、NOTSORTED が自動的に使用されます。NOTSORTED の自動設定は、ASCII と EBCDIC の動作環境間で入力形式または出力形式を転送する場合に発生します。この状況に問題がある場合、次を行います。

- PROC FORMAT ステートメントの CNTLOUT=オプションを使用し、出力制御データセットを作成します。
- CPORT プロシジャを使用して、制御データセットに対し転送ファイルを作成します。
- 対象の動作環境で CIMPORT プロシジャを使用して、転送ファイルをインポートします。
- 対象の動作環境で、PROC FORMAT を CNTLIN=オプションと使用し、インポートされた制御データセットから出力形式と入力形式を作成します。

UPCASE
すべての生データ値を可能な範囲と比較する前に大文字に変換します。UPCASE を使用する場合、指定する値または範囲が大文字であることを確認してください。

value-range-set(s)
生データと、生データがなる値を指定します。*value-range-set(s)*には、次のうち 1 つ以上が可能です。

value-or-range-1<, *value-or-range-2* …>=*informatted-value* | [*existing-informat*]

入力形式は生データを等号の右側の *informatted-value* の値に変換します。

value-or-range
“値や範囲の指定” (874 ページ)を参照してください。

informatted-value
value-or-range の生データがなる値です。*informatted-value* に対し、次の形式のうちいずれかを使用します。

'character-string'
最大 32,767 文字長の文字列です。通常、入力形式を使用して生データを変換する場合に *character-string* は文字変数の値になります。*character-string* を *informatted-value* に使用するのは、文字入力形式を作成しているときだけです。*character-string* を一重引用符または二重引用符で囲まない場合、INVALUE ステートメントは引用符がそこにあるとみなします

16 進リテラルの場合、使用できるのは、最大 32,767 入力文字または最大 16,382 表現文字(表現文字ごとに 2 つの 16 進文字)です。

number
入力形式適用値になる数値です。通常、入力形式を使用して生データを変換する場合に *number* は、数値変数の値になります。*number* を *informatted-value* に使用するのは、数値入力形式を作成しているときだけです。*number* の最大値は、ホストの動作環境によって異なります。

ERROR
指定範囲のデータ値を無効なデータとして扱います。SAS は欠損値を変数に割り当て、データ行を SAS ログに印刷し、警告メッセージを発行します。

SAME
入力形式で生データがその他の値として変換されないようにします。たとえば、次の GROUP.入力形式は値 01 から 20 までを変換し、結果として数字 1 から 20 までを割り当てます。その他すべての値には、欠損値が割り当てられます。

```
invalue group 01-20= _same_
              other= .;
```

existing-informat
SAS によって提供されている入力形式または既存するユーザー定義の入力形式です。作成している入力形式では既存する入力形式を使用して、等号の左側の *value-or-range* に一致する生データを変換します。既存する入力形式を使用する場合、その入力形式名を角かっこ([date9.]など)、またはかっこと縦棒((|date9.|)など)で囲みます。既存する入力形式の名前を一重引用符で囲まないでください。

ヒント 最良の方法として、value-range-set の既存する入力形式を指定する場合、DEFAULT=オプションを使用して常にデフォルト値を指定します。

例

例 1: 生データ値に対する文字入力形式の作成

$GENDER.文字入力形式は、生データ値 **F** と **M** を文字値**'1'**と**'2'**にそれぞれ交換します。

```
invalue $gender 'F'='1'
                'M'='2';
```

ドル記号の接頭辞は、入力形式が文字データを変換することを示します。

例 2: 文字値および数値または値範囲の作成

数値入力形式の作成時に、*value-or-range* に対して文字列または数字を指定できます。たとえば、TRIAL.入力形式は **A** から **M** までを並び替える文字列を数字 1 に、**N** から **Z** までを並び替える文字列を数字 2 に変換します。入力形式は引用符なしの範囲 1–3000 を数値範囲として扱います。これには 1 から 3000 までのすべての数字が含まれます。

```
invalue trial 'A'-'M'=1
              'N'-'Z'=2
               1-3000=3;
```

例 3: _ERROR_と_SAME_を使用した入力形式の作成

CHECK.入力形式は_ERROR_と_SAME_を使用して、値 1 から 4、99 を変換します。その他すべての値は無効です。

```
invalue check 1-4=_same_
               99=.
            other=_error_;
```

数値入力形式を使用して値または範囲に対応しない文字列を変換する場合、エラーメッセージが表示されます。

PICTURE ステートメント

数字を印刷するためのテンプレートを作成します。

ヒント: 最良の方法として、value-range-set の既存の出力形式を指定する場合、DEFAULT=オプション (869 ページ)を使用して常にデフォルト値を指定します。

DATATYPE=オプションを使用している場合、DEFAULT=オプションを使用して、これらの文字のフォーマットに十分な大きさのデフォルト出力形式幅を設定します。DEFAULT=オプションを設定しない場合、出力形式のデフォルト幅は等号の右側の最大値の幅となります。

参照項目: *SAS 出力形式と入力形式: リファレンス* と *SAS 各国語サポート(NLS): リファレンスガイド* で、SAS によって提供されている出力形式に関するドキュメントについて参照してください。

例: “例 2: ピクチャ形式の作成” (886 ページ)
“例 4: ピクチャ形式の埋め込み” (890 ページ)
“例 14: 英語以外の言語の出力形式の作成” (913 ページ)

構文

PICTURE *name* <(*format-option(s)*)>
<*value-range-set-1* <(*picture-1-option(s)*)>
<*value-range-set-2* <(*picture-2-option(s)*)>> …>;

オプション引数の要約

Control the attributes of each picture in the format

FILL='*character*'
: フォーマットされた値を完了する文字を指定します。

MULTIPLIER=*n*
: フォーマットされる前に変数の値に掛ける数字を指定します。

NOEDIT
: 数字が数値セレクタではなく、メッセージ文字になるように指定します。

PREFIX='*prefix*'
: フォーマットされた値の前に置く文字接頭辞を指定します。

Control the attributes of the format

DATATYPE=DATE | TIME | DATETIME | DATETIME_UTIL
: ピクチャでディレクトリをテンプレートとして使用して、日付値、時間値、日時値をフォーマットできます。

DECSEP='*character*'
: 数字の小数部分に対して区切り文字を指定します。

DEFAULT=*length*
: ピクチャのデフォルト長を指定します。

DIG3SEP='*character*'
: 数字に対して 3 桁の区切り文字を指定します。

FUZZ=*fuzz-factor*
: 値を範囲に一致させるための誤差定数を指定します。

LANGUAGE=

日付、時間、日時のピクチャで置き換えられる平日、月に使用される言語を指定します。

MAX=*length*
出力形式に対し最大長を指定します。

MIN=*length*
出力形式に対し最小長を指定します。

MULTILABEL
同一の値または重複した値を含む可能性のある複数の *values-or-range* 値へのラベルの割り当てが可能になります。

NOTSORTED
定義順で値または範囲を保存します。

ROUND
フォーマットする前に、値を四捨五入して整数にします。

Control the template for printing

value-range-set
1 つ以上の変数値と、それらの値を印刷するためのテンプレートを指定します。

必須引数

name
作成中の出力形式に名前を付けます。

制限事項 ユーザー定義の出力形式の名前を SAS によって提供されている出力形式の名前にすることはできません。

要件 名前は有効な SAS 名である必要があります。数値出力形式名の長さには最大 32 文字が可能です。文字出力形式名の長さには最大 31 文字が可能で、最後を数字にすることはできません。文字出力形式を作成している場合、最初の文字としてドル記号($)を使用します。このため、文字入力形式は 31 文字に制限されています。SAS 名に関する情報については、“Rules for Words and Names in the SAS Language” (*SAS Language Reference: Concepts*)を参照してください。

操作 出力形式名の最大長は、VALIDFMTNAME=システムオプションによって制御されます。詳細については、*SAS システムオプション: リファレンス*を参照してください。

ヒント 名前の後にピリオドを使用して、出力形式を後で参照します。ただし、VALUE ステートメントでは出力形式名の後にピリオドを置かないでください。

オプション引数

DATATYPE=DATE | TIME | DATETIME | DATETIME_UTIL
ピクチャでディレクトリをテンプレートとして使用して、日付値、時間値、日時値をフォーマットできます。ピクチャ形式で使用するディレクティブに基づいて、DATE、TIME、DATETIME、DATETIME_UTIL のいずれかを指定します。次の定義とリストについては、ディレクティブ (857 ページ)*ピクチャ*での説明を参照してください。

操作 DATATYPE=DATETIME に設定すると、日時の時間が 00:00:00 から 23:59:59 になります。DATATYPE=DATETIME_UTIL に設定すると、日時の時間が 00:00:01 から 24:00:00 の間になります。

ヒント 数値欠損値をフォーマットする場合、結果のラベルは ERROR となります。プログラムに欠損値をチェックする句を追加することにより、ERROR ラベルを削除できます。

DEFAULT=*length*
ピクチャのデフォルト長を指定します。出力形式と変数を関連付ける際に特定の長さを指定しない場合、DEFAULT=の値が *picture* の長さになります。

デフォルト 最も長いピクチャ値の長さ

範囲 1-32767

ヒント DATATYPE=オプションを使用している場合は、DEFAULT=オプションを使用して、これらの文字のフォーマットに十分な大きさのデフォルト出力形式幅を設定します。

DECSEP='*character*'
数字の小数部分に対して区切り文字を指定します。

デフォルト .(小数点)

DIG3SEP='*character*'
数字に対して 3 桁の区切り文字を指定します。

デフォルト , (カンマ)

FILL='*character*'
フォーマットされた値を完了する文字を指定します。

有効桁数が出力形式の長さより少ない場合、出力形式はフォーマットされた値を完了つまり幅に合わせる必要があります。

- 数値セレクタとしてゼロを指定すると、出力形式は *character* を使用して、フォーマットされた値を幅に合わせます。
- ゼロ以外の数値セレクタを指定すると、出力形式はゼロを使用してフォーマットされた値を幅に合わせます。FILL=オプションは影響しません。

配置されている最終有効桁にマップする数値セレクタの左側に表示されるカンマなどのその他の文字がピクチャに含まれている場合、文字は埋め文字または先頭のゼロによって置き換えられます。

デフォルト ' ' (ブランク)

制限事項 FILL=オプションは、値に出力形式を設定する関数を使用する場合は、有効になりません。

操作 同一のピクチャで FILL=オプションと PREFIX=オプションを使用する場合、出力形式は接頭辞を置いてから埋め字を入力します。

例 “例 4: ピクチャ形式の埋め込み” (890 ページ)

FUZZ=*fuzz-factor*
値を範囲に一致させるための誤差定数を指定します。数字が範囲に完全には一致しない、または厳密には範囲内になくてもどちらかの端の *fuzz-factor* 内にあれ

ば、出力形式はそれを一致とみなします。たとえば、次の VALUE ステートメントは誤差定数 2 を使用する LEVELS.出力形式を作成します。

```
value levels (fuzz=.2) 1='A'
                       2='B'
                       3='C';
```

FUZZ=.2 は、変数値が範囲のどちらかの端の値.2 内にある場合、出力形式が変数値の印刷に対応するフォーマットされた値を使用することを意味します。そのため、LEVELS.形式出力は値 2.1 を B としてフォーマットします。

デフォルト 1E−12 (数値出力形式)

ヒント VALUE ステートメントを使用して数値出力形式を作成するとき、FUZZ=0 を指定して記憶域を保存します。

ゼロ以外の誤差定数は、非常に近いが完全には一致しない数値でのみ使用します。*誤差定数*が 1 つの範囲の終わりに追加され、次の範囲の先頭から削除された場合に、これらの範囲が重なると、結果は予測不能なものになることがあります。値は、バイナリ検索で一致した最初の範囲におかれます。

最良の方法は、数値出力形式で<除外演算子を使用する場合は、FUZZ=0 を使用します。

LANGUAGE=

日付、時間、日時のピクチャで置き換えられる平日、月に使用される言語を指定します。サポートされていない、または無効な言語を指定すると、英語が使用されます。

LANGUAGE=オプションの有効値は次のとおりです。

アフリカーンス語	英語	マケドニア語	スペイン語
カタロニア語	フィンランド語	ノルウェー語	スウェーデン語
クロアチア語	フランス語	ポーランド語	Swiss_French
チェコ語	ドイツ語	ポルトガル語	Swiss_German
デンマーク語	ハンガリー語	ロシア語	
オランダ語	イタリア語	スロベニア語	

デフォルト シングルバイトの文字セットの場合、DFLANG=システムオプションで指定される言語

ダブルイトおよび UTF-8 の文字セットの場合、LOCALE=システムオプションで指定される言語

ヒント ユーザー定義の出力形式を、LANGUAGE=オプションでサポートされる言語以外の言語で使用するには、LOCALE=システムオプションをその言語のロケールに設定します。PROC FORMAT では、LANGUAGE=オプションを指定しないでください。ピクチャ形式の言語は、ロケール設定によって決定されます。ロケールのリストについては、“LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

参照項目 “DFLANG= System Option: UNIX, Windows, and z/OS” (*SAS National Language Support (NLS): Reference Guide*)

MAX=*length*
出力形式に対し最大長を指定します。出力形式と変数を関連付ける際に、MAX= 値を超える幅を指定できません。

デフォルト 40

範囲 1–32767

MIN=*length*
出力形式に対し最小長を指定します。

デフォルト 1

範囲 1–32767

MULTILABEL
同一の値または重複した値を含む可能性のある複数の *values-or-range* 値へのラベルの割り当てが可能になります。ラベルは value-range-set の等号の右側のピクチャ定義によって決定されるフォーマットされた値です。ここでは、MULTILABEL の使用方法の例を示します。

次の PICTURE ステートメントでは、MULTILABEL オプションの 2 つの使用を示します。いずれの場合も、数値出力形式は、ラベルとして割り当てられます。最初の PICTURE ステートメントは複数のラベルを単一の値に割り当てます。複数のラベルは単一の値範囲に割り当てることもできます。2 番目の PICTURE ステートメントは、ラベルを重複した値範囲に割り当てます。MULTILABEL オプションにより、複数のラベルを重複した値に割り当てることができます。

```
picture abc (multilabel)
    1000='9,999'
    1000='9999';

  picture overlap (multilabel)
    /* without decimals */
    0-999='999'
    1000-9999='9,999'

     /* with decimals */
    0-9='9.999'
    10-99='99.99'
    100-999='999.9';
```

PROC MEANS、PROC SUMMARY、PROC TABULATE などのマルチラベル有効プロシジャだけが、複数のラベルを使用できます。その他すべてのプロシジャと DATA ステップは、1 次ラベルのみ認識します。

指定エントリに対する *primary label* は、最初の値に割り当てられるフォーマットされた値(ピクチャに基づく)、またはすべての値(等号の左側の)が順番に並べ替えられるときにエントリに一致する、または含まれる range-of-values (等号の左側)です。次に例を示します。

- 最初の PICTURE ステートメントでは 1000 の 1 次ラベルは 1,000 です。ピクチャ 9,999 が 1000 に割り当てられている最初の値であるためです。1000 の第 2 ラベルは 1000 で、これは 9999 ピクチャに基づいています。

- 2 番目の PICTURE ステートメントでは、5 の 1 次ラベルは範囲 0–9 に割り当てられている 9.999 ピクチャに基づく 5.000 です。0–9 が 5 を含む値の最初の範囲であるためです。5 の 2 次ラベルは 005 です。範囲 0–999 が範囲 0–9 の後に順に発生するためです。

複数のラベルを 1 つの値に割り当てる際は、注意が必要です。

value-range-sets の割り当て時に NOTSORTED オプションを使用しない限り、value-range-sets は並べ替え順に保存されます。この順序により、MULTILABEL 出力形式の value-range-sets の処理時に予期せぬ結果が発生する可能性があります。次に例を示します。

2 番目の PICTURE ステートメントでは、15 の 1 次ラベルは 015 で、15 の 2 次ラベルは 15.00 です。範囲 0–999 が範囲 10–99 の前に順に発生するためです。15 の 1 次ラベルで 99.99 出力形式を使用する場合、PICTURE ステートメントで範囲 10–99 を 0–99 に変更します。範囲 0–99 は範囲 0–999 の前に順に発生し、必要な結果が生成されます。

制限事項 単一の出力形式または入力形式に対して作成できるラベルの最大数は、255 です。

MULTIPLIER=*n*

フォーマットされる前に変数の値に掛ける数字を指定します。MULTIPLIER=オプションの値は乗算の結果と *value-range-set* の *picture* 部分の数値セレクタによって異なります。たとえば、次の PICTURE ステートメントは、変数値 1600000 を **`$1.6M`** としてフォーマットする MILLION.出力形式を作成します。

```
picture million low-high='09.9M'
        (prefix='$' mult=.00001);
```

1600000 に最初に.00001 を掛けると、16 になります。小数点の後に数値セレクタがあることに注意してください。値 16 は、右で始まるピクチャに置かれます。値 16 は 09.9 を上書きし、結果は 01.6 となります。先頭のゼロが削除され、最終結果は 1.6M となります。

低-高の値が'000M'である場合、結果は 16M となります。

別名 MULT=

デフォルト 10^{n}.*n* はピクチャの最初の小数点の後の桁数です。たとえば、データに値 123.456 が含まれていて、それを'999.999'のピクチャを使用して印刷するとします。この出力形式は、123.456 に 10^{3} を掛けて求め値 123456 が、フォーマットされた値 123.456 になります。

制限事項 MULT=オプションは、値に出力形式を設定する関数を使用する場合は、有効になりません。

例 “例 2: ピクチャ形式の作成” (886 ページ)

“例 3: 大きなドル金額のピクチャ形式の作成” (888 ページ)

NOEDIT

数字が数値セレクタではなく、メッセージ文字になるように指定します。つまり、出力形式は数字をピクチャで表示されるように印刷します。たとえば、次の PICTURE ステートメントは 1000 を超える変数値を **`>1000 miles`** としてフォーマットする MILES.出力形式を作成します。

```
picture miles 1-1000='0000'
          1000<-high='>1000 miles'(noedit);
```

制限事項 NOEDIT=オプションは、値に出力形式を設定する関数を使用する場合は、有効になりません。

NOTSORTED
定義順で値または範囲を保存します。NOTSORTED を指定しない場合、値または範囲はデフォルトで並び替え順序で保存され、SAS では特定の値が含まれる範囲を位置付けるためのバイナリ検索アルゴリズムが使用されます。NOTSORTED を指定する場合、SAS は各範囲を一致するまで定義順に検索します。

次のうちいずれかが true の場合、NOTSORTED を使用します。

- 発生する特定範囲の尤度がわかっており、処理時間を削減するために、入力形式を使用してそれらの範囲を最初に検索する。
- FMTLIB オプションを使用して出力形式の説明を出力する際に、範囲を定義する順序を保持する。
- ORDER=DATA オプションと PRELOADFMT オプションを使用して PROC MEANS、PROC SUMMARY または PROC TABULATE のクラス変数を分析する際に、範囲を定義する順序を保持する。

値の分布が一様または不明な場合、または値の数が比較的小さい場合は、NOTSORTED を使用しないでください。NOTSORTED が指定されていないときに SAS が使用するバイナリ検索アルゴリズムにより、これらの条件下の検索のパフォーマンスが最適化されます。

CPORT プロシジャと CIMPORT プロシジャを使用して標準照合順序が異なる動作環境間で入力形式または出力形式を転送する場合、NOTSORTED が自動的に使用されます。NOTSORTED の自動設定は、ASCII と EBCDIC の動作環境間で入力形式または出力形式を転送する場合に発生します。この状況に問題がある場合、次を行います。

- PROC FORMAT ステートメントの CNTLOUT=オプションを使用し、出力制御データセットを作成します。
- CPORT プロシジャを使用して、制御データセットに対し転送ファイルを作成します。
- 対象の動作環境で CIMPORT プロシジャを使用して、転送ファイルをインポートします。
- 対象の動作環境で、PROC FORMAT を CNTLIN=オプションと使用し、インポートされた制御データセットから出力形式と入力形式を作成します。

PREFIX='*prefix*'
フォーマットされた値の前に置く文字接頭辞を指定します。接頭辞は、値の最初の有効桁の前に置かれます。ゼロ数値セレクタを使用する必要があります。使用しない場合、接頭辞は使用されません。

PREFIX=は通常、先頭の通貨記号とマイナス記号の印刷に使用します。たとえば、PAY.出力形式は、変数値 25500 を**$25,500.00** と印刷します。

```
picture pay
   low-high='000,009.99' (prefix='$');
```

デフォルト no prefix

制限事項 PREFIX=オプションは、値に出力形式を設定する関数を使用する場合は、有効になりません。

操作 同一のピクチャで FILL=オプションと PREFIX=オプションを使用する場合、出力形式は接頭辞を置いてから埋め字を入力します。

例 “例 2: ピクチャ形式の作成” (886 ページ)

“例 4: ピクチャ形式の埋め込み” (890 ページ)

注意 **ピクチャの幅が値と接頭辞の両方を含めるのに十分でない場合、出力形式は接頭辞を切り捨て、または省略し、これによりデータが不正確になります。**

ROUND

フォーマットする前に、値を四捨五入して整数にします。ROUND オプションを指定しない場合、出力形式は変数値に乗数を掛け、小数部(ある場合)を切り捨て、定義するテンプレートに従って結果を印刷します。ROUND オプションを指定する場合、出力形式は変数値に乗数を掛け、その結果を四捨五入して整数にしてから、その値にテンプレートに従ってフォーマットします。FUZZ=オプションも指定されている場合、四捨五入は、値が属する範囲の決定に誤差定数が使用された後に発生します。

ヒント ROUND オプションは、値.5 を次に最も大きい整数に四捨五入します。

注意 **数字を四捨五入することにより桁が数字に追加される場合、ピクチャが追加される桁に対し十分広い必要があります。** たとえば、数字.996 のピクチャは ‘99’ (prefix ‘.’ mult=100)になります。数字を四捨五入し、それに 100 を掛けると、結果の数字は 100 になります。ピクチャが適用されると、結果は不正確な数字、.00 となります。四捨五入するときに数字が正しいものになるように、ピクチャ幅をより大きな数字を表示するのに十分なものにしてください。

value-range-set

1 つ以上の変数値と、それらの値を印刷するためのテンプレートを指定します。*value-range-set* には、次の形式があります。

value-or-range-1 <, value-or-range-2, …>='picture'

value-or-range

“値や範囲の指定” (874 ページ)を参照してください。

picture

数値変数の値をフォーマットするためのテンプレートを指定します。ピクチャは、一重引用符で囲まれた一連の文字です。ピクチャの最大長は 40 文字です。ピクチャは、数値セレクタ、メッセージ文字、ディレクティブの 3 つの種類の文字で指定されます。1 つのピクチャに可能な数値セレクタは最大 16 です。

digit selectors

数値に対し位置を定義する数値文字(0 から 9 まで)です。数値セレクタがゼロ以外のピクチャ形式は変数値の先頭のゼロを印刷します。ピクチャ数値セレクタが 0 の場合は変数値の先頭のゼロを印刷しません。ピクチャ形式に数値セレクタが含まれている場合、数値セレクタはピクチャの最初の文字である必要があります。

注 このセクションでは 9 をゼロ以外の数値セレクタとして使用します。

message characters

ピクチャで指定されているように印刷する非数値文字です。次の PICTURE ステートメントには、数値セレクタ(99)とメッセージ文字(`illegal day value`)が含まれています。DAYS.出力形式に非ゼロ数値セレクタがあるた

め、値は先頭のゼロとともに印刷されます。特殊範囲 OTHER は、指定範囲(1 から 31 まで)内にない値に対しメッセージ文字を印刷します。

```
picture days
          01-31='99'
          other='99-illegal day value';
```

例 値 02 と 67 は、のように印刷されます

```
                              02
          67-illegal day value
```

directives

ピクチャで日付値、時間値、日時値をフォーマットするために使用できる特殊文字です。

注: ディレクティブを使用できるのは、PICTURE ステートメントで DATATYPE=オプション (850 ページ)を指定するときだけです。DATATYPE=オプションの値が、使用するディレクティブの種類に適切であることを確認してください。不適切な値を使用する場合、データに出力形式は適用されません。たとえば、%a ディレクティブには DATATYPE=DATE を使用します。

使用できるディレクティブは次のとおりです。

%a
曜日の省略名(Wed など)。

%A
曜日の完全名(Wednesday など)。

%b
月の省略名(JAN や Jan など)。大文字小文字は、SAS セッションロケールで決まります。

ヒント 英語の場合、常に先頭文字のみを大文字(Jan など)にして月の省略名を作成するには、ディレクティブ%3B を使用します。

%<*n*>B
n がディレクティブに含まれない場合は月の完全名(January など)。*n* には月名に表示する文字数を指定します。それに対し、一部のロケールでは、%b ディレクティブで、大文字の 3 文字の月の省略名が作成されます。

制限事項 *n* は、DBCS および Unicode SAS セッションでサポートされません。

例 %3B と指定すると、October の月を表す Oct が作成されます。

%C
ブランクを埋め込んだ長い月名(January から December まで)(December など)。

%d
月の日付。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0d など)。

%e
先頭のスペースを含む 2 文字の 10 進数の月の日付(" 1"- "31") (“ 2” など)。

%F
ブランクを埋め込んだ曜日の完全名。

%G
4 桁の 10 進数の年(2008 など)。1 月 1 日を含む週の 4 日以上が新年にかかっている場合、新年の第 1 週とみなされます。その他の場合は前年の最終週となり、年は前年とみなされます。

%H
時間(24 時間)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0H など)。

ヒント DATETYPE=DATETIME に設定すると、SAS は日時の時間に 00:00–23:59 を使用します。DATETYPE=DATETIME_UTIL に設定すると、SAS は日時の時間に 00:00:01–24:00:00 を使用します。24:00:00 はその日の終わりの午前 0 時です。00:00:00 という時刻は時間の範囲に含まれていないため、この値を使用すると前の日の 24:00:00 に変換されます。DATETIME を指定すると、00:00:00 は新しい日の午前 0 時になり、24:00:00 は次の日の午前 0 時になります。

例 “例 5: 24 時間形式から 00:00:01–24:00:00 形式への変更” (892 ページ)

%I
時間(12 時間)。

別名 %i

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0I など)。

%j
先頭のゼロを含む 10 進数の年の日付(1–366)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0j など)。

%m
月(1–12)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0m など)。

%M
分(0–59)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0M など)。

%n
10 進数での期間の日数(最大 10 桁) (25 など)。

制限事項 このディレクティブは、DBCS と Unicode の SAS セッションには対応していません。

%o
ブランクが埋め込まれた月(1-12) (" 2"など)。

%p
a.m.または p.m.と同じです

%q
1、2、3、4 など、四半期の省略文字列。

%Q
Quarter1、Quarter2、Quarter3、Quarter4 などの四半期の文字列。

%s
10 進数での小数点以下の秒数(.39555 など)。フォーマットされた桁数は、出力形式使用時に指定される小数点の右側の桁数です。小数点以下の秒数は四捨五入され、小数点以下の秒数に指定された桁数に調整されます。

制限事項 このディレクティブは、DBCS と Unicode の SAS セッションには対応していません。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0s など)。

%S
可能なうるう秒に使用可能な秒数(0–59)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0S など)。

例 58 と 59.07

%u
1 桁の 10 進数の曜日(1–7 (Monday - Sunday)) (Sunday=7 など)。

%U
10 進数の年の週数(0–53)。日曜日は週の最初の日とみなされます。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0U など)。

%V
第 1 週の開始日が第 1 月曜日の週数(01–53)。第 1 週の最小日数は 4 です。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0SV など)。

%w
1 桁の 10 進数の曜日(0–6 (Sunday から Saturday)) (Sunday=0 など)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0w など)。

%W
第 1 週の開始日が第 1 月曜日の週数(0–53)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0W など)。

%y
世紀なしの年(0–99) (93 など)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0y など)。

%Y
4 桁の 10 進数の世紀を含む年(1970–2069) (1994 など)。

%z
UTC タイムゾーンオフセット。

%Z
タイムゾーン名。

%%
%文字。

ヒント コードをプログラムに追加して、欠損値の表示方法を指示します。

操作 出力形式定義で LANGUAGE=と PICTURE=を指定すると、出力形式では英語とヨーロッパ言語のみサポートされます。LANGUAGE=オプションによってサポートされているもの以外の言語のユーザー定義出力形式を使用するには、PICTURE=ステートメントを使用します。LANGUAGE=オプションを指定しないでください。ピクチャ形式の言語は、ロケール設定によって決定されます。

詳細

ピクチャ形式の作成: 手順

このセクションでは、先頭にゼロを付けないピクチャ形式の作成方法を説明します。サンプルデータセットでは、変数 Amount のデフォルト印刷で 1 から-1 までの数字に先頭のゼロが含まれています。PICTURE ステートメントで、1 から–1 までの数値の先頭のゼロを削除する 2 つの似たような出力形式を定義します。2 つの出力形式の違いは、NOZEROSR.出力形式では ROUND オプションを指定して数字を四捨五入し、NOZEROS.出力形式では四捨五入しないという点です。

次のプログラムでは、サンプルデータセットを作成、並べ替え、印刷します。

```
data sample;
   input Amount;
   datalines;
-2.051
-.05
-.017
  0
.093
.54
.556
6.6
14.63
0.996
-0.999
-45.00
```

```
;
run;


proc sort data=sample;
   by amount;
run;

proc print data=sample;
   title 'Default Printing of the Variable Amount';
run;
```

Default Printing of the Variable Amount

Obs	Amount
1	-45.000
2	-2.051
3	-0.999
4	-0.050
5	-0.017
6	0.000
7	0.093
8	0.540
9	0.556
10	0.996
11	6.600
12	14.630

次は、NOZEROSR.および NOZEROS.出力形式を指定する PROC FORMAT ステップです。どちらの出力形式でも、フォーマットされた値で先頭のゼロは削除されます。NOZEROSR.出力形式では、ROUND オプションが指定されて数値が四捨五入されます。NOZEROS.出力形式では、四捨五入は実行されません。

```
libname library 'SAS-library';

proc format;
   picture nozerosR (round fuzz=0)
     low -  -1      =  '000.00' (prefix='-')
     -1 < - < -.99  =  '0.99'  (prefix='-' mult=100)
     -0.99 <-<  0   =  '99'    (prefix='-.' mult=100)
                0   =  '9.99'
      0 <  -< .99   =  '99'    (prefix='.' mult=100)
      0.99 - < 1    =  '0.99'  (mult=100)
      1  - high     =  '09.99';

   picture nozeros (fuzz=0)
     low -  -1      =  '000.00' (prefix='-')
```

```
      -1 < - < -.99  =  '0.99'  (prefix='-.' mult=100)
      -0.99 <-<  0   =  '99'    (prefix='-.' mult=100)
                 0   =  '9.99'
       0 <  -< .99   =  '99'    (prefix='.' mult=100)
       0.99 - < 1    =  '0.99'  (prefix='.' mult=100)
       1  - high     =  '09.99';

run;
```

次の表に、各範囲からの値をフォーマットする方法について説明します。各ステップの詳細については、表 27.3 (864 ページ)を参照してください。

表 27.2 *ピクチャ形式の作成*

ステップ	PICTURE ステートメントの処理のルール	例は次のとおりです。
1	値の範囲を決定し、ピクチャを使用します。	2 番目の範囲で、除外演算子<がハイフンの両側に表示され、範囲から−1 と-.99 が削除されます。3 番目の範囲では、0 と.99 が削除されます。4 番目の範囲では、1 が削除されます。 除外演算子が使用されるため、FUZZ=0 オプションが指定されています。
2	数値の絶対値を取得します。	絶対値が使用されているため、マイナス記号に接頭辞を付けるために負数には別の範囲とピクチャが必要です。
3	数字に MULT=値を掛けます。MULT=オプションを指定しない場合、PICTURE ステートメントはデフォルト値を使用します。デフォルトは 10^n です。*n* は、ピクチャの小数点*の右側の数値セレクタの数です (ステップ 6 で数値セレクタについてさらに説明しています)。	MULT=値の指定は、0 から 1 までの数字と 0 から-1 までの数字に必要です。小数点がこれらの範囲のピクチャに表示されないためです。MULT=のデフォルトは 1 になるため、有効桁の切り捨てにより MULT=値が指定されません (切り捨てについては次のステップで説明します)。MULT=値が指定されていない 3 つの範囲に対し、MULT=値のデフォルトは 100 になります。対応するピクチャに小数点の右側に 2 つの数値セレクタが含まれているためです。MULT=値が適用された後、すべての有効桁は小数点の左側に移動されます。
4	数値が大きい整数の 10^{-8} 内である場合、数値は切り上げられます。この演算は、ROUND オプションが実行される前に実行されます。 ROUND オプションが有効になっています。この出力形式では、小数点の後の数字が.5 以上の場合、小数点の後の数字は次に大きい整数まで切り上げられます。	例ではすべての有効桁が小数点の左に移動されるようにする MULT=値を使用するため、有効桁は失われません。ゼロは切り捨てられます。 205.1 は、205 まで切り捨てられます。 55.6 は、56 に切り上げられます。 99.6 は、100 に切り上げられます。 四捨五入は 5 と 660 で実行されません。
4a	ROUND オプションが実行されない場合、小数点の後の数字が切り捨てられます。	205.1 は、205 に切り捨てられます。 55.6 は、55 に切り捨てられます。 99.6 は、99 に切り捨てられます。

ステップ	PICTURE ステートメントの処理のルール	例は次のとおりです。
5	数字を文字列に変換します。数字がピクチャよりも短い場合、文字列の長さはピクチャの数値セレクタの数と同じです。文字列に先頭のゼロを埋め込みます(結果は *Zw.* 出力形式の使用と同じになります。*Zw.* については、*SAS 出力形式と入力形式: リファレンス* の SAS 出力形式に関するセクションを参照してください)。	205 は、文字列 00205 になります。 5 は、文字列 05 になります。 56 は、文字列 56 になります。 100 は、文字列 100 になります。 660 は、文字列 0660 になります。 ピクチャが数字より長い場合、出力形式は先頭のゼロを各値に追加します。出力形式は先頭のゼロを文字列 56 と 100 に追加しません。これは、対応するピクチャに同じ数の数値セレクタが含まれているためです。
5a		205 は、文字列 00205 になります。 5 は、文字列 05 になります。 55 は、文字列 55 になります。 99 は、文字列 099 になります。 660 は、文字列 0660 になります。 ピクチャが数字より長い場合、出力形式は先頭のゼロを各値に追加します。出力形式は先頭のゼロを文字列 55 に追加しません。これは、対応するピクチャに同じ数の数値セレクタが含まれているためです。
6	文字列をピクチャに適用します。出力形式は文字列の最右の *n* 文字のみマップします。*n* はピクチャの数値セレクタの数です。そのため、ピクチャに文字列の文字に合うだけの十分な数値セレクタが含まれていることを確認することは重要です。 出力形式は最右の *n* 文字を取得した後、これらの文字をピクチャへ左から右にマップします。文字列に先頭のゼロが含まれている場合は、ゼロまたは非ゼロの数値セレクタを選択することが重要です。文字列の先頭のゼロのうちいずれかが非ゼロ数値セレクタにマップする場合、そのゼロとすべての後続の先頭のゼロおよびメッセージ文字は、フォーマットされた値の一部になります。先頭のゼロがすべてゼロ数値セレクタにマップする場合、先頭のゼロまたはメッセージ文字のいずれもフォーマットされた値の一部にはなりません。出力形式は文字列の先頭のゼロをブランクと置き換えます。**	00205 は、2.05 にマップされます。 05 は、05 にマップされます。 56 は、56 にマップされます。 100 は、1.00 にマップされます。 0660 は、6.60 にマップされます。 先頭のゼロは各文字列 **00205** と **0660** から削除されます。これは先頭ゼロがピクチャのゼロ数値セレクタにマップするためです。

ステップ	PICTURE ステートメントの処理のルール	例は次のとおりです。
6a		00205 は、2.05 にマップされます。 05 は、05 にマップされます。 55 は、55 にマップされます。 099 は、99 にマップされます。 0660 は、6.60 にマップされます。 先頭のゼロは各文字列 `00205`、`099`、`0660` から削除されます。これは先頭ゼロがピクチャのゼロ数値セレクタにマップするためです。 0.99 ピクチャのピリオド(.)メッセージ文字は削除されます。これは、先頭ゼロがゼロ数値セレクタにマップするためです。 ピリオドメッセージ文字が削除されるため、範囲 0.99 – < 1 の出力形式定義では、NOZEROS.出力形式で 10 進数をフォーマットするために'.'の接頭辞が必要です。
7	PREFIX=オプションで指定される文字に接頭辞を付けます。ピクチャに数値セレクタが含まれている場合、ピクチャは数値セレクタで始まる必要があるため、PREFIX=オプションは必要です。そのため、ピクチャを小数点、マイナス記号、または数値セレクタではない、その他の文字で開始することはできません。	PREFIX=オプションは、フォーマットされた値 `-2.05`、`-.05` および `.56` で表わされているように、小数点とマイナス記号を回復します。
7a		PREFIX=オプションは、フォーマットされた値 `-2.05`、`-.05`、`.55`、`.99` で表されているように、小数点とマイナス記号を回復します。

* PREFIX=オプションの小数点は、ピクチャの一部ではありません。

** FILL=オプションを使用して、フォーマットされた値の一部となるブランク以外の文字を指定できます。

表 27.3 *さまざまな値をフォーマットするためのステップ*

ステップ	アクション	-2.051	-.05	.556	.996	6.6
1	範囲	low – –1	–0.99 < – < 0	0 < – < .99	0.99 – < 1	1 – high
	ピクチャ	000.00	99	99	0.99	09.99
2	絶対値	2.051	.05	.556	.996	6.6
3	MULT=	$2.051\times10^2=205.1$	$.05\times100=5$	$.556\times100=55.6$	$.996\times100=99.6$	$6.6\times10^2=660$
4	四捨五入	205	5	56	100	660
4a	四捨五入なし	205	5	55	99	660

ステップ	アクション	-2.051	-.05	.556	.996	6.6
5	文字列、四捨五入	00205	05	56	100	0660
5a	文字列、四捨五入なし	00205	05	55	099	0660
6	テンプレート、四捨五入	2.05	05	56	1.00	6.60
6a	テンプレート、四捨五入なし	2.05	05	55	99	6.60
7	接頭辞、四捨五入	prefix='–'	prefix='–.'	prefix='.'	none	none
7a	接頭辞、四捨五入なし	prefix='–'	prefix='–.'	prefix='.'	prefix='.'	none
	フォーマットされた結果、四捨五入	-2.05	–.05	.56	1.00	6.60
	フォーマットされた結果、四捨五入なし	-2.05	–.05	.55	.99	6.60

次の PROC PRINT ステップは、NOZEROSR.出力形式と NOZEROS.出力形式を SAMPLE の AMOUNT 変数と関連付けます。最初の出力には、四捨五入の結果が表示されます。

```
proc print data=sample;
   format amount nozerosr.;
   title 'Formatting the Variable Amount';
   title2 'with the NOZEROSR. Format Using Rounding';
run;

proc print data=sample;
   format amount nozeros.;
   title 'Formatting the Variable Amount';
   title2 'with the NOZEROS. Format, No Rounding';
run;
```

Formatting the Variable Amount with the NOZEROSR. Format Using Rounding

Obs	Amount
1	-45.00
2	-2.05
3	-1.00
4	-.05
5	-.02
6	00
7	.09
8	.54
9	.56
10	1.00
11	6.60
12	14.63

Formatting the Variable Amount with the NOZEROS. Format, No Rounding

Obs	Amount
1	-45.00
2	-2.05
3	-.99
4	-.05
5	-.01
6	00
7	.09
8	.54
9	.55
10	.99
11	6.60
12	14.63

注意:
ピクチャには、接頭辞と数字に対して十分な幅がある必要があります。この例では、値-45.00 に NOZEROS.が適用された場合、最初の範囲 low - -1 内にあり、その範囲のピクチャは接頭辞の付いたマイナス記号と数字を表示するには広さが十分でないため、結果は 45.00 になります。

注意:
数字を四捨五入することにより桁が数字に追加される場合、ピクチャが追加される桁に対し十分広い必要があります。たとえば、数字.996 のピクチャは'99' (prefix '.' mult=100)になります。数字を四捨五入し、それに 100 を掛けると、結果の数字は 100 になります。ピクチャが適用されると、結果は不正確な数字、.00 となります。四捨五入するときに数字が正しいものになるように、ピクチャ幅をより大きな数字を表示するのに十分なものにしてください。

ピクチャなしの指定

この PICTURE ステートメントは、ピクチャを含まない *picture-name* 出力形式を作成します。

```
picture picture-name;
```

この出力形式を使用すると、デフォルト SAS 出力形式を値に適用する効果があります。

SELECT ステートメント

FMTLIB オプションと CNTLOUT=オプションによって処理するためのエントリを選択します。

制限事項: 1 つの SELECT ステートメントだけを PROC FORMAT ステップに表示できます。
SELECT ステートメントと EXCLUDE ステートメントを同じ PROC FORMAT ステップ内で使用することはできません。

例: "例 11: 入力形式と出力形式の説明の印刷" (907 ページ)

構文

SELECT *entry(s)*;

必須引数

entry(s)
処理対象のカタログエントリを 1 つ以上指定します。カタログエントリの名前は、保存する入力形式または出力形式の名前と同じです。入力形式と出力形式の名前、文字と数値の入力形式と出力形式の名前をそれぞれ同じにすることができるため、SELECT ステートメントで入力形式と出力形式を指定する際は特定の接頭辞を使用する必要があります。SELECT ステートメントでエントリを指定する際は、次の規則に従います。

- ドル記号($)の付いた文字出力形式を含むエントリ名を前に置きます。
- アットマークとドル記号の付いた文字出力形式を含むエントリ名(@$*entry-name* など)を前に置きます。
- アットマーク(@)の付いた数値入力形式を含むエントリ名を前に置きます。
- 接頭辞の付かない数値出力形式を含むエントリ名を指定します。

詳細

名前指定のショートカット

コロン(:)とハイフン(-)ワイルドカード文字を使用して、エントリを選択できます。たとえば、次の SELECT ステートメントは、文字 **a** で始まるすべての出力形式または入力形式を選択します。

```
select a:;
```

また、次の SELECT ステートメントは、**apple** から **pear** まで(両端の値を含む)にアルファベット順に発生するすべての出力形式または入力形式を選択します。

```
select apple-pear;
```

カタログが読み取り/更新モードで開かれる場合の FMTLIB と CNTLOUT=オプションの影響

SELECT ステートメントで FMTLIB オプションと CNTLOUT=オプションを使用し、カタログが読み込みモードまたは更新モードのいずれかで開かれるかを示します。次の規則が適用されます。

- SELECT ステートメントを使用し、FMTLIB オプションまたは CNTLOUT=オプションを指定しない場合、PROC FORMAT はカタログが更新モードで開かれているとみなします。
- SELECT ステートメントを使用し、FMTLIB オプションまたは CNTLOUT=オプションを指定する場合、カタログは読み取りアクセス向けに開かれています。
- SELECT ステートメントを FMTLIB オプションまたは CNTLOUT=オプションを指定せずに使用し、SAS プログラムにカタログへの書き込みアクセス権限がない場合、次のエラーが SAS ログに書き込まれます。

```
ERROR: User does not have appropriate authorization level
       for file libref.FORMATS.CATALOG.
```

VALUE ステートメント

変数値の印刷に使用する文字列を指定する出力形式を作成します。

参照項目: *SAS 出力形式と入力形式: リファレンス* で、SAS 出力形式に関するドキュメントについて参照してください。

例: “例 6: 文字値の出力形式の作成” (893 ページ)
“例 8: 標準 SAS 出力形式とカラー背景を使用した、日付の出力形式の書き出し” (898 ページ)
“例 13: 文字列の範囲の書き出し” (911 ページ)

構文

VALUE <$>*name* <(*format-option(s)*)> <*value-range-set(s)*>;

オプション引数の要約

DEFAULT=*length*
出力形式のデフォルト長を指定します。

FUZZ=*fuzz-factor*

値を範囲に一致させるための誤差定数を指定します。

MAX=*length*
出力形式に対し最大長を指定します。

MIN=*length*
出力形式に対し最小長を指定します。

MULTILABEL
複数のラベルまたは外部値を内部値に割り当てることができます。

NOTSORTED
定義順で値または範囲を保存します。

value-range-set(s)
フォーマットされた値への値または値範囲の割り当てを指定します。

必須引数

name
作成中の出力形式に名前を付けます。出力形式として使用するために FCMP プロシジャを使用して関数を作成した場合、*name* は、かっこのない関数名になります。

制限事項 ユーザー定義の出力形式の名前は、SAS によって提供されている出力形式の名前と同じにすることはできません。

出力形式名の最後を数字にすることはできません。

要件 名前は有効な SAS 名である必要があります。数値出力形式名の長さは最大 32 文字です。文字出力形式名の長さは最大 31 文字です。文字出力形式を作成している場合、ドル記号($)を最初の文字として使用します。

操作 出力形式名の最大長は、VALIDFMTNAME=システムオプションによって制御されます。詳細については、*SAS システムオプション: リファレンス*を参照してください。

ヒント 名前の後にピリオドを使用して、出力形式を後で参照します。ただし、VALUE ステートメントでは出力形式名の後ろにピリオドを使用しないでください。

参照項目 “関数を使用した値のフォーマッティング” (875 ページ)

オプション引数

DEFAULT=*length*
出力形式のデフォルト長を指定します。出力形式と変数を関連付ける際に特定の長さを指定しない場合、DEFAULT=の値が出力形式の長さになります。

デフォルト 等号の右側に割り当てられる最も長いラベルの長さ

範囲 1-32767

ヒント 最良の方法として、出力形式をラベルとして指定する場合は、常に DEFAULT=オプションを指定します。

FUZZ=*fuzz-factor*
値を範囲に一致させるための誤差定数を指定します。数字が範囲に完全には一致しない、または厳密には範囲内になくても *fuzz-factor* 内にあれば、出力形式は

それを一致とみなします。たとえば、次の VALUE ステートメントは誤差定数 2 を使用する LEVELS.出力形式を作成します。

```
value levels (fuzz=.2) 1='A'
                       2='B'
                       3='C';
```

FUZZ=.2 は、変数値が範囲のどちらかの端の値.2 内にある場合、出力形式が変数値の印刷に対応するフォーマットされた値を使用することを意味します。そのため、LEVELS.出力形式は値 2.1 を B としてフォーマットします。

デフォルト 1E−12 (数値出力形式)と 0 (文字出力形式)

ヒント VALUE ステートメントを使用して数値出力形式を作成するとき、FUZZ=0 を指定して記憶域を保存します。

ゼロ以外の誤差定数は、非常に近いが完全には一致しない数値でのみ使用します。バイナリ検索を実行するために、範囲は並べ替え順に内部保存されます(NOTSORTED オプションが使用されていない場合)。誤差定数が 1 つの範囲の終わりに追加され、次の範囲の先頭から削除れた場合に、これらの範囲が重なると、結果は予測不能なものになることがあります。値は、バイナリ検索で一致した最初の範囲に置かれます。除外演算子は、このバイナリ検索アルゴリズムを無効にするには不十分です。最良の方法として、除外演算子を使用する場合は、FUZZ=0 または NOTSORTED オプションを設定します。

最良の方法は、数値出力形式で<除外演算子を使用する場合は、FUZZ=0 を使用します。

MAX=*length*
出力形式に対し最大長を指定します。出力形式と変数を関連付ける際に、MAX=値を超える幅を指定できません。

デフォルト 40

範囲 1–32767

MIN=*length*
出力形式に対し最小長を指定します。

デフォルト 1

範囲 1–32767

MULTILABEL
複数のラベルまたは外部値を内部値に割り当てることができます。次の VALUE ステートメントでは、MULTILABEL オプションの 2 つの使用を示します。最初の VALUE ステートメントは複数のラベルを単一の内部値に割り当てます。複数のラベルは単一の内部値範囲に割り当てることもできます。2 番目の VALUE ステートメントは、ラベルを重複した内部値範囲に割り当てます。MULTILABEL オプションにより、複数のラベルを重複した内部値に割り当てることができます。

```
value one (multilabel)
    1='ONE'
    1='UNO'
    1='UN';
```

```
value agefmt (multilabel)
  15-29='below 30 years'
  30-50='between 30 and 50'
  51-high='over 50 years'
  15-19='15 to 19'
  20-25='20 to 25'
  25-39='25 to 39'
  40-55='40 to 55'
  56-high='56 and above';
```

PROC MEANS、PROC SUMMARY、PROC TABULATE などのマルチラベル有効プロシジャだけが、複数のラベルを使用できます。その他すべてのプロシジャと DATA ステップは、1 次ラベルのみ認識します。

指定エントリに対する 1 次ラベルは、最初の内部値に割り当てられる外部値、またはすべての内部値が順に並べ替えられるときにそのエントリに一致する、またはそのエントリを含む内部値の範囲です。次に例を示します。

- 最初の VALUE ステートメントでは、1 に対する 1 次ラベルは ONE です。ONE が 1 に割り当てられている最初の外部値であるためです。1 に対する 2 次ラベルは UNO と UN です。
- 2 番目の VALUE ステートメントでは、33 に対する 1 次ラベルは **`25 to 39`** です。範囲 25–39 が 33 を含む内部値の連続した最初の範囲であるためです。33 に対する 2 次ラベルは **`between 30 and 50`** です。範囲 30–50 が範囲 25–39 の後に連続して発生するためです。

制限事項 単一の出力形式に対して作成できるラベルの最大数は 255 です。

NOTSORTED

定義順で値または範囲を保存します。NOTSORTED を指定しない場合、値または範囲はデフォルトで並び替え順序で保存され、SAS では特定の値が含まれる範囲を位置付けるためのバイナリ検索アルゴリズムが使用されます。NOTSORTED を指定する場合、SAS は各範囲を一致するまで定義順に検索します。

次のうちいずれかが true の場合、NOTSORTED を使用します。

- 発生する特定範囲の尤度がわかっており、処理時間を削減するために、入力形式を使用してそれらの範囲を最初に検索する。
- FMTLIB オプションを使用して出力形式の説明を出力する際に、範囲を定義する順序を保持する。
- ORDER=DATA オプションと PRELOADFMT オプションを使用して PROC MEANS、PROC SUMMARY または PROC TABULATE のクラス変数を分析する際に、範囲を定義する順序を保持する。

値の分布が一様または不明な場合、または値の数が比較的小さい場合は、NOTSORTED を使用しないでください。NOTSORTED が指定されていないときに SAS が使用するバイナリ検索アルゴリズムにより、これらの条件下の検索のパフォーマンスが最適化されます。

CPORT プロシジャと CIMPORT プロシジャを使用して標準照合順序が異なる動作環境間で出力形式を転送する場合、NOTSORTED が自動的に使用されます。NOTSORTED の自動設定は、ASCII と EBCDIC の動作環境間で出力形式を転送する場合に発生します。この状況に問題がある場合、次を行います。

- PROC FORMAT ステートメントの CNTLOUT=オプションを使用し、出力制御データセットを作成します。
- CPORT プロシジャを使用して、制御データセットに対し転送ファイルを作成します。

- 対象の動作環境で CIMPORT プロシジャを使用して、転送ファイルをインポートします。
- 対象の動作環境で、PROC FORMAT を CNTLIN=オプションと使用し、インポートされた制御データセットから出力形式を作成します。

value-range-set(s)
フォーマットされた値への値または値範囲の割り当てを指定します。*value-range-set(s)*には、次の形式があります。

value-or-range-1 <, *value-or-range-2*, …>=*existing-format* | [*existing-format*]

等号の左側の変数値は、等号の右側に文字列として印刷します。等号の左側の各 *value-or-range* の最大長は 32,767 文字です。

value-or-range
value-or-range の指定方法については、“値や範囲の指定” (874 ページ)を参照してください。

formatted-value
等号の左側に表示される変数値の印刷値になる文字列を指定します。フォーマットされた値は、文字出力形式または数値出力形式を作成しているかどうかに関係なく、常に文字列です。

フォーマットされた値には、最大 32,767 文字可能です。16 進リテラルの場合、使用できるのは、最大 32,767 入力文字または最大 16,382 表現文字(表現文字ごとに 2 つの 16 進文字)です。ただし、一部のプロシジャではフォーマットされた値の最初の 8 文字または 16 文字だけを使用します。

要件 フォーマットされた値を一重引用符か二重引用符で囲む必要があります。次の例に、二重引用符で囲まれたフォーマットされた値を示します。

```
value $ score
   'M'="Male"
   'F'="Female";
```

フォーマットされた値に一重引用符が含まれている場合、値を二重引用符で囲みます。

```
value sect
   1="Smith's class"
   2="Leung's class";
```

ヒント 数値変数をフォーマットすると、これらの変数の使用は算術演算子に含まれません。SAS では、算術演算に対して保存された値が使用されます。

existing-format
SAS によって提供されている出力形式または既存するユーザー定義の出力形式を指定します。作成している出力形式では既存する出力形式を使用して、等号の左側の *value-or-range* に一致する生データを変換します。

既存する出力形式を使用することは、出力形式をネストすることと考えられます。ネストされたレベルとは、出力形式 B をフォーマットされた値として使用して出力形式 A を作成している場合、プロシジャは A を作成するために既存する出力形式を 1 つだけ使用する必要があるということです。

要件 既存する出力形式を使用する場合、その出力形式名を角かっこ([date9.]など)、またはかっこと縦棒((|date9.|)など)で囲みます。既存する出力形式の名前を一重引用符で囲まないでください。

ヒント 出力形式の複数レベルのネストは避けます。レベルの追加ごとにリソース要件が非常に増加する可能性があります。

最良の方法として、*value-range-set* の既存する出力形式を指定する場合、DEFAULT=オプション (869 ページ)を使用して常にデフォルト値を指定します。

例

例 1: 選択した州の郵便番号を印刷するための出力形式の作成

$STATE.文字出力形式は、選択した州の郵便番号を印刷します。

```
value $state 'Delaware'='DE'
             'Florida'='FL'
             'Ohio'='OH';
```

変数値 **Delaware** は **DE**、変数値 **Florida** は **FL**、変数値 **Ohio** は **OH** と印刷されます。$STATE.出力形式は、ドル記号で始まります。

注: 範囲指定は、大文字と小文字を区別します。上記の$STATE.出力形式では、値 **OHIO** は指定範囲のいずれにも一致しません。データ値が大文字、小文字であるかがわからない場合、1 つの解決策としてデータ値で UPCASE 関数を使用し、範囲にすべての大文字を指定します。

例 2: 数値を文字値として書き出す

数値出力形式 ANSWER.は、値 1 と 2 を yes と no にそれぞれ書き出します。

```
value answer 1='yes'
             2='no';
```

例 3: 範囲なしを指定

この VALUE ステートメントは、範囲が指定されていない *format-name* 出力形式を作成します。

```
value format-name;
```

この出力形式を使用すると、デフォルト SAS 出力形式を値に適用する効果があります。

例 4: ラベルとしての出力形式を使用する形式の作成

このプログラムでは、MYfmt.形式を作成し、該当する年に基づいて日付がフォーマットされます。

```
data test;
  do Date='01jan2006'd to '31dec2013'd;
    do j=1 to rannor(0)*100;
       output;
    end;
  end;
run;
proc format;
   value MYfmt
        /* Format dates prior to 31DEC2011 using only a year. */
        low-'31DEC2011'd=[year4.]

        /* Format 2012 dates using the month and year. */
```

```
        '01jan2012'd-'31DEC12'd=[monyy7.]

        /* Format dates 01JAN2013 and beyond using the day, month, and year. */
        '01JAN2013'd-high=[date9.]

        /* Catch missing values. */
        other='n/a';
run;

proc freq data=test;
      table date /missing;
      format date myfmt.;
run;
```

値や範囲の指定

ステートメント INVALUE、PICTURE、VALUE の構文で示されているように、値を *value-range-sets* として指定する必要があります。等号の左側で、その他の値に変換する値を指定します。等号の右側で、左側の値になる値を指定します。このセクションでは、等号の左側の値を表す *value-or-range* に使用できるさまざまな出力形式について説明します。等号の右側の値を指定する方法については、該当するステートメントの“必須引数”セクションを参照してください。

ステートメント INVALUE、PICTURE、VALUE では、等号の左側の数値を使用できます。文字入力形式では、数値範囲は文字列として扱われます。INVALUE と VALUE では、等号の左側の文字列を使用することもできます。

構文で示されているように、*value-or-range* の複数の発生を *value-range-set* ごとに、それぞれをカンマで区切ることができます。*value-or-range* の発生は、それぞれ次のうちいずれかです。

value
単一値(12、**'CA'**など)。文字出力形式および文字入力形式の場合、文字値を一重引用符で囲みます。

キーワード OTHER=を単一値として使用できます。OTHER は、その他の値または範囲に一致しないすべての値に一致します。出力形式が値をフォーマットする関数でない限り、出力形式を OTHER=の値として使用してユーザー定義の出力形式をネストできません。詳細については、“関数を使用した値のフォーマッティング” (875 ページ)を参照してください。例については、“例 6: 文字値の出力形式の作成” (893 ページ)と“例 16: 出力形式として使用する関数の作成” (920 ページ)を参照してください。

range
値のリスト(12–68、**'A'-'Z'**)など)。文字列を含む範囲の場合、各文字列を一重引用符で囲む必要があります。たとえば、A から Z までの文字列を含む範囲を指定する場合、**A** と **Z** を一重引用符で囲んで、**'A'-'Z'**を範囲として指定します。

'A-Z'を指定すると、プロシジャはそれを最初の文字が **A**、2 番目の文字がハイフン(-)、3 番目の文字が **Z** の 3 文字の文字列として解釈します。

数値ユーザー定義入力形式では、プロシジャは *value-range-set* の左側の引用符で囲まない数値範囲を数値範囲として解釈します。文字ユーザー定義入力形式では、プロシジャは **value-range-set** の左側の引用符で囲まない数値範囲を文字列として解釈します。たとえば、文字入力形式では、範囲 **12-86** は **'12'-'86'**と解釈されます。

範囲の 1 つの値として LOW または HIGH を使用できます。範囲 LOW-HIGH を使用してすべての値を含めることができます。たとえば、次は有効な範囲です。

```
low-'ZZ'
35-high
low-high
```

数値範囲で、LOW には、欠損値を除く、最小の数値が含まれます。HIGH には、範囲内で最大値が含まれます。文字範囲では、LOW に欠損値が含まれます。

未満(<)記号を使用して、値を範囲から除外できます。範囲の最初の値を除外する場合は、その値の後に<除外演算子を挿入します。範囲の最後の値を除外する場合は、その値の前に<除外演算子を挿入します。たとえば、次の範囲に 0 は含まれません。

```
0<-100
```

同様に、次の範囲に 100 は含まれません。

```
0-<100
```

ヒント <除外演算子を使用して範囲に値を挿入する場合、数値出力形式では VALUE ステートメントで FUZZ=0 オプションを使用します。文字出力形式では、FUZZ=0 がデフォルトであるためこの設定は不要です。

範囲の上限の値が別の範囲の下限に表示されていて、<除外演算子を使用しない場合、PROC FORMA はその値を最初の範囲に割り当てます。たとえば、次の範囲では、値 **AJ** は最初の範囲の一部です。

```
'AA'-'AJ'=1 'AJ'-'AZ'=2
```

この例では、値 **AJ** を 2 番目の範囲に含めるため、最初の範囲で<除外演算子を使用します。

```
'AA'-<'AJ'=1 'AJ'-'AZ'=2
```

範囲に値が重複する場合、VALUE ステートメントに MULTILABEL オプションが指定されていない限り、PROC FORMAT はエラーメッセージを返します。たとえば、範囲が **'AA'-'AK'=1 'AJ'-'AZ'=2** の場合、エラーが発生します。

value-or-range にはそれぞれ最大 32,767 文字が可能です。*value-or-range* が 32,767 文字を超える場合、プロシジャは最初の 32,767 文字を処理した後にその値を切り捨てます。

注: 等号の左側のすべての値を表示する必要はありません。これらの値は、デフォルトの入力形式または出力形式を使用して変換されます。たとえば、次の VALUE ステートメントは、98.6 のすべての発生を **NORMAL** と印刷する TEMP.出力形式を作成します。

```
value temp 98.6='NORMAL';
```

値が 96.9 の場合、印刷される結果は **96.9** となります。

関数を使用した値のフォーマッティング

SAS では、値で関数を最初に実行することにより、値をフォーマットする方法を提供しています。値をフォーマットする関数を使用して、カスタマイズされた出力形式を作成できます。たとえば、SAS では日付を四半期を使用してフォーマットする YYQ*w*.、YYQ*xw*.、YYQR*w*.、YYQR*xw*.の 4 つの出力形式を提供しています。これらの出力形式のいずれも Q1、Q2、Q3、Q4 を使用する要件を満たしていません。日付と SAS 出力形式に基づいて Q1、Q2、Q3、Q4 値を作成する FCMP プロシジャを使用して関数

を作成してから、その関数を FORMAT プロシジャで使用して出力形式を作成できます。

関数を作成し、それを使用して値をフォーマットするためのステップは、次のとおりです。

1. FCMP プロシジャを使用して、関数を作成します。
2. OPTIONS ステートメントを使用して、CMPLIB=システムオプションで関数の場所を指定することにより、関数を利用可能にします。
3. FORMAT プロシジャを使用して、新しい出力形式を作成します。
4. その新しい出力形式を SAS プログラムで使用します。

次に例を示します。

```
/* Create a function that creates the value Qx from a formatted value. */

proc fcmp outlib=work.functions.smd;

   function qfmt(date) $;
      length qnum $4;
      qnum=put(date,yyq4.);
      if substr(qnum,3,1)='Q'
         then return(substr(qnum,3,2));
      else return(qnum);
   endsub;
run;

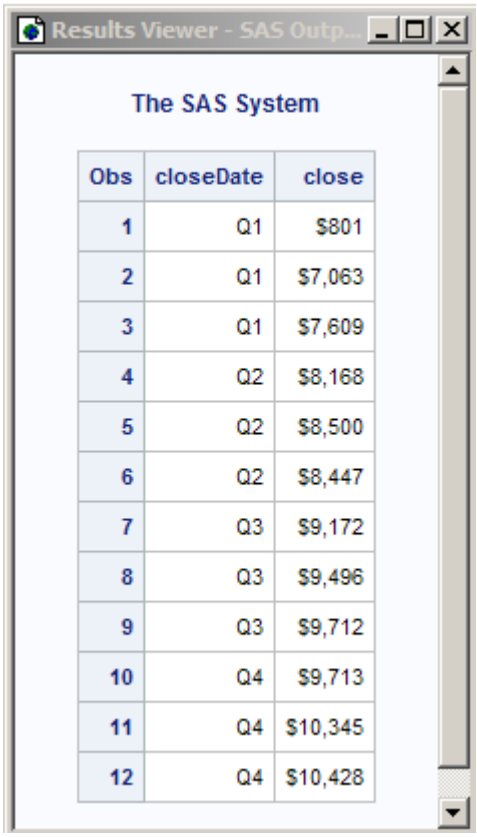
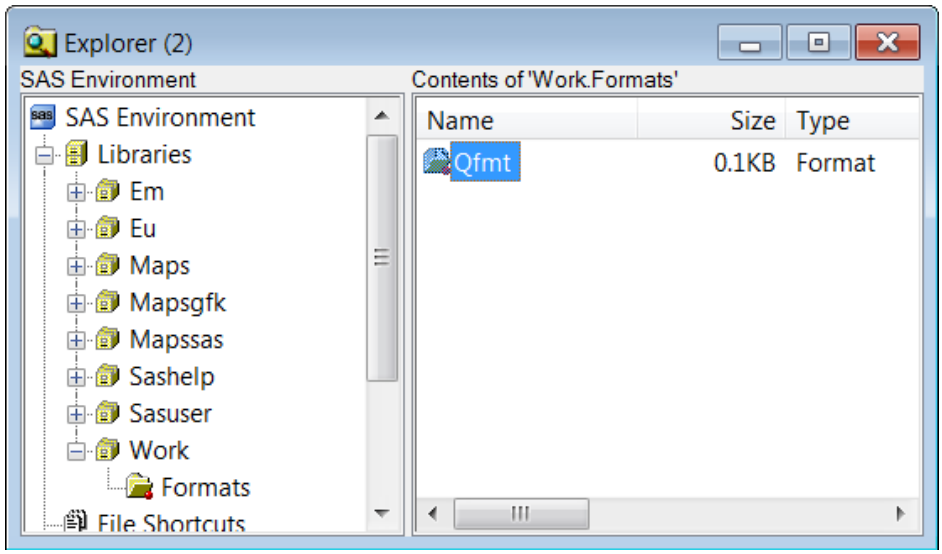
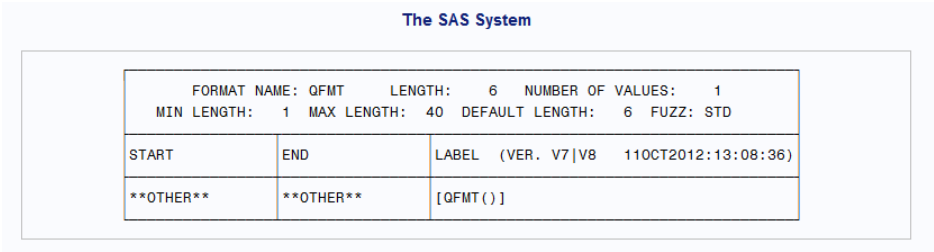
/* Make the function available to SAS. */

options cmplib=(work.functions);

/* Create a format using the function created by the FCMP procedure. */

proc format;
   value qfmt
         other=[qfmt()]; run;

/* Use the format in a SAS program.  */

data djia2013;
      input closeDate date7. close;
      datalines;
01jan13 800.86
02feb13 7062.93
02mar13 7608.92
01apr13 8168.12
01may13 8500.33
01jun13 8447.00
01jul13 9171.61
03aug13 9496.28
01sep13 9712.28
01oct13 9712.73
02nov13 10344.84
02dec13 10428.05
run;
```

```
proc print data=djia2013;
format closedate qfmt. close dollar9.;
run;
```

出力は次のとおりです。

アウトプット 27.1 *四半期ごとのダウジョーンズ終値*



Obs	closeDate	close
1	Q1	$801
2	Q1	$7,063
3	Q1	$7,609
4	Q2	$8,168
5	Q2	$8,500
6	Q2	$8,447
7	Q3	$9,172
8	Q3	$9,496
9	Q3	$9,712
10	Q4	$9,713
11	Q4	$10,345
12	Q4	$10,428

関数出力形式を OTHER=に対する値として使用できます。

SAS エクスプローラを使用して出力形式定義を表示する

出力形式を作成した後、SAS エクスプローラを使用して出力形式の定義を表示できます。

1. **表示** ⇨ **エクスプローラ**を選択します。
2. 出力形式フォルダを開きます。デフォルトの出力形式フォルダを表示するには、**ライブラリ** ⇨ **Work** の順に展開し、**出力形式**を選択します。



3. コンテンツペインの出力形式名の次のアクションのうちいずれかを実行します。
   - 出力形式名をダブルクリックする
   - 出力形式名を右クリックして、**表示を選択する**

   出力形式の説明が**結果ビューア**ウィンドウに表示されます。



The SAS System

FORMAT NAME: QFMT LENGTH: 6 NUMBER OF VALUES: 1 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 6 FUZZ: STD		
START	END	LABEL (VER. V7|V8 11OCT2012:13:08:36)
OTHER	**OTHER**	[QFMT()]

結果:FORMAT プロシジャ

出力制御データセット

出力制御データセットには、入力形式または出力形式を説明する情報が含まれています。出力制御データセットには、多くの使用があります。たとえば、出力制御データセットは、DATA ステップを使用して編集し、値範囲をプログラムで変更できます。または DATA ステップを使用してサブセット化し、新しい出力形式と入力形式を作成できます。また、出力制御データセットを作成し、CPORT プロシジャを使用してデータセットの転送ファイルを作成して、出力形式と入力形式を別の動作環境に移動できます。次に、移動先の動作環境で CIMPORT および FORMAT プロシジャを使用して、その環境に出力形式と入力形式を作成します。

PROC FORMAT ステートメントの CNTLOUT=オプションを使用して、出力制御データセットを作成します。出力制御データット、または出力制御データセットからの一連のオブザベーションを、CNTLIN=オプションを指定した後続の PROC FORMAT ステップの入力制御データセットとして使用できます。

出力制御データセットには、LIBRARY=カタログの各入力形式または各出力形式の各値または各範囲に対するオブザベーションが含まれています。データセットは、PROC FORMAT ステップで作成された各出力形式と各入力形式に関するグローバル情報、または各範囲と各値に関する特定情報を提供する変数から構成されています。

出力制御データセットの変数は、次のとおりです。

DEFAULT
出力形式または入力形式のデフォルト長を示す数値変数を指定します。

END
範囲の終了値を提供する文字変数を指定します。

EEXCL
範囲の終了値が除外されるかどうかを示す文字変数を指定します。有効な値は次のとおりです。

Y
範囲の終了値が除外されるように指定します。

N
範囲の終了値が除外されないように指定します。

FILL
ピクチャ形式に対し、値が FILL=オプションの値である数値変数を指定します。

FMTNAME
値が出力形式名または入力形式名である文字変数を指定します。

FUZZ
値が FUZZ=オプションの値である数値変数を指定します。

HLO
出力形式または入力形式に関する範囲情報を含む文字変数を指定します。次の有効値は、組み合わせで表示できます。

F
値と使用される標準 SAS 出力形式または入力形式を指定します。

H
範囲の終了値が HIGH になるように指定します。

I
数値入力形式範囲を指定します。

J
入力形式の位置合わせを指定します。

L
範囲の開始値が LOW になるように指定します。

M
MULTILABEL オプションが有効になるように指定します。

N
出力形式または入力形式に範囲が含まれないように指定します。OTHER=範囲なしを含みます。

O
範囲が OTHER になるように指定します。

R
ROUND オプションが有効になるように指定します。

S
NOTSORTED オプションが有効になるように指定します。

U
UPCASE オプションが入力形式に使用されるように指定します。

LABEL
値が出力形式または入力形式と関連付けられている文字変数を指定します。

LENGTH
値が LENGTH=オプションの値である数値変数を指定します。

MAX
値が MAX=オプションの値である数値変数を指定します。

MIN
値が MIN=オプションの値である数値変数を指定します。

MULT
値が MULT=オプションの値である数値変数を指定します。

NOEDIT
ピクチャ形式に対し、値が NOEDIT オプションが有効であるかどうかを示す数値変数を指定します。有効な値は次のとおりです。

1
NOEDIT オプションが有効になるように指定します。

0
NOEDIT オプションが有効にならないように指定します。

PREFIX
ピクチャ形式に対し、値が PREFIX=オプションの値である文字変数を指定します。

SEXCL
範囲の開始値が除外されるかどうかを示す文字変数を指定します。有効な値は次のとおりです。

Y
範囲の開始値が除外されるように指定します。

N
範囲の開始値が除外されないように指定します。

START
範囲の開始値を提供する文字変数を指定します。

TYPE
出力形式の種類を示す文字変数を指定します。可能な値は次のとおりです。

C
文字出力形式を指定します。

I
数値入力形式を指定します。

J
文字入力形式を指定します。

N
数値出力形式(ピクチャを除く)を指定します。

P
ピクチャ形式を指定します。

次の出力に、FORMAT プロシジャの例で作成されたすべての入力形式と出力形式に関する情報を含む出力制御データセットを示します。

アウトプット 27.2 PROC FORMAT 例の出力制御データセット

Results Viewer - SAS Output

An Output Control Data Set

Obs	FMTNAME	START	END	LABEL	MIN	MAX	DEFAULT	LENGTH	FUZZ	PREFIX	MULT	FILL	NOEDIT	TYPE	SEXCL	EEXCL	HLO	DECSEP	DIG3SEP	DATATYPE	LANGUAGE
1	BENEFIT	LOW	14609	WORDDATE20.	1	40	20	20	1E-12		0.00		0	N	N	N	LF				
2	BENEFIT	14610	HIGH	** Not Eligible **	1	40	20	20	1E-12		0.00		0	N	N	N	H				
3	SALARY	LOW	HIGH	00,000,000.00	1	40	13	13	1E-12	$	100.00	*	0	P	N	N	LH	.	,		
4	USCURRENCY	LOW	HIGH	000,000	1	40	7	7	1E-12	$	1.61		0	P	N	N	LH	.	,		
5	CITY	BR1	BR1	Birmingham UK	1	40	14	14	0		0.00		0	C	N	N					
6	CITY	BR2	BR2	Plymouth UK	1	40	14	14	0		0.00		0	C	N	N					
7	CITY	BR3	BR3	York UK	1	40	14	14	0		0.00		0	C	N	N					
8	CITY	US1	US1	Denver USA	1	40	14	14	0		0.00		0	C	N	N					
9	CITY	US2	US2	Miami USA	1	40	14	14	0		0.00		0	C	N	N					
10	CITY	**OTHER**	**OTHER**	INCORRECT CODE	1	40	14	14	0		0.00		0	C	N	N	O				
11	EVALUATION	C	C	1	1	40	1	1	0		0.00		0	I	N	N					
12	EVALUATION	E	E	2	1	40	1	1	0		0.00		0	I	N	N					
13	EVALUATION	N	N	0	1	40	1	1	0		0.00		0	I	N	N					
14	EVALUATION	O	O	4	1	40	1	1	0		0.00		0	I	N	N					
15	EVALUATION	S	S	3	1	40	1	1	0		0.00		0	I	N	N					

SELECT ステートメントまたは EXCLUDE ステートメントを使用して、出力制御データセットで表される出力形式または入力形式を制御できます。詳細については、“SELECT ステートメント” (867 ページ)と “EXCLUDE ステートメント” (843 ページ)を参照してください。

入力制御データセット

PROC FORMAT ステートメントの CNTLIN=オプションを使用して、入力制御データセットを指定します。FORMAT プロシジャは入力制御データセットのデータを使用して、入力形式と出力形式を作成します。そのため、INVALUE ステートメント、PICTURE ステートメント、VALUE ステートメントを記述せずに入力形式と出力形式を作成できます。

入力制御データセットに、次の特性が含まれている必要があります。

- 数値出力形式と文字出力形式の両方に対し、データセットに変数 FMTNAME、START、LABEL が含まれている必要があります。これらの変数については、“出力制御データセット” (878 ページ)を参照してください。その他の変数は、常に必要であるというわけではありません。
- 文字出力形式と文字入力形式を作成している場合、出力形式名または入力形式名のいずれかをドル記号($)で始めるか、値が `C` の TYPE 変数を指定する必要があります。
- PICTURE ステートメント出力形式を作成している場合、値 `P` の TYPE 変数を指定する必要があります。
- 入力値の範囲を含む出力形式を作成する場合、END 変数を指定する必要があります。範囲値が非包含の場合、変数 SEXCL と EEXCL の値はそれぞれ `Y` である必要があります。包含がデフォルトです。

各出力形式に対するオブザベーションがグループ化されると、入力制御データセットから複数の出力形式を作成できます。

CNTLIN=オプションが指定されている同じ PROC FORMAT ステップで、VALUE ステートメント、INVALUE ステートメント、または PICTURE ステートメントを使用できます。VALUE ステートメント、INVALUE ステートメント、または PICTURE ステートメントは CNTLIN=オプションが作成しているものと同じ入力形式または出力形式を作成している場合、入力形式または出力形式を作成し、CNTLIN=データセットは使用されません。ただし、VALUE、INVALUE または PICTURE を使用して入力形式または出力形

式を作成し、同じ PROC FORMAT ステップで CNTLIN=を使用して異なる入力形式または出力形式を作成できます。

入力制御データセットに関する例については、“例 10: データセットからの出力形式の作成” (903 ページ)を参照してください。

プロシジャの出力

FORMAT プロシジャは、PROC FORMAT ステートメントで FMTLIB オプションまたは PAGE オプションを指定するときだけ、出力を印刷します。出力は、LIBRARY=オプションで指定されるカタログの各出力形式エントリまたは各入力形式エントリに対するテーブルです。出力には、グローバル情報、出力形式または入力形式に対して定義される各値または範囲に関する詳細も含まれています。SELECT ステートメントまたは EXCLUDE ステートメントを使用して、FMTLIB 出力で表される出力形式と入力形式を制御できます。詳細については、“SELECT ステートメント” (867 ページ)と“EXCLUDE ステートメント” (843 ページ)を参照してください。例については、“例 11: 入力形式と出力形式の説明の印刷” (907 ページ)を参照してください。

次の出力で表示される FMTLIB 出力には、“例 6: 文字値の出力形式の作成” (893 ページ)で作成される$CITY.出力形式と、“例 9: 生の文字データを数値に変換する” (900 ページ)で作成される EVALUATION.入力形式の説明が含まれています。

アウトプット 27.3 *FMTLIB オプションによる PROC FORMAT からの出力*

FORMAT NAME: $CITY LENGTH: 14 NUMBER OF VALUES: 6
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 14 FUZZ: 0

START	END	LABEL (VER. V7|V8 12OCT2012:11:33:26)
BR1	BR1	Birmingham UK
BR2	BR2	Plymouth UK
BR3	BR3	York UK
US1	US1	Denver USA
US2	US2	Miami USA
OTHER	**OTHER**	INCORRECT CODE

INFORMAT NAME: @EVALUATION LENGTH: 1
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 1 FUZZ: 0

START	END	INVALUE(VER. 9.4 12OCT2012:11:34:18)
C	C	1
E	E	2

INFORMAT NAME: @EVALUATION LENGTH: 1
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 1 FUZZ: 0

START	END	INVALUE (CONT'D)
N	N	0
O	O	4
S	S	3

フィールドは次に、出力に表示される順序で左から右に説明されます。

INFORMAT NAME または **FORMAT NAME**
入力形式または出力形式の名前です。入力形式名はアットマーク(@)で始まります。

LENGTH
入力形式または出力形式の長さです。PROC FORMAT は長さを次の方法で決定します。

- 文字入力形式の場合、LENGTH の値は、等号の左側の最も長い生データ値の長さです。
- 数値入力形式の場合、次が true です。
  - 等号の左側のすべての値が数値の場合、LENGTH は 12 です。
  - LENGTH は、等号の左側の最も長い生データ値と同じになります。

- 出力形式の場合、LENGTH の値は等号の右側の最も長い値の長さになります。

$CITY.の出力では LENGTH は 14 です。最も長いピクチャが 14 文字であるためです。

@EVALUATION.の出力では長さは 1 です。1 が等号の左側の最も長い生データ値であるためです。

NUMBER OF VALUES
入力形式または出力形式と関連付けられている値または範囲の数。NOZEROS.には 4 つの範囲、EVAL.には 5 つの範囲がそれぞれ含まれています。

MIN LENGTH
入力形式または出力形式の最小長。MIN=オプションで異なる最小長を指定しない限り、MIN LENGTH の値は 1 です。

MAX LENGTH
入力形式または出力形式の最大長。MAX=オプションで異なる最大長を指定しない限り、MAX LENGTH の値は 40 です。

DEFAULT LENGTH
INVALUE フィールドまたは LABEL フィールドの最長値、または DEFAULT=オプションの値。

FUZZ
誤差定数。入力形式の場合、FUZZ は常に 0 です。出力形式の場合、FUZZ=オプションを使用しない場合はこのフィールドの値は STD です。STD はデフォルトの誤差値を表します。

START
範囲の開始値。FMTLIB は、START 列と END 列の値の最初の 16 文字だけを印刷します。

END
範囲の終了値。値が範囲から除外されると、除外記号(<)が START と END の値の後に表示されます。

INVALUE
入力形式に対してのみ表示され、入力形式のある値が含まれます。

LABEL
LABEL は入力形式に対してのみ表示され、フォーマットされた値またはピクチャのいずれかが含まれます。SAS バージョン番号と、出力形式または入力形式が作成された日付は、INVALUE または LABEL の後にかっこで囲まれます。

注: バージョン番号 V7|V8 が表示される場合、出力形式はこれらのバージョンと互換性があります。これらの以前のリリースと互換性がない場合、出力形式を作成したリリースが表示されます。バージョン V9 では長い(8 文字を超える)出力形式名と入力形式名をサポートしていますが、V7|V8 ではサポートしていません。

NOZEROS.などのピクチャ形式の場合、LABEL セクションには PREFIX=、FILL=および MULT=の値が含まれています。これらの値を表記するため、FMTLIB は各オプションを表すための文字 **P**、**F**、および **M** の後に値を印刷します。たとえば、LABEL セクションでは、**P-.**は、接頭辞値はハイフンで、その後にピリオドが続くことを示します。

FMTLIB は、LABEL 列の 40 文字のみ印刷します。

例: FORMAT プロシジャ

例 1: サンプルデータセットの作成

詳細

このセクションのいくつかの例では、PROCLIB.STAFF データセットを使用します。また、これらの例で作成される入力形式と出力形式の多くは Library.Formats に保存されます。“出力制御データセット” (878 ページ)で示される出力データセットには、これらの入力形式と出力形式の説明が含まれています。

変数は、アメリカ合衆国とイギリスにサイトのある企業に勤務する従業員の小さなサブセットに関するものです。データには、各従業員に対する名前、ID 番号、給与(イギリスポンド)、場所、雇用日が含まれています。

プログラム

```
libname proclib 'SAS-library';

data proclib.staff;
   infile datalines dlm='#';
   input Name & $16. IdNumber $ Salary
         Site $ HireDate date8.;
   format hiredate date8.;
   datalines;
Capalleti, Jimmy#  2355# 21163# BR1# 30JAN13
Chen, Len#         5889# 20976# BR1# 18JUN06
Davis, Brad#       3878# 19571# BR2# 20MAR04
Leung, Brenda#     4409# 34321# BR2# 18SEP94
Martinez, Maria#   3985# 49056# US2# 10JAN93
Orfali, Philip#    0740# 50092# US2# 16FEB03
Patel, Mary#       2398# 35182# BR3# 02FEB90
Smith, Robert#     5162# 40100# BR5# 15APR06
Sorrell, Joseph#   4421# 38760# US1# 19JUN11
Zook, Carla#       7385# 22988# BR3# 18DEC10
;
```

プログラムの説明

```
libname proclib 'SAS-library';
```

データセット PROCLIB.STAFF を作成します。 INPUT ステートメントは、名前 Name、IdNumber、Salary、Site、HireDate を DATALINES ステートメントの後に表示される変数に割り当てます。FORMAT ステートメントは、標準 SAS 出力形式 DATE7.を変数 HireDate に割り当てます。

```
data proclib.staff;
   infile datalines dlm='#';
   input Name & $16. IdNumber $ Salary
         Site $ HireDate date8.;
   format hiredate date8.;
```

```
   datalines;
Capalleti, Jimmy#  2355# 21163# BR1# 30JAN13
Chen, Len#         5889# 20976# BR1# 18JUN06
Davis, Brad#       3878# 19571# BR2# 20MAR04
Leung, Brenda#     4409# 34321# BR2# 18SEP94
Martinez, Maria#   3985# 49056# US2# 10JAN93
Orfali, Philip#    0740# 50092# US2# 16FEB03
Patel, Mary#       2398# 35182# BR3# 02FEB90
Smith, Robert#     5162# 40100# BR5# 15APR06
Sorrell, Joseph#   4421# 38760# US1# 19JUN11
Zook, Carla#       7385# 22988# BR3# 18DEC10
;
```

例 2: ピクチャ形式の作成

要素: PROC FORMAT ステートメントオプション
LIBRARY=
PICTURE ステートメントオプション
MULT=
PREFIX=
LIBRARY ライブラリ参照名
LOW キーワードと HIGH キーワード

データセット: 例 1 の PROCIB.STAFF

詳細

この例では、PICTURE ステートメントを使用して、データセット PROCLIB.STAFF の変数 Salary の値を U.S.ドルで印刷する出力形式を作成します。

プログラム

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';

options nodate pageno=1 linesize=80 pagesize=40;

   proc format library=library;

      picture uscurrency low-high='000,000' (mult=1.61 prefix='$');
   run;

proc print data=proclib.staff noobs label;

   label salary='Salary in U.S. Dollars';
   format salary uscurrency.;

   title 'PROCLIB.STAFF with a Format for the Variable Salary';
run;
```

プログラムの説明

2 つの SAS ライブラリ参照(PROCLIB と LIBRARY)を割り当てます。 この場合、ライブラリ参照 LIBRARY を割り当てると便利です。PROC FORMAT を使用すると、LIBRARY

ライブラリ参照名で参照されるライブラリの入力形式と出力形式が自動的に検索されるためです。

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';
```

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

ユーザー定義の出力形式がカタログ Library.Formats に保存されるように指定します。 LIBRARY=オプションは、PROC FORMAT で作成する出力形式または入力形式を含む SAS カタログを指定します。LIBRARY という名前のライブラリを作成する際、LIBRARY 内に FORMATS という名前のカタログが自動的に作成されます。

```
    proc format library=library;
```

USCURRENCY.ピクチャ形式を定義します。 PICTURE ステートメントは、数字を印刷するためのテンプレートを作成します。LOW-HIGH によりすべての値が範囲に含まれます。MULT=ステートメントオプションは、各値に 1.61 を掛けるように指定します。PREFIX=ステートメントは、US ドル記号をフォーマットする数字に追加します。ピクチャには数値セレクタが給与に 5 つ、ドル記号接頭辞に 1 つ、合計 6 つ含まれています。

```
       picture uscurrency low-high='000,000' (mult=1.61 prefix='$');
    run;
```

PROCLIB.STAFF データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。LABEL オプションは列ヘッダーの変数名ではなく、変数ラベルを使用します。

```
proc print data=proclib.staff noobs label;
```

Salary 変数に対しラベルと出力形式を指定します。 LABEL ステートメントは、レポートの変数に対する特定のラベルを置き換えます。この場合、“Salary in US Dollars”はこの印刷ジョブに対してのみ変数 Salary に置き換えられます。FORMAT ステートメントは、このプロシジャステップの期間に対し USCurrency.出力形式と変数名 Salary を関連付けます。

```
    label salary='Salary in U.S. Dollars';
    format salary uscurrency.;
```

タイトルを指定します。

```
    title 'PROCLIB.STAFF with a Format for the Variable Salary';
run;
```

出力

アウトプット 27.4 *変数 Salary の出力形式を含む PROCLIB.STAFF*

PROCLIB.STAFF with a Format for the Variable Salary

Name	IdNumber	Salary in U.S. Dollars	Site	HireDate
Capalleti, Jimmy	2355	$34,072	BR1	30JAN09
Chen, Len	5889	$33,771	BR1	18JUN06
Davis, Brad	3878	$31,509	BR2	20MAR04
Leung, Brenda	4409	$55,256	BR2	18SEP94
Martinez, Maria	3985	$78,980	US2	10JAN93
Orfali, Philip	0740	$80,648	US2	16FEB03
Patel, Mary	2398	$56,643	BR3	02FEB90
Smith, Robert	5162	$64,561	BR5	15APR06
Sorrell, Joseph	4421	$62,403	US1	19JUN11
Zook, Carla	7385	$37,010	BR3	18DEC10

例 3: 大きなドル金額のピクチャ形式の作成

要素: PICTURE ステートメントオプション
MULT

出力形式: BIGMONEY.

詳細

この例では、PICTURE ステートメントの MULT オプションを使用して、それぞれ 100 万ドル、億ドル、または兆ドルを示す M、B、T を表示するドルをフォーマットします。この例では、出力形式定義で指数表記と小数点表記を使用します。

ヒント セントを含まないドル金額を使用するため、四捨五入は不要です。ドル金額にセントが含まれる場合は、PICTURE ステートメントで ROUND オプションを使用して金額を四捨五入します。詳細については、“ROUND” (856 ページ)を参照してください。

プログラム

```
proc format;
   picture bigmoney (fuzz=0)
      1E06-<1000000000='0000 M' (prefix='$' mult=.000001)
      1E09-<1000000000000='0000 B' (prefix='$' mult=1E-09)
      1E12-<1000000000000000='0000 T' (prefix='$' mult=1E-012);
run;
```

```
data mult;
   do i=5 to 12;
      x=16**i;
      put x=comma20. x= bigmoney.;
   end;
run;
```

プログラムの説明

BIGMONEY 出力形式を作成します。 BIGMONEY.出力形式では、100 万ドル、億ドル、兆ドルをフォーマットする 3 つの value-range sets を定義します。1E06 は 100 万、1E09 は 1 億、1E12 は 1 兆です。<除外演算子で、範囲に後続の数値を含めるかどうかを指定します。除外演算子を使用して正確な数値になるようには、FUZZ=0 を使用するのが最もよい方法です。100 万ドルの場合、範囲は 1,000,000 から 999,999,999 です。等号の右側で指定されるラベルでは、4 つのゼロと数値セレクタを使用します。ゼロの数値セレクタでは、先頭のゼロを印刷しないことを指定します。最初の数値セレクタは、値が 3 桁の場合に$接頭辞記号を印刷するために必要です。MULT=オプションの値を.000001 に設定して、1E-06 (100 万分の 1)を作成することもできます。値に、100 万分の 1、1 億分の 1、1 兆分の 1 の乗数を掛けると、100 万ドル、億ドル、兆ドルの数値が返されます。

```
proc format;
   picture bigmoney (fuzz=0)
      1E06-<1000000000='0000 M' (prefix='$' mult=.000001)
      1E09-<1000000000000='0000 B' (prefix='$' mult=1E-09)
      1E12-<1000000000000000='0000 T' (prefix='$' mult=1E-012);
run;
```

ドルとしてフォーマットする大きな数字を生成します。

```
data mult;
   do i=5 to 12;
      x=16**i;
      put x=comma20. x= bigmoney.;
   end;
run;
```

LOG

ログ 27.1 *フォーマットされた 100 万ドル、億ドル、兆ドル金額*

```
x=1,048,576 x=$1 M x=16,777,216 x=$16 M x=268,435,456 x=$268 M x=4,294,967,296 x=
$4 B x=68,719,476,736 x=$68 B x=1,099,511,627,776 x=$1 T x=17,592,186,044,416 x=
$17 T x=281,474,976,710,656 x=$281 T
```

プログラム

```
proc format;
   picture bigmoney (fuzz=0)
      1E06-<1000000000='0000.99 M' (prefix='$' mult=.0001)
      1E09-<1000000000000='0000.99 B' (prefix='$' mult=1E-07)
      1E12-<1000000000000000='0000.99 T' (prefix='$' mult=1E-010);
run;

data mult;
   do i=5 to 12;
```

```
      x=16**i;
      put x=comma20. x= bigmoney.;
   end;
run;
```

プログラムの説明

このプログラムでは、BIGMONEY.出力形式を変更し、小数点値を追加してより精度の高い数値を表示します。

BIGMONEY 出力形式を変更します。 より精度の高い数値を表示するには、ピクチャ値および MULT=値を変更します。2 桁の小数点値を表示するには、ピクチャに.99 を追加します。2 桁の小数点値を計算するには、MULT=オプションの値を 100 万分の 1 から 1 万分の 1 に減らします。16^5 に.0001 を掛けると、結果は 104.8576 になります。小数点値は切り捨てられ、ピクチャの右側から 104 が配置されます。フォーマットされた結果の値は 1.04 M です。

```
proc format;
   picture bigmoney (fuzz=0)
      1E06-<1000000000='0000.99 M' (prefix='$' mult=.0001)
      1E09-<1000000000000='0000.99 B' (prefix='$' mult=1E-07)
      1E12-<1000000000000000='0000.99 T' (prefix='$' mult=1E-010);
run;
```

ドルとしてフォーマットする大きな数字を生成します。

```
data mult;
   do i=5 to 12;
      x=16**i;
      put x=comma20. x= bigmoney.;
   end;
run;
```

LOG

ログ 27.2 *より高い精度でフォーマットされた大きなドル金額*

```
x=1,048,576 x=$1.04 M x=16,777,216 x=$16.77 M x=268,435,456 x=$268.43 M
x=4,294,967,296 x=$4.29 B x=68,719,476,736 x=$68.71 B x=1,099,511,627,776 x=
$1.09 T x=17,592,186,044,416 x=$17.59 T x=281,474,976,710,656 x=$281.47 T
```

例 4: ピクチャ形式の埋め込み

要素: PICTURE ステートメントオプション
FILL=
PREFIX=

詳細

この例では、次のタスクを実行します。

- フォーマットされた値への指定文字の接頭辞指定
- 先頭のブランクへの指定文字の入力

- FILL=オプションと PREFIX=オプションの間の相互作用の表示

プログラム

```
data pay;
   input Name $ MonthlySalary;
   datalines;
Liu   1259.45
Lars  1289.33
Kim   1439.02
Wendy 1675.21
Alex  1623.73
;

proc format;
   picture salary low-high='00,000,000.00' (fill='*' prefix='$');
run;

proc print data=pay noobs;
   format monthlysalary salary.;

   title 'Printing Salaries for a Check';
run;
```

プログラムの説明

PAY データセットを作成します。 PAY データセットには、各従業員に対する月次給与が含まれます。

```
data pay;
   input Name $ MonthlySalary;
   datalines;
Liu   1259.45
Lars  1289.33
Kim   1439.02
Wendy 1675.21
Alex  1623.73
;
```

SALARY.ピクチャ形式を定義し、ピクチャの入力方法を指定します。 FILL=および PREFIX= PICTURE ステートメントオプションが同一のピクチャに表示されるとき、出力形式は接頭辞を置いてから、埋め字を入力します。SALARY.出力形式はピクチャに埋め字を入力しますが、これはピクチャに数値セレクタとしてゼロが含まれているためです。ピクチャの最左のカンマは埋め字と置き換えられます。

```
proc format;
   picture salary low-high='00,000,000.00' (fill='*' prefix='$');
run;
```

PAY データセットを印刷します。 NOOBS オプションはオブザベーション番号の印刷を行いません。FORMAT ステートメントは、SALARY.出力形式と変数 MonthlySalary を一時的に関連付けます。

```
proc print data=pay noobs;
   format monthlysalary salary.;
```

タイトルを指定します。

```
    title 'Printing Salaries for a Check';
run;
```

出力

アウトプット 27.5 *給与の確認用印刷*

Printing Salaries for a Check

Name	Monthly Salary
Liu	****$1,259.45
Lars	****$1,289.33
Kim	****$1,439.02
Wendy	****$1,675.21
Alex	****$1,623.73

例 5: 24 時間形式から 00:00:01–24:00:00 形式への変更

要素: PICTURE ステートメントオプション
DATATYPE=DATETIME_UTIL

詳細

午前 0 時を 24:00 と表現する必要がある場合、または日時の時間範囲 00:00:01–24:00:00 を使用する必要がある場合。DATATYPE=DATETIME と設定した場合の時間値の範囲は 00:00:00 から 23:59:59 です。この例では、オプション DATATYPE=DATETIME_UTIL を使用して 00:00:01 から 24:00:00 の範囲で時間を表現し、00:00:00 を使用した場合日付が変更される様子を示します。

プログラム

```
proc format;
  picture  hour (default=19)
          other='%Y-%0m-%0d %0H:%0M:%0S' (datatype=datetime_util);
run;

data _null_;
  x = '01jul2015:00:00:01'dt; put x=hour.;
  x = '01jul2015:00:00:00'dt; put x=hour.;
run;
```

プログラムの説明

DATATYPE=DATETIME_UTIL オプションを設定して、時間範囲 00:00:01–24:00:00 を使用します。

```
proc format;
```

```
  picture  hour (default=19)
          other='%Y-%0m-%0d %0H:%0M:%0S' (datatype=datetime_util);
run;
```

日付の値を比較します。最初の日時値は 00:00:01 の範囲内にあり、日付は July 1 と示されます。2 番目の日時値は 00:00:01 から 24:00:00 の範囲内にないため、前の日の午前 0 時として結果が表示されます。

```
data _null_;
  x = '01jul2015:00:00:01'dt; put x=hour.;
  x = '01jul2015:00:00:00'dt; put x=hour.;
run;
```

ログ

ログ 27.3 *日付範囲 00:00:01–24:00:00 の使用*

```
x=2015-07-01 00:01:00 x=2015-06-30 24:00:00
```

例 6: 文字値の出力形式の作成

要素: VALUE ステートメントオプション
OTHER

データセット: PROCLIB.STAFF

出力形式: 例 2 の USCURRENCY.

詳細

この例では、VALUE ステートメントを使用して、文字変数の値を異なる文字列として印刷する文字出力形式を作成します。

プログラム

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';


   proc format library=library;

      value  $city 'BR1'='Birmingham UK'
                   'BR2'='Plymouth UK'
                   'BR3'='York UK'
                   'US1'='Denver USA'
                   'US2'='Miami USA'
                   other='INCORRECT CODE';
   run;

proc print data=proclib.staff noobs label;

   label salary='Salary in U.S. Dollars';

   format salary uscurrency. site $city.;

   title 'PROCLIB.STAFF with a Format for the Variables';
```

```
   title2 'Salary and Site';
run;
```

プログラムの説明

---

2 つの SAS ライブラリ参照(PROCLIB と LIBRARY)を割り当てます。 この場合、ライブラリ参照 LIBRARY を割り当てると便利です。PROC FORMAT を使用すると、LIBRARY ライブラリ参照名で参照されるライブラリの入力形式と出力形式が自動的に検索されるためです。

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';
```

ユーザー定義の出力形式が保存される Library.Formats という名前のカタログを作成します。 LIBRARY=オプションは作成する出力形式に対し永久保存場所を指定します。指定されたライブラリに FORMAT という名前のカタログも作成します。LIBRARY=を使用しない場合、作成する出力形式と入力形式は Work.Formats というカタログに一時的に保存されます。

```
proc format library=library;
```

$CITY.出力形式を定義します。 特殊コード BR1、BR2 などは、対応する市の名前に変換されます。キーワード OTHER は、リストされている市コード値のいずれにも一致しないデータセットの値が値 INCORRECT CODE に変換されるように指定します。

```
   value  $city 'BR1'='Birmingham UK'
                'BR2'='Plymouth UK'
                'BR3'='York UK'
                'US1'='Denver USA'
                'US2'='Miami USA'
                other='INCORRECT CODE';
run;
```

PROCLIB.STAFF データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。LABEL オプションは列ヘッダーの変数名ではなく、変数ラベルを使用します。

```
proc print data=proclib.staff noobs label;
```

Salary 変数に対しラベルを指定します。 LABEL ステートメントは、名前 SALARY にラベル“Salary in U.S. Dollars”を代入します。

```
label salary='Salary in U.S. Dollars';
```

Salary と Site に対し出力形式を指定します。 FORMAT ステートメントは、USCURRENCY.出力形式を変数 SALARY に一時的に関連付け、また出力形式 $CITY.を変数 SITE に一時的に関連付けます。

```
format salary uscurrency. site $city.;
```

タイトルを指定します。

```
   title 'PROCLIB.STAFF with a Format for the Variables';
   title2 'Salary and Site';
run;
```

出力

アウトプット 27.6 *Salary と Site に対してフォーマットされた変数を含む PROCLIB.STAFF*

PROCLIB.STAFF with a Format for the Variables Salary and Site

Name	IdNumber	Salary in U.S. Dollars	Site	HireDate
Capalleti, Jimmy	2355	$34,072	Birmingham UK	30JAN09
Chen, Len	5889	$33,771	Birmingham UK	18JUN06
Davis, Brad	3878	$31,509	Plymouth UK	20MAR04
Leung, Brenda	4409	$55,256	Plymouth UK	18SEP94
Martinez, Maria	3985	$78,980	Miami USA	10JAN93
Orfali, Philip	0740	$80,648	Miami USA	16FEB03
Patel, Mary	2398	$56,643	York UK	02FEB90
Smith, Robert	5162	$64,561	INCORRECT CODE	15APR06
Sorrell, Joseph	4421	$62,403	Denver USA	19JUN11
Zook, Carla	7385	$37,010	York UK	18DEC10

例 7: 欠落した変数値と欠落していない変数値の出力形式の作成

要素: VALUE ステートメント
VALUE ステートメントオプション
OTHER

データセット: EDUCATION

詳細

EDUCATION データセットは複数の状態に対してドロップアウト率と計算スコアをレポートし、状態ごとに領域を示します。

この例では、VALUE ステートメントを使用して、計算スコアの欠落した値すべてに対してテキスト値 n/a が作成されます。欠落していない計算スコア値はすべて、5.1 形式を使用してフォーマットされます。

次に例では、領域ごとに各状態のドロップアウト率と計算スコアが出力されます。

プログラム

```
options obs=20;


proc format;
   value myfmt .='n/a' other=[5.1];
```

```
run;

proc sort data=education;
   by region;
run;

proc print data=education;
   by region;
   var state dropOutRate mathScore;
   format mathScore myfmt.;
run;
```

プログラムの説明

出力するオブザベーションの数を設定します。

```
options obs=20;
```

Mathscore 変数値の出力形式を作成します。 VALUE ステートメントを使用して、Mathscore 変数用の出力形式 myfmt.を作成します。プログラムで欠落した Mathscore 値が検出されると、その値は n/a としてフォーマットされます。Mathscore のその他のすべての値は 5.1 形式を使用してフォーマットされます。

```
proc format;
   value myfmt .='n/a' other=[5.1];
run;
```

データをソートして出力します。 PROC SORT を使用して、領域ごとにデータセットをソートします。領域ごとにデータを出力するには、PROC PRINT BY ステートメントに領域変数を指定します。状態、ドロップアウト率、および計算スコアをレポートするには、VAR ステートメントを使用し、state、dropOutRate、mathScore の各変数を指定します。最後に、FORMAT ステートメントを使用して SAS に myfmt.出力形式を使用して mathScore 変数をフォーマットするように指示します。

```
proc sort data=education;
   by region;
run;

proc print data=education;
   by region;
   var state dropOutRate mathScore;
   format mathScore myfmt.;
run;
```

出力

アウトプット 27.7 *領域内の状態ごとのドロップアウト率と計算スコア*

The SAS System

Region=MW

Obs	State	DropoutRate	Math Score
1	Illinois	21.5	260.0
2	Indiana	13.8	267.0
3	Iowa	13.6	278.0
4	Kansas	17.9	n/a

Region=NE

Obs	State	DropoutRate	Math Score
5	Connecticut	16.8	270.0
6	Delaware	28.5	261.0
7	Maine	22.5	n/a
8	Maryland	26.0	260.0

Region=SE

Obs	State	DropoutRate	Math Score
9	Alabama	22.3	252.0
10	Arkansas	11.5	256.0
11	Florida	38.5	255.0
12	Georgia	27.9	258.0
13	Kentucky	32.7	256.0
14	Louisiana	43.1	246.0

Region=W

Obs	State	DropoutRate	Math Score
15	Alaska	35.8	n/a
16	Arizona	31.2	259.0
17	California	32.7	256.0
18	Colorado	24.7	267.0
19	Hawaii	18.3	251.0
20	Idaho	21.8	272.0

例 8: 標準 SAS 出力形式とカラー背景を使用した、日付の出力形式の書き出し

要素: VALUE ステートメントオプション
HIGH

他の要素: PROC PRINT ステートメント
VAR ステートメントの STYLE オプション

データセット: PROCLIB.STAFF

出力形式: 例 2 の USCURRENCY.

出力形式: 例 3 の$CITY.

詳細

この例では、フォーマットされた値として SAS によって提供される既存の出力形式と、日付に基づくカラーコード値を使用します。

タスクには、次が含まれます。

- 数値出力形式の作成
- 出力形式のネスト
- 標準 SAS 出力形式を使用した出力形式の書き出し
- カラースキームを使用した日付のフォーマット

プログラム

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';

proc format library=library;

   value benefit
      low-'31DEC2008'd=[worddate20.]
      '01JAN2008'd-high='  ** Not Eligible **';

   value color;
      low-'31DEC2008'd='light green'
      '01JAN2009'd-high='light red';
run;

proc print data=proclib.staff noobs label;
   var name idnumber salary site;
   var hiredate /style=[background=color.];

   label salary='Salary in U.S. Dollars';

   format salary uscurrency. site $city. hiredate benefit.;

   title 'PROCLIB.STAFF with a Format for the Variables';
   title2 'Salary, Site, and HireDate';
run;
```

プログラムの説明

このプログラムでは、BENEFIT.という出力形式を定義します。これにより 31DEC2008 以前に雇用された従業員を区別します。このプログラムの目的は、福利厚生を受ける資格のある従業員を 2008 年 12 月 31 日以前の雇用日に基づいて示すことです。雇用日がそれより後のその他すべての従業員は福利厚生の資格なしとしてリストされます。

---

2 つの SAS ライブラリ参照(PROCLIB と LIBRARY)を割り当てます。 この場合、ライブラリ参照 LIBRARY を割り当てると便利です。PROC FORMAT を使用すると、LIBRARY ライブラリ参照名で参照されるライブラリの入力形式と出力形式が自動的に検索されるためです。

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';
```

BENEFIT.出力形式をカタログ LIBRARY.FORMATS に保存します。 LIBRARY=オプションは作成する出力形式に対し永久保存場所 LIBRARY を指定します。ILIBRARY=を使用しない場合、作成する出力形式と入力形式は Work.Formats というカタログに一時的に保存されます。

```
proc format library=library;
```

BENEFIT.出力形式の最初の範囲を定義します。 最初の範囲では、31DEC2008 以前に雇用された従業員とその日より後に雇用された従業員を区別します。キーワード LOW と SAS 日付定数'31DEC2008'd は、2008 年 12 月 31 日以前に発生するすべての日付値を含む最初の範囲を作成します。この範囲内の値に対し、WORDDATE*w*.出力形式が適用されます。SAS 日付定数の詳細については、次をを参照してください。“Dates, Times, and Intervals” (*SAS Language Reference: Concepts*)WORDDATE 出力形式の詳細については、“WORDDATEw. Format” (*SAS Formats and Informats: Reference*)を参照してください。

```
   value benefit
      low-'31DEC2008'd=[worddate20.]
      '01JAN2008'd-high='  ** Not Eligible **';
```

範囲の色を定義します。 同じ日付範囲を使用して、日付に基づいて福利厚生を受ける資格のある従業員が、薄緑でカラーコーディングされます。福利厚生を受ける資格のない従業員は、明るい赤でカラーコーディングされます。

```
   value color;
      low-'31DEC2008'd='light green'
      '01JAN2009'd-high='light red';
run;
```

PROCLIB.STAFF データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。LABEL オプションは列ヘッダーの変数名ではなく、変数ラベルを使用します。VAR ステートメントで、印刷する変数を指定します。2 番目の VAR ステートメントでは、STYLE=オプションを使用して、Hiredate 変数の背景色として color.出力形式を指定します。

```
proc print data=proclib.staff noobs label;
   var name idnumber salary site;
   var hiredate /style=[background=color.];
```

Salary 変数に対しラベルを指定します。 LABEL ステートメントは、名前 SALARY にラベル“Salary in U.S. Dollars”を代入します。

```
   label salary='Salary in U.S. Dollars';
```

Salary、Site、Hiredate に対し出力形式を指定します。FORMAT ステートメントは、USCURRENCY.出力形式(“例 2: ピクチャ形式の作成” (886 ページ)で作成)とSALARY、$CITY.出力形式(“例 6: 文字値の出力形式の作成” (893 ページ)で作成)とSITE、BENEFIT.出力形式と HIREDATE を関連付けます。

```
      format salary uscurrency. site $city. hiredate benefit.;
```

タイトルを指定します。

```
      title 'PROCLIB.STAFF with a Format for the Variables';
      title2 'Salary, Site, and HireDate';
   run;
```

出力

アウトプット 27.8 *変数 Salary、Site、HireDate に対する出力形式を含む PROCLIB.STAFF*

PROCLIB.STAFF with a Format for the Variables Salary, Site, and HireDate

Name	IdNumber	Salary in U.S. Dollars	Site	HireDate
Capalleti, Jimmy	2355	$34,072	Birmingham UK	** Not Eligible **
Chen, Len	5889	$33,771	Birmingham UK	June 18, 2006
Davis, Brad	3878	$31,509	Plymouth UK	March 20, 2004
Leung, Brenda	4409	$55,256	Plymouth UK	September 18, 1994
Martinez, Maria	3985	$78,980	Miami USA	January 10, 1993
Orfali, Philip	0740	$80,648	Miami USA	February 16, 2003
Patel, Mary	2398	$56,643	York UK	February 2, 1990
Smith, Robert	5162	$64,561	INCORRECT CODE	April 15, 2006
Sorrell, Joseph	4421	$62,403	Denver USA	** Not Eligible **
Zook, Carla	7385	$37,010	York UK	** Not Eligible **

例 9: 生の文字データを数値に変換する

要素: INVALUE ステートメント

詳細

この例では、INVALUE ステートメントを使用して、数値と文字の生データを数値データに変換する数値入力形式を作成します。

プログラム

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';

proc format library=library;
```

```
   invalue evaluation 'O'=4
                      'S'=3
                      'E'=2
                      'C'=1
                      'N'=0;
run;

data proclib.points;
   input EmployeeId $ (Q1-Q4) (evaluation.,+1);
   TotalPoints=sum(of q1-q4);
   datalines;
2355 S O O S
5889 2 2 2 2
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E C C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;

proc print data=proclib.points noobs;

   title 'The PROCLIB.POINTS Data Set';
run;
```

プログラムの説明

このプログラムは、グレードをポイントとして合計するレポートを生成できるように、アルファベット順の従業員評価グレードを年に 4 回数値に変換します。

PROCLIB と LIBRARY の 2 つの SAS ライブラリ参照を設定します。

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';
```

EVALUATION.入力形式をカタログ Library.Formats に保存します。

```
proc format library=library;
```

数値入力形式 EVALUATION を作成します。 INVALUE ステートメントは指定値を変換します。文字 O (Outstanding)、S (Superior)、E (Excellent)、C (Commendable)、および N (None)はそれぞれ数字 4、3、2、1、0 に対応しています。

```
   invalue evaluation 'O'=4
                      'S'=3
                      'E'=2
                      'C'=1
                      'N'=0;
run;
```

PROCLIB.POINTS データセットを作成します。 DATALINES ステートメントの直後に続くインストリームデータには、四半期(Q1–Q4)の各従業員に対する一意の ID 番号(EmployeeId)と賞与評価が含まれています。データ行にリストされている賞与評価値の一部は数字です。その他は文字値です。文字値がデータ行にリストされている場合、EVALUATION.入力形式は値 O を 4 に、値 S を 3 などに変換します。生データ値 0 から 4 は、入力形式の定義で参照されないため、そのまま読み込まれます。文字値

を数字に変換することにより、年間の各従業員に対する賞与ポイントの合計数を計算できるようになります。TotalPoints は、賞与ポイントの合計数です。

```
data proclib.points;
   input EmployeeId $ (Q1-Q4) (evaluation.,+1);
   TotalPoints=sum(of q1-q4);
   datalines;
2355 S O O S
5889 2 2 2 2
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E C C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;
```

PROCLIB.POINTS データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=proclib.points noobs;
```

タイトルを指定します。

```
   title 'The PROCLIB.POINTS Data Set';
run;
```

出力

アウトプット 27.9 *PROCLIB.POINT データセット*

The PROCLIB.POINTS Data Set

EmployeeId	Q1	Q2	Q3	Q4	TotalPoints
2355	3	4	4	3	14
5889	2	2	2	2	8
3878	1	2	2	2	7
4409	0	1	1	1	3
3985	3	3	3	2	11
0740	3	2	2	3	10
2398	2	2	1	1	6
5162	1	1	1	2	5
4421	3	2	2	2	9
7385	1	1	1	0	3

例 10: データセットからの出力形式の作成

要素: PROC FORMAT ステートメントオプション
CNTLIN=
入力制御データセット

詳細

この例では、SAS データセットから出力形式を作成する方法を示します。

手順は次のとおりです。

- 入力制御データセトからの出力形式の作成
- 既存する SAS データセットからの入力制御データセットの作成

プログラム

```
data scale;
   input begin: $char2. end: $char2. amount: $char2.;
   datalines;
0   3    0%
4   6    3%
7   8    6%
9   10   8%
11  16   10%
;

data ctrl;
   length label $ 11;

   set scale(rename=(begin=start amount=label)) end=last;

   retain fmtname 'PercentageFormat' type 'n';

   output;

   if last then do;
      hlo='O';
      label='***ERROR***';
      output;
   end;
run;

proc print data=ctrl noobs;

   title 'The CTRL Data Set';
run;
```

プログラムの説明

scale という名前の一時データセットを作成します。 データ行の BEGIN、END という最初の 2 つの変数は、出力形式の範囲を指定するために使用されます。AMOUNT という 3 つ目の変数には、出力形式のフォーマットされた値として使用されるパーセントが含まれます。この 3 つの変数はすべて、PROC FORMAT 入力制御データセットに必要な文字変数です。

```
data scale;
   input begin: $char2. end: $char2. amount: $char2.;
   datalines;
0   3    0%
4   6    3%
7   8    6%
9   10   8%
11  16   10%
;
```

入力制御データセット CTRL を作成し、LABEL 変数の長さを設定します。 LENGTH ステートメントにより、LABEL 変数がラベル***ERROR***に合う十分な長さになります。

```
data ctrl;
   length label $ 11;
```

変数名を変更し、ファイルの終わりフラグを作成します。 データセット CTRL は WORK.SCALE から派生します。RENAME=は、BEGIN と AMOUNT をそれぞれ START と LABEL に名前変更します。END=オプションは、変数 LAST を作成します。この変数の値は、最後のオブザベーションの処理時に 1 に設定されます。

```
   set scale(rename=(begin=start amount=label)) end=last;
```

固定値を使用して、変数 Fmtname と Type を作成します。 RETAIN ステートメントはこの場合、割り当てステートメントよりも効率的です。RETAIN はプログラムデータベクトルの Fmtname と Type の値を保持し、DATA ステップの反復ごとに値を書き込む必要をなくします。Fmtname は名前 PercentageFormat を指定します。これは、入力制御データセットが作成する出力形式です。Type 変数は、入力制御データセットが数値出力形式を作成するように指定します。

```
   retain fmtname 'PercentageFormat' type 'n';
```

出力データセットにオブザベーションを書き込みます。

```
   output;
```

“その他”カテゴリを作成します。 このアプリケーションに有効な値は 0–16 だけであるため、その他の値(欠損値など)はエラーとしてユーザーに示される必要があります。IF ステートメントは DATA ステップが入力データセットからの最後のオブザベーションを処理した後にのみ実行します。IF の実行時には、HLO は範囲が OTHER であることを示す値 0 を受け取ります。LABEL は値***ERROR***を受け取ります。OUTPUT ステートメントはこれらの値をデータセットの最後のオブザベーションとして書き込みます。HLO には、その他すべてのオブザベーションに対する欠損値が含まれます。

```
   if last then do;
      hlo='O';
      label='***ERROR***';
      output;
   end;
run;
```

制御データセット CTRL を印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=ctrl noobs;
```

タイトルを指定します。

```
   title 'The CTRL Data Set';
```

```
run;
```

出力

アウトプット 27.10 *CTRL データセット*

The CTRL Data Set

label	start	end	fmtname	type	hlo
0%	0	3	PercentageFormat	n	
3%	4	6	PercentageFormat	n	
6%	7	8	PercentageFormat	n	
8%	9	10	PercentageFormat	n	
10%	11	16	PercentageFormat	n	
ERROR	11	16	PercentageFormat	n	O

作成した出力形式をカタログ Work.Formats に保存し、出力形式のソースを指定します。 CNTLIN=オプションは、データセット CTRL が出力形式 PercentageFormat のソースになるように指定します。

```
proc format library=work cntlin=ctrl;
run;
```

数値入力形式 EVALUATION を作成します。 INVALUE ステートメントは指定値を変換します。文字 O (Outstanding)、S (Superior)、E (Excellent)、C (Commendable)、および N (None)はそれぞれ数字 4、3、2、1、0 に対応しています。

```
proc format library=library;
   invalue evaluation 'O'=4
                      'S'=3
                      'E'=2
                      'C'=1
                      'N'=0;
run;
```

WORK.POINTS データセットを作成します。 DATALINES ステートメントの直後に続くインストリームデータには、四半期(Q1–Q4)の各従業員に対する一意の ID 番号(EmployeeId)と賞与評価が含まれています。データ行にリストされている賞与評価値の一部は数字です。その他は文字値です。文字値がデータ行にリストされている場合、Evaluation.入力形式は値 O を 4 に、値 S を 3 などに変換します。生データ値 0 から 4 は、入力形式の定義で参照されないため、そのまま読み込まれます。文字値を数字に変換することにより、年間の各従業員に対する賞与ポイントの合計数を計算できるようになります。TotalPoints は、賞与ポイントの合計数です。欠損値が TotalPoints の欠損値になるように、加算演算子が SUM 関数の代わりに使用されます。

```
data points;
   input EmployeeId $ (Q1-Q4) (evaluation.,+1);
   TotalPoints=q1+q2+q3+q4;
   datalines;
2355 S O O S
5889 2 . 2 2
```

```
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E   C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;
```

WORK.POINTS に対しレポートを生成し、PERCENTAGEFORMAT.出力形式と TotalPoints 変数を関連付けます。 DEFINE ステートメントはその関連付けを実行します。TotalPoints のフォーマットされた値を含む列では、別名 Pctage を使用しています。別名を使用することにより、変数を出力形式とデフォルト出力形式を使用してそれぞれ 1 回ずつ、合計 2 回印刷できます。REPORT プロシジャの詳細については、55 章, “REPORT プロシジャ,” (1580 ページ)を参照してください。

```
proc report data=work.points nowd headskip split='#';
   column employeeid totalpoints totalpoints=Pctage;
   define employeeid / right;
   define totalpoints / 'Total#Points' right;
   define pctage / format=PercentageFormat12. 'Percentage' left;
   title 'The Percentage of Salary for Calculating Bonus';
run;
```

出力

アウトプット 27.11 *賞与を計算するための給与のパーセント*

The Percentage of Salary for Calculating Bonus

EmployeeId	Total Points	Percentage
2355	14	10%
5889	.	***ERROR***
3878	7	6%
4409	3	0%
3985	11	10%
0740	10	8%
2398	.	***ERROR***
5162	5	3%
4421	9	8%
7385	3	0%

例 11: 入力形式と出力形式の説明の印刷

要素: PROC FORMAT ステートメントオプション
FMTLIB
SELECT ステートメント

出力形式: 例 4 の BENEFIT.

入力形式: 例 6 の EVALUATION.

詳細

この例では、入力形式と出力形式の説明を印刷する方法を示します。説明には、入力および出力される値が示されます。

プログラム

```
libname library 'SAS-library';

proc format library=library fmtlib;

   select @evaluation benefit;

   title 'FMTLIB Output for the BENEFIT. Format and the';
   title2 'EVALUATION. Informat';
run;
```

プログラムの説明

LIBRARY という名前の SAS ライブラリ参照を設定します。

```
libname library 'SAS-library';
```

EVALUATION.と BENEFIT の説明を印刷します。FMTLIB オプションは、LIBRARY=オプションが指定するカタログの出力形式と入力形式に関する情報を印刷します。LIBRARY=LIBRARY は Library.Formats カタログを指します。

```
proc format library=library fmtlib;
```

入力形式と出力形式を選択します。SELECT ステートメントは前の例で作成された EVALUATION.と BENEFIT.を選択します。EVALUATION.の前のアットマーク(@)は、EVALUATION.が入力形式であることを示します。

```
   select @evaluation benefit;
```

タイトルを指定します。

```
   title 'FMTLIB Output for the BENEFIT. Format and the';
   title2 'EVALUATION. Informat';
run;
```

出力

アウトプット 27.12 *BENEFIT 出力形式と EVALUATION 入力形式の FMTLIB 出力*

FMTLIB Output for the BENEFIT. Format and the EVALUATION. Informat

FORMAT NAME: BENEFIT LENGTH: 20 NUMBER OF VALUES: 2
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 20 FUZZ: STD

START	END	LABEL (VER. V7|V8 17SEP2012:15:53:27)
LOW	17897	[WORDDATE20.]
17898	HIGH	** Not Eligible **

INFORMAT NAME: @EVALUATION LENGTH: 1
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 1 FUZZ: 0

START	END	INVALUE(VER. 9.4 17SEP2012:16:10:24)
C	C	1
E	E	2
N	N	0
O	O	4
S	S	3

例 12: (永久)出力形式の読み込み

要素: PROC FORMAT ステートメントオプション
LIBRARY=

他の要素: FMTSEARCH=システムオプション

データセット: SAMPLE

この例では、LIBRARY=オプションと FMTSEARCH=システムオプションを使用して、Work.Formats または Library.Formats 以外のカタログに保存されている出力形式を保存し、取得します。

プログラム

```
libname proclib 'SAS-library';

proc format library=proclib;

picture nozeros (fuzz=0)
           low -  -1  = '000.00'(prefix='-')
       -1 < - < -.99  = '0.99' (prefix='-.' mult=100)
       -0.99 < - < 0  = '99'    (prefix='-.' mult=100)
                  0  = '0.99'
        0 < - < .99   = '99'    (prefix='.'  mult=100)
           0.99 - <1  = '0.99'  (prefix='.'  mult=100)
            1  - high = '00.99';
run;
```

```
options  fmtsearch=(proclib);

data sample;
  input Amount;
   datalines;
-2.051
-.05
-.017
  0
.093
.54
.556
6.6
14.63
0.996
-0.999
-45.00
;
run;

proc print data=sample;
   format amount nozeros.;

   title1 'Retrieving the NOZEROS. Format from PROCLIB.FORMATS';
   title2 'The SAMPLE Data Set';
run;
```

プログラムの説明

PROCLIB という名前の SAS ライブラリ参照を設定します。

```
libname proclib 'SAS-library';
```

NOZEROS.出力形式を PROCLIB.FORMATS カタログに保存します。

```
proc format library=proclib;
```

NOZEROS.出力形式を作成します。 PICTURE ステートメントはピクチャ形式 NOZEROS. を定義します。“詳細” (860 ページ)を参照してください。

```
picture nozeros (fuzz=0)
           low -  -1  = '000.00'(prefix='-')
       -1 < - < -.99  = '0.99' (prefix='-.' mult=100)
       -0.99 < - < 0  = '99'    (prefix='-.' mult=100)
                  0  = '0.99'
        0 < - < .99   = '99'    (prefix='.'  mult=100)
           0.99 - <1  = '0.99'  (prefix='.'  mult=100)
            1  - high = '00.99';
run;
```

PROCLIB.FORMATS カタログをユーザー定義の出力形式の検索に使用される検索パスに追加します。 FMTSEARCH=システムオプションは、検索パスを定義します。FMTSEARCH=システムオプションにはライブラリ参照名のみ必要です。FMTSEARCH=は、カタログ名が表示されていない場合はカタログ名を FORMATS とみなします。FMTSEARCH=オプションがないと、NOZEROS.出力形式は見つかりません。詳細については、“FMTSEARCH= System Option” (*SAS System Options: Reference*)を参照してください。

```
options  fmtsearch=(proclib);
```

サンプルデータセットを作成します。

```
data sample;
  input Amount;
   datalines;
-2.051
-.05
-.017
  0
.093
.54
.556
6.6
14.63
0.996
-0.999
-45.00
;
run;
```

SAMPLE データセットを印刷します。FORMAT ステートメントは、NOZEROS.出力形式と Amount 変数を関連付けます。

```
proc print data=sample;
   format amount nozeros.;
```

タイトルを指定します。

```
   title1 'Retrieving the NOZEROS. Format from PROCLIB.FORMATS';
   title2 'The SAMPLE Data Set';
run;
```

出力

アウトプット 27.13 *Retrieving the NOZEROS.Format from PROCLIB.FORMATS*

Retrieving the NOZEROS. Format from PROCLIB.FORMATS
The SAMPLE Data Set

Obs	Amount
1	-2.05
2	-.05
3	-.01
4	.00
5	.09
6	.54
7	.55
8	6.60
9	14.63
10	.99
11	-.99
12	45.00

例 13: 文字列の範囲の書き出し

要素: VALUE ステートメント

データセット: PROCLIB.STAFF

この例では、出力形式を作成し、文字列を含む範囲を使用する方法を示します。

プログラム

```
libname proclib'SAS-library';

data train;
   set proclib.staff(keep=name idnumber);
run;

proc print data=train noobs;

   title 'The TRAIN Data Set without a Format';
run;
```

プログラムの説明

```
libname proclib'SAS-library';
```

PROCLIB.STAFF データセットから TRAIN データセットを作成します。PROCLIB.STAFF は“例 1: サンプルデータセットの作成” (885 ページ)で作成されました。

```
data train;
   set proclib.staff(keep=name idnumber);
run;
```

データセット TRAIN を出力形式なしで印刷します。NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=train noobs;
```

タイトルを指定します。

```
   title 'The TRAIN Data Set without a Format';
run;
```

出力

アウトプット 27.14 *出力形式のない TRAIN データセット*

The TRAIN Data Set without a Format

Name	IdNumber
Capalleti, Jimmy	2355
Chen, Len	5889
Davis, Brad	3878
Leung, Brenda	4409
Martinez, Maria	3985
Orfali, Philip	0740
Patel, Mary	2398
Smith, Robert	5162
Sorrell, Joseph	4421
Zook, Carla	7385

出力形式を Work.Formats に保存します。LIBRARY=オプションが表示されないため、出力形式は Work.Formats に保存され、現在の SAS セッションにのみ利用可能です。

```
proc format;
```

**$SKILLTEST.出力形式を作成します。**$SKILLTEST.出力形式は各従業員の ID 番号と割り当てられているスキルテストを出力します。従業員は、姓によって TEST A、TEST B、TEST C のいずれかを受ける必要があります。除外演算子(<)は、範囲の最終値を除外します。そのため、最初の範囲には姓が A から D で始まる従業員が、2 番目の範囲には姓が E から M で始まる従業員が含まれます。最後の範囲のチルダ(~)は、Z で始まる文字列全体を含めるために必要です。

```
   value $skilltest  'a'-<'e','A'-<'E'='Test A'
                     'e'-<'m','E'-<'M'='Test B'
                     'm'-'z~','M'-'Z~'='Test C';
run;
```

TRAIN データセットのレポートを生成します。 DEFINE ステートメントの FORMAT=オプションは、$SKILLTEST.出力形式と Name 変数を関連付けます。Name のフォーマットされた値を含む列は、別名 Test を使用しています。別名を使用することにより、変数を出力形式とデフォルト出力形式を使用してそれぞれ 1 回ずつ、合計 2 回印刷できます。詳細については、55 章, “REPORT プロシジャ” (1579 ページ)を参照してください。

```
proc report data=train nowd headskip;
   column name name=test idnumber;
   define test / display format=$skilltest. 'Test';
   define idnumber / center;
   title 'Test Assignment for Each Employee';
run;
```

出力

アウトプット 27.15 *各従業員のテスト割り当て*

Test Assignment for Each Employee

Name	Test	IdNumber
Capalleti, Jimmy	Test A	2355
Chen, Len	Test A	5889
Davis, Brad	Test A	3878
Leung, Brenda	Test B	4409
Martinez, Maria	Test C	3985
Orfali, Philip	Test C	0740
Patel, Mary	Test C	2398
Smith, Robert	Test C	5162
Sorrell, Joseph	Test C	4421
Zook, Carla	Test C	7385

例 14: 英語以外の言語の出力形式の作成

要素: PICTURE ステートメントオプション
DATATYPE=
LANGUAGE=

他の要素: LOCALE=システムオプション

詳細

この例では、次のタスクを実行します。

- DATATYPE=ステートメントオプションを使用して日付値と日時値をフォーマットするためのディレクティブを使用してピクチャ形式を作成します。
- LOCALE=システムオプションを使用して、ドイツ語のロケールを指定します。
- ピクチャ形式を使用して、日付値と日時値をドイツ語のログに印刷します。
- LANGUAGE=French を指定するピクチャ形式を使用して、フランス語の日時値をログに印刷します。

プログラム

```
proc format;
   picture mdy(default=8) other='%0d%0m%Y' (datatype=date);
   picture langtsda (default=50) other='%A, %d %B, %Y' (datatype=date);
   picture langtsdt (default=50) other='%A, %d,%B, %Y %H %M %S'
                                 (datatype=datetime);
   picture langtsfr (default=50) other='%A, %d %B, %Y %H %M %S'
                                 (datatype=datetime language=french);
   picture alltest (default=100)
           other='%a %A %b %B %d %H %I %j %m %M %p %S %w %U %y %%'
           (datatype=datetime);
run;

option locale = de_DE;

data _null_ ;
   a= 18903;
   b = 1633239000;
   put a= mdy.;
   put a= langtsda.;
   put b= langtsdt.;
   put b= langtsfr.;
   put b= alltest.;
run ;
```

プログラムの説明

PICTURE ステートメントを使用して、出力形式を作成します。 各 PICTURE ステートメントは、ディレクティブを使用してフォーマットする日付値と日時値を指定します。%A は週の完全名を印刷します。%B は月の完全名を印刷します。%d は月の日を印刷します。%Y は年を印刷します。%H は時間(24 時間)を印刷します。%M は分を印刷します。%S は秒を印刷します。最初の 3 つの出力形式は、日付と日時を LOCALE=システムオプションの現在値によって指定される言語で印刷します。LANGTSFT.出力形式は、日時をフランス語で印刷します。その他のディレクティブについては、PICTURE ステートメント (849 ページ)を参照してください。

```
proc format;
   picture mdy(default=8) other='%0d%0m%Y' (datatype=date);
   picture langtsda (default=50) other='%A, %d %B, %Y' (datatype=date);
   picture langtsdt (default=50) other='%A, %d,%B, %Y %H %M %S'
                                 (datatype=datetime);
   picture langtsfr (default=50) other='%A, %d %B, %Y %H %M %S'
                                 (datatype=datetime language=french);
   picture alltest (default=100)
```

```
            other='%a %A %b %B %d %H %I %j %m %M %p %S %w %U %y %%'
            (datatype=datetime);
run;
```

LOCALE=システムオプションを設定します。 de_DE は、ドイツ語のロケール値です。

```
option locale = de_DE;
```

日付値と日時値をドイツ語とフランス語で印刷します。 DATA ステップは、SAS ログに 2011 年 10 月 3 日、05:30:00 AM に関する日付および日時の情報を印刷します。LANGTSFR.出力形式を使用してフォーマットされている b の値を除くすべての値がドイツ語で書き出されます。この出力形式は、日時値をフランス語で印刷します。

```
data _null_ ;
   a= 18903;
   b = 1633239000;
   put a= mdy.;
   put a= langtsda.;
   put b= langtsdt.;
   put b= langtsfr.;
   put b= alltest.;
run ;
```

ドイツ語とフランス語のピクチャ形式出力を表示する SAS ログ

```
1    proc format; 2       picture mdy(default=8) other='%0d%0m%Y'
(datatype=date); NOTE:Format MDY has been output.3       picture langtsda
(default=50) other='%A, %d %B, %Y' (datatype=date); NOTE:Format LANGTSDA has
been output.4       picture langtsdt (default=50) other='%A, %d,%B, %Y %H %M %S'
5                                    (datatype=datetime); NOTE:Format LANGTSDT
has been output.6       picture langtsfr (default=50) other='%A, %d %B, %Y %H %M
%S' 7                                  (datatype=datetime language=french);
NOTE:Format LANGTSFR has been output.8       picture alltest (default=100)
9               other='%a %A %b %B %d %H %I %j %m %M %p %S %w %U %y %%'
10              (datatype=datetime); NOTE:Format ALLTEST has been output.11
run; NOTE:PROCEDURE FORMAT used (Total process time): real time           0.03
seconds cpu time            0.03 seconds 12 13   option locale = de_DE; 14 15
data _null_ ; 16      a= 18903; 17      b = 1633239000; 18      put a= mdy.;
19      put a= langtsda.; 20      put b= langtsdt.; 21      put b= langtsfr.;
22      put b= alltest.; 23   run ; a=03102011 a=Montag, 3 Oktober, 2011
b=Montag, 3,Oktober, 2011 5 30 0 b=Lundi, 3 octobre, 2011 5 30 0 b=Mo  Montag
Okt Oktober 3 5 5 276 10 30 AM 0 2 40 11 %
```

例 15: ロケール固有の出力形式カタログの作成

要素: PROC FORMAT LOCALE オプション
FMTSEARCH=システムオプション

詳細

この例では、英語とルーマニア語の 2 つの言語で出力形式を作成する方法と、英語とルーマニアの出力形式カタログにアクセスして、これらの 2 つの言語でデータセットを印刷する方法を示します。この例は、SAS セッションのエンコードが、ルーマニアロケールをサポートする latin 2 エンコードである場合に最も適切に動作します。

プログラム

```
/*no locale information*/
proc format lib=work.formats;
value age low - 5 = 'baby'
6 - 12 = 'child'
13 - 15 = 'teen'
16 - 30 = 'youth'
31 - 50 = 'midlife'
51 - high = 'older';
run;

options locale=ro_RO;

proc format lib = work.formats locale;
value age low - 5 = 'Copil'
6 - 12 = 'Copil'
13 - 15 = 'Adolescent'
16 - 30 = 'Tineretului'
31 - 50 = 'Asta vrei'
51 - high = 'Mai vechi';
run;


options fmtsearch=(work/locale);

/* Set the locale back to English(US) */
options locale=en_US;

data datatst;
input age sex $;
attrib age format= age.;
cards;
5 M
6 F
12 M
13 F
15 M
16 F
30 M
35 F
51 M
100 F
;
run;
/* Use the English format catalog*/
title "Locale is English, Use the Original Format Catalog";
proc print data=datatst; run;

/* Use the Romanian format catalog*/
options locale=ro_RO;
title 'Locale is ro_RO, Use the Romanian Format Catalog';
proc print data=datatst;run;
```

プログラムの説明

AGE.出力形式を英語で作成します。

```
/*no locale information*/
proc format lib=work.formats;
value age low - 5 = 'baby'
6 - 12 = 'child'
13 - 15 = 'teen'
16 - 30 = 'youth'
31 - 50 = 'midlife'
51 - high = 'older';
run;
```

ローケルを変更し、AGE.出力形式をロケール固有のカタログに作成します。 LOCALE=システムオプションを使用して、ロケールをルーマニアロケールに変更します。PROC FORMAT ステートメントの LOCALE オプションで、ルーマニア言語に対応する、現在のロケールを表す出力形式カタログを作成することを指定します。

```
options locale=ro_RO;

proc format lib = work.formats locale;
value age low - 5 = 'Copil'
6 - 12 = 'Copil'
13 - 15 = 'Adolescent'
16 - 30 = 'Tineretului'
31 - 50 = 'Asta vrei'
51 - high = 'Mai vechi';
run;
```

ロケール固有の出力形式カタログを出力形式検索パスに追加します。 FMTSEARCH=システムオプションでは、検索する出力形式カタログを指定します。複数のロケール固有カタログを作成できるため、検索リストの libref に/LOCALE を追加すると、現在のロケールに関連付けられたカタログが検索されます。

```
options fmtsearch=(work/locale);
```

データセットを作成し、英語の出力形式カタログを使用して印刷します。 LOCALE=システムオプションで、ロケールを英語に設定します。

```
/* Set the locale back to English(US) */
options locale=en_US;

data datatst;
input age sex $;
attrib age format= age.;
cards;
5 M
6 F
12 M
13 F
15 M
16 F
30 M
35 F
51 M
100 F
;
run;
/* Use the English format catalog*/
```

```
title "Locale is English, Use the Original Format Catalog";
proc print data=datatst; run;
```

ルーマニア語の出力形式カタログを使用して、データセットを印刷します。LOCALE=システムオプションを使用して、ロケールをルーマニアロケールに設定します。

```
/* Use the Romanian format catalog*/
options locale=ro_RO;
title 'Locale is ro_RO, Use the Romanian Format Catalog';
proc print data=datatst;run;
```

次は、英語とルーマニア語の出力形式カタログを使用して印刷されたデータセットを示しています。

アウトプット 27.16 *英語とルーマニア語の出力形式カタログを使用して印刷されたデータセット*

Locale is English, Use the Original Format Catalog

Obs	age	sex
1	baby	M
2	child	F
3	child	M
4	teen	F
5	teen	M
6	youth	F
7	youth	M
8	midlife	F
9	older	M
10	older	F

Locale is ro_RO, Use the Romanian Format Catalog

Obs	age	sex
1	Copil	M
2	Copil	F
3	Copil	M
4	Adolescent	F
5	Adolescent	M
6	Tineretului	F
7	Tineretului	M
8	Asta vrei	F
9	Mai vechi	M
10	Mai vechi	F

例 16: 出力形式として使用する関数の作成

要素: PROC FCMP ステートメント
CMPLIB=システムオプション
PROC FORMAT ステートメント
出力形式機能としての関数

詳細

この例では、気温をセ氏から力氏、力氏からセ氏に変換する関数を作成します。プログラムでは、関数を 1 つの DATA ステップで関数として、別の DATA ステップで出力形式として使用します。

プログラム

```
proc fcmp outlib=library.functions.smd;
   function ctof(c) $;
      return(cats(((9*c)/5)+32,'F'));
   endsub;

   function ftoc(f) $;
      return(cats((f-32)*5/9,'C'));
   endsub;

run;

options cmplib=(library.functions);

data _null_;
   f=ctof(100);
   put f=;
run;

proc format;
   value ctof (default=10) other=[ctof()];
   value ftoc (default=10) other=[ftoc()];
 run;

data _null_;
   c=100;
   put c=ctof.;
   f=212;
   put f=ftoc.;
run;
```

プログラムの説明

気温をセ氏から力氏、力氏からセ氏に変更する関数を作成します。 FCMP プロシジャは、セ氏気温を力氏、力氏気温をセ氏に変換する CTOF 関数を作成します。

```
proc fcmp outlib=library.functions.smd;
   function ctof(c) $;
      return(cats(((9*c)/5)+32,'F'));
```

```
   endsub;

   function ftoc(f) $;
      return(cats((f-32)*5/9,'C'));
   endsub;

run;
```

関数ライブラリにアクセスします。 CMPLIB システムオプションにより、プログラムのコンパイラ中に関数を含めることができます。

```
options cmplib=(library.functions);
```

その関数を SAS プログラムの関数として使用します。

```
data _null_;
   f=ctof(100);
   put f=;
run;
```

関数を使用して、ユーザー定義の出力形式を作成します。 出力形式の名前は、関数の名前です。関数を出力形式として使用する際に、OTHER キーワードによって示されるように出力形式をネストできます。

```
proc format;
   value ctof (default=10) other=[ctof()];
   value ftoc (default=10) other=[ftoc()];
 run;
```

関数を出力形式として使用します。 この DATA ステップは、PUT ステートメントを使用して気温をフォーマットします。ここで出力形式を PUT ステートメントの変数に割り当てます。

```
data _null_;
   c=100;
   put c=ctof.;
   f=212;
   put f=ftoc.;
run;
```

出力:ログ

ログ 27.4 *出力形式として使用する関数を作成した後の SAS ログ*

```
323  proc fcmp outlib=library.functions.smd; 324     function ctof(c) $;
325        return(cats(((9*c)/5)+32,'F')); 326     endsub; 327 328     function
ftoc(f) $; 329        return(cats((f-32)*5/9,'C')); 330     endsub; 331 332
run; NOTE:Function ftoc saved to library.functions.smd.NOTE:Function ctof saved
to library.functions.smd.NOTE:PROCEDURE FCMP used (Total process time): real
time           17.59 seconds cpu time            1.26 seconds 333 334  options
cmplib=(library.functions); 335 336  data _null_; 337     f=ctof(100); 338
put f=; 339  run; f=212F NOTE:DATA statement used (Total process time): real
time           0.50 seconds cpu time            0.01 seconds 340 341  proc
format; 342       value ctof (default=10) other=[ctof()]; NOTE:Format CTOF has
been output.343       value ftoc (default=10) other=[ftoc()]; NOTE:Format FTOC
has been output.344       run; NOTE:PROCEDURE FORMAT used (Total process time):
real time           0.00 seconds cpu time            0.00 seconds 345 346  data
_null_; 347     c=100; 348     put c=ctof.; 349     f=212; 350     put f=ftoc.;
351  run; c=212F f=100C
```

例 17: 出力形式を使用してドリルダウンテーブルを作成する

要素: VALUE ステートメント

他の要素: PROC PRINT FORMAT ステートメント

詳細

この例では、アメリカ合衆国の 5 つの州に関する人口情報を含む HTML テーブルを作成します。州の名前は、州の Web サイトへのリンクになっています。リンクは、州名をフォーマットするためのユーザー定義の出力形式を使用して作成されます。この例では、次を行います。

- 州の人口情報を含むデータセットの作成
- 値が HTML リンク(&<a&>)要素である VALUE ステートメントを使用したユーザー定義の出力形式の作成
- HTML ファイルの名前と HTML ファイルのタイトルの定義
- ユーザー定義の出力形式を使用した HTML テーブルの印刷

プログラム

```
data mydata;
   format population comma12.0;
   label st='State';
   label population='Population';
   input st $ 1-2 population;
   year=2000;
   datalines;
VA  7078515
NC  8049313
SC  4012012
GA  8186453
FL 15982378
;
run;

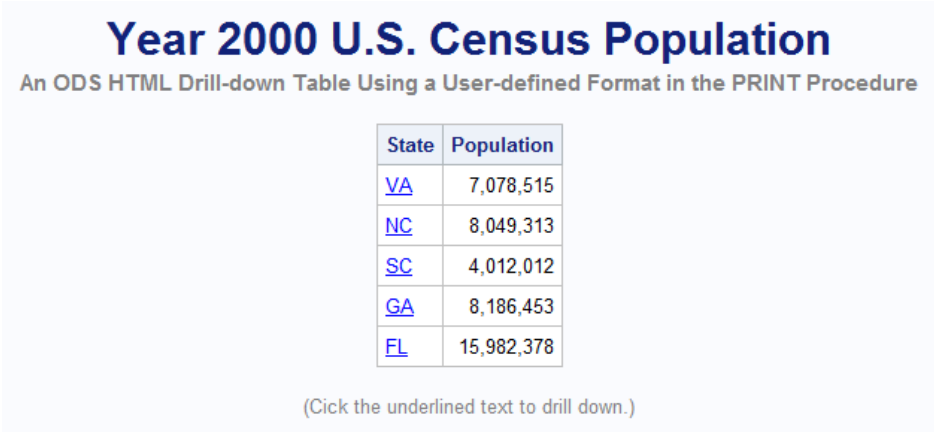
proc format;
value $COMPND
'VA'='<a href=http://www.va.gov>VA</a>'
'NC'='<a href=http://www.nc.gov>NC</a>'
'SC'='<a href=http://www.sc.gov>SC</a>'
'GA'='<a href=http://www.ga.gov>GA</a>'
'FL'='<a href=http://www.fl.gov>FL</a>';
run;

ods html file="c:\mySAS\html\Drilldown.htm"
 (title="An ODS HTML Drill-down Table Using a User-defined Format in the PRINT
 Procedure");

title h=.25in "Year 2000 U.S. Census Population";
title2 color=gray "An ODS HTML Drill-down Table Using a User-defined Format in
the PRINT Procedure";
footnote color=gray "(Click the underlined text to drill down.)";
```

```
options nodate;
proc print data=mydata label noobs;
 var st population;
 format st $compnd. ;
run;

ods html close;
ods html;
```

プログラムの説明

データセットを作成します。 mydata DATA ステップは、2000 年に行われた国勢調査に基づいたアメリカ合衆国の 5 つの州の人口に関する情報を含むデータセットを作成します。作成される変数は、国勢調査の年、州の略称、州の人口に関するデータを割り当てます。

```
data mydata;
   format population comma12.0;
   label st='State';
   label population='Population';
   input st $ 1-2 population;
   year=2000;
   datalines;
VA  7078515
NC  8049313
SC  4012012
GA  8186453
FL 15982378
;
run;
```

$COMPND.出力形式を作成します。 $COMPND.出力形式は、各州を州の各 Web サイトへのリンクとしてフォーマットします。

```
proc format;
value $COMPND
'VA'='<a href=http://www.va.gov>VA</a>'
'NC'='<a href=http://www.nc.gov>NC</a>'
'SC'='<a href=http://www.sc.gov>SC</a>'
'GA'='<a href=http://www.ga.gov>GA</a>'
'FL'='<a href=http://www.fl.gov>FL</a>';
run;
```

テーブルファイル名とテーブルタイトルを設定します。 ODS HTML FILE=オプションは、HTML 出力が保存されるディレクトリとファイル名を指定します。

```
ods html file="c:\mySAS\html\Drilldown.htm"
 (title="An ODS HTML Drill-down Table Using a User-defined Format in the PRINT
 Procedure");

title h=.25in "Year 2000 U.S. Census Population";
title2 color=gray "An ODS HTML Drill-down Table Using a User-defined Format in
the PRINT Procedure";
footnote color=gray "(Click the underlined text to drill down.)";
```

テーブルを印刷して、HTML 出力先を閉じ、再度開きます。 PRINT プロシジャは出力形式 $COMPND.を使用して、州名をフォーマットします。フォーマットされた名前は、州の各 Web サイトへのリンクとなります。ODS HTML ステートメントは、後の出力が作成したばかりの HTML ファイルを上書きしないように、HTML 出力先を閉じ、再度開きます。

```
options nodate;
proc print data=mydata label noobs;
 var st population;
 format st $compnd. ;
run;

ods html close;
ods html;
```

出力

アウトプット 27.17 *出力形式を使用した HTML テーブルでのドリルダウンテキストの作成*



Year 2000 U.S. Census Population

An ODS HTML Drill-down Table Using a User-defined Format in the PRINT Procedure

State	Population
VA	7,078,515
NC	8,049,313
SC	4,012,012
GA	8,186,453
FL	15,982,378

(Cick the underlined text to drill down.)

28 章

FSLIST プロシジャ

概要: FSLIST プロシジャ

FSLIST プロシジャを使用して、SAS セッション内で SAS データセット以外の外部ファイルを参照できます。ファイルは対話型のウィンドウに表示されるため、このプロシジャで、ファイルコンテンツを検証するための非常に便利な手法を使用できます。また、**FSLIST** ウィンドウから SAS テキストエディタを使用するウィンドウにテキストをコピーできます。

構文: FSLIST プロシジャ

PROC FSLIST FILEREF=*file-specification* | UNIT=*nn* <*options*>;
PROC FSLIST DDNAME=*file-specification* | UNIT=*nn* <*options*>;
PROC FSLIST DD=*file-specification* | UNIT=*nn* <*options*>;

ステートメント	タスク
“PROC FSLIST ステートメント”	FSLIST プロシジャを開始し、参照する外部ファイルを指定します

PROC FSLIST ステートメント

SAS セッション内で SAS データセット以外の外部ファイルを参照できます。

構文

PROC FSLIST FILEREF=*file-specification* | UNIT=*nn* <*options*>;
PROC FSLIST DDNAME=*file-specification* | UNIT=*nn* <*options*>;
PROC FSLIST DD=*file-specification* | UNIT=*nn* <*options*>;

オプション引数の要約

CAPS | NOCAPS
: FIND コマンドの検索文字列の処理方法を制御します。

CC | FORTCC | NOCC
: キャリッジコントロール文字が表示のフォーマットに使用されるかどうかを示します。

HSCROLL=*n* | HALF | PAGE
: LEFT コマンドと RIGHT コマンドに対するデフォルトの水平スクロール幅を示します。

NOBORDER
: **FSLIST** ウィンドウの境界の側面と下部を非表示にします。

NUM | NONUM
: シーケンス番号の表示を制御します。

OVP | NOOVP
: 重ね打ちのキャリッジコントロールコードが有効かどうかを示します。

必須引数

FILEREF | DDNAME | DD=*file-specification*
参照する外部ファイルを指定します。*File-specification* は次のいずれかにすることができます。

'external-file'
: 外部ファイルに対する完全な動作環境ファイル指定(一部の動作環境では完全修飾パス名と呼ばれる)です。名前を引用符で囲む必要があります。

fileref
: 以前外部ファイルに割り当てられたファイル参照名です。FILENAME ステートメントを使用して、ファイル参照名と実際のファイル名を関連付けることができます。詳細については、“FILENAME Statement” (*SAS Statements: Reference*)を参照してください。

UNIT=*nn*
参照する外部ファイルの FORTRAN 形式の論理ユニット番号を定義します。このオプションは、参照するファイルに形式 FT*nn*F001 のファイル参照名が含まれているときに便利です。*nn* は、UNIT=引数で指定される論理ユニット番号です。たとえば、次のように指定できます。`proc fslist unit=20;` instead of
`proc fslist fileref=ft20f001;`

オプション引数

CAPS | NOCAPS
FIND コマンドの検索文字列の処理方法を制御します。

CAPS
検索文字列を大文字に変換します。引用符で囲まれている場合は変換されません。たとえば、このオプションを有効にすると、コマンド `find nc` によって、`nc` ではなく、`NC` の発生が検出されます。小文字を検索するには、次のように検索文字列を引用符で囲みます。`find 'nc'`

NOCAPS
この場合、変換は行われません。FIND コマンドは検索文字と完全に一致するテキスト文字列のみを検索します。

デフォルトは NOCAPS です。**FSLIST** ウィンドウで CAPS コマンドを使用して、ファイル参照時にプロシジャの動作を変更できます。

CC | FORTCC | NOCC
キャリッジコントロール文字が表示のフォーマットに使用されるかどうかを示します。このオプションに対して、次の値のいずれかを指定できます。

CC
動作環境のネイティブキャリッジコントロール文字を使用します。

FORTCC
FORTRAN 形式のキャリッジコントロールを使用します。外部ファイルの各行の最初の列は表示されません。この列の文字は、キャリッジコントロールコードとして解釈されます。FSLIST プロシジャは、次のキャリッジコントロール文字を認識します。

+
ゼロ行をスキップして印刷(重ね打ち)。

blank
1 行をスキップして印刷(シングルスペース)。

0
2 行をスキップして印刷(ダブルスペース)。

-
3 行をスキップして印刷(トリプルスペース)。

1
新しいページに移動して印刷。

NOCC
キャリッジコントロール文字を通常のテキストとして処理します。

FSLIST プロシジャがファイルの属性から、ファイルにキャリッジコントロール情報が含まれていることを判別できれば、表示されているテキストのフォーマットにそのキャリッジコントロール情報が使用されます。この場合、CC オプションがデフォルトになります。それ以外の場合は、ファイルのコンテンツ全体がテキストとして処理されます。この場合、NOCC オプションがデフォルトになります。

注: 一部の動作環境では、FORTRAN 形式のキャリッジコントロールがネイティブのキャリッジコントロールになります。このような環境の場合、FORTCC オプションと CC オプションの動作は同じになります。

HSCROLL=*n* | HALF | PAGE
LEFT コマンドと RIGHT コマンドに対するデフォルトの水平スクロール幅を示します。有効な値は次のとおりです。

n
デフォルトのスクロール幅を *n* 列に設定します。

HALF
デフォルトのスクロール幅をウィンドウ幅の半分に設定します。

PAGE
デフォルトのスクロール幅をウィンドウ幅に設定します。

デフォルトは HSCROLL=HALF です。HSCROLL コマンドを **FSLIST** ウィンドウで使用して、デフォルトのスクロール幅を変更できます。

NOBORDER
FSLIST ウィンドウの境界の側面と下部を非表示にします。このオプションを使用すると、通常は境界に使用される列と行にテキストを表示できます。

NUM | NONUM
レコード長が 80 で、列 73 から 80 までにシーケンス番号を含むファイルの行シーケンス番号の表示を制御します。NUM は行シーケンス番号を表示します。NONUM はこれらを非表示にします。

デフォルト NONUM

OVP | NOOVP
重ね打ちのキャリッジコントロールコードが有効かどうかを示します。

OVP
プロシジャによって重ね打ちコードが有効になり、このコードが検出されると、以前の行を上書きして現在の行が印刷されるようになります。

NOOVP
プロシジャによって重ね打ちコードが無視され、表示されている個別の行にファイルの各行が印刷されるようになります。

FSLIST コマンド

FSLIST セッションを SAS ウィンドウから開始します。コマンドでファイル参照名またはファイル名を使用して、参照するファイルを指定できます。キャリッジコントロール情報がどのように解釈されるかを指定することもできます。

構文

FSLIST <* | ?| *file-specification* <*carriage-control-option* <*overprinting-option*>>>

引数なし

これらの 3 つの引数をいずれも指定しない場合、外部のファイル名を選択できる選択ウィンドウが表示されます。

任意引数

*
参照するファイル名をさまざまな FSLIST プロシジャオプションとともに指定できるダイアログボックスが開かれます。ダイアログボックスで、物理ファイル名、ファイル参照名、ディレクトリ名のいずれかを指定できます。ディレクトリ名を指定すると、

ディレクトリ内のファイルの選択リストが表示され、そこから対象ファイルを選択できます。

?

参照する外部ファイルを選択できる選択ウィンドウが開きます。ウィンドウの選択リストには、現在の SAS セッションで識別されるすべての外部ファイル(ファイル参照名が定義されているすべてのファイル)が含まれます。

ファイルを選択するには、該当するファイル参照名の上にカーソルを合わせて、ENTER を押します。

注 現在の SAS セッション内で定義されるファイル参照名のみが選択リストに表示されます。一部の動作環境では、SAS 外のファイル参照名を割り当てることができます。そのようなファイル参照名は、FSLIST コマンドによって表示される選択リストには表示されません。

選択ウィンドウは、ファイル参照名が現在の SAS セッションで定義されていない場合は開きません。代わりに、FSLIST コマンドでファイル名を入力するように指示するエラーメッセージが書き込まれます。

file-specification

参照する外部ファイルを特定します。*File-specification* は次のいずれかにすることができます。

'external-file'

外部ファイルに対する完全な動作環境ファイル指定(一部の動作環境では完全修飾パス名と呼ばれる)。名前を引用符で囲む必要があります。

指定したファイルが見つからない場合、利用可能なすべてのファイル参照名が示された選択ウィンドウが表示されます。

fileref

現在外部ファイルに割り当てられているファイル参照名。現在定義されていないファイル参照名を指定すると、利用可能なすべてのファイル参照名が示された選択ウィンドウが表示されます。選択ウィンドウのエラーメッセージには、指定されたファイル参照名が定義されていないことが示されます。

file-specification を FSLIST コマンドで指定すると、次のキャリッジコントロールまたは重ね打ちオプションを使用することもできます。?引数が使用されている場合、または引数が使用されない場合、これらのオプションは無効になります。

CC | FORTCC | NOCC

キャリッジコントロール文字が表示のフォーマットに使用されるかどうかを示します。

FSLIST プロシジャがファイルの属性から、ファイルにキャリッジコントロール情報が含まれていることを判別できれば、表示されているテキストのフォーマットにそのキャリッジコントロール情報が使用されます。この場合、CC オプションがデフォルトになります。それ以外の場合は、ファイルのコンテンツ全体がテキストとして処理されます。この場合、NOCC オプションがデフォルトになります。

このオプションに対して、次の値のいずれかを指定できます。

CC

動作環境のネイティブキャリッジコントロール文字を使用します。

FORTCC

FORTRAN 形式のキャリッジコントロールを使用します。詳細については、PROC FSLIST ステートメントの FORTCC オプションに関する説明を参照してください。

NOCC
キャリッジコントロール文字を通常のテキストとして処理します。

OVP | NOOVP
重ね打ちのキャリッジコントロールコードが有効かどうかを示します。OVP により、重ね打ちコードが有効になります。NOOVP で無効になります。OVP オプションは、NOCC が有効な場合は無視されます。

デフォルト NOOVP

FSLIST ウィンドウの使用

FSLIST ウィンドウの概要

FSLIST ウィンドウには、参照専用でファイルが表示されます。ファイルは **FSLIST** ウィンドウでは編集できません。ただし動作環境によっては、次のいずれかの方法でテキストを **FSLIST** ウィンドウからペーストバッファにコピーできます。

- マウスを使用してテキストを選択し、**編集**メニューから**コピー**を選択します。
- グローバル MARK および STORE コマンドを使用します。

動作環境によっては、SAS/FSP ソフトウェアの **FSLETTER** ウィンドウなど、SAS テキストエディタを使用する SAS ウィンドウや、テキストの貼り付けを行えるその他のアプリケーションに、コピーしたテキストを貼り付けることができます。

コマンドウィンドウまたはコマンド行でコマンドを使用し、**FSLIST** ウィンドウを制御できます。

FSLIST ウィンドウコマンド

グローバルコマンド

FSLIST ウィンドウでは、*SAS/FSP Procedures Guide* で説明されているすべてのグローバルコマンドを使用できます。

スクロールコマンド

n
テキストの行 *n* がウィンドウの最上部にくるようにウィンドウをスクロールします。コマンドウィンドウまたはコマンド行に対象となる行番号を入力して、ENTER を押します。*n* がファイルの行数を超える場合、ファイルの最後の数行はウィンドウの最上部に表示されます。

BACKWARD <*n*|HALF | PAGE | MAX>
ファイルの最初の行に向かって垂直にスクロールします。次のスクロール幅を指定できます。

n
指定された行数分だけ上にスクロールします。

HALF
ウィンドウの行数の半分だけ上にスクロールします。

PAGE
ウィンドウの行数分だけ上にスクロールします。

MAX
ファイルの最初の行が表示されるまで上にスクロールします。

スクロール幅が明示的に指定されていない場合、ウィンドウは最新の VSCROLL コマンドで指定された幅の分だけスクロールされます。デフォルトの VSCROLL 幅は PAGE です。

BOTTOM
ファイルの最後の行が表示されるまで下にスクロールします。

FORWARD <*n*|HALF | PAGE | MAX>
ファイルの最後に向かって垂直にスクロールします。次のスクロール幅を指定できます。

n
指定された行数分だけ下にスクロールします。

HALF
ウィンドウの行数の半分だけ下にスクロールします。

PAGE
ウィンドウの行数分だけ下にスクロールします。

MAX
ファイルの最初の行が表示されるまで下にスクロールします。

スクロール幅が明示的に指定されていない場合、ウィンドウは最新の VSCROLL コマンドで指定された幅の分だけスクロールされます。デフォルトの VSCROLL 幅は PAGE です。クロール幅に関係なく、このコマンドはファイルの最終行を超えてスクロールしません。

HSCROLL <*n*|HALF | PAGE>
LEFT コマンドと RIGHT コマンドに対するデフォルトの水平スクロール幅を設定します。次のスクロール幅を指定できます。

n
デフォルトのスクロール幅を指定された列の数に設定します。

HALF
デフォルトのスクロール幅をウィンドウの列数の半分に設定します。

PAGE
デフォルトのスクロール幅をウィンドウの列数に設定します。

デフォルトの HSCROLL 幅は HALF です。

LEFT <*n*|HALF | PAGE | MAX>
テキストの左余白に向かって水平にスクロールします。このコマンドは、ファイル幅がウィンドウ幅を超えない限り、無視されます。次のスクロール幅を指定できます。

n
指定された列数分だけ左にスクロールします。

HALF
ウィンドウの列数の半分だけ左にスクロールします。

PAGE
ウィンドウの列数分だけ左にスクロールします。

MAX
テキストの左余白が表示されるまで左にスクロールします。

スクロール幅が明示的に指定されていない場合、ウィンドウは最新の HSCROLL コマンドで指定された幅の分だけスクロールされます。デフォルトの HSCROLL 幅

は HALF です。スクロール幅に関係なく、このコマンドはテキストの左余白を超えてスクロールしません。

RIGHT <*n*|HALF | PAGE | MAX>
テキストの右余白に向かって水平にスクロールします。このコマンドは、ファイル幅がウィンドウ幅を超えない限り、無視されます。次のスクロール幅を指定できます。

n
指定された列数分だけ右にスクロールします。

HALF
ウィンドウの列数の半分だけ右にスクロールします。

PAGE
ウィンドウの列数分だけ右にスクロールします。

MAX
テキストの右余白が表示されるまで右にスクロールします。

スクロール幅が明示的に指定されていない場合、ウィンドウは最新の HSCROLL コマンドで指定された幅の分だけスクロールされます。デフォルトの HSCROLL 幅は HALF です。スクロール幅に関係なく、このコマンドはテキストの右余白を超えてスクロールしません。

TOP
ファイルのテキストの最初の行が表示されるまで上にスクロールします。

VSCROLL <*n* | HALF | PAGE>
FORWARD コマンドと BACKWARD コマンドに対するデフォルトの垂直スクロール幅を設定します。次のスクロール幅を指定できます。

n
デフォルトのスクロール幅を指定した行数に設定します。

HALF
デフォルトのスクロール幅をウィンドウの行数の半分に設定します。

PAGE
デフォルトのスクロール幅をウィンドウの行数に設定します。

デフォルトの VSCROLL 幅は PAGE です。

検索コマンド

BFIND <*search-string* <PREFIX | SUFFIX | WORD>>
ファイル内の指定された文字列の以前の発生を検索します。現在のカーソル位置から始まり、ファイルの始めに向かって後方に処理されます。*search-string* 値は、ブランクが埋め込まれている場合は引用符で囲む必要があります。

FIND コマンドが以前に発行されていた場合、BFIND コマンドを引数なしで使用し、逆方向での検索を繰り返すことができます。

PROC FSLIST ステートメントの CAPS オプションおよび CAPS ON コマンドによって、文字列が引用符で囲まれている場合を除き、検索文字列が大文字に変換されて検索されます。詳細については、FIND コマンドの説明を参照してください。

デフォルトで、BFIND コマンドは指定した文字列の発生を検索します。これは、文字列がその他の文字列に埋め込まれている場合でも同様です。次のオプションのいずれかを使用して、コマンドの動作を変更できます。

PREFIX
検索文字列が語句の始めに発生したテキスト文字列にのみ一致するようにします。

SUFFIX
検索文字列が語句の最後に発生したテキスト文字列にのみ一致するようにします。

WORD
検索文字列が個別語句のテキスト文字列にのみ一致するようにします。

RFIND コマンドを使用して、最新の BFIND コマンドを繰り返すことができます。

CAPS <ON | OFF>
FIND、BFIND、RFIND コマンドが検索文字列の一致を検索する方法を制御します。デフォルトで、FIND、BFIND および RFIND コマンドは入力された検索文字列と完全に一致するテキスト文字列のみを検索します。CAPS コマンドを発行すると、文字列が引用符で囲まれている場合を除き、FIND、BFIND および RFIND コマンドは検索文字列を大文字に変換してから検索します(表示されているテキストは影響を受けません)。引用符で囲まれている文字列は影響を受けません。

たとえば、CAPS ON コマンドを発行すると、次の両方のコマンドによって **NC** の発生が検索されますが、**nc** の発生は検索されません:**find NC**, **find nc**。ON 引数または OFF 引数を省略すると、CAPS コマンドはトグルとして機能し、オフの場合は属性がオンに、オンの場合は属性がオフになります。

FIND *search-string* <NEXT | FIRST | LAST | PREV | ALL> <PREFIX | SUFFIX | WORD>
ファイル内の指定された *search-string* の発生を検索します。*search-string* は、ブランクが埋め込まれている場合は引用符で囲む必要があります。

search-string のテキストは、文字と大文字/小文字の両方に関して、ファイルのテキストと一致している必要があります。たとえば、次のコマンドは、**raleigh** の発生を検索します:**find raleigh**.次のコマンドは **Raleigh** の発生を検索します:**find Raleigh**。

CAPS オプションと PROC FSLIST ステートメントを一緒に使用する場合、または CAPS ON コマンドをウィドウで発行する場合、検索文字列は大文字に変換されてから検索されます。ただし、検索文字列が引用符で囲まれている場合は変換されません。この場合、コマンド **find raleigh** は、ファイル内でテキスト **RALEIGH** のみを検索します。テキスト **Raleigh** を検索するには、代わりにコマンド **find 'Raleigh'**を使用する必要があります。

次のオプションのいずれかを追加して、FIND コマンドの動作を変更できます。

ALL
ウィンドウのメッセージ行のファイル内で文字列が発生した合計数をレポートし、最初に発生した箇所にカーソルを移動します。

FIRST
ファイル内で文字列が最初に発生した箇所にカーソルを移動します。

LAST
ファイル内で文字列が最後に発生した箇所にカーソルを移動します。

NEXT
ファイル内で文字列が次に発生する箇所にカーソルを移動します。

PREV
ファイル内で文字列が前に発生した箇所にカーソルを移動します。

デフォルトのオプションは NEXT です。

デフォルトで、FIND コマンドは、指定した文字列が他の文字列に埋め込まれている場合でも、その文字列の発生を検索します。次のオプションのいずれかを使用して、コマンドの動作を変更できます。

PREFIX
検索文字列が語句の始めに発生したテキスト文字列にのみ一致するようにします。

SUFFIX
検索文字列が語句の最後に発生したテキスト文字列にのみ一致するようにします。

WORD
検索文字列が個別語句のテキスト文字列にのみ一致するようにします。

FIND コマンドを発行すると、RFIND コマンドを使用した文字列の次の発生の検索、または BFIND コマンドを使用した前の発生の検索を繰り返すことができます。

RFIND
最新の FIND コマンドを繰り返します。現在のカーソル位置から始まり、ファイルの最後に向かって前方に処理されます。

表示コマンド

COLUMN <ON | OFF>
FSLIST ウィンドウのメッセージ行の下に列のルーラーを表示します。ルーラーは、特定の文字が存在する列を判別する必要がある場合に便利です。ON または OFF の指定を省略すると、COLUMN コマンドはトグルとして機能し、オフの場合はルーラーをオンにし、オンの場合はルーラーをオフにします。

HEX <ON | OFF>
FSLIST ウィンドウの特殊な 16 進数表示形式を制御します。16 進形式がオンの場合、ファイルからの各文字行が表示の 3 行を占めるようになります。最初の行は文字として表示される行です。表示の次の 2 行には、テキスト行の文字に対する動作環境の文字コードの 16 進値が表示されます。16 進値は垂直に表示されます。最も重要なバイトは最上部に表示されます。ON または OFF の指定を省略すると、HEX コマンドはトグルとして機能し、オフの場合は 16 進表記オンにし、オンの場合は 16 進表記をオフにします。

NUMS <ON | OFF>
行番号がウィンドウの左側に表示されるかどうかを制御します。デフォルトで、行番号は表示されません。行番号がオンの場合、ウィンドウのテキストが右や左にスクロールされても、表示の左側に表示されたままになります。ON 引数または OFF 引数を省略すると、NUMS コマンドはトグルとして機能し、オフの場合は行番号をオンにし、オンの場合は行番号をオフにします。

その他のコマンド

BROWSE *fileref* | '*actual-filename*' <CC | FORTCC | NOCC <OVP | NOOVP>>
現在のファイルを閉じ、**FSVIEW** ウィンドウで指定されたファイルを表示します。以前にファイルと関連付けられたファイル参照名か、引用符で囲まれた実際のファイル名のいずれかを指定できます。BROWSE コマンドは、FSLIST コマンドと同じキャリッジコントロールオプションを受け入れることもできます。詳細については、“任意引数” (928 ページ)を参照してください。

END
FSLIST ウィンドウを閉じ、FSLIST セッションを終了します。

HELP <*command*>
FSLIST プロシジャに関する情報と、**FSLIST** ウィンドウで使用可能なコマンドに関する情報が示される **Help** ウィンドウを開きます。特定の **FSLIST** ウィンドウコマンドに関する情報を取得するには、対象のコマンドの名前を使用して HELP コマンドに従います。

KEYS

opens the **FSLIST** ウィンドウのファンクションキー定義を参照および編集するための **KEYS** ウィンドウが開きます。**FSLIST** ウィンドウのデフォルトキー定義は Sashelp.Fsp カタログの FSLIST.KEYS エントリに保存されます。

KEYS ウィンドウでキー定義を変更すると、新しい FSLIST.KEYS エントリが個人の PROFILE カタログ(Sasuser ライブラリが割り当てられていない場合は Sasuser.Profile または Work.Profile)に作成されます。

FSLIST プロシジャが開始すると、個人の PROFILE カタログの FSLIST.KEYS エントリのキー定義が最初に検索されます。エントリが存在しない場合、SASHELP.FSP カタログのデフォルトのエントリが使用されます。

29 章

GROOVY プロシジャ

概要: GROOVY プロシジャ

Groovy は、Java Virtual Machine (JVM)で実行する動的な言語です。PROC GROOVY により、SAS コードが Groovy コードを JVM で実行できるようになります。

PROC GROOVY は、SAS コードの一部として作成された Groovy ステートメントを実行できます。また、PROC GROOVY コマンドで指定したファイル内のステートメントを実行できます。Groovy ステートメントを解析して Groovy Class オブジェクトにし、これらのオブジェクトを実行するか、またはその他の PROC GROOVY ステートメント、Java DATA Step Objects で利用できるようにすることができます。PROC GROOVY を使用して、CLASSPATH 環境変数を追加の CLASSPATH 文字列、または jar ファイルへのファイル参照名で更新することもできます。

注:

- PROC GROOVY とサブミットされる Groovy コードはプロセスの所有者として実行し、プロセスの所有者と同じリソース(ファイルシステム、ネットワークなど)へアクセス権があります。リソースへの Groovy コードアクセスにより、SAS コードが Stored Process Server などのマルチユーザーサーバー内で実行中のときに問題が発生する可能性があります。この機能に対する一部の制御を管理者に付与するため、PROC GROOVY は NOXCMD オプションがオフの場合にのみ実行します。すべての SAS サーバーでは、NOXCMD オプションがオンに設定されています。

- SAS ログに Java によって出力されるテキストの最初のバイトのパーセント文字(%)の使用は、SAS によって予約されます。Java テキスト行の最初のバイトでパーセント文字を出力する必要がある場合、別のパーセント文字を直後に続ける必要があります(%%)。
- PROC GROOVY は、SAS システムオプション THREADS | NOTHREADS はサポートしていません。ただし、PROC GROOVY でサブミットする Groovy コードにより、JVM のスレッド処理を使用できます。

構文: GROOVY プロシジャ

制限事項: NOXCMD オプションがオンの場合は PROC GROOVY は機能しません。

操作: SAS サーバーがロック状態のときは、PROC GROOVY は実行されません。詳細については、“SAS Processing Restrictions for Servers in a Locked-Down State” (*SAS Language Reference: Concepts*)を参照してください。

PROC GROOVY <*classpath option(s)*>;
  ADD *classpath option(s)*;
  EVALUATE <(LOAD | PARSEONLY | NORUN)>
    "*Groovy statement string*" <*argument(s)*>;
  EXECUTE <(LOAD | PARSEONLY | NORUN)>
    Groovy filename | fileref <*argument(s)*>;
  SUBMIT <(LOAD | PARSEONLY | NORUN)> <*argument(s)*>;
    Groovy statement(s)
  ENDSUBMIT;
  CLEAR;
QUIT;

ステートメント	タスク	例
“PROC GROOVY ステートメント”	SAS コードが Groovy コードを JVM で実行できるようにします	Ex. 1
“ADD ステートメント”	現行の CLASSPATH 環境変数に、一定のクラスパスを追加します	
“EVALUATE ステートメント”	引用符で囲まれた Groovy ステートメントを解析し、Script で Run メソッドを呼び出します	
“EXECUTE ステートメント”	引用符で囲まれたパスまたはファイル参照名として指定されるファイルのコンテンツを読み込み、Script で Run メソッドを呼び出します	
“SUBMIT ステートメント”	SUBMIT コマンドと ENDSUBMIT コマンド間の Groovy ステートメントを解析し、Script で Run メソッドを呼び出します	
“ENDSUBMIT ステートメント”	SUBMIT コマンドで始まる Groovy ステートメントを終了します。	

ステートメント	タスク	例
“CLEAR ステートメント”	バインドを空にし、Groovy クラスローダーをアンロードします。	

PROC GROOVY ステートメント

SAS コードが Groovy コードを JVM で実行できるようにします。

構文

PROC GROOVY <*classpath option(s)*>;

オプション引数

classpath options
次のうちいずれかになります。

CLASSPATH=
引用符で囲まれた CLASSPATH 文字列、または現在のクラスパスに追加される特定の JAR ファイルへのファイル参照名を指定します。このパスは、ユーザーの CLASSPATH 環境変数にあるパスの後に検索されます。

別名 PATH=

SASJAR=<*version*=> | <*range*=>
現在のクラスパスに追加する必要がある Versioned Jar Repository (VJR)の jar を識別する引用符で囲まれた文字列を指定します。VERSION 値と RANGE 値は任意です。次の例のように、RANGE は VERSION よりも優先されます。

```
ADD SASJAR="sas.core";
ADD SASJAR="sas.core" version="903000.9.0.20100810190000_v930";
ADD SASJAR="sas.core" range="[0,909000]";
```

注: SAS JAR ファイルには、SAS のバージョンにまたがるソース互換性の保証はありません。この jar の将来のバージョンは、通知なしに変わる可能性があります。続行される機能については、SAS テクニカルサポートに連絡してください。

詳細

PROC GROOVY は、現在のユーザーの CLASSPATH 環境変数をそのクラスパスを構築するための基盤として使用します。CLASSPATH オプションと SASJAR オプションを使用して、パスを現在のクラスパスに追加できます。

クラスのロード時は、パスが次の順序で検索されます。

1. プロセス開始時の CLASSPATH 環境変数
2. ADD CLASSPATH ステートメントと ADD SASJAR ステートメントで実行順に追加されたパス

ADD ステートメント

現行の CLASSPATH 環境変数に一定のクラスパスを追加します。

構文

ADD *classpath option(s)*;

必須引数

classpath option(s)
次のうちいずれかになります。

CLASSPATH=
引用符で囲まれた CLASSPATH 文字列、または現在のクラスパスに追加される特定の JAR ファイルへのファイル参照名を指定します。このパスは、ユーザーの CLASSPATH 環境変数にあるパスの後に検索されます。

別名 PATH=

SASJAR=<*version*=> | <*range*=>
現在のクラスパスに追加する必要がある Versioned Jar Repository (VJR)の JAR ファイルを識別する引用符で囲まれた文字列を指定します。VERSION 値と RANGE 値は任意です。次の例のように、RANGE は VERSION よりも優先されます。

```
ADD SASJAR="sas.core";
ADD SASJAR="sas.core" version="903000.9.0.20100810190000_v930";
ADD SASJAR="sas.core" range="[0,909000]";
```

注: SAS JAR ファイルには、SAS のバージョンにまたがるソース互換性の保証はありません。この JAR ファイルの将来のバージョンは、通知なしに変わる可能性があります。続行される機能については、SAS テクニカルサポートに連絡してください。

詳細

ADD ステートメントは、指定のクラスパスを現在の CLASSPATH 環境変数に追加します。

CLASSPATH または SASJAR を少なくとも 1 つ指定する必要があります。複数の CLASSPATH または SASJAR を指定できます。

EVALUATE ステートメント

引用符で囲まれた文字列で groovy.lang.Script オブジェクトに付与される Groovy ステートメントを解析し、Script で Run メソッドを呼び出します。

構文

EVALUATE <(LOAD | PARSEONLY | NORUN)>
"*Groovy statement string*" <*argument(s)*>;

必須引数

Groovy statement string
EVALUATE コマンドによって解析される Groovy ステートメント文字列を指定します。

オプション引数

LOAD | PARSEONLY | NORUN
Groovy ステートメントを解析して groovy.lang.Script オブジェクトにしますが、実行しません。引数は相互に別名となります。

argument(s)
評価中のコードに渡される引数を指定します。

詳細

EVALUATE ステートメントは、引用符で囲まれた文字列で付与される Groovy ステートメントを解析して groovy.lang.Script オブジェクトにし、Script で Run メソッドを呼び出します。LOAD、PARSEONLY、NORUN オプションのいずれかが存在する場合、このステートメントは Groovy ステートメントを解析して Class オブジェクトにしますが、実行しません。Groovy コードによって定義されるクラスは、PROC GROOVY ステートメントまたは Java DATA Step Objects で使用できます。

EVAL は、EVALUATE ステートメントの別名です。

EXECUTE ステートメント

引用符で囲まれた文字列パスまたはファイル参照名として指定されるファイルのコンテンツを読み込みます。

構文

EXECUTE <(LOAD | PARSEONLY | NORUN)>
Groovy filename | *fileref* <*argument(s)*>;

必須引数

Groovy filename
EXECUTE ステートメントによって解析される Groovy ファイルの名前を指定します。

fileref
EXECUTE ステートメントによって解析されるファイル参照名の名前を指定します。

オプション引数

LOAD | PARSEONLY | NORUN
指定された Groovy ファイルまたはファイル参照名の Groovy ステートメントを解析して groovy.lang.Script オブジェクトにしますが、実行しません。引数は相互に別名となります。

argument(s)
実行中のコードに渡される引数を指定します。

詳細

EXECUTE ステートメントは、引用符で囲まれた文字列パスまたはファイル参照名として指定されるファイルのコンテンツを読み込みます。コンテンツは解析されて groovy.lang.Script オブジェクトになり、Run メソッドが Script で呼び出されます。LOAD、PARSEONLY、NORUN オプションのいずれかが存在する場合、このステートメントはファイルコンテンツを Class オブジェクトに解析しますが、実行しません。Groovy コードによって定義されるクラスは、PROC GROOVY ステートメントまたは Java DATA Step Objects で使用できます。

EXEC は、EXECUTE ステートメントの別名です。

注: EXEC PARSEONLY ステートメントを使用してファイルを Class にコンパイルした場合、そのファイルへの変更が将来の EXEC PARSEONLY コマンドによって有効化されるように、CLASS ステートメントをサブミットする必要があります。CLEAR ステートメントをサブミットしない場合、EXEC PARSEONLY ステートメントの発行後にファイルに行った変更は、EXEC PARSEONLY ステートメントの後続サブミットによって含まれません。再ロード可能なスクリプトを使用する必要がない場合、GroovyScriptEngine Class を使用できます。

SUBMIT ステートメント

SUBMIT コマンドは SUBMIT コマンドと ENDSUBMIT コマンドの間の Groovy ステートメントを解析して groovy.lang.Script オブジェクトにし、Script で Run メソッドを呼び出します。

構文

SUBMIT <(LOAD | PARSEONLY | NORUN)> <*argument(s)*>;
Groovy statement(s)
ENDSUBMIT;

必須引数

Groovy statement(s)
SUBMIT ステートメントによって解析されて groovy.lang.Script オブジェクトになる Groovy ステートメントを指定します。

オプション引数

LOAD | PARSEONLY | NORUN
Groovy ステートメントを解析して groovy.lang.Script オブジェクトにしますが、実行しません。引数は相互に別名となります。

argument(s)
サブミット中のコードに渡される引数を指定します。

詳細

SUBMIT ステートメントは SUBMIT コマンドと ENDSUBMIT コマンドの間の Groovy ステートメントを解析して groovy.lang.Script オブジェクトにし、Script で Run メソッドを呼び出します。LOAD、PARSEONLY、NORUN オプションのいずれかが存在する場合、このステートメントは Groovy ステートメントを解析して Class オブジェクトにしますが、実行しません。Groovy コードによって定義されるクラスは、PROC GROOVY ステートメントまたは Java DATA Step Objects で使用できます。

注:

- ENDSUBMIT ステートメントのみが 1 行にあり、その前にはブランクスペースのみがある必要があります。
- マクロ代替は、SUBMIT コマンドと NDSUBMIT コマンド間で無効化されます。
- 複数行サブミットコマンドを含む PROC GROOVY は、マクロ内では使用できません。

ENDSUBMIT ステートメント

SUBMIT コマンドで始まる Groovy ステートメントを終了します。

構文

ENDSUBMIT;

詳細

SUBMIT コマンドで始まる Groovy ステートメントを終了します。

注: ENDSUBMIT ステートメントのみが 1 行にあり、その前にはブランクスペースのみがある必要があります。

CLEAR ステートメント

バインドを空にし、Groovy クラスローダーをアンロードします。

構文

CLEAR;

詳細

CLEAR ステートメントはバインドを空にし、Groovy クラスローダーをアンロードします。このステートメントの実行時は、バインドに保存されている変数が使用できなくなります。Groovy クラスローダーにロードされるクラスも使用できなくなります。

RESET は、CLEAR ステートメントの別名です。

注: CLEAR ステートメントと RESET ステートメントのいずれも、System.Properties コレクションまたは CLASSPATH をリセットしません。

特殊変数

PROC GROOVY には、BINDING、ARGS、EXPORTS、SHELL の 4 つの特殊変数があります。これらの変数を実行中の Groovy コードで使用できるようにします。これらの変数を実行中の Groovy コードで使用できるようにします。

BINDING

BINDING 特殊変数は、PROC GROOVY の実行間のオブジェクトの状態の共有に使用されます。スコープなしで作成される変数、またはバインドに明示的に保存される変数によって生成されます。BINDING は、このセクションで説明されているその他すべての特殊変数も保持します。バインドは、CLEAR コマンドによってクリアできます。

```
proc groovy;
  eval "a = 42";
  eval "binding.b = 84";
  eval "binding.setProperty( 'c', 168 )";
quit;
proc groovy;
  eval "println ""----> ${binding.getProperty('a')}""";
  eval "println ""----> ${b}""";
  eval "println ""----> ${binding.c}""";
quit;
```

ARGS

引数は、バインドの ARGS 特殊変数の Groovy コードに渡されます。

```
proc groovy;
  eval "args.each{ println ""----> ev ${it}"" }" "arg1" "arg2" "arg3";

  exec "args.groovy" "arg1" "arg2" "arg3";

  submit "arg1" "arg2" "arg3";
    args.each{
      println "----> su ${it}"
    }
  endsubmit;
quit;
```

EXPORTS

EXPORTS 特殊変数には、バインドにマップが含まれます。キーまたは値のペアをこのマップに追加すると、PROC GROOVY の終了時に SAS マクロ変数が作成されます。Groovy は大文字と小文字を区別しますが、マクロは区別しません。ケースのみ異なる 2 つのキーがマップに存在する場合、SAS マクロにエクスポートされるキーは決定されません。java.util.Map から継承するオブジェクトを含むバインドの EXPORTS 変数を置き換えることもできます。変数を置き換えると、そのオブジェクトのすべてのキーまたは値のペアがエクスポートされます。

```
proc groovy;
  eval "exports.fname = ""first name""";
  eval "binding.exports.lname = ""last name""";
  eval "exports.put('state', 'NC')";
quit;

data _NULL_;
  put "----> &fname &lname: &state";
run;

proc groovy;
```

```
  submit;
    exports = [fname:"first name", lname: "last name", state: "NC"]
  endsubmit;
quit;

data _NULL_;
  put "----> &fname &lname: &state";
run;
```

SHELL

バインドの SHELL 特殊変数は、現在のスクリプトのコンパイルに使用された groovy.lang.GroovyShell に設定されます。この例で execution.groovy ファイルに行われた変更がコードの後続実行で反映されるようにするには、CLEAR ステートメントをサブミットする必要があります。

```
proc groovy;
  eval "shell.run(
              new File(""execution.groovy""),
              [] as String[] )";
quit;
```

注: 変更時に自動的にリロードされる Groovy スクリプトが必要な場合、GroovyScriptEngine クラスの新しいインスタンスを作成します。

例: 分類の定義

要素: PROC GROOVY ステートメントオプション
CLASSPATH
SUBMIT ステートメントオプション
PARSEONLY
SUBMIT ステートメント
ENDSUBMIT ステートメント

次の例に、PROC GROOVY を使用したクラスの定義方法を示します。

プログラム

Groovy コードがデフォルトで実行されます。スクリプトに実行可能なコードが含まれていない場合、エラーが返されます。次の例ではクラスを定義しますが、実行可能なコードがなく、エラーが返されます。

```
proc groovy classpath=cp;
  submit;
    class Speaker {
      def say( word ) {
        println "----> \"${word}\""
      }
    }
  endsubmit;
quit;
```

プログラム

次の例では、**main** メソッドを含めることによる実行可能なクラスの定義方法を示します。

```
proc groovy classpath=cp;
  submit;
    class Speaker {
      def Speaker() {
        println "----> ctor"
      }
      def main( args ) {
        println "----> main"
      }
    }
  endsubmit;
quit;
```

プログラム

次の例では、PARSEONLY オプションを使用した、実行呼び出しを回避する方法を示します。この新しいクラスは、PROC GROOVY の別の実行で使用できます。

```
proc groovy classpath=cp;
  submit parseonly;
    class Speaker {
      def say( word ) {
        println "----> \"${word}\""
      }
    }
  endsubmit;
quit;

proc groovy classpath=cp;
  eval "s = new Speaker(); s.say( ""Hi"" )";
quit;
```

30 章

HADOOP プロシジャ

概要: HADOOP プロシジャ

HADOOP プロシジャを使用すると、Apache Hadoop コードを実行することで SAS が Hadoop データとやりとりできます。Apache Hadoop は、Java で作成されたオープンソースフレームワークで、データ記憶域と大量データの分散処理を提供します。

PROC HADOOP は、Hadoop JobTracker とインターフェイスでつながっています。Hadoop JobTracker は、クラスタ内の特定のノードに対するタスクを管理する、Hadoop 内のサービスです。PROC HADOOP を使用すると、次をサブミットできます。

- Hadoop Distributed File System (HDFS)コマンド
- MapReduce プログラム
- Pig 言語コード

構文: HADOOP プロシジャ

制限事項: PROC HADOOP は、z/OS 動作環境でサポートされません。

SAS がロック状態のときは、PROC HADOOP は使用できません。サーバー管理者は、ロックダウン状態でもこのプロシジャにアクセスできるように再有効化できます。LOCKDOWN ENABLE_AMS=ステートメントを使用して FILENAME Hadoop アクセスメソッドが再有効化されると、PROC HADOOP は自動的に再有効化されます。詳細については、“SAS Processing Restrictions for Servers in a Locked-Down State” (*SAS Language Reference: Concepts*)を参照してください。

要件: Java Runtime Environment(JRE) 1.6 以上

SAS 9.4 メンテナンスリリース 3 でサポートされる Hadoop ディストリビューションは、Cloudera CDH 4.7、Cloudera CDH 5.2、Hortonworks HDP 1.3.2、Hortonworks HDP 2.1、IBM InfoSphere BigInsights 3.0、MapR 3.1、MapR 4.0.1、Pivotal HD 1.1.1、Pivotal HD 2.1 です。

操作: SAS 9.4 メンテナンスリリース 3 で Hadoop クラスタに接続するには、まず SAS クライアントマシンから Hadoop クラスタ構成ファイルにアクセスできる必要があります。これには、SAS クライアントマシンからアクセスできる任意の物理的な場所に構成ファイルをコピーして、SAS 環境変数 SAS_HADOOP_CONFIG_PATH をその場所に設定します。あるいは、複数の Hadoop クラスタ構成ファイルからプロパティをマージして単一の構成ファイルを作成し、PROC HADOOP ステートメントの CFG=引数を使用してその構成ファイルを特定することもできます。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

HDFS コマンド、MapReduce プログラム、および Pig 言語コードを Java API を使用してサブミットするには、SAS クライアントマシンからアクセスできる物理的な場所に Hadoop ディストリビューション JAR ファイルをコピーする必要があります。SAS 環境変数 SAS_HADOOP_JAR_PATH に Hadoop JAR ファイルの場所を設定する必要があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

WebHDFS を経由して HDFS コマンドをサブミットするには、SAS 環境変数 SAS_HADOOP_RESTFUL 1 が設定されている必要があります。また、Hadoop 構成ファイル hdfs-site.xml に、WebHDFS の場所のプロパティが含まれている必要があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

SAS 9.4 メンテナンスリリース 3 で MapReduce プログラムと Pig 言語コードを Apache Oozie RESTful API を使用してサブミットするには、SAS 環境変数 SAS_HADOOP_RESTFUL 1 を定義する必要があります。また、SAS 環境変数 SAS_HADOOP_CONFIG_PATH を、hdfs-site.xml 構成ファイルと core-site.xml 構成ファイルが置かれている場所に設定する必要があります。hdfs-site.xml ファイルには、WebHDFS の場所のプロパティが含まれている必要があります。また、Oozie 固有のプロパティを構成ファイルに指定して、PROC HADOOP ステートメントの CFG=引数でその構成ファイルを特定できるようにする必要もあります。Oozie 固有のプロパティには、oozie_http_port、fs.default.name、および mapred.job.tracker があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

PROC HADOOP <*hadoop-server-option(s)*>;
HDFS <*hadoop-server-option(s)* > < *hdfs-command-option(s)*>;
MAPREDUCE <*hadoop-server-option(s)*> <*mapreduce-option(s)*>;
PIG <*hadoop-server-option(s)*> <*pig-code-option(s)*>;
PROPERTIES <*configuration-properties*>;

ステートメント	タスク	例
"PROC HADOOP ステートメント"	Hadoop サーバーへのアクセスを制御します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
"HDFS ステートメント"	Hadoop Distributed File System (HDFS)コマンドをサブミットします。	Ex. 1, Ex. 2
"MAPREDUCE ステートメント"	MapReduce プログラムを Hadoop クラスタにサブミットします。	Ex. 3
"PIG ステートメント"	Pig 言語コードを Hadoop クラスタにサブミットします。	Ex. 4
"PROPERTIES ステートメント"	構成プロパティをサブミットします。	Ex. 5

PROC HADOOP ステートメント

Hadoop サーバーへのアクセスを制御します。

例: "例 1: HDFS コマンドのサブミット" (962 ページ)
"例 3: MapReduce プログラムのサブミット" (964 ページ)
"例 4: Pig 言語コードのサブミット" (966 ページ)

構文

PROC HADOOP <*hadoop-server-option(s)*>;

オプション引数の要約

AUTHDOMAIN='*authentication-domain*'
Hadoop サーバーに接続するために認証ドメインメタデータオブジェクトの名前を指定します。

CFG=*fileref* | '*external-file*'
Hadoop サーバーに接続するために使用する Hadoop 構成ファイルを指定します。

MAXWAIT=*wait-interval*
WebHDFS を使用する場合の HTTP ステータスの応答時間を指定します。

PASSWORD='*password*'
Hadoop サーバーのユーザー ID のパスワードです。

USERNAME='*ID*'
Hadoop サーバーの認証ユーザー ID です。

VERBOSE
SAS ログに表示される追加のメッセージを有効にします。

Hadoop サーバーオプション

次のオプションで Hadoop サーバーへのアクセスを制御します。次のオプションはすべての HADOOP プロシジャステートメントで指定できます。

AUTHDOMAIN='*authentication-domain*'
Hadoop サーバーに接続するために認証ドメインメタデータオブジェクトの名前を指定します。認証情報(ユーザー ID とパスワード)を明示的に指定しなくても、認証ドメインは認証情報を参照します。

管理者は、SAS Management Console の User Manager でユーザー定義を作成するときに、認証ドメイン定義を作成します。認証ドメインは、Hadoop サーバーへのアクセスを提供する 1 つ以上のログインメタデータオブジェクトに関連付けられます。メタデータオブジェクトは、SAS が SAS Metadata Server を呼び出し、認証情報を返すことで、解決されます。

要件 *authentication-domain* 名を一重引用符または二重引用符で囲みます。

認証ドメインと関連付けられたログイン定義はメタデータリポジトリに格納する必要があります。また、メタデータオブジェクト仕様を解決するには、Metadata Server を実行している必要があります。

操作 AUTHDOMAIN=を指定する場合は、USERNAME=と PASSWORD=を指定しないでください。

参照項目 認証ドメインの作成と使用の詳細については、*SAS Intelligence Platform: Security Administration Guide* の認証情報の管理に関する説明を参照してください。

CFG=*fileref* | '*external-file*'
Hadoop サーバーに接続するために使用する Hadoop 構成ファイルを指定します。構成ファイルには、fs.defaultFS などのファイルシステムプロパティを含む Hadoop システム情報のエントリが格納されます。構成ファイルは、Hadoop の core-site.xml ファイルのコピーであることもできます。ただし、HDFS のフェイルオーバーが有効な Hadoop クラスタで動作している場合は、Hadoop の core-site.xml と hdfs-site.xml とを組み合わせたファイルを作成する必要があります。構成ファイルには、指定したサーバーの名前と JobTracker アドレスが指定されている必要があります。

fileref
Hadoop 構成ファイルに割り当てられる SAS ファイル参照を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'*external-file*'
XML ドキュメントの物理的な場所です。完全パス名とファイル名を含みます。最大長は、200 文字です。

要件 物理名を一重引用符または二重引用符で囲みます。

別名 OPTIONS=

要件 ファイルは XML ドキュメントである必要があります。

MAXWAIT=*wait-interval*
WebHDFS を使用する場合の HTTP ステータスの応答時間を指定します。

デフォルト 40000 ミリ秒

要件 環境変数 SAS_HADOOP_RESTFUL 1 が設定されている必要があります。

ヒント ログにタイムアウトメッセージが報告されている場合は、MAXWAIT=を使用して待機時間を増やします。

PASSWORD='*password*'
Hadoop サーバーのユーザー ID のパスワードです。ユーザー ID とパスワードは、CFG=で指定された一連のオプションに追加されます。

別名 PASS=

操作 PASSWORD=を指定するには、USERNAME=も指定する必要があります。

USERNAME='*ID*'
Hadoop サーバーの認証ユーザー ID です。ユーザー ID とパスワードは、CFG=で指定された一連のオプションに追加されます。

別名 USER=

VERBOSE
SAS ログに表示される追加のメッセージを有効にします。VERBOSE は、優れたエラー診断ツールです。SAS を呼び出すときにエラーメッセージが表示された場合は、このオプションを使用して、システムオプションの指定にエラーがあるかどうか確認できます。

HDFS ステートメント

Hadoop Distributed File System (HDFS)コマンドをサブミットします。

制限事項: HDFS ステートメントは、1 回の呼び出しで 1 つの操作のみ実行できます。

操作: HDFS コマンドを Java API を使用してサブミットするには、SAS クライアントマシンからアクセスできる物理的な場所に Hadoop ディストリビューション JAR ファイルをコピーする必要があります。SAS 環境変数 SAS_HADOOP_JAR_PATH に Hadoop JAR ファイルの場所を設定する必要があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

WebHDFS を経由して HDFS コマンドをサブミットするには、SAS 環境変数 SAS_HADOOP_RESTFUL 1 が設定されている必要があります。さらに、Hadoop 構成ファイルに WebHDFS の場所のプロパティが含まれている必要があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

例: “例 1: HDFS コマンドのサブミット” (962 ページ)
“例 2: ワイルドカード文字を使用した HDFS コマンドのサブミット” (963 ページ)

構文

HDFS <*hadoop-server-option(s)* > < *hdfs-command-option(s)*>;

オプション引数の要約

CAT='*HDFS-file*' <ONLY=*n*> <OUT='*output-location*'> <RECURSE>
<SHOW_FILENAME>
指定した 1 つ以上のファイルの内容を表示します。

CHMOD='*HDFS-file*' PERMISSION=<'>*value*<'> <RECURSE>
1 つ以上の HDFS ファイルに対するファイルアクセス権限を変更します。

COPYFROMLOCAL='*local-file*' OUT='*output-location*' <DELETESOURCE>
<OVERWRITE> <RECURSE>
指定されたローカルファイルを HDFS パスの出力先にコピーします。

COPYTOLOCAL='*HDFS-file*' OUT='*output-location*' <DELETESOURCE>
<KEEPCRC> <OVERWRITE> <RECURSE>
指定された HDFS ファイルをローカルファイルの出力先にコピーします。

DELETE='*HDFS-file*' <NOWARN> <RECURSE>
指定された HDFS ファイルを削除します。

LS='*HDFS-pathname*' <OUT=*output-location*><RECURSE>
指定された HDFS パス名にあるファイルをリストします。

MKDIR='*HDFS-pathname*'
指定された HDFS パス名を作成します。完全な HDFS パス名を指定します。

RENAME='*HDFS-file*' OUT='*output-location*'
指定された HDFS ファイルの名前を変更します。

HDFS コマンドオプション

次のオプションは、HDFS とやりとりするコマンドをサポートします。1 つの HDFS ステートメントに 1 つの操作のみを含めます。

**CAT='*HDFS-file*' <ONLY=*n*> <OUT='*output-location*'> <RECURSE>
<SHOW_FILENAME>**
指定した 1 つ以上のファイルの内容を表示します。

'*HDFS-file*'
パス名、またはパス名とファイル名を指定します。ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して 1 つ以上の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

ONLY=*n*
ファイルの最初から、指定した行数のみを表示します。たとえば、`only=10` を指定するとファイルの最初の 10 行が表示されます。このオプションは、ファイルの内容を判別するときに役立ちます。

OUT='*output-location*'
内容の出力先を指定します。マシンの外部にあるファイルにすることも、FILENAME ステートメントで割り当てられたファイル参照名にすることもできます。デフォルトで、出力先は SAS log になります。

RECURSE
指定されたパス名にあるすべてのファイル、およびサブディレクトリにあるすべてのファイルの内容が表示されるように指定します。RECURSE は、指定された HDFS ファイルがディレクトリの場合のみ有効です。

SHOW_FILENAME
出力にファイルの名前を含めます。たとえば、`hdfs cat='/tmp/*.txt' show_filename only=10 recurse;`のように指定すると、SAS ログにその

ファイルの名前と、/tmp ディレクトリとそのすべてのサブディレクトリで見つかったすべての.txt ファイルの最初の 10 行が表示されます。

要件 WebHDFS を経由して Hadoop に接続する必要があります。

注 SAS 9.4 メンテナンスリリース 3 では、CAT=オプションを使用できます。

CHMOD='*HDFS-file*' PERMISSION=<'>*value*<'> <RECURSE>
1 つ以上の HDFS ファイルに対するファイルアクセス権限を変更します。

'*HDFS-file*'
パス名、またはパス名とファイル名を指定します。ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

PERMISSION=*value*
owner、group、および user という 3 つのレベルの権限を表す値を指定します。3 つすべての権限レベルが必要です。read、write、および execute (rwx)という記号表記または 8 進数表記で権限を指定できます。

- rwx 記号表記の場合は、9 文字を使用します。最初の 3 文字のセットは所有者が実行できる事柄を表し、2 番目のセットはグループが実行できる事柄を表し、3 番目のセットはユーザーが実行できる事柄を表します。3 文字のそれぞれのセットについて、最初の位置には r または- (読み取り権限を示す)、2 番目の位置には w または- (書き込み権限を示す)、3 番目の位置には x または- (実行権限を示す)を指定する必要があります。たとえば **permission=rwxr-xr-x** のように指定すると、所有者には読み取り、書き込み、および実行権限があり、グループメンバには読み取りおよび実行権限があり、ユーザーには読み取りおよび実行権限があることになります。
- 8 進法表記では、3 つの数字を使用します。各数字は所有者、グループ、ユーザーの権限を表します。各数字は 0 から 7 までにする必要があります。8 進数表記は、rwx 記号表記と同じ数値を表します。つまり、4 は r、2 は w、1 は x、0 は-をそれぞれ表します。たとえば **permission=755;**のように指定すると、所有者には読み取り、書き込み、および実行権限があり、グループメンバには読み取りおよび実行権限があり、ユーザーには読み取りおよび実行権限があることになります。

RECURSE
指定されたパス名にあるすべてのファイルとディレクトリ、およびサブディレクトリにあるすべてのファイルとディレクトリに対するアクセス権限を変更するように指定します。RECURSE は、指定された HDFS ファイルがディレクトリの場合のみ有効です。たとえば、**hdfs chmod='/tmp' permission=755 recurse;**のように指定すると、指定されたディレクトリと、そのディレクトリ内にあるすべてのファイルとサブディレクトリに対する権限が変更されます。

要件 WebHDFS を経由して Hadoop に接続する必要があります。

注 SAS 9.4 メンテナンスリリース 3 では、CHMOD=オプションを使用できます。

COPYFROMLOCAL='*local-file*' OUT='*output-location*' <DELETESOURCE> <OVERWRITE> <RECURSE>
指定されたローカルファイルを HDFS パスの出力先にコピーします。

'*local-file*'
完全パス名とファイル名を指定します。SAS 9.4 メンテナンスリリース 3 では、ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数

の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

OUT='*output-location*'
コピーされたファイルの出力先を指定します。これは完全な HDFS パス名およびファイル名です。

DELETESOURCE
コピーコマンドの後に入力ソースファイルを削除します。

OVERWRITE
既存の出力先を上書きするように指定します。

RECURSE
指定されたパス名にあるすべてのファイル、およびサブディレクトリにあるすべてのファイルをコピーするように指定します。RECURSE は、指定されたファイルがディレクトリの場合のみ有効です。

注 SAS 9.4 メンテナンスリリース 3 では、RECURSE オプションを使用できます。

COPYTOLOCAL='*HDFS-file*' OUT='*output-location*' <DELETESOURCE> <KEEPCRC> <OVERWRITE> <RECURSE>
指定された HDFS ファイルをローカルファイルの出力先にコピーします。

'*HDFS-file*'
完全パス名とファイル名を指定します。SAS 9.4 メンテナンスリリース 3 では、ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

OUT='*output-location*'
コピーされたファイルの出力先を指定します。これはマシンの外部ファイルです。

DELETESOURCE
コピーコマンドの後に入力ソースファイルを削除します。

KEEPCRC
コピーコマンドの後に巡回冗長チェック(CRC)ファイルをローカル出力先に保存します。CRC ファイルは、OUT=オプションで指定された場所と同じ場所に保存されます。CRC ファイルは、コピーされているファイルの正確さを確認するために使用されます。デフォルトで、CRC ファイルは削除されます。

OVERWRITE
既存の出力先を上書きするように指定します。

RECURSE
指定されたパス名にあるすべてのファイル、およびサブディレクトリにあるすべてのファイルをコピーするように指定します。RECURSE は、指定された HDFS ファイルがディレクトリの場合のみ有効です。

注 SAS 9.4 メンテナンスリリース 3 では、RECURSE オプションを使用できます。

DELETE='*HDFS-file*' <NOWARN> <RECURSE>
指定された HDFS ファイルを削除します。

HDFS-file
パス名、またはパス名とファイル名を指定します。ファイル名を含めると、そのファイルのみが削除されます。ファイル名を含めない場合は、指定されたパス名にあるすべてのファイルと、サブディレクトリにあるすべてのファイルが削除

されます。SAS 9.4 メンテナンスリリース 3 では、ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

NOWARN
存在しないファイルを削除しようとすると表示される警告メッセージを非表示にします。

LS='*HDFS-pathname*' <OUT=*output-location*><RECURSE>
指定された HDFS パス名にあるファイルをリストします。各ファイルの出力はその権限、ユーザー ID、ユーザー ID グループ、ファイルサイズ、作成日、作成時刻、ファイル名で構成されます。

HDFS-pathname
パス名を指定します。ワイルドカード文字を使用して、パス名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

OUT=*output-location*
ファイルリストの出力先を指定します。これは、HDFS パス名とファイル名、ローカルファイル、または FILENAME ステートメントで割り当てられたファイル参照名にすることができます。デフォルトで、出力先は SAS log になります。

RECURSE
指定されたパス名にあるファイル、およびサブディレクトリにあるすべてのファイルをリストするように指定します。RECURSE は、指定されたファイルがディレクトリの場合のみ有効です。

デフォルト デフォルトの出力先は SAS log になります。

要件 WebHDFS を経由して Hadoop に接続する必要があります。

注 SAS 9.4 メンテナンスリリース 3 では、LS=オプションを使用できます。

MKDIR='*HDFS-pathname*'
指定された HDFS パス名を作成します。完全な HDFS パス名を指定します。

RENAME='*HDFS-file*' OUT='*output-location*'
指定された HDFS ファイルの名前を変更します。

'*HDFS-file*'
名前を変更するパス名とファイル名を指定します。

OUT='*output-location*'
新しい HDFS のパス名とファイル名を指定します。

MAPREDUCE ステートメント

MapReduce プログラムを Hadoop クラスタにサブミットします。

要件 Hadoop サーバーに MapReduce プログラムをサブミットするには、Hadoop 構成ファイルに MapReduce (MR1)または MapReduce 2 (MR2)と YARN を実行するためのプロパティが含まれている必要があります。

操作: MapReduce プログラムを Java API を使用してサブミットするには、SAS クライアントマシンからアクセスできる物理的な場所に Hadoop ディストリビューション JAR ファイルをコピーする必要があります。SAS 環境変数 SAS_HADOOP_JAR_PATH に Hadoop JAR フ

ァイルの場所を設定する必要があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

SAS 9.4 メンテナンスリリース 3 で MapReduce プログラムを Apache Oozie RESTful API を使用してサブミットするには、SAS 環境変数 SAS_HADOOP_RESTFUL 1 を定義する必要があります。また、SAS 環境変数 SAS_HADOOP_CONFIG_PATH を、hdfs-site.xml 構成ファイルと core-site.xml 構成ファイルが置かれている場所に設定する必要があります。hdfs-site.xml ファイルには、WebHDFS の場所のプロパティが含まれている必要があります。また、Oozie 固有のプロパティを構成ファイルに指定して、PROC HADOOP ステートメントの CFG=引数でその構成ファイルを特定できるようにする必要もあります。Oozie 固有のプロパティには、oozie_http_port、fs.default.name、および mapred.job.tracker があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

例: “例 3: MapReduce プログラムのサブミット” (964 ページ)

構文

MAPREDUCE <*hadoop-server-option(s)*> <*mapreduce-option(s)*>;

オプション引数の要約

COMBINE='*class-name*'
: combiner クラスの名前をドット表記で指定します。

DELETERESULTS
: MapReduce の結果を削除するように指定します。

GROUPCOMPARE='*class-name*'
: grouping comparator (GroupComparator)クラスの名前をドット表記で指定します。

INPUT='*HDFS-pathname*'
: MapReduce 入力ファイルへの HDFS パス名を指定します。

INPUTFORMAT='*class-name*'
: input format クラスの名前をドット表記で指定します。

JAR='*external-file(s)*'
: MapReduce プログラムと名前の付いたクラスを含む JAR ファイルの場所を指定します。

MAP='*class-name*'
: map クラスの名前をドット表記で指定します。

OUTPUT='*HDFS-pathname*'
: Oozie RESTful API を介して Hadoop サーバーに接続している場合、MapReduce 出力用に新しい HDFS パス名を指定します。

OUTPUTFORMAT='*class-name*'
: output format クラスの名前をドット表記で指定します。

OUTPUTKEY='*class-name*'
: output key クラスの名前をドット表記で指定します。

OUTPUTVALUE='*class-name*'
: ドット表記で表された output value クラスの名前です。

PARTITIONER='*class-name*'
: partitioner クラスの名前をドット表記で指定します。

REDUCE='*class-name*'
: reducer クラスの名前をドット表記で指定します。

REDUCETASKS='*integer*'
: reduce タスクの数を指定します。

REPLACE
Oozie RESTful API を介して Hadoop に接続している場合、Oozie アプリケーションで既存のワークフローまたは JAR ファイルを上書きしてから、作業ディレクトリにコピーするように指定します。

SORTCOMPARE='*class-name*'
sort comparator クラスの名前をドット表記で指定します。

WORKINGDIR='*HDFS-pathname*'
HDFS 作業ディレクトリパス名の名前を指定します。

MapReduce オプション

COMBINE='*class-name*'
combiner クラスの名前をドット表記で指定します。

DELETERESULTS
MapReduce の結果を削除するように指定します。SAS 9.4 メンテナンスリリース 3 では、Oozie RESTfulAPI を介して Hadoop に接続している場合、この引数によって既存の出力ディレクトリが削除されてから Oozie ジョブが開始するように指定されます。

GROUPCOMPARE='*class-name*'
grouping comparator (GroupComparator)クラスの名前をドット表記で指定します。

INPUT='*HDFS-pathname*'
MapReduce 入力ファイルへの HDFS パス名を指定します。

INPUTFORMAT='*class-name*'
input format クラスの名前をドット表記で指定します。

JAR='*external-file(s)*'
MapReduce プログラムと名前の付いたクラスを含む JAR ファイルの場所を指定します。完全パス名とファイル名を含みます。

要件 各場所は一重引用符または二重引用符で囲みます。

MAP='*class-name*'
map クラスの名前をドット表記で指定します。map クラスには、キー値とマップされた値の組み合わせで構成される要素が含まれます。

OUTPUT='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop サーバーに接続している場合、MapReduce 出力用に新しい HDFS パス名を指定します。

要件 MapReduce 出力先を指定する必要があります。

物理名を一重引用符または二重引用符で囲みます。

OUTPUTFORMAT='*class-name*'
output format クラスの名前をドット表記で指定します。

OUTPUTKEY='*class-name*'
output key クラスの名前をドット表記で指定します。

OUTPUTVALUE='*class-name*'
ドット表記で表された output value クラスの名前です。

PARTITIONER='*class-name*'
partitioner クラスの名前をドット表記で指定します。partitioner クラスは、中間マップ出力のキーの区分を制御します。

REDUCE='*class-name*'
reducer クラスの名前をドット表記で指定します。reduce クラスは、キーを共有する中間値のセットを小さい値のセットに縮小します。

REDUCETASKS='*integer*'
reduce タスクの数を指定します。

REPLACE
Oozie RESTful API を介して Hadoop に接続している場合、Oozie アプリケーションで既存のワークフローまたは JAR ファイルを上書きしてから、作業ディレクトリにコピーするように指定します。デフォルトで、PROC HADOOP では既存のワークフローまたは JAR ファイルは置き換えられません。

注 SAS 9.4 メンテナンスリリース 3 では、REPLACE オプションを使用できます。

SORTCOMPARE='*class-name*'
sort comparator クラスの名前をドット表記で指定します。

WORKINGDIR='*HDFS-pathname*'
HDFS 作業ディレクトリパス名の名前を指定します。

要件 SAS 9.4 メンテナンスリリース 3 では、Oozie RESTful API を介して Hadoop に接続している場合にこの引数が必要となり、Oozie ワークフローアプリケーションディレクトリ用の HDFS パス名が指定されます。

HDFS-pathname の名前を一重引用符または二重引用符で囲みます。

PIG ステートメント

Pig 言語コードを Hadoop クラスタにサブミットします。

操作: Pig 言語コードを Java API を使用してサブミットするには、SAS クライアントマシンからアクセスできる物理的な場所に Hadoop ディストリビューション JAR ファイルをコピーする必要があります。SAS 環境変数 SAS_HADOOP_JAR_PATH に Hadoop JAR ファイルの場所を設定する必要があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

SAS 9.4 メンテナンスリリース 3 で Pig 言語コードを Apache Oozie RESTful API を使用してサブミットするには、SAS 環境変数 SAS_HADOOP_RESTFUL 1 を定義する必要があります。また、SAS 環境変数 SAS_HADOOP_CONFIG_PATH を、hdfs-site.xml 構成ファイルと core-site.xml 構成ファイルが置かれている場所に設定する必要があります。hdfs-site.xml ファイルには、WebHDFS の場所のプロパティが含まれている必要があります。また、Oozie 固有のプロパティを構成ファイルに指定して、PROC HADOOP ステートメントの CFG=引数でその構成ファイルを特定できるようにする必要もあります。Oozie 固有のプロパティには、oozie_http_port、fs.default.name、および mapred.job.tracker があります。詳細については、*SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS* を参照してください。

例: “例 4: Pig 言語コードのサブミット” (966 ページ)

構文

PIG <*hadoop-server-option(s)*> <*pig-code-option(s)*>;

オプション引数の要約

CODE=*fileref* | '*external-file*'
実行する Pig 言語コードを含むソースを指定します。

DELETERESULTS
Oozie RESTful API を介して Hadoop サーバーに接続している場合、既存の出力先を削除してから Oozie ジョブを開始するように指定します。

OUTPUT='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop サーバーに接続している場合、既存の出力先を削除してから Oozie ジョブを開始するように指定します。

PARAMETERS=*fileref* | '*external-file*'
Pig コードが実行されるときに、引数として渡されるパラメータを含むソースを指定します。

REGISTERJAR='*external-file(s)*'
実行する Pig スクリプトを含む JAR ファイルの場所を指定します。

REPLACE
Oozie RESTful API を介して Hadoop に接続している場合、Oozie アプリケーションで既存のワークフローまたは JAR ファイルを上書きしてから、作業ディレクトリにコピーするように指定します。

WORKINGDIR='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop に接続している場合、Oozie ワークフローアプリケーションディレクトリ用に HDFS パス名を指定します。

Pig コードオプション

CODE=*fileref* | '*external-file*'
実行する Pig 言語コードを含むソースを指定します。

fileref
ソースファイルに割り当てられる SAS ファイル参照名です。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'*external-file*'
ソースファイルの物理的な場所です。完全パス名とファイル名を指定します。

要件 物理名を一重引用符または二重引用符で囲みます。

DELETERESULTS
Oozie RESTful API を介して Hadoop サーバーに接続している場合、既存の出力先を削除してから Oozie ジョブを開始するように指定します。

操作 DELETERESULTS オプションは OUTPUT=オプションとともに使用します。

注 SAS 9.4 メンテナンスリリース 3 では、DELETERESULTS オプションを使用できます。

OUTPUT='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop サーバーに接続している場合、既存の出力先を削除してから Oozie ジョブを開始するように指定します。

要件 物理名を一重引用符または二重引用符で囲みます。

操作 OUTPUT=オプションは DELETERESULTS オプションとともに使用します。

注 SAS 9.4 メンテナンスリリース 3 では、OUTPUT=オプションを使用できます。

PARAMETERS=*fileref* | '*external-file*'
Pig コードが実行されるときに、引数として渡されるパラメータを含むソースを指定します。

fileref
ソースファイルに割り当てられる SAS ファイル参照名です。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'*external-file*'
ソースファイルの物理的な場所です。完全パス名とファイル名を指定します。

要件 物理名を一重引用符または二重引用符で囲みます。

REGISTERJAR='*external-file(s)*'
実行する Pig スクリプトを含む JAR ファイルの場所を指定します。完全パス名とファイル名を指定します。

要件 各場所を一重引用符または二重引用符で囲みます。

REPLACE
Oozie RESTful API を介して Hadoop に接続している場合、Oozie アプリケーションで既存のワークフローまたは JAR ファイルを上書きしてから、作業ディレクトリにコピーするように指定します。デフォルトで、PROC HADOOP では既存のワークフローまたは JAR ファイルは置き換えられません。

注 SAS 9.4 メンテナンスリリース 3 では、REPLACE オプションを使用できます。

WORKINGDIR='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop に接続している場合、Oozie ワークフローアプリケーションディレクトリ用に HDFS パス名を指定します。

要件 Oozie RESTFUL API を介して Hadoop に接続している場合、この引数は必須です。

HDFS-pathname の名前を一重引用符または二重引用符で囲みます。

注 SAS 9.4 メンテナンスリリース 3 では、WORKINGDIR=オプションを使用できます。

PROPERTIES ステートメント

構成プロパティを Hadoop サーバーにサブミットします。

別名: PROP

例: “例 5: 構成プロパティのサブミット” (967 ページ)

構文

PROPERTIES '*configuration-property-1*' <'*configuration-property-2*'> …;

必須引数

configuration-property
Hadoop 構成ファイルで指定できるプロパティを指定します。

要件 各プロパティを一重引用符または二重引用符で囲み、各値も一重引用符またはは二重引用符で囲みます。たとえば、`prop 'mapred.job.tracker'='xxx.us.company.com:8021' 'fs.default.name'='hdfs://xxx.us.company.com:8020';`のようになります。

HADOOP プロシジャの使用

Hadoop Distributed File System コマンドのサブミット

Hadoop Distributed File System (HDFS)は、Hadoop フレームワーク用の分散型でスケーラブルなポータブルファイルシステムです。HDFS は、大量のデータをネットワーク上に分散して格納し、多くのクライアントによってデータへのアクセスを提供する設計になっています。

PROC HADOOP HDFS ステートメントは、Hadoop サーバーに HDFS コマンドをサブミットします。HDFS コマンドは、Hadoop シェルコマンドと同様に、HDFS とのやりとりを行い、ファイルを操作します。HDFS コマンドのリストについては、“HDFS ステートメント” (951 ページ)を参照してください。

MapReduce プログラムのサブミット

MapReduce は、開発者が大量のデータを処理するプログラムを作成するための並列処理フレームワークです。MapReduce フレームワークには、次の 2 つの主要な関数があります。

- map 関数は処理対象のデータを独立したチャンクに分離します
- reduce 関数はデータに関する分析を実行します

PROC HADOOP MAPREDUCE ステートメントでは、MapReduce プログラムを Hadoop クラスタにサブミットします。詳細については、“MAPREDUCE ステートメント” (955 ページ)を参照してください。

Pig 言語コードのサブミット

Apache Pig 言語は、Hadoop で使用される MapReduce プログラムをサブミットする高レベルのプログラミング言語です。

PROC HADOOP PIG ステートメントは、Pig 言語コードを Hadoop クラスタにサブミットします。詳細については、“PIG ステートメント” (958 ページ)を参照してください。

構成プロパティのサブミット

Hadoop 構成ファイルを指定するのではなく、PROC HADOOP PROPERTIES ステートメントで構成プロパティをサブミットできます。詳細については、“PROPERTIES ステートメント” (960 ページ)を参照してください。

例: HADOOP プロシジャ

例 1: HDFS コマンドのサブミット

要素: SAS_HADOOP_CONFIG_PATH 環境変数
SAS_HADOOP_JAR_PATH 環境変数
PROC HADOOP ステートメント
HDFS ステートメント

他の要素: OPTIONS ステートメント
SET システムオプション

詳細

この PROC HADOOP の例では、HDFS コマンドを Hadoop サーバーにサブミットします。ステートメントで、ディレクトリの作成、ディレクトリの削除、HDFS からローカルの出力先へのファイルのコピーを実行します。

プログラム

```
options set=SAS_HADOOP_CONFIG_PATH="\\sashq\root\u\abcdef\cdh45p1";
options set=SAS_HADOOP_JAR_PATH="\\sashq\root\u\abcdef\cdh45";

proc hadoop username='sasabc' password='sasabc' verbose;

   hdfs mkdir='/user/sasabc/new_directory';

   hdfs delete='/user/sasabc/temp2_directory';

   hdfs copytolocal='/user/sasabc/testdata.txt'
        out='C:\Users\sasabc\Hadoop\testdata.txt' overwrite;
run;
```

プログラムの説明

SAS_HADOOP_CONFIG_PATH 環境変数と SAS_HADOOP_JAR_PATH 環境変数を定義します。 OPTIONS ステートメントには、環境変数を定義するための SET システムオプションが含まれています。環境変数によって Hadoop クラスタ構成ファイルと Hadoop JAR ファイルの場所が設定されるため、SAS セッションで必要なファイルを使用できるようになります。

```
options set=SAS_HADOOP_CONFIG_PATH="\\sashq\root\u\abcdef\cdh45p1";
options set=SAS_HADOOP_JAR_PATH="\\sashq\root\u\abcdef\cdh45";
```

PROC HADOOP ステートメントを実行します。 PROC HADOOP ステートメントで、Hadoop サーバーのユーザー ID とパスワードを識別し、SAS ログに追加メッセージを書き込めるようにする VERBOSE オプションを指定して、Hadoop サーバーへのアクセスを制御します。

```
proc hadoop username='sasabc' password='sasabc' verbose;
```

HDFS パス名を作成します。最初の HDFS ステートメントでは、HDFS パス名を作成する MKDIR=オプションを指定しています。

```
hdfs mkdir='/user/sasabc/new_directory';
```

HDFS ファイルを削除します。2 つ目の HDFS ステートメントでは、HDFS ファイルを削除する DELETE=オプションを指定しています。

```
hdfs delete='/user/sasabc/temp2_directory';
```

HDFS ファイルをコピーします。3 つ目の HDFS ステートメントでは、コピーする HDFS ファイルを指定する COPYTOLOCAL=オプション、ローカルマシン上の出力先を指定する OUT=オプション、および出力先が存在する場合はその場所を指定して上書きする OVERWRITE オプションを指定しています。

```
hdfs copytolocal='/user/sasabc/testdata.txt'
     out='C:\Users\sasabc\Hadoop\testdata.txt' overwrite;
run;
```

例 2: ワイルドカード文字を使用した HDFS コマンドのサブミット

要素: SAS_HADOOP_CONFIG_PATH 環境変数
SAS_HADOOP_JAR_PATH 環境変数
PROC HADOOP ステートメント
HDFS ステートメント
ワイルドカード文字

他の要素: OPTIONS ステートメント
SET システムオプション

詳細

この PROC HADOOP の例では、HDFS コマンドを Hadoop サーバーにサブミットします。ステートメントによって、指定されたファイルの内容が表示され、1 つの HDFS ファイルに対する権限が変更され、指定された HDFS パス名にあるファイルがリストされます。

プログラム

```
options set=SAS_HADOOP_CONFIG_PATH="\\sashq\root\u\abcdef\cdh45p1";
options set=SAS_HADOOP_JAR_PATH="\\sashq\root\u\abcdef\cdh45";

proc hadoop username='sasabc' password='sasabc' verbose;

   hdfs cat='/user/sasabc/*';

   hdfs chmod='/user/sasabc/' permission=rwxr-xr-x;

   hdfs ls='/user/sasabc/*';
run;
```

プログラムの説明

SAS_HADOOP_CONFIG_PATH 環境変数と SAS_HADOOP_JAR_PATH 環境変数を定義します。 OPTIONS ステートメントには、環境変数を定義するための SET システムオプションが含まれています。環境変数によって Hadoop クラスタ構成ファイルと Hadoop JAR ファイルの場所が設定されるため、SAS セッションで必要なファイルを使用できるようになります。

```
options set=SAS_HADOOP_CONFIG_PATH="\\sashq\root\u\abcdef\cdh45p1";
options set=SAS_HADOOP_JAR_PATH="\\sashq\root\u\abcdef\cdh45";
```

PROC HADOOP ステートメントを実行します。 PROC HADOOP ステートメントで、Hadoop サーバーのユーザー ID とパスワードを識別し、SAS ログに追加メッセージを書き込めるようにする VERBOSE オプションを指定して、Hadoop サーバーへのアクセスを制御します。

```
proc hadoop username='sasabc' password='sasabc' verbose;
```

HDFS ファイルの内容を表示します。 最初の HDFS ステートメントで、HDFS ファイルの内容を表示する CAT=オプションを指定します。ワイルドカード文字*は、1 つ以上の文字がマッチされるように指定します。ディレクトリ/user/sasabc/に含まれているすべてのファイルが SAS ログに表示されます。

```
   hdfs cat='/user/sasabc/*';
```

ファイルアクセス権限を変更します。 2 番目の HDFS ステートメントで、指定された HDFS パス名に対するファイルアクセス権限を変更する CHMOD=オプションを指定します。ファイルアクセス権限により、所有者に読み取り、書き込み、および実行権限が付与され、グループメンバに読み取りおよび実行権限が付与され、ユーザーに読み取りおよび実行権限が付与されます。

```
   hdfs chmod='/user/sasabc/' permission=rwxr-xr-x;
```

HDFS パス名にあるファイルをリストします。 3 番目の HDFS ステートメントで、指定された HDFS パス名にあるファイルを SAS ログにリストする LS=オプションを指定します。ワイルドカード文字*は、1 つ以上の文字がマッチされるように指定します。ディレクトリ/user/sasabc/に含まれているすべてのファイルが SAS ログに表示されます。各ファイルの出力はその権限、ユーザー ID、ユーザー ID グループ、ファイルサイズ、作成日、作成時刻、ファイル名で構成されます。

```
   hdfs ls='/user/sasabc/*';
run;
```

例 3: MapReduce プログラムのサブミット

要素: PROC HADOOP ステートメント
MAPREDUCE ステートメント

他の要素: FILENAME ステートメント

詳細

この PROC HADOOP の例では、MapReduce プログラムを Hadoop サーバーにサブミットします。この例では、Hadoop MapReduce のアプリケーションである WordCount を使用します。WordCount はテキスト入力ファイルを読み込み、各行を単語に分割し、単語数をカウントしてから、単語数を出力テキストファイルに書き込みます。

プログラム

```
filename cfg 'C:\Users\sasabc\Hadoop\sample_config.xml';

proc hadoop cfg=cfg username='sasabc' password='sasabc' verbose;

   mapreduce input='/user/sasabc/architectdoc.txt'

      output='/user/sasabc/outputtest'

      jar='C:\Users\sasabc\Hadoop\jars\WordCount.jar'

      outputkey='org.apache.hadoop.io.Text'

      outputvalue='org.apache.hadoop.io.IntWritable'

      reduce='org.apache.hadoop.examples.WordCount$IntSumReducer'

      combine='org.apache.hadoop.examples.WordCount$IntSumReducer'

      map='org.apache.hadoop.examples.WordCount$TokenizerMapper';
run;
```

プログラムの説明

ファイル参照を Hadoop 構成ファイルに割り当てます。 FILENAME ステートメントで、Sample_Config.xml という名前の Hadoop 構成ファイルの物理的な場所にファイル参照 CFG を割り当てます。

```
filename cfg 'C:\Users\sasabc\Hadoop\sample_config.xml';
```

PROC HADOOP ステートメントを実行します。 PROC HADOOP ステートメントで、CFG= オプションを使用して Hadoop 構成ファイルを参照し、Hadoop サーバーのユーザー ID とパスワードを識別し、SAS ログへ書き込まれる追加のメッセージを有効にする VERBOSE オプションを指定して、Hadoop サーバーへのアクセスを制御します。

```
proc hadoop cfg=cfg username='sasabc' password='sasabc' verbose;
```

MapReduce プログラムをサブミットします。 MAPREDUCE ステートメントには、いくつかのオプションが含まれています。INPUT=では、ArchitectDoc.txt という名前の入力 Hadoop ファイルの HDFS パス名とファイル名を指定しています。

```
   mapreduce input='/user/sasabc/architectdoc.txt'
```

HDFS パス名を作成します。 OUTPUT=では、OutputTest という名前のプログラム出力先の HDFS パス名を指定しています。

```
      output='/user/sasabc/outputtest'
```

JAR ファイルを指定します。 JAR=では、WordCount.jar という名前の MapReduce プログラムを含む JAR ファイルを指定しています。

```
      jar='C:\Users\sasabc\Hadoop\jars\WordCount.jar'
```

output key クラスを指定します。 OUTPUTKEY=では、output key クラスの名前を指定しています。org.apache.hadoop.io.Text クラスは、標準の UTF8 エンコードを使用してテキストを保存し、テキストを比較するメソッドを提供します。

```
      outputkey='org.apache.hadoop.io.Text'
```

output value クラスを指定します。 OUTPUTVALUE=では、output value クラスの名前 org.apache.hadoop.io.IntWritable を指定しています。

```
        outputvalue='org.apache.hadoop.io.IntWritable'
```

reducer クラスを指定します。 REDUCE=では、reducer クラスの名前 org.apache.hadoop.examples.WordCount$IntSumReducer を指定しています。

```
        reduce='org.apache.hadoop.examples.WordCount$IntSumReducer'
```

combiner クラスを指定します。 COMBINE=では、combiner クラスの名前 org.apache.hadoop.examples.WordCount$IntSumReducer を指定しています。

```
        combine='org.apache.hadoop.examples.WordCount$IntSumReducer'
```

map クラスを指定します。 MAP=では、map クラスの名前 org.apache.hadoop.examples.WordCount$TokenizerMapper を指定しています。

```
        map='org.apache.hadoop.examples.WordCount$TokenizerMapper';
   run;
```

例 4: Pig 言語コードのサブミット

要素: PROC HADOOP ステートメント
PIG ステートメント

他の要素: FILENAME ステートメント

詳細

この PROC HADOOP の例では、Pig 言語コードを Hadoop クラスタにサブミットします。次に、実行する Pig 言語コードを示します。

```
A = LOAD '/user/sasabc/testdata.txt' USING PigStorage(',')
    AS (customer_number,account_number,tax_id,date_of_birth,status,
        residence_country_code,marital_status,email_address,phone_number,
        annual_income,net_worth_amount,risk_classification);
B = FILTER A BY marital_status == 'Single';
store B into '/user/sasabc/output_customer' USING PigStorage(',');
```

プログラム

```
   filename cfg 'C:\Users\sasabc\hadoop\sample_config.xml';

   filename code 'C:\Users\sasabc\hadoop\sample_pig.txt';

   proc hadoop cfg=cfg username='sasabc' password='sasabc' verbose;

      pig code=code registerjar='C:\Users\sasabc\Hadoop\jars\myudf.jar';
   run;
```

プログラムの説明

ファイル参照を Hadoop 構成ファイルに割り当てます。 最初の FILENAME ステートメントで、Sample_Config.xml という名前の Hadoop 構成ファイルの物理的な場所にファイル参照 CFG を割り当てます。

```
   filename cfg 'C:\Users\sasabc\hadoop\sample_config.xml';
```

ファイル参照を Pig 言語コードに割り当てます。 2 つ目の FILENAME ステートメントでは、Sample_Pig.txt という名前の Pig 言語コードを含むファイル(前述を参照)の物理的な場所にファイル参照 CODE を割り当てています。

```
filename code 'C:\Users\sasabc\hadoop\sample_pig.txt';
```

PROC HADOOP ステートメントを実行します。 PROC HADOOP ステートメントは、CFG=オプションを使用して Hadoop 構成ファイルを参照し、USERNAME=および PASSWORD=オプションを使用して Hadoop サーバーのユーザー ID とパスワードを識別し、SAS ログへの追加のメッセージを有効にする VERBOSE オプションを指定して、Hadoop サーバーへのアクセスを制御します。

```
proc hadoop cfg=cfg username='sasabc' password='sasabc' verbose;
```

PIG ステートメントを実行します。 PIG ステートメントには、Pig 言語コードを含むファイルの物理的な場所に割り当てられる SAS ファイル参照 CODE を指定する CODE=オプションと、実行する Pig スクリプトを含む JAR ファイルを指定する REGISTERJAR=オプションが含まれています。

```
   pig code=code registerjar='C:\Users\sasabc\Hadoop\jars\myudf.jar';
run;
```

例 5: 構成プロパティのサブミット

要素: PROC HADOOP ステートメント
PROPERTIES ステートメント
MAPREDUCE ステートメント

詳細

この PROC HADOOP の例では、MapReduce プログラムを Hadoop サーバーにサブミットします。PROC HADOOP ステートメントで Hadoop 構成ファイルを指定するのではなく、PROPERTIES ステートメントで構成プロパティがサブミットされています。

プログラム

```
proc hadoop username='sasabc' password='sasabc' verbose;

   prop 'mapred.job.tracker'='xxx.us.company.com:8021'
      'fs.default.name'='hdfs://xxx.us.company.com:8020';

   mapreduce jar="&mapreducejar."
      input="&inputfile."
      output="&outdatadir."
      deleteresults;
run;
```

プログラムの説明

PROC HADOOP ステートメントを実行します。 PROC HADOOP ステートメントで、Hadoop サーバーのユーザー ID とパスワードを識別し、SAS ログに追加メッセージを書き込めるようにする VERBOSE オプションを指定して、Hadoop サーバーへのアクセスを制御します。

```
proc hadoop username='sasabc' password='sasabc' verbose;
```

構成プロパティをサブミットします。 PROPERTIES ステートメントでは、Hadoop サーバーに接続するためにサーバーの名前と JobTracker アドレスを指定するプロパティをサブミットしています。

```
   prop 'mapred.job.tracker'='xxx.us.company.com:8021'
      'fs.default.name'='hdfs://xxx.us.company.com:8020';
```

MapReduce プログラムをサブミットします。 MAPREDUCE ステートメントには、いくつかのオプションが含まれています。

```
   mapreduce jar="&mapreducejar."
      input="&inputfile."
      output="&outdatadir."
      deleteresults;
run;
```

31 章

HDMD プロシジャ

概要: HDMD プロシジャ

HDMD プロシジャの動作

PROC HDMD を使用して、HDFS に保存されているファイルのコンテンツについて記述する、XML ベースのメタデータを生成します。このメタデータによって、SAS/ACCESS Interface to Hadoop および SAS ハイパフォーマンスプロシジャで、Hive などの中間メタデータリポジトリを介さずに、Hadoop データを直接読み込むことができます。

ファイルリーダー

PROC HDMD を使用すると、次の形式で表形式の HDFS ファイルを記述できます。

- 固定レコード長(バイナリ)データ
- 区切りテキスト

- XML エンコードされたテキスト

PROC HDMD は、カスタム MapReduce リーダー(Java ベース)をファイルに関連付けることもできます。カスタムリーダーは、PROC HDMD 構文を使用して記述できる区切りテキストまたは固定長バイナリレコードを生成できます。カスタム MapReduce リーダーは、Hadoop 用 SAS ハイパフォーマンスプロシジャでのみ使用できます。現在、SAS/ACCESS Interface to Hadoop では、カスタムリーダーは使用されていません。

概念: HDMD プロシジャ

Hive から独立したデータアクセス

HDFS_METADIR=接続オプションを指定すると、SAS/ACCESS は Hive に接続しません。データには、HDFS を介してアクセスします。SAS は、HDFS ファイルおよびテーブルの XML ベースのメタデータ記述を作成し、使用できます。XML ベースのメタデータは、PROC HDMD を使用して作成できます。PROC HDMD が生成する、XML ベースのメタデータを記述するためのファイルの種類は SASHDMD(product_table.sashdmd など)です。このメタデータには、*SASHDMD ディスクリプタ*という名前もあります。

HiveQL データ定義言語(DDL)と同様、SASHDMD ディスクリプタは、HDFS ファイルまたはテーブルの列について記述しており、ファイルまたはテーブルの場所が含まれます。1 つのファイルについて記述している場合、SASHDMD ディスクリプタには、完全な HDFS ファイルパスが含まれます。

/corp/files/product_codes.dat

テーブルについて記述している場合、SASHDMD ディスクリプタには、HDFS ディレクトリが含まれます。

/corp/tables/purchases/franchise_201

Hive と同様、ディレクトリには、構造が同じファイルが含まれることが想定されています。これらのファイルの列のレイアウトは同じです。

/corp/tables/purchases/franchise_201/income_2012-01-02.dat

/corp/tables/purchases/franchise_201/income_2012-01-03.dat

/corp/tables/purchases/franchise_201/income_2012-01-04.dat

この例では、PROC HDMD は、この収入テーブルデータの SASHDMD ディスクリプタを作成しています。収入テーブルには、製品コード、購入量、価格、合計購入額(税込)の 4 つのカンマ区切りの列が含まれます。

```
libname hdplib hive server=mysrv1
   user=myusr1 pass=mypwd1
   hdfs_metadir="/corp/metadata"
   hdfs_datadir="/corp/tables/purchases/franchise_201";

proc hdmd name=hdplib.meta_income
   file_format=delimited encoding=utf8 sep=','
   data_file='file01.ebcd'
;

column product_code int;
column quantity_purchased int;
```

```
column price real;
column total_purchase_amount real;
run;
```

構文: HDMD プロシジャ

要件 PROC HDMD プロシジャステップでは、COLUMN ステートメントが少なくとも 1 つ必要です。

参照項目: “LIBNAME Statement Specifics for Hadoop” (*SAS/ACCESS for Relational Databases: Reference*)

例:
“例 1: 区切りファイルからの Hadoop メタデータの作成”
“例 2: 区切りファイル用にメタデータを定義”
“例 3: 列ヘッダーが含まれる区切りファイル用にメタデータを定義”
“例 4: バイナリデータからの列の抽出”
“例 5: MVS バイナリデータからの列の抽出”
“例 6: 中国語エンコーディングのバイナリファイルからの列の抽出”
“例 7: XML データからの列の抽出”
“例 8: DESCRIBE オプションの使用”
“例 9: カスタムリーダーの定義とデータファイルの使用”

PROC HDMD <*Hadoop-metadata-options*>;
COLUMN *column-specification(s)*;

PROC HDMD ステートメント

Hive で登録されていないテーブルまたはファイルに対して XML メタデータを生成します。

構文

PROC HDMD <*Hadoop-metadata-options*>;
COLUMN *column-specification(s)* ;

Hadoop メタデータオプション

これらのオプションは、メタデータの生成方法を制御します。

BYTE_ORDER=< LITTLEENDIAN | BIGENDIAN >
数値データを、(PC のように)最下位バイトから保存するか、最上位バイトから保存するかを指定します。

種類	オプション
デフォルト	クライアント
適用対象	BINARY

DATA_FILE='*input-filename*'
Hadoop エンジン LIBNAME=ステートメントの HDFS_DATADIR=オプションを基準にした、入力データファイルの相対パスを指定します。

種類	必須
デフォルト	none
適用対象	BINARY, DELIMITED, XML

ENCODING=*encoding*
入力データファイルまたはフォルダのテキストのエンコーディングを指定します。

種類	オプション
デフォルト	UTF8(値を指定しない場合)
適用対象	BINARY

FILE_FORMAT=*file-format*
Hadoop 用 SAS Embedded Process に渡す入力データの形式を指定します。

file-format には、次のいずれかの値を使用できます。

BINARY
固定長レコードを含むファイルを指定します。このファイルでは、数値データは、マシン固有のバイナリ形式で保存されます。

DELIMITED
テキストベースのデータを含むファイルを指定します。このファイルのフィールドは、特定の区切り文字で区切られています。区切られたレコードの長さはさまざまです。

参照項目 “SEP='*character-separator*'” (974 ページ)

XML
テキストベースのデータファイルを XML 形式で指定します。

種類	必須
別名	FILE_FMT=
デフォルト	none
適用対象	BINARY, DELIMITED, XML

FILE_TYPE='*custom-input-file-type*'
MapReduce フレームワークで使用されているファイルの種類を指定して、データを SAS Embedded Process にロードします。ファイルの種類は、SAS 提供の MapReduce 入力形式クラスの名前にマップされます。特別なファイルの種類に対して SAS が提供する入力形式クラスは、特定の入力リーダーを作成します。たとえば、ファイルの種類が DELIMITED の場合は、MapReduce 入力形式クラス com.sas.access.hadoop.ep.delimited.DelimitedInputFormat にマップされます。

種類	オプション
デフォルト	none
適用対象	BINARY, DELIMITED, XML

要件 カスタムシーケンスのファイルの種類を使用するには、INPUT_CLASS=のほか、FILE_TYPE=CUSTOM_SEQUENCE も指定する必要があります。

操作 入力クラスを指定すると、FILE_TYPE=はデフォルトで CUSTOM に設定されます。

FROM=*Hive-table*
データベース内スコアリングに使用する Hive テーブルの名前を指定します。SAS によって、メタデータファイル *Hive-table*.sashdmd が、ターゲットのメタデータディレクトリに作成されます。

例
```
proc hdmd name=hdfs.&modelnm. from=hive.&modelnm.; run;
```

HEADER_LINES=*n*
区切りファイルの解析中にスキップされた行の数を指定します。

種類 オプション

デフォルト none

適用対象 DELIMITED

INPUT_CLASS='*java.class*'
使用する Java カスタム MapReduce リーダーを実装する完全修飾クラス名を指定します。

種類 オプション

デフォルト none

適用対象 BINARY, DELIMITED, XML

要件 クラスは、Hadoop サーバーのクラスパスに含まれている必要があります。

操作 DBCREATE_EXTERNAL_TABLE= LIBNAME オプションは、このオプションを無視します。

MANAGED
ファイルのメタデータが削除されたときに、そのファイルが削除されることを指定します(たとえば、PROC DELETE を使用)。

種類 オプション

デフォルト デフォルトでは、データファイルは管理されません。つまり、メタデータが削除されたときに、ファイルは削除されません。

適用対象 BINARY, DELIMITED, XML

操作 DBCREATE_EXTERNAL_TABLE= LIBNAME オプションは、このオプションを無視します。

NAME=*libref.filename*
作成するメタデータファイルの名前を指定します。HDFS_METADIR=接続オプションは、メタデータの場所を指定します。

種類 必須

デフォルト none

適用対象 BINARY, DELIMITED, XML

要件 *libref* は、HDFS_METADIR=オプションと HDFS_DATADIR=オプションが指定された、有効な Hadoop エンジン参照名である必要があります。

RECORD_LENGTH=*record-length*
BINARY ファイルのレコード長を指定します。

種類 必須

デフォルト none

適用対象 BINARY

ROW_TAG='*row-tag*'
入力 XML のレコードを特定する XML タグを指定します。

種類 必須

デフォルト none

適用対象 XML

制限事項 このオプションでは、大文字と小文字が区別されます。

SEP='*character-separator*'
区切り入力ファイルのレコードの列を区切る文字を指定します。値を指定する方法を次に示します。

- SEP=^A
- SEP=','
- SEP=TAB
- SEP=^Z
- SEP='09'x
- SEP=32

種類 必須

デフォルト ^A(値を指定しない場合)

範囲 U+0001 から U+007F までの Unicode 範囲の 1 文字のみを指定できます。

適用対象 DELIMITED

制限事項 このオプションの値は、TEXT_QUALIFIER=の文字と同じ文字にすることはできません。また、改行('0a'x)にすることもできません。

TEXT_QUALIFIER='*character-qualifier*'
入力データファイルまたはフォルダのテキスト修飾子を指定します。値を指定する方法を次に示します。

種類	オプション
デフォルト	none
範囲	U+0001 から U+007F までの Unicode 範囲の 1 文字のみを指定できます。
適用対象	DELIMITED
制限事項	このオプションの値は、SEP=の文字と同じ文字にすることはできません。また、改行('0a'x)にすることもできません。
要件	一重引用符で囲まれた二重引用符(' " ')または二重引用符で囲まれた一重引用符(" ' ")を指定する必要があります。

COLUMN ステートメント

1 つ以上の列の仕様を提供します。

要件 次のうち 1 つ以上のステートメントが必要です。

構文

COLUMN <*name*> <*data-type*><*column-options*>;

列の仕様

column-options
列オプションを 1 つ以上指定します。

BYTES=*byte-lenth*
BINARY ファイルについては、レコード内でデータが占めるバイト数を指定します。

種類	必須
デフォルト	none
適用対象	BINARY

CTYPE=*ctype*
BINARY ファイルについては、レコードに保存する実際のバイナリデータの種類を指定します。有効なバイナリデータの種類を次に示します。

- char
- double
- float
- int8
- int16
- int32
- int64

- uint8
- uint16
- uint32
- uint64

種類 オプション

デフォルト none

適用対象 BINARY

ENCODING=*encoding*
BINARY ファイルについては、文字データのエンコーディングを指定します(ファイル全体のエンコーディングと異なる場合)。

種類 オプション

デフォルト none

適用対象 BINARY

FORMAT=*format-specification*
列に関連付けられている出力形式を指定します。

種類 オプション

デフォルト none

適用対象 BINARY, DELIMITED, XML

参照項目 *SAS 出力形式と入力形式: リファレンス*

INFORMAT=*informat-specification*
入力データを読み込むときに使用する入力形式を指定します。

種類 オプション

デフォルト none

適用対象 BINARY, DELIMITED, XML

参照項目 *SAS 出力形式と入力形式: リファレンス*

OFFSET=*bytes*
レコードの列データのオフセットを指定します。

種類 必須

デフォルト none

適用対象 BINARY

TAG='*tag*'
列データを囲む XML 要素を指定します。

種類 必須

デフォルト	none
適用対象	XML

data-type
有効なデータ型を指定します。

- BIGINT
- CHAR(*n*)
- DATE
- DOUBLE
- INT
- REAL
- SMALLINT
- TIME[(prec)]
- TIMESTAMP[(prec)]
- TINYINT
- VARCHAR(*n*)

種類	必須
デフォルト	none
適用対象	BINARY, DELIMITED, XML
操作	Hadoop エンジンは、すべての数値の種類を DOUBLE に変換します。

name
列の名前を指定します。

種類	必須(指定されている場合)
デフォルト	none
適用対象	BINARY, DELIMITED

例: HDMD プロシジャ

例 1: 区切りファイルからの Hadoop メタデータの作成

HDMD プロシジャを使用して、Hadoop ファイルまたはファイルのディレクトリにメタデータを作成できます。この例は、3 つの列が含まれるカンマ区切りのファイルで始まります。

```
Name,Age,Weight John,32,180 Jane,27,112 Tim,54,210
```

取得された各列のデータ型を割り当てて、メタデータを作成する方法を次に示します。

```
libname hdplib hive server=mysrv1_cluster1
user=myusr1 pass=mypwd1
/* connection options */
config='/user/configs/hadoop_cluster1.xml'
hdfs_tempdir='/corp/tempdir'
hdfs_metadir='/corp/metadata'
hdfs_datadir='/corp/tables/purchases;

proc hdmd name=hdplib.people
   file_format=delimited sep=',' encoding=utf8
   data_file='people.csv' header_lines=1;
column name char(8);
column age int;
column weight int;
run;
```

例 2: 区切りファイル用にメタデータを定義

この例では、Hadoop LIBNAME ステートメントを使用して、区切りファイル用にメタデータを定義します。ファイルには次のデータが含まれます。

```
12.34 23f45 "This shows quotes" 4.5 2013-05-05 unquoted 11:12:13 09:05:12.2345 "2013-03-
```

ファイルの解析には、ブランク文字と二重引用符が使用されます。

```
libname hdplib hive server=mysrv1_cluster1
user=myusr1 pass=mypwd1
/* connection options */
config='/user/configs/hadoop_cluster1.xml'
hdfs_tempdir='/corp/tempdir'
hdfs_metadir='/corp/metadata'
hdfs_datadir='/corp/tables/purchases;

proc hdmd name=hdp.foo file_format=delimited
encoding=utf8 sep='20'x text_qualifier='"'
data_dir='franchise_201';
column col1 double file_format=dollar6.2;
column col2 int informat=hex5.;
column 'col3 has a blank'n char(20) file_format=$revers20.;
column col4 real;
column col5 date;
column col5 varchar(42);
column col6 time;
column col7 time(4);
column col8 timestamp(8);
column col9 tinyint;
column col10 smallint;
column col11 bigint;
run;
```

例 3: 列ヘッダーが含まれる区切りファイル用にメタデータを定義

この例は、2 つの列が含まれるカンマ区切りのファイルで始まります。

```
ID,Full Name 1,"Doe, John" 2,"Smith, Sally"
```

HDMD プロシジャの構文を次に示します。

```
proc hdmd name=hdplib.text_qualifer_example
   file_format=delimited sep=',' text_qualifier='"'
   header_lines=1 input_dir="text_qualifier.csv"
;

column id double;
column fullname char(32);

run;
```

結果の PROC PRINT 出力を次に示します。

```
proc print data=hdplib.text_qualifer_example;run;
```

```
The SAS System

Obs    id    fullname
1    1    Doe, John
2    2    Smith, Sally
```

例 4: バイナリデータからの列の抽出

この例では、2 つの列がバイナリファイルから抽出されます。

```
proc hdmd name=hdp.foo file_format=binary
record_length=80
data_file='foo.bin';
column size double ctype=double bytes=8;
column id bigint ctype=int64 offset=8;
column name char(42) ctype=char offset=16 bytes=42;
run;
```

例 5: MVS バイナリデータからの列の抽出

この例は前の例と似ています。ただし、データは MVS から抽出されているため、このデータには、DOUBLE データ型と EBCDIC 文字が含まれます。

```
proc hdmd name=hdp.foo file_format=binary
record_length=80 encoding=ebcdic037
data_file='foo.bin';
column size double ctype=double bytes=8 informat=s370frb8.;
column name char(42) ctype=char offset=8 bytes=42 encoding=ebcdic037;
run;
```

例 6: 中国語エンコーディングのバイナリファイルからの列の抽出

この例では、2 つの列がバイナリ 80 ASCII ファイルの先頭から抽出されます。文字の列は中国語でエンコードされます。

```
proc hdmd name=hdp.foo file_format=binary
record_length=80 encoding=utf8
```

```
data_file='foo.bin';
column size double ctype=double bytes=8;
column name char(42) ctype=char offset=8 bytes=42 encoding='euc-cn';
run;
```

例 7: XML データからの列の抽出

この例では、2 つの列が、次の XML タグを含む XML ファイルから抽出されます。

```
<row><Size>42.5</Size><Name>Julius Caesar</Name></row>
```

HDMD プロシジャの構文を次に示します。

```
proc hdmd name=hdp.foo file_format=xml
record_tag='row'
data_file='foo.xml';
column size double tag='Size';
column name char(42) tag='Name';
run;
```

例 8: DESCRIBE オプションの使用

この例では、HDMD プロシジャを使用して、構文と出力について記述します。

```
proc hdmd name=libname.data
   file_format=delimited sep=','
   input_dir="mydata.csv"
;

column I   double;
column J   double;
column W   double;
run;

proc hdmd name=libname.data describe;
run;
```

結果のログファイルを次に示します。

```
1103  proc hdmd name=libname.data 1104  describe; 1105  run;
```

結果の出力を次に示します。

```
PROC HDMD NAME=HDPLIB.REAL2 FILE_FORMAT=DELIMITED ENCODING=UTF8 SEP=','
BYTE_ORDER=LITTLEENDIAN FILE_TYPE=DELIMITED DATA_DIR='/user/data/csv/data/
mydasta.csv' META_DIR='/user/data/csv/meta'; COLUMN /* 1 */ I DOUBLE OFFSET=0
BYTES=8 CTYPE=DOUBLE; COLUMN /* 2 */ J DOUBLE OFFSET=0 BYTES=8 CTYPE=DOUBLE;
COLUMN /* 3 */ W DOUBLE OFFSET=0 BYTES=8 CTYPE=DOUBLE;
```

例 9: カスタムリーダーの定義とデータファイルの使用

PROC HDMD の実行前に、この OPTION SET ステートメントを入力し、SAS 内で SAS_HADOOP_JAR_PATH 環境変数を定義することで、この例が開始されます。変数は、Hadoop JAR ファイルを含むフォルダを指定します。

```
option set=SAS_HADOOP_JAR_PATH="/users/SAS/JARS:/users/SAS/JARS/cdh420";
```

その後、LIBNAME ステートメントを定義して、Hadoop を指定します。

```
libname hdplib hive
   user=myusr1 pw=mypwd1 server="mysrv1"
   HDFS_TEMPDIR="/user/hdmddemo/temp"
   HDFS_DATADIR="/user/hdmddemo/data"
   HDFS_METADIR="/user/hdmddemo/meta"
   DBCREATE_TABLE_EXTERNAL=NO
   CONFIG="/user/mycfg1.xml";
```

最後に、HDMD ファイルを作成します。

```
proc hdmd name=hdplib.peopleseq
   file_format=delimited sep=tab
   file_type=custom_sequence
   input_class=
      'com.abc.hadoop.ep.inputformat.sequence.PeopleCustomSequenceInputFormat'
   data_file='people.seq'
;

column name varchar(20);
column sex varchar(1);
column age int;
column height double;
column weight double;
run;
```

32 章

HTTP プロシジャ

概要: HTTP プロシジャ

PROC HTTP は、ハイパーテキスト転送プロトコル(HTTP)要求を発行します。SAS 9.4 メンテナンスリリース 3 以降、PROC HTTP では、制約のない一連のメソッドが許可されています。この PROC HTTP は、SAS の以前のリリースでサポートされている標準メソッドのほかに、HTTP/1.1 標準に準拠し、ターゲットのウェブサーバーによって認識される任意のメソッドを受け入れます。また、永続的な接続、Cookie のキャッシング、

EXPECT_100_CONTINUE サポートなどの HTTP/1.1 機能も実装しており、認証の種類を指定できます。入力データは引用符付き文字列で指定するか、ファイル参照名からサブミットできます。カスタム要求ヘッダーは HEADERS ステートメントで名前と値のペアとして指定するか、ファイル参照名から完全にフォーマットされた入力ファイルをサブミットして指定できます。

これをサポートするウェブサーバーについては、プロシジャはデフォルトで接続キャッシングと Cookie キャッシングを使用します。プロシジャ内で、この 2 つの種類のキャッシングの動作を切り替えて、キャッシュをクリアするには、プロシジャの引数を指定します。または、マクロ変数を使用して、Cookie キャッシングをオフにします。

認証の指定機能を使用すると、要求に対して 1 つ以上の認証の種類を指定できます。

メンテナンスリリース 3 より前の SAS 9.4 ソフトウェアリリースでは、標準 HTTP メソッドがこのプロシジャによってサポートされています。入力をサブミットして、ファイル参照名から出力を受け取ります。カスタム要求ヘッダーと応答ヘッダーは、完全にフォーマットされた入力ファイルをサブミットすることでサポートされます。この入力ファイルは、ファイル参照名を介して要求に追加するすべてのカスタムヘッダー行を指定します。ユーザー名とパスワードが指定されている場合は、基本認証が実行されます。

構文: HTTP プロシジャ

制限事項: SAS がロック状態のときは、HTTP プロシジャは使用できません。サーバー管理者は、ロック状態でもこのプロシジャにアクセスできるように再有効化できます。LOCKDOWN ENABLE_AMS=ステートメントを使用して FILENAME、URL アクセスメソッドが再有効化されると、HTTP プロシジャは自動的に再有効化されます。

参照項目: “SAS Processing Restrictions for Servers in a Locked-Down State” (*SAS Language Reference: Concepts*)

SAS Intelligence Platform: Application Server Administration Guide *の“LOCKDOWN Statement”も参照してください。*

PROC HTTP URL="*URL-to-target*" <*option(s)*>;
HEADERS "*HeaderName*"="*HeaderValue*" <"*HeaderName-n*"="*HeaderValue-n*">;

ステートメント	タスク	例
“PROC HTTP ステートメント”	HTTP 要求の発行	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10, Ex. 11, Ex. 12, Ex. 13, Ex. 14, Ex. 15
“HEADERS ステートメント”	HTTP 要求の要求ヘッダーの指定	Ex. 10

PROC HTTP ステートメント

要求を発行するウェブサービスを起動します。

例:

構文

PROC HTTP URL="*URL-to-target*"
<METHOD="*http-method*">
<authentication-type-options>
<caching-options>
<header-options>
<proxy-server-connection-options>
<web-server-authentication-options>
<EXPECT_100_CONTINUE>
<HTTP_TOKENAUTH>
<IN="*string*" | *fileref*>
<OUT=*fileref*>
;

オプション引数の要約

EXPECT_100_CONTINUE
ターゲットサーバー側に要求を受け入れる用意があるかどうかを、クライアントが判断できるようにします。

HTTP_TOKENAUTH
SAS コンテンツサーバーへのアクセスに使用できるワンタイムパスワードをメタデータサーバーから生成します。

IN="*string*" | *fileref*
入力データを指定します。

METHOD="*http-method*"

HTTP メソッドを指定します。

OUT=*fileref-to-response-data*
出力が書き込まれるファイル参照名を指定します。

PROXYPORT="*proxy-port-number*"
HTTP プロキシサーバーのポートを指定します。

Authenticate to Web Server

WEBAUTHDOMAIN="*web-credentials-from-metadata*"
ウェブ認証ドメインを指定します。

WEBPASSWORD="*basic-authentication-password*"
基本認証のためのパスワードを指定します。

WEBUSERNAME="*basic-authentication-name*"
基本認証のためのユーザー名を指定します。

Connect to Proxy Server

PROXYHOST="*proxy-host-name*"
HTTP プロキシサーバーのインターネットホスト名を指定します。

PROXYPASSWORD="*proxy-passwd*"
HTTP プロキシサーバーのパスワードを指定します。

PROXYUSERNAME="*proxy-user-name*"
HTTP プロキシサーバーのユーザー名を指定します。

Disable Shared Connection and Cookie Caching

CLEAR_CACHE
HTTP 要求の実行前に、共有接続キャッシュと Cookie キャッシュの両方がクリアされるように指定します。

CLEAR_CONN_CACHE
HTTP 要求の実行前に、共有接続キャッシュがクリアされるように指定します。

CLEAR_COOKIE_CACHE
HTTP 要求の実行前に、共有 Cookie キャッシュがクリアされるように指定します。

NO_CONN_CACHE
このプロシジャ実行の接続キャッシングを無効にします。

NO_COOKIE_CACHE
キャッシュされた Cookie がこのプロシジャ実行に使用されないように指定します。

Specify Authentication Type

AUTH_ANY
接続サーバーに対する認証に、任意の種類の認証を使用できることを指定します。

AUTH_BASIC
接続サーバーに対する認証に、ユーザー ID 認証が使用されるように指定します。

AUTH_NEGOTIATE
接続サーバーに対する認証に、NTLM、Kerberos またはその他の種類の HTTP 認証が使用されるように指定します。

AUTH_NTLM
接続サーバーに対する認証に、NTLM 認証が使用されるように指定します。

PROXY_AUTH_BASIC
プロキシサーバー経由のユーザー ID 認証が実行されるように指定します。

PROXY_AUTH_NEGOTIATE
プロキシサーバー経由の NTLM、Kerberos またはその他の種類の HTTP 認証が実行されるように指定します。

PROXY_AUTH_NTLM
プロキシサーバー経由の NTLM 認証が実行されるように指定します。

Specify HTTP Headers

CT="*content-type*"
要求ヘッダーの送信する HTTP コンテンツの種類を指定します。

HEADERIN=*fileref-to-request-header-file*
出力形式 *key:value* の要求ヘッダーごとに 1 行を含むテキストファイルへのファイル参照名を指定します。

HEADEROUT_OVERWRITE
リダイレクトの発生時にウェブサーバーによって送信された最後のヘッダーブロックのみが、応答ヘッダーで記録されるようにします。

HEADEROUT=*fileref-to-response-header-file*
応答ヘッダーが出力形式 *key:value* で書き込まれるテキストファイルへのファイル参照名を指定します。

必須引数

URL="*URL-to-target*"
HTTP 要求のエンドポイントを識別する完全修飾 URL パスを指定します。

注 PROC HTTP に渡される URL は、URL エンコードされていると見なされます。確実かつ適切にエンコードするには、ターゲットウェブサーバーに適した接続クラスを使用します。たとえば、Amazon Web Services には AWSV4Signer クラスを使用します。または、RFC3986 の説明に従って、予約された文字をエンコードします。

ヒント SAS 9.4 メンテナンスリリース 3 以降、プロトコルを指定する必要はありません。パス(たとえば、`"httpbin.org"`)を設定した場合、使用される実際の URL は `http://httpbin.org` です。

任意引数

AUTH_ANY
ユーザー名とパスワードを指定した場合は、そのユーザー名とパスワードが、接続サーバーの認証に使用されます。それ以外の場合は、使用可能な他の認証フォームが使用されます。AUTH_ANY を指定した場合の動作は、プロシジャステートメントで AUTH_NEGOTIATE、AUTH_NTLM および AUTH_BASIC を指定する場合と同じです。

デフォルト 認証の種類が指定されていない場合は、これがデフォルトの認証の種類として使用されます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

ヒント HTTP サーバーには複数回アクセスする可能性があるため、データが複数回アップロードされないように、EXPECT_100_CONTINUE を指定します。

AUTH_BASIC
接続サーバーの認証に、ユーザー ID 認証が使用されるように指定します。ユーザー名とパスワードは、WEBUSERNAME 引数と WEBPASSWORD 引数で指定されます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 で有効です。

AUTH_NTLM
接続サーバーに対する認証に、NTLM 認証が使用されるように指定します。現在のユーザー ID に権限がある場合は、認証が確立されます。

制限事項 NTLM は、現在、Windows クライアントでのみ使用できます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 で有効です。

例 “例 12: 認証の種類を指定する要求” (1002 ページ)

AUTH_NEGOTIATE
接続サーバーに対する認証に、NTLM、Kerberos またはその他の種類の HTTP 認証が使用されるように指定します。現在のユーザー ID に権限がある場合は、認証が確立されます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

例 “例 12: 認証の種類を指定する要求” (1002 ページ)

CLEAR_CACHE
HTTP 要求の実行前に、共有接続キャッシュと Cookie キャッシュの両方がクリアされるように指定します。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

CLEAR_CONN_CACHE
HTTP 要求の実行前に、共有接続キャッシュがクリアされるように指定します。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

例 “例 14: 接続キャッシングの無効化” (1003 ページ)

CLEAR_COOKIE_CACHE
HTTP 要求の実行前に、共有 Cookie キャッシュがクリアされるように指定します。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

CT="*content-type*"
HEADERIN=引数とともに使用され、要求ヘッダーで設定される HTTP コンテンツの種類を指定します。コンテンツの種類には、受信側のユーザーエージェントがユーザーにデータを表示するために十分であるように、本体に含まれるデータを記述します。

コンテンツの種類の指定例は次のとおりです。

```
CT="Text/HTML; charset=ISO-8859-4"

CT="Text/plain; charset=us-ascii"

CT="Application/x-www-form-urlencoded"
```

注 SAS 9.4 メンテナンスリリース 3 以降、この引数は、前のバージョンの SAS ソフトウェアとの互換性のためだけにサポートされています。以降、CT=のかわりに、“HEADERS ステートメント” (992 ページ)を使用します。

EXPECT_100_CONTINUE
要求本文が含まれる要求メッセージを送信するクライアントが、ターゲットサーバー側にその要求を受け入れる用意があるかどうかを、要求ヘッダーに基づいて判断できるようにします。大きなデータを送信する場合、不要なデータ転送を行いたくないときに、EXPECT_100_CONTINUE を使用します。詳細については、http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.2.3 を参照してください。

該当要素 PUT で最もよく使用される、IN=引数を指定する HTTP 要求。

操作 この引数は、HEADEROUT=引数とともに使用されます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

例 “例 13: EXPECT_100_CONTINUE を指定する PUT” (1002 ページ)

HEADERIN=*fileref-to-request-header-file*
出力形式 *key:value* の要求ヘッダーごとに 1 行を含むテキストファイルへのファイル参照名を指定します。

z/OS 固有
z/OS 動作環境では、HEADERIN=ファイルは変数レコード長で作成する必要があります。

注 SAS 9.4 メンテナンスリリース 3 以降、この引数は、前のバージョンの SAS ソフトウェアとの互換性のためだけにサポートされています。以降、HEADERIN=のかわりに、“HEADERS ステートメント” (992 ページ)を使用します。

注意 **HEADERS ステートメントと HEADERIN=引数の両方を指定しないでください。**両方のオプションを一緒に指定した場合の動作は定義されていません。

HEADEROUT=*fileref-to-response-header-file*
応答ヘッダーが出力形式 *key:value* で書き込まれるテキストファイルへのファイル参照名を指定します。

例 “例 8: 応答ヘッダーを取得する POST” (999 ページ)

“例 13: EXPECT_100_CONTINUE を指定する PUT” (1002 ページ)

HEADEROUT_OVERWRITE
HEADEROUT=引数とともに使用され、リダイレクトの発生時にウェブサーバーによって送信された最後のヘッダーブロックのみが、応答ヘッダーで記録されるようにします。

例 “例 9: HEADEROUT_OVERWRITE を指定する GET” (999 ページ)

HTTP_TOKENAUTH
SAS コンテンツサーバーへのアクセスに使用できるワンタイムパスワードをメタデータサーバーから生成します。

IN="*string*" | *fileref*
入力データを指定します。SAS 9.4 メンテナンスリリース 3 以降、引用符付き文字列またはファイル参照名で入力データを指定できます。前の SAS リリースでは、ファイル参照名を指定する必要があります。

要件 POST メソッドおよび PUT メソッドが使用されている場合、このオプションは必須です。

例 “例 4: 入力データを文字列として指定” (996 ページ)

METHOD="*http-method*"
HTTP メソッドを指定します。標準メソッドには、HEAD、TRACE、GET、POST、PUT、DELETE が含まれます。SAS 9.4 メンテナンスリリース 3 以降、このメソッドには制約がありません。HTTP/1.1 標準に準拠し、ターゲットのウェブサーバーによって認識される任意のメソッドが受け入れ可能です。詳細については、HTTP/1.1 の仕様(www.w3.org)を参照してください。

デフォルト SAS 9.4 メンテナンスリリース 3 以降、METHOD 引数を省略し、IN 引数を指定しない場合、デフォルトメソッドは GET です。METHOD を省略し、IN 引数を指定した場合、メンテナンスリリース 3 より前の SAS リリースでは、デフォルトメソッドは POST です。

制限事項 SAS 9.4 メンテナンスリリース 3 より前のソフトウェアリリースでは、標準メソッドのみがサポートされます。

例 “例 2: 単純な PUT 要求” (995 ページ)

“例 10: HEADERS ステートメントを使用する GET” (1001 ページ)

“例 11: 非標準のメソッド” (1001 ページ)

NO_CONN_CACHE
この HTTP 要求の接続キャッシングを無効にします。接続は、指定した接続パラメータによって確立されます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

NO_COOKIE_CACHE
キャッシュされた Cookie がこの HTTP 要求に使用されないように指定します。この引数を使用しても、"Cookie"ヘッダーによって Cookie を手動で送信することは可能です。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

OUT=*fileref-to-response-data*
出力が書き込まれる場所を示すファイル参照名を指定します。

例 “例 2: 単純な PUT 要求” (995 ページ)

PROXY_AUTH_BASIC
プロキシサーバー経由のユーザー ID 認証が実行されるように指定します。ユーザー名とパスワードは、PROXYUSERNAME 引数と PROXYPASSWORD 引数で指定されます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

PROXY_AUTH_NTLM
プロキシサーバー経由の NTLM 認証が実行されるように指定します。現在のユーザー ID に権限がある場合は、認証が確立されます。

制限事項 NTLM は、現在、Windows クライアントでのみ使用できます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

PROXY_AUTH_NEGOTIATE
プロキシサーバー経由の NTLM、Kerberos またはその他の種類の HTTP 認証が実行されるように指定します。現在のユーザー ID に権限がある場合は、認証が確立されます。

注 この引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

PROXYHOST="*proxy-host-name*"
HTTP プロキシサーバーのインターネットホスト名を指定します。次のフォームの名前をお勧めします。

```
protocol://host-name:port-number
```

プロトコルが名前に含まれない場合、デフォルトのプロトコルは `http://`です。ポート番号が指定されていない場合、デフォルトのポート番号は `80` です。

例 “例 6: HTTP 要求で指定されたプロキシ” (998 ページ)

PROXYPASSWORD="*proxy-passwd*"
HTTP プロキシサーバーのパスワードを指定します。

ヒント パスワードは、プロキシサーバーで認証情報が必要な場合のみ要求されます。

PROC PWENCODE によって生成されるエンコーディングがサポートされています。

例 “例 7: 認証が必要なプロキシ” (998 ページ)

PROXYPORT="*proxy-port-number*"
HTTP プロキシサーバーのポートを指定します。

注 PROXYPORT=を使用して、プロキシサーバーのポートを指定することはお勧めしません。“PROXYHOST="*proxy-host-name*"” (991 ページ)を参照してください。PROXYPORT=は、前のバージョンの SAS ソフトウェアとの互換性のためにサポートされています。PROXYPORT=が使用されている場合、そのポートは、PROXYHOST=に埋め込まれているすべてのポートに優先します。

PROXYUSERNAME="*proxy-user-name*"
HTTP プロキシサーバーのユーザー名を指定します。

ヒント ユーザー名は、プロキシサーバーで認証情報が必要な場合のみ要求されます。

例 “例 7: 認証が必要なプロキシ” (998 ページ)

WEBAUTHDOMAIN="*web-credentials-from-metadata*"
ウェブ認証ドメインを指定します。指定すると、ユーザー名とパスワードが指定された認証ドメインのメタデータから取得されます。

WEBPASSWORD="*basic-authentication-password*"
基本認証のためのパスワードを指定します。

ヒント PROC PWENCODE によって生成されるエンコーディングがサポートされています。

WEBUSERNAME="*basic-authentication-name*"
基本認証のためのユーザー名を指定します。

HEADERS ステートメント

HTTP 要求の要求ヘッダーを指定します。

サポート: すべての HTTP メソッド

注: このステートメントは、SAS 9.4 メンテナンスリリース 3 で有効です。
SAS 9.4 メンテナンスリリース 3 以降では、PROC HTTP CT=引数および HEADERIN=引数ではなく、HEADERS ステートメントを使用してください。

例: “例 10: HEADERS ステートメントを使用する GET” (1001 ページ)

構文

HEADERS "*HeaderName*"="*HeaderValue*" <"*HeaderName-n*"="*HeaderValue-n*">

必須引数

"HeaderName"="HeaderValue"
ヘッダー名とその値を表す名前と値のペアです。HeaderName は、標準ヘッダー名またはカスタムヘッダー名です。ヘッダーフィールド定義については、HTTP/1.1 仕様(www.w3.org)を参照してください。

注: ヘッダー名ではコロン(:)を指定しないでください。名前と値のペアは、次のフォームに自動的に変換されます。

```
HeaderName : HeaderValue
```

詳細

HEADERS ステートメントを使用すると、プロシジャ要求でヘッダー値を簡単に指定できます。完全にフォーマットされた入力ファイルをファイル参照名で指定する必要はありません。HEADERS ステートメントを使用して、値がメソッドのデフォルト値と異なる場合にアップロードするドキュメントのコンテンツの種類と文字のセットを指定します。

HTTP メソッド	デフォルトのコンテンツの種類
POST	application/x-www-form-urlencoded
PUT	application/octet-stream

ハイパーテキスト転送プロトコルセキュア(HTTPS)の使用

HTTP セキュリティ:TLS とデータ暗号化

Transport Layer Security (TLS)とその前身である Secure Sockets Layer (SSL)は、データを暗号化することによりウェブブラウザとウェブサーバーが保護された接続によって通信できるようにします。ブラウザとサーバーでは、データが送信前に暗号化されます。受信するブラウザまたはサーバーでは、データが処理前に解読されます。

注: TLS についての説明は、その前身のプロトコルである SSL にもすべて当てはまります。

HTTPS プロトコルを使用して PROC HTTP 呼び出しを実行

HTTP(HTTPS)経由の安全な通信が、SSLCALISTLOC システムオプションを使用して構成された System Trusted Root CA バンドルまたは Trusted Root CA バンドルのいずれかによって制御されます。SSLCALISTLOC システムオプションの詳細については、*Encryption in SAS* を参照してください。“例 3: TLS を使用した単純な POST 要求” (996 ページ)PROC HTTP 要求でシステムオプションがどのように指定されているかを示します。

基本認証以外の認証の使用

SAS 9.4 メンテナンスリリース 3 以降、PROC HTTP を使用すると、認証の種類を指定することができます。認証の種類を指定する機能は、要求を成功させるのに必要な認証の種類があらかじめわかっている場合に便利です。適切な種類を指定すると、認証についてプロシジャがネゴシエートする必要がなくなり、プロシジャの実行が最適化されます。たとえば、サーバーがサポートする認証が Kerberos だけであることがわかっている場合は、AUTH_NEGOTIATE 引数を指定することをお勧めします。NTLM 認証しかサポートしていない場合は、AUTH_NTLM を指定します。

認証の種類を指定しない場合、デフォルトの種類(メンテナンスリリース 3 より前の SAS リリースで使用できる認証の種類)は AUTH_ANY です。AUTH_ANY は、AUTH_NTLM、AUTH_NEGOTIATE および AUTH_BASIC を要求で一緒に指定するのと同じです。AUTH_NTLM 認証が最初に試行された後(Windows のみ)、AUTH_NEGOTIATE などが試行されますが、どの認証の種類が使用されるかは、最終的にはサーバーが決定します。接続先のサーバーが NTLM 認証プロトコルまたは Kerberos 認証プロトコルをサポートしている場合、通常は、ユーザー名とパスワードを指定する必要はありません。現在のユーザー ID に権限がある場合は、認証が確立されます。

EXPECT_100_CONTINUE は、複数回サーバーにアクセスする必要がある要求を最適化するためにサポートされます。この引数を指定すると、データが複数回アップロードされるのを防ぐことができます。

AUTH_NEGOTIATE と AUTH_NTLM を使用するときは、接続キャッシングをオフにしないでください。この認証の種類では、接続キャッシングを有効にしておく必要があります。

HTTP_TOKENAUTH を使用すると、PROC HTTP から SAS コンテンツサーバーにアクセスできます。その際、ユーザー名とパスワードを入力する必要はありません。

WEBAUTHDOMAIN もユーザー名とパスワードの代わりに使用されます。ただし、指定されたウェブ認証ドメインに対して、ユーザー名とパスワードを保存するメタデータエントリを設定する必要があります。

回線ロギング

回線ロギングは、ネットワーク上に流れるパケットの情報をそのままロギングします。この情報は、通常はダンプと呼ばれます。回線のダンプを使用すると、サーバーにどのような情報が送信され、サーバーによってどのような情報が送り返されたのかを表示できます。表示されるのは生のデータであるため、回線ダンプはプログラムのデバッグにも有用です。

SAS 9.4 メンテナンスリリース 3 以降、HTTP 固有のメッセージのログへの書き込みには、ロガー APP.TK.HTTPC が使用されます。ロガーによって生成される回線ダンプは、ロガー APP.TK.HTTPC を DEBUG レベル以上に設定することで有効にできます。SAS 9.4 より前のバージョンでは、HTTP 固有のメッセージのログへの書き込みには、ロガー HTTP が使用されます。このロガー HTTP を、DEBUG レベル以上に設定します。DEBUG レベルでは、受信および送信データの最初の 64 バイトがロギングされます。TRACE レベルでは、すべてのデータがログに書き込まれます。TRACE レベルでは、パフォーマンスが大幅に低下することに注意してください。

詳細については、*SAS Logging: Configuration and Programming Reference* の SAS Logging を参照してください。

PROC HTTP でエンコーディングを使用する

応答はセッションエンコーディングにエンコードされません。要求に使用するエンコーディングを提供し、コンテンツの種類を設定する必要があります。

PROC HTTP マクロ変数

SAS 9.4 メンテナンスリリース 3 以降、PROC HTTP によって 3 つの自動マクロ変数が生成され、デフォルトの PROC HTTP を設定したり、その設定を変更したりできます。

PROCHTTP_PROXY="*proxy-server-name-and-port-number*";
PROC HTTP 要求のデフォルトのプロキシサーバーを設定します。一度設定されたら、PROC HTTP 要求で PROXYHOST=引数を指定しない限り、指定されたプロキシサーバーによって、SAS セッションのすべての PROC HTTP 要求に対するプロキシが確立されます。プロシジャの引数で指定された値は、マクロ変数で指定された値を無効にします。マクロ変数とは異なる値で PROXYHOST=引数を指定して、異なるプロキシサーバーを要求に対して使用します。要求に対してプロキシの使用を無効にするには、値を指定せずに PROXYHOST=を指定します。詳細については、“例 5: マクロ変数のプロキシセット” (997 ページ)を参照してください。

PROCHTTP_NOCOOKIES= blank | *integer*;
PROC HTTP 要求で Cookie キャッシングをグローバルに制御できるようにします。マクロ変数を省略するか、値を指定せずにマクロ変数を指定すると、Cookie キャッシングが有効になります(Cookie キャッシングはデフォルトでオンです)。Cookie キ

ャッシングをグローバルに無効にするには、ゼロ以外の値をマクロ変数で指定します。特定の PROC HTTP 要求に対して Cookie キャッシングを無効にするには、PROC HTTP 要求で NO_COOKIE_CACHE 引数だけでなく、CLEAR_COOKIE_CACHE 引数または CLEAR_CACHE 引数も指定します。

マクロ変数は、%LET ステートメントで設定されます。Cookie キャッシングを無効にした場合は、%SYMDEL ステートメントでマクロ変数をシンボルテーブルから削除できます。“例 15: Cookie キャッシングをグローバルに無効化” (1004 ページ)を参照してください。

例: HTTP プロシジャ

例 1: 単純な GET 要求

要素: METHOD=引数
URL=引数
OUT=引数

詳細

この例では、GET 要求を行います。GET は、PROC HTTP で作成できる最も一般的で単純な要求です。

プログラム

```
filename resp TEMP;

proc http
   method="GET"
   url="http://httpbin.org/get"
   out=resp;
run;
```

例 2: 単純な PUT 要求

要素: IN=引数
OUT=引数
HEADEROUT ステートメント

詳細

この例では、ローカルネットワークで、サーバーへの単純な PUT メソッド呼び出しを行います。アップロードするファイルは、IN=引数でファイル参照名によって特定されます。応答ヘッダーおよび出力ヘッダーは、ファイル参照名に書き込まれます。

プログラム

```
filename resp TEMP;
filename headout TEMP;

filename input "fileToUpload.data";

proc http
   method="PUT"
   url="http://httpbin.org/put"
   in=input
   out=resp
   headerout=headout;
run;
```

例 3: TLS を使用した単純な POST 要求

要素: SSLCALISTLOC=システムオプション

詳細

この例では、TLS を使用するサーバーへの POST 要求を行います。SSLCALISTLOC=システムオプションは、信頼できる接続の確立に使用される証明書を構成します。

注: SAS セッションの開始時にロードされる sasv9.cfg ファイルに、この SSLCALISTLOC=システムオプションを追加することもできます。

プログラム

```
options set= SSLCALISTLOC="/home/sas/sascacertsbundle.pem";

filename out "u:\prochttp\Testware\Test_out.txt";

proc http
   url="http://httpbin.org/post"
   method="POST"
   in="text to write out"
   out=out;
run;
```

例 4: 入力データを文字列として指定

要素: IN= "*string*"

注: IN=引数での入力文字列の指定は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

SAS 9.4 メンテナンスリリース 3 以降、PROC HTTP IN=引数は、引用符付き入力文字列またはファイル参照名を受け入れて、入力データをサブミットします。入力を文字列で指定すると、テキストポストおよびフォームベースのポストがさらに送信しやすくなり

ます。この例では、http://httpbin.org/forms/post にあるフォームをサブミットします。応答は、応答ファイルに書き込まれます。

プログラム

```
filename resp TEMP;

proc http
   url="http://httpbin.org/post"
   in='custname=Sas+User&custtel=919-555-5555&custemail=sas.user%40
   sas.com&size=medium&topping=cheese&delivery=12%3A00&comments=Dont+Drop+It'
   out=resp;
run;

data _null_;
   infile resp;
   input;
   put _infile_;
run;
```

これは、Resp ファイルのコンテンツです。

```
{ "args":{}, "data":"", "files":{}, "form":{ "comments":"Dont Drop It",
"custemail":"sas.user@sas.com", "custname":"Sas User",
"custtel":"919-555-5555", "delivery":"12:00", "size":"medium",
"topping":"cheese" }, "headers":{ "Accept":"*/*", "Content-Length":"133",
"Content-Type":"application/x-www-form-urlencoded", "Host":"httpbin.org", "User-
Agent":"SAS/9", }, "json": null, "origin":"149.173.1.80, 104.129.194.85",
"url":"http://httpbin.org/post" }
```

例 5: マクロ変数のプロキシセット

要素: PROCHTTP_PROXYHOST=マクロ変数
IN= "*string*"

注: PROCHTTP_PROXYHOST=マクロ変数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

この例では、外部サーバーへのメソッド呼び出しを行い、プロキシサーバーの使用を要求します。プロキシサーバーのインターネットホスト名とポート番号は、PROCHTTP_PROXYHOST マクロ変数で指定されます。プロキシはマクロ変数で設定されるため、以降 SAS セッションで行われるすべての HTTP 要求で使用できます。POST へのパラメータは、IN=引数で指定されたテキスト文字列から読み込まれます。応答は、ファイル参照名 OUT に書き込まれます。

プログラム

```
%let PROCHTTP_PROXY="http://myproxy:889";

filename out "u:\prochttp\Testware\ProxyTest_out.txt";
```

```
proc http
   url="http://httpbin.org/post"
   method="post"
   in="text to write out"
   out=out;
run;
```

例 6: HTTP 要求で指定されたプロキシ

要素: IN="*string*"
PROXYHOST=引数

注: PROXYHOST=引数は、メンテナンスリリース 3 より前の SAS 9.4 ソフトウェアリリースでプロキシサーバーを指定する唯一の方法です。

詳細

この例では、PROXYHOST 引数を使用してプロキシを指定し外部サーバーに接続します。PROXYHOST 引数で指定された値は、PROCHTTP_PROXYHOST=マクロ変数の値に優先します(設定されている場合)。PROXYHOST 引数が再度指定されない限り、以降の HTTP 要求に対してはグローバルプロキシが使用されます。

この例では、“例 5: マクロ変数のプロキシセット” (997 ページ)と同じ要求を行います。

プログラム

```
%let PROCHTTP_PROXY="http://myproxy:889";

filename out "u:\prochttp\Testware\ProxyTest_out.txt";

proc http
   url="http://httpbin.org/post"
   method="post"
   in="text to write out"
   out=out
   proxyhost="http://myproxy2:776";
run;
```

例 7: 認証が必要なプロキシ

要素: IN="*string*"
PROCHTTP_PROXYHOST=マクロ変数
PROXYPASSWORD=引数
PROXYUSERNAME=引数

注: PROCHTTP_PROXYHOST マクロ変数と、IN=引数でのテキスト文字列のサポートは、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

この例では、POST 要求を行い、認証を必要とするプロキシサーバーを使用します。プロキシサーバーは、PROCHTTP_PROXYHOST=マクロ変数で設定されます。プロキシサーバーに対する認証の証明書は、プロシジャ引数として指定されます。この例では、“例 5: マクロ変数のプロキシセット” (997 ページ)と同じ要求を行います。

注: メンテナンスリリース 3 より前の SAS リリースを使用するときは、PROXYHOST= 引数でプロキシサーバーを指定する必要があります。

プログラム

```
%let PROCHTTP_PROXY="http://myproxy:889";

filename out "u:\prochttp\Testware\ProxyTest_out.txt";

proc http
   url="http://httpbin.org/post"
   method="post"
   in="text to write out"
   out=out
   proxyusername="your-user-name"
   proxypassword="your-password";
run;
```

例 8: 応答ヘッダーを取得する POST

要素: IN="*string*"
HEADEROUT=引数

詳細

この例では、“例 5: マクロ変数のプロキシセット” (997 ページ)と同じ POST 要求を行いますが、headerOut.txt というファイルの応答ヘッダー取得します。

プログラム

```
%let PROCHTTP_PROXY="http://myproxy:889";

filename out "u:\prochttp\Testware\ProxyTest_out.txt";
filename hdrout "u:\prochttp\Testware\headerOut.txt";

 proc http
   url="http://httpbin.org/post"
   method="post"
   in="text to write out"
   out=out
   headerout=hdrout;
run;
```

例 9: HEADEROUT_OVERWRITE を指定する GET

要素: HEADEROUT 引数

HEADEROUT_OVERWRITE 引数

注: HEADEROUT_OVERWRITE 引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

この例は、HEADEROUT_OVERWRITE 引数の効果を示しています。GET 要求は、出力先に到達する前に 2 回リダイレクトされます。HEADEROUT_OVERWRITE では、最後の出力ヘッダーのみが記録されます。

リダイレクト後の標準の HEADEROUT 出力の例

```
filename hdrs "u:\prochttp\Testware\GetHdr_out.txt";
filename out "u:\prochttp\Testware\GetTest_out.txt";

proc http
   url="http://httpbin.org/redirect/2"
   method="GET"
   headerout=hdrs
   out=out;
run;
```

これは GetHdr_out.txt のコンテンツです。

```
HTTP/1.1 302 FOUND Server: nginx Date:Mon, 20 Apr 2015 14:19:52 GMT Content-
Type: text/html; charset=utf-8 Content-Length:247 Connection: keep-alive
Location:/relative-redirect/1 Access-Control-Allow-Origin:* Access-Control-
Allow-Credentials: true HTTP/1.1 302 FOUND Server: nginx Date:Mon, 20 Apr 2015
14:19:53 GMT Content-Type: text/html; charset=utf-8 Content-Length:0
Connection: keep-alive Location:/get Access-Control-Allow-Origin:* Access-
Control-Allow-Credentials: true HTTP/1.1 200 OK Server: nginx Date:Mon, 20 Apr
2015 14:19:53 GMT Content-Type: application/json Content-Length:195 Connection:
keep-alive Access-Control-Allow-Origin:* Access-Control-Allow-Credentials: true
```

HEADEROUT_OVERWRITE での HEADEROUT 要求の例

```
filename hdrs "u:\prochttp\Testware\GetHdr2_out.txt";
filename out "u:\prochttp\Testware\GetTest2_out.txt";

proc http
 url="http://httpbin.org/redirect/2"
method="GET"
headerout=hdrs
out=out
HEADEROUT_OVERWRITE;
run;
```

これは GetHdr2_out.txt のコンテンツです。

```
HTTP/1.1 200 OK Server: nginx Date:Mon, 20 Apr 2015 14:22:48 GMT Content-Type:
application/json Content-Length:195 Connection: keep-alive Access-Control-Allow-
Origin:* Access-Control-Allow-Credentials: true
```

例 10: HEADERS ステートメントを使用する GET

要素: HEADERS ステートメント
GET メソッド

注: HEADERS ステートメントは、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

HEADERS ステートメントを指定する GET メソッド要求の例を示します。SAS 9.4 メンテナンスリリース 3 以降、GET は、IN 引数が指定されていないときのデフォルトのメソッドです。

プログラム

```
filename resp TEMP;

proc http
   url="http://httpbin.org/headers"
   out=resp;
 headers
   "Accept"="application/json";
run;

data _null_;
   infile resp;
   input;
   put _infile_;
run;
```

出力は次のようになります。

```
  "headers":{ "Accept":"*/*,application/json", "Host":"httpbin.org", "User-
Agent":"SAS/9", } }
```

例 11: 非標準のメソッド

要素: METHOD 引数

注: 非標準のメソッドは、SAS 9.4 メンテナンスリリース 3 以降でサポートされています。

詳細

この例では、MKCOL WEBDAV http メソッドをサブミットします。出力は、Resp という名前の一時ファイルに書き込まれます。非標準のメソッドに入力および出力の要件はありません。ターゲットサーバーからデータが返されたときに、有効な OUT を指定していれば、データは OUT ファイル参照名に書き込まれます。Resp に書き込まれる出力を次に示します。

プログラム

```
filename resp TEMP;

proc http
   url="http://hostname/directory/"
   method="MKCOL"
   out=resp;
run;
```

例 12: 認証の種類を指定する要求

要素: AUTH_NEGOTIATE 引数
AUTH_NTLM 引数

注: 認証の種類を指定する機能は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

この例では、PROC HTTP 要求の認証の種類を指定します。2 つの認証の種類が指定され、Negotiate と NTLM 認証のみが許可されていることを示しています。

プログラム

```
proc http
   url="http://securesite.com"
   AUTH_NEGOTIATE
   AUTH_NTLM;
run;
```

例 13: EXPECT_100_CONTINUE を指定する PUT

要素: EXPECT_100_CONTINUE 引数
HEADEROUT=引数

注: EXPECT_100_CONTINUE 引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

この例では、EXPECT_100_CONTINUE ヘッダーを指定します。

プログラム

```
filename resp TEMP;
filename hdrs TEMP;

proc http
   url="http://httpbin.org/put"
   method="PUT"
   in='Some Put Data'
   out=resp
   headerout=hdrs
```

```
   EXPECT_100_CONTINUE;
run;

data _null_;
   infile hdrs;
   input;
   put _infile_;
run;

data _null_;
   infile resp;
   input;
   put _infile_;
run;
```

HDRS の出力は次のようになります。

```
HTTP/1.1 100 Continue HTTP/1.1 200 OK Server: gunicorn/18.0 Date:Mon, 24 Nov
2014 20:18:29 GMT Content-Type: application/json Content-Length:652 Access-
Control-Allow-Origin:* Access-Control-Allow-Credentials: true X-Cache:MISS from
transproxy Via:1.1 vegur, 1.1 transproxy (squid) Connection: keep-alive
```

Resp ファイルの出力は次のようになります。

```
{ "args":{}, "data":"Some Put Data", "files":{}, "form":{}, "headers":
{ "Accept":"*/*", "Content-Length":"13", "Content-Type":"application/octet-
stream", "Host":"httpbin.org", "User-Agent":"SAS/9", "Xxpect":"100-
continue", }, "json": null, "origin":"149.173.1.80, 104.129.194.85",
"url":"http://httpbin.org/put" }
```

例 14: 接続キャッシングの無効化

要素: CLEAR_CONN_CACHE 引数
NO_CONN_CACHE 引数

注: これらの引数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

SAS 9.4 メンテナンスリリース 3 以降、PROC HTTP では以前の HTTP 要求の接続パラメータを使用して、以降の接続を確立します。CLEAR_CONN_CACHE 引数と NO_CONN_CACHE 引数を指定すると、要求を実行する前に接続キャッシュがクリアされ、今後使用するために現在の接続がキャッシングされなくなります。

プログラム

```
proc http
   url="http://httpbin.org/get"
   CLEAR_CONN_CACHE  /* Clear the connection cache */
   NO_CONN_CACHE;    /* Do not use connection cache for this execution */
run;
```

例 15: Cookie キャッシングをグローバルに無効化

要素: PROCHTTP_NOCOOKIES=マクロ変数

注: PROCHTTP_NOCOOKIES=マクロ変数は、SAS 9.4 メンテナンスリリース 3 以降で有効です。

詳細

SAS 9.4 メンテナンスリリース 3 以降、Cookie キャッシングはデフォルトで有効になっています。この例では、Cookie キャッシングをグローバルに無効にします。Cookie キャッシングを無効にするとき、PROCHTTP_NOCOOKIES=マクロ変数で指定する値は重要でありません(値を指定する場合)。

プログラム

```
%let PROCHTTP_NOCOOKIES=1;
```

33 章

IMPORT プロシジャ

概要:IMPORT プロシジャ

IMPORT プロシジャは外部データソースからデータを読み取り、それを SAS データセットに書き出します。Base SAS 9.4 では、JMP ファイルと区切り文字で区切られたファイルをインポートできます。

区切り文字で区切られたファイルでは、区切り文字(ブランク、カンマ、タブなど)がデータ値の列を区切ります。SAS/ACCESS Interface to PC Files のライセンスがある場合、追加の外部データソースに次のようなファイルを含めることができます。Microsoft Access データベースファイル、Microsoft Excel ファイルと Lotus スプレッドシート。詳細については、*SAS/ACCESS Interface to PC Files: Reference* を参照してください。

SAS 9.4 以降では、JMP 7 以降のファイルからデータをインポートでき、JMP 変数が最大 255 文字長になりました。値ラベルを SAS 出力形式カタログにインポートすることもできます。拡張属性が自動的に使用されるようになり、META=ステートメントがサポートされなくなりました。詳細については、“JMP Files” (*SAS/ACCESS Interface to PC Files: Reference*)を参照してください。

IMPORT プロシジャは実行時に入力ファイルを読み込み、データを指定された SAS データセットに書き出します。デフォルトでは、変数名が最初の行に現れることを IMPORT プロシジャは仮定しています。プロシジャでは変数を数えるために最初の 20

行をスキャンし、各変数について適切な入力形式と出力形式の決定を試みます。IMPORT プロシジャのステートメントを使用すると次のことが行えます。

- 変数の種類と長さを決定するために SAS がスキャンする行の数(GUESSINGROWS=)
- SAS がデータの読み込みを開始する行(DATAROW=)
- SAS が変数名を抽出するかどうかの変更(GETNAMES=)

これらのステートメントは、デフォルト値を変更するためにも使用できます。

IMPORT プロシジャは区切り文字で区切られたファイルの読み込み時に、DATA ステップを生成して、データをインポートします。入力データソース固有のオプションやステートメントを指定して、結果を制御します。IMPORT プロシジャは指定の出力 SAS データセットを生成し、インポートに関する情報を SAS ログに書き出します。ログには、IMPORT プロシジャによって生成された DATA ステップコードが表示されます。

プロシジャの実行後にコードを修正する必要がある場合は、RECALL コマンドを発行(または F4 キーを押下)して、生成された DATA ステップをリコールします。この段階で、INFILE ステートメントからオプションを追加または削除でき、またデータの INFORMAT、FORMAT、INPUT ステートメントをカスタマイズできます。

このメソッドを使用して入力形式を修正した場合は、出力形式の同じ変数についても修正を行ってください。入力形式と出力形式のすべての変数は、種類(文字または数値)も同じである必要があります。さらに種類が文字である場合は、データの表示時に切り捨てが起こらないような長さの変数を出力形式に割り当てる必要があります。たとえば、文字変数の長さが 400 文字であるのに、出力形式に`$char50` が割り当てられている場合、データの表示時には最初の 50 文字しか表示されません。

PROC IMPORT コードをリコールするには、RECALL コマンドを再度発行(または F4 キーをもう一度押下)します。

注: デフォルトでは、IMPORT プロシジャは、区切り文字で区切られたファイルを可変レコード長のファイルとして読み込みます。外部ファイルが固定長のファイル形式を使用している場合は、INFILE ステートメントに RECFM=F と LRECL=オプションを含んだ SAS DATA ステップを使用します。INFILE ステートメントの RECFM=オプションの詳細については、*SAS ステートメント: リファレンス*を参照してください。

データのインポートには、**インポートウィザード**または**外部ファイルインターフェイス**(EFI)も使用できます。これらは、外部データソースをインポートするためのステップをガイドします。インポートウィザードを使用して、IMPORT プロシジャステートメントを生成し、これをファイルに保存して後から使用できます。

インポートウィザードまたは EFI を SAS ウィンドウ環境から開くには、**ファイル ⇨ データのインポート**を選択します。インポートウィザードまたは EFI の詳細については、Base SAS オンラインヘルプおよびドキュメントを参照してください。詳細と例については、“Using the SAS Import and Export Wizards” (*SAS/ACCESS Interface to PC Files: Reference*)を参照してください。

ヒント **区切り文字で区切られたファイルのホスト間での共有:**IMPORT プロシジャを使用して区切り文字で区切られたファイルが SAS に読み込まれる場合、各行はホストに固有の行末区切り文字で終了している必要があります。あるホストで作成された区切り文字で区切られたファイルを他のホストで共有する場合、デフォルトの行末の区切り文字が一致しないことがあります。この場合、新しいホストでの行末区切り文字をファイルで指定する必要があります。

- Microsoft Windows の場合:デフォルトの改行区切り文字は、キャリッジリターン/改行(CRLF)です。UNIX または Linux で作成されたファイルを読み込むには、FILENAME ステートメントに TERMSTR=LF オプションを使用します。FILENAME ステートメントの詳細については、*SAS ステートメント: リファレンス*を参照してください。

- UNIX または Linux の場合:デフォルトの行末の区切り文字は改行(LF)です。Windows で作成されたファイルを読み込むには、FILENAME ステートメントに TERMSTR=CRLF オプションを使用します。FILENAME ステートメントの詳細については、*SAS ステートメント: リファレンス*を参照してください。

構文: IMPORT プロシジャ

制限事項: IMPORT プロシジャは次の動作環境で使用できます。

- Microsoft Windows
- UNIX または Linux

ファイルのパス名には、最大で 201 文字を使用できます。

操作: パーセント記号(%)の付いたすべてのデータは、誤って解釈されないように文字データとみなされます。パーセントデータは、誤って解釈される可能性があるため、文字データとみなされます。

参照項目: “ANYDTDTMw. Informat” (*SAS Formats and Informats: Reference*)

PROC IMPORT

DATAFILE="*filename*" | TABLE="*tablename*"

OUT=<*libref.*>*SAS data set* <(*SAS data set option(s)*)>

<DBMS=*identifier*> <REPLACE>;

statements for importing from delimited files

DATAROW=*n*;

DELIMITER=*char* | '*nn*'x;

GETNAMES=YES | NO;

GUESSINGROWS=*n* | MAX;

statements for importing from JMP files

DBENCODING=*12-char SAS encoding-value*;

FMTLIB=<*libref.*>*format-catalog*;

META=*libref.member-data-set*;

ステートメント	タスク	例
“PROC IMPORT ステートメント”	外部データファイルを SAS データセットにインポートします	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DATAROW ステートメント”	区切りテキストファイルの特定行からのデータ読み込みを開始します	Ex. 3
“DBENCODING ステートメント”	JMP ファイルに使用するエンコーディング文字セットを示します	
“DELIMITER ステートメント”	入力ファイルのデータ列を区切る区切り文字を指定します	Ex. 1, Ex. 3, Ex. 4
“FMTLIB ステートメント”	値ラベルを指定の SAS 出力形式カタログに保存します	

ステートメント	タスク	例
“GETNAMES ステートメント”	入力ファイルの最初の行のデータ値から SAS 変数名を生成します	Ex. 1, Ex. 2
“GUESSINGROWS ステートメント”	スキャンする入力ファイルの行数を指定して、変数に適切なデータの種類と長さを決定します	
“META ステートメント”	JMP メタデータ情報を指定の SAS データセットに保存します	

PROC IMPORT ステートメント

外部データファイルを SAS データセットにインポートします

構文

PROC IMPORT

DATAFILE="*filename* " | TABLE="*tablename* "
OUT=<*libref .*>*SAS data set* <(*SAS data set option(s)*)>
<DBMS=*identifier*> <REPLACE>;

オプション引数の要約

DBMS=*identifier*
: インポートするデータの種類を指定します。

REPLACE
: 既存する SAS データセットを上書きします。

SAS data set option(s)
: SAS データセットオプションを指定します。

必須引数

DATAFILE="*filename*" | "*fileref*"
入力 PC ファイル、スプレッドシート、または区切り外部ファイルの完全パスとファイル名またはファイル参照名を指定します。ファイル参照名は、出力ファイルの物理的な場所に関連付けられた SAS 名です。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。FILENAME ステートメントの詳細については、*SAS ステートメント: リファレンス*を参照してください。PC ファイルフォーマットの詳細については、次を参照してください。*SAS/ACCESS Interface to PC Files: Reference*

ファイル参照名を指定した場合、または完全パスとファイル名にパスのバックスラッシュなどの特殊文字、小文字やスペースが含まれていない場合、引用符を省略することができます。

制限事項 IMPORT プロシジャは、DISK 以外の FILENAME ステートメントのデバイスの種類またはアクセスメソッドをサポートしていません。たとえば、IMPORT プロシジャは一時外部ファイルを作成する TEMP デバイスの種類をサポートしていません。

IMPORT プロシジャがデータをインポートできるのは、SAS でそのデータの種類がサポートされている場合のみです。SAS では数値および文字の種類のデータはサポートされていますが、たとえばバイナリオブジェクトはサポートされていません。インポートするデータの種類が SAS でサポートされていない場合、IMPORT プロシジャはデータを正しくインポートできないことがあります。多くの場合、プロシジャは可能な限り最善の方法でデータを変換します。ただし、種類によっては変換できないものもあります。

操作 デフォルトでは、IMPORT プロシジャは、区切り文字で区切られたファイルを可変レコード長のファイルとして読み込みます。外部ファイルが固定長のファイル形式を使用している場合は、INFILE ステートメントに RECFM=F と LRECL=オプションを含んだ SAS DATA ステップを使用します。詳細については、INFILE ステートメントを参照してください。

fileref を使用してインポートする区切り文字で区切られたファイルを指定する場合、FILENAME ステートメントで LRECL=オプションを指定した場合を除き、論理レコード長(LRECL)のデフォルトは 256 になります。IMPORT プロシジャでサポートされている最大 LRECL は、32767 です。

区切り文字で区切られたファイルの場合、最初の 20 行がスキャンされ、変数の属性が決定されます。GUESSINGROWS=ステートメントを使用して、スキャンされる行数を増やすことができます。すべての値は、文字列として読み込まれます。日時出力形式または数値入力形式がデータ値に適用可能な場合、その種類は数値として宣言されます。その他の場合は、その種類は文字のままです。

例 “例 1: 区切り文字で区切られたファイルのインポート” (1015 ページ)

“例 2: ファイル参照名を使用した特殊区切り文字で区切られたファイルのインポート” (1017 ページ)

“例 3: タブ区切り文字で区切られたファイルのインポート” (1019 ページ)

“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1022 ページ)

OUT=<*libref.*>*SAS data set*

出力 SAS データセットを 1 レベルまたは 2 レベルの SAS 名(ライブラリとメンバ名)で識別します。指定した SAS データセットが存在しない場合は、IMPORT プロシジャが作成します。1 レベルの名前を指定すると、デフォルトで IMPORT プロシジャは USER ライブラリ(割り当てられている場合)、または WORK ライブラリ(USER が割り当てられていない場合)のいずれかを使用します。

SAS 9.4 の最初の管理リリースでは、VALIDMEMNAME=EXTEND システムオプションも指定されている場合、SAS データセット名に単一引用符を含めることができます。VALIDMEMNAME=を使用すると、SAS データセット名など、特定の SAS メンバの名前に関するルールが拡張されます。詳細については、*SAS 言語リファレンス: 解説編*の SAS データセット名、表示名、アイテムストア名に関する情報を参照してください。

例 “例 1: 区切り文字で区切られたファイルのインポート” (1015 ページ)

“例 2: ファイル参照名を使用した特殊区切り文字で区切られたファイルのインポート” (1017 ページ)

“例 3: タブ区切り文字で区切られたファイルのインポート” (1019 ページ)

“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1022 ページ)

TABLE="*tablename*"
入力 DBMS テーブルの名前を指定します。名前に特殊文字(疑問符など)、小文字、またはスペースが含まれていない場合は、引用符を省略できます。DBMS テーブル名では大文字と小文字が区別される場合もあります。

要件 DBMS テーブルをインポートするには、SAS/ACCESS Interface to PC Files のライセンスが必要です。

DBMS テーブルのインポート時には、DBMS オプションを指定しなければなりません。

オプション引数

DBMS=*identifier*
インポートするデータの種類を指定します。Base SAS にある JMP ファイル (DBMS=JMP)、または区切り文字で区切られたファイルをインポートできます。JMP ファイルはバージョン 7 以降である必要があり、JMP 変数名の長さは最大で 255 文字までです。SAS では、32,767 以上の変数がある JMP ファイルのインポートがサポートされています。

タブ区切り文字で区切られたファイルをインポートするには、識別子として TAB を指定します。.CSV で終わらないその他の区切り文字で区切られたファイルをインポートするには、識別子として DLM を指定します。拡張子が.CSV のカンマ区切りファイルの場合、DBMS=は任意です。IMPORT プロシジャは、カンマ区切り文字で区切られたファイルの拡張子を.CSV と認識します。

参照項目 表 21.1 (679 ページ) には、このオプションの ID の詳細が記載されています。

“SAS LIBNAME Statement for PC Files: Options” (*SAS/ACCESS Interface to PC Files: Reference*)には、SAS/ACCESS Interface to PC Files を使用する場合のその他の DBMS 値のリストがあります。

例 “例 1: 区切り文字で区切られたファイルのインポート” (1015 ページ)

“例 2: ファイル参照名を使用した特殊区切り文字で区切られたファイルのインポート” (1017 ページ)

“例 3: タブ区切り文字で区切られたファイルのインポート” (1019 ページ)

“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1022 ページ)

REPLACE
既存する SAS データセットを上書きします。REPLACE を省略すると、IMPORT プロシジャは既存するデータセットを上書きしません。

注意:
既存の SAS 世代データセットに出力するために REPLACE オプションを付けて IMPORT プロシジャを使用すると、最新(base)の世代データセットまたは世代データセットのグループが削除されます。

IMPORT プロシジャに REPLACE オプションを付けて使用して、既存の世代データセットに書き込みを行う場合、次のいずれかを実行します。

- 世代数を増加または減少するために GENMAX=データセットオプションを使用すると、既存のすべての世代が削除されて新規に単一のベース世代データセットに置き換えられます。
- GENMAX=データセットオプションを省略すると、既存のすべての世代は削除され同じ名前の新規に単一のデータセットに置き換えられますが、これは世代データセットではありません。

上記のかわりに SAS DATA ステップで REPLACE=データセットオプションを使用して、永久 SAS データセットを置き換え、SAS データセットの世代グループを保持します。詳細については、"Understanding Generation Data Sets" (*SAS Language Reference: Concepts*)を参照してください。

例 "例 1: 区切り文字で区切られたファイルのインポート" (1015 ページ)

"例 2: ファイル参照名を使用した特殊区切り文字で区切られたファイルのインポート" (1017 ページ)

"例 3: タブ区切り文字で区切られたファイルのインポート" (1019 ページ)

"例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート" (1022 ページ)

SAS data set option(s)

SAS データセットオプションを指定します。たとえば、結果の SAS データセットにパスワードを割り当てるには、ALTER=、PW=、READ=または WRITE=データセットオプションを使用できます。指定の条件を満たすデータのみインポートするには、WHERE データセットオプションを使用できます。

制限事項 区切り外部ファイル、カンマ区切り外部ファイル、タブ区切り外部ファイルのインポート時は、データセットオプションを指定できません。

参照項目 *SAS データセットオプション: リファレンス*

DATAROW ステートメント

区切りテキストファイルの指定行番号からのデータの読み込みを開始します。

デフォルト: When GETNAMES=NO:1, when GETNAMES=YES:2

制限事項: GETNAMES=NO の場合、DATAROW は 1 以上である必要があります。
GETNAMES=YES の場合、DATAROW は 2 以上である必要があります。

操作: DATAROW ステートメントは、区切り文字で区切られたファイルにのみ有効です。

参照項目: "GETNAMES ステートメント" (1013 ページ)

例: "例 3: タブ区切り文字で区切られたファイルのインポート" (1019 ページ)

構文

DATAROW=*n*;

必須引数

n
IMPORT プロシジャがデータの読み込みを開始する入力ファイルの行番号を指定します。

DBENCODING ステートメント

JMP ファイルに使用するエンコーディング文字セットを示します。

操作: DBENCODING ステートメントは、DBMS=JMP の場合のみ有効です。

構文

DBENCODING=*12-char SAS encoding-value*;

必須引数

12-char SAS encoding-value
JMP ファイルと使用するエンコーディングを示します。エンコーディングによって、文字セットの各文字が一意の数値表現にマッピングされ、コードポイントのテーブルが構成されます。各文字は、エンコーディングごとに異なる数値表現を使用できます。この値には最大で 12 文字を使用できます。

DELIMITER ステートメント

入力ファイルのデータ列を区切る区切り文字を指定します。

デフォルト: 空欄

操作: DBMS=DLM を指定した場合は、DELIMITER=ステートメントも指定する必要があります。

例: “例 1: 区切り文字で区切られたファイルのインポート” (1015 ページ)

構文

DELIMITER=*char* | '*nn*'x;

必須引数

***char* | 'nn'x**
入力ファイルのデータ列を区切る区切り文字を指定します。区切り文字は、単一文字または 16 進値として指定できます。たとえば、データ列をアンパサンドで区切る場合、DELIMITER='&'を指定します。

DELIMITER=を省略する場合、IMPORT プロシジャは区切り文字をスペースとみなします。

FMTLIB ステートメント

値ラベルを指定の SAS 出力形式カタログに保存します。

操作: FMTLIB ステートメントは、DBMS=JMP の場合のみ有効です。

構文

FMTLIB=*<libref.>format-catalog*;

必須引数

<libref.>format-catalog
値ラベルが保存される出力形式カタログを指定します。

GETNAMES ステートメント

IMPORT プロシジャが入力ファイルの最初の行のデータ値から SAS 変数名を生成するかどうかを指定します。

デフォルト: YES

制限事項: IMPORT プロシジャと同時に使用する場合にのみ有効です。

VALIDVARNAME=ANY が使用された場合、GETNAMES=はデータ値の接頭辞にアンダースコアを使用しません。

操作: GETNAMES ステートメントは、区切り文字で区切られたファイルにのみ有効です。

例: “例 1: 区切り文字で区切られたファイルのインポート” (1015 ページ)

“例 2: ファイル参照名を使用した特殊区切り文字で区切られたファイルのインポート” (1017 ページ)

“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1022 ページ)

構文

GETNAMES=YES | NO;

必須引数

YES | NO

YES は、インポートした区切り文字で区切られたファイルの最初の行にあるデータ値から、IMPORT プロシジャによって SAS 変数名が生成されることを指定します。

NO は、IMPORT プロシジャによって SAS 変数名が VAR1、VAR2 などのように生成されることを指定します。

注: 入力ファイルの最初の行のデータ値が読み込まれ、これにブランクなど SAS 名で無効な特殊文字が含まれている場合、SAS はその文字をアンダースコアに変換します。たとえば、変数名 `Occupancy Code` は、SAS 変数名 `Occupancy_Code` になります。SAS 変数名は数字からは始められないため、GETNAMES=により、その値の最初の文字を置き換えるのではなく、変数名の接頭辞にアンダースコアを追加します。たとえば、`2014.CHANGES` は `_2014_CHANGES` になります。

GUESSINGROWS ステートメント

スキャンするファイルの行数を指定して、変数に適切なデータの種類と長さを決定します。

デフォルト: 20

制限事項: この値は、DATAROW に対して指定された値より大きい必要があります。

操作: GUESSINGROWS ステートメントは、区切り文字で区切られたファイルにのみ有効です。

構文

GUESSINGROWS=*n* | MAX;

必須引数

n

IMPORT プロシジャがスキャンする入力ファイルの行数を示し、変数の適切なデータの種類と長さを決定します。範囲は 1 から 2147483647 (または MAX)です。スキャンデータプロセスは、行 1 から GUESSINGROWS オプションによって指定される番号までをスキャンします。

注: SAS レジストリでデフォルトの行の値を変更できます。SAS コマンドラインで、`regedit` と入力します。**レジストリエディタ**が開いたら、**Products** ⇨ **BASE** ⇨ **EFI** ⇨ **GuessingRows** を選択します。

MAX

2147483647 のかわりに指定されます。最大値を指定すると、パフォーマンスに悪影響を及ぼすことがあります。

META ステートメント

JMP メタデータ情報を指定の SAS データセットに保存します。(廃止予定)

操作: META ステートメントは、DBMS=JMP の場合のみ有効です。

構文

META=*libref.member-data-set*;

必須引数

libref.member-data-set

メタデータ情報が書き込まれる SAS データセットを指定します。

META ステートメントは、JMP ファイルのインポートではサポートされなくなり、無視されます。代わりに、拡張属性が自動的に使用されます。拡張属性による JMP ファイルのインポート時には、拡張属性は新規 SAS データセットに自動的に割り当てられます。

プログラム内に META ステートメントが残っていることもあり、拡張属性に置換されたことを示す NOTE をログに生成します。

例: IMPORT プロシジャ

例 1: 区切り文字で区切られたファイルのインポート

要素: PROC OPTIONS ステートメントオプション
DATAFILE=
DBMS=
GETNAMES=
OUT=
REPLACE

他の要素: DELIMITER=ステートメント
OPTIONS ステートメント
PRINT プロシジャ

詳細

この例では、次の区切り外部ファイルをインポートし、WORK.MYDATA という名前の一時 SAS データセットを作成します。

```
Region&State&Month&Expenses&Revenue Southern&GA&JAN2001&2000&8000 Southern&GA&FEB2001&12
```

プログラム

```
options nodate ps=60 ls=80;

proc import datafile="C:\My Documents\myfiles\delimiter.txt"
            dbms=dlm
            out=mydata
            replace;

     delimiter='&';

     getnames=yes;

run;


proc print data=mydata;
run;
```

プログラムの説明

システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。LINESIZE=オプションで出力行長さを指定し、PAGESIZE=オプションで出力ページの行数を指定します。

```
options nodate ps=60 ls=80;
```

入力ファイルを指定します。 入力ファイルには区切り文字で区切られたファイルを指定します。既存のデータセットが存在する場合、置き換えられます。出力 SAS データセットを識別します。

```
proc import datafile="C:\My Documents\myfiles\delimiter.txt"
            dbms=dlm
            out=mydata
            replace;
```

区切り文字として& (アンパサンド)を指定します。

```
       delimiter='&';
```

データの最初の行から変数名を生成します。

```
       getnames=yes;

run;
```

出力データセットを印刷します。

```
proc print data=mydata;
run;
```

ログの例

SAS ログには、正常なインポートに関する情報が表示されます。この例の場合、次の一部のログにあるように、IMPORT プロシジャが SAS DATA ステップを生成します。

ログ 33.1 *SAS データセットを作成するためにインポートされた外部ファイル*

```
1    options nodate ps=60 ls=80; proc import datafile="C:\My 1  !Documents
\myfiles\delimiter.txt" dbms=dlm out=mydata replace; delimiter='&' 1  !
delimiter='&'; getnames=yes;run; 2     /
************************************************************************ 3     *
PRODUCT:   SAS 4     *   VERSION:   9.3 5     *   CREATOR:   External File
Interface 6     *   DATE:      31JAN11 7     *   DESC:      Generated SAS
Datastep Code 8     *   TEMPLATE SOURCE:(None Specified.)9
************************************************************************/
10       data WORK.MYDATA    ; 11       %let _EFIERR_ = 0; /* set the ERROR
detection macro variable */ 12       infile 'C:\My Documents\myfiles
\delimiter.txt' delimiter = '&' MISSOVER 12 !DSD lrecl=32767 firstobs=2 ;
13          informat Region $8.; 14          informat State $2.; 15
informat Month MONYY7.; 16          informat Expenses best32.; 17
informat Revenue best32.; 18          format Region $8.; 19          format
State $2.; 20          format Month MONYY7.; 21          format Expenses
best12.; 22          format Revenue best12.; 23       input 24
Region $ 25                  State $ 26                  Month
27                  Expenses 28                  Revenue 29       ; 30
if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection 30 ! macro
variable */ 31       run; NOTE:The infile 'C:\My Documents\myfiles
\delimiter.txt' is:Filename=C:\My Documents\myfiles\delimiter.txt,
RECFM=V,LRECL=32767,File Size (bytes)=254, Last Modified=31Jan2011:11:44:29,
Create Time=31Jan2011:11:44:29 NOTE:7 records were read from the infile 'C:\My
Documents\myfiles\delimiter.txt'.The minimum record length was 29.The maximum
record length was 30.NOTE:The data set WORK.MYDATA has 7 observations and 5
variables.NOTE:DATA statement used (Total process time): real time
0.06 seconds cpu time            0.03 seconds 7 rows created in WORK.MYDATA from
C:\My Documents\myfiles\delimiter.txt.NOTE:WORK.MYDATA data set was successfully
created.
```

出力例

アウトプット 33.1 *データセット Work.MyData*

The SAS System

Obs	Region	State	Month	Expenses	Revenue
1	Southern	GA	JAN2001	2000	8000
2	Southern	GA	FEB2001	1200	6000
3	Southern	FL	FEB2001	8500	11000
4	Northern	NY	FEB2001	3000	4000
5	Northern	NY	MAR2001	6000	5000
6	Southern	FL	MAR2001	9800	13500
7	Northern	MA	MAR2001	1500	1000

例 2: ファイル参照名を使用した特殊区切り文字で区切られたファイルのインポート

要素: PROC OPTIONS ステートメントオプション
DATAFILE=
DBMS=
GETNAMES=
OUT=
REPLACE

他の要素: FILENAME ステートメント
PRINT プロシジャ

詳細

この例では、次のスペース区切り文字で区切られたファイルをインポートし、Work.States という名前の一時 SAS データセットを作成します。

```
Region State Capital Bird South Georgia Atlanta 'Brown Thrasher' South 'North Carolina'
```

プログラム

```
filename stdata 'c:\temp\state_data.txt' lrecl=100;


proc import datafile=stdata
        dbms=dlm
        out=states
        replace;

     delimiter=' ';
     getnames=yes;
run;
```

```
proc print data=states;
run;
```

プログラムの説明

ファイル名を指定します。

```
filename stdata 'c:\temp\state_data.txt' lrecl=100;
```

入力ファイルを指定します。 入力ファイルには区切り文字で区切られたファイルを指定します。既存のデータセットが存在する場合、置き換えられます。出力 SAS データセットを識別します。

```
proc import datafile=stdata
        dbms=dlm
        out=states
        replace;
```

DELIMITER ステートメントに空白の値を指定します。GETNAMES ステートメントを使用して、データの最初の行から変数名を生成します。

```
     delimiter=' ';
     getnames=yes;
run;
```

データセットを出力します。

```
proc print data=states;
run;
```

ログの例

SAS ログには、正常なインポートに関する情報が表示されます。この例の場合、次の一部のログにあるように、IMPORT プロシジャが SAS DATA ステップを生成します。

ログ 33.2 *ファイル参照名を使用した特殊区切り文字で区切られたファイルのインポート*

```
324  filename stdata 'c:\myfiles\state_data.txt' lrecl=100; 325 326  proc import
datafile=stdata 327              dbms=dlm 328              out=states
329              replace; 330       delimiter=' '; 331       getnames=yes; 332
333  run; 334   /
************************************************************************ 335   *
PRODUCT:   SAS 336   *   VERSION:   9.4 337   *   CREATOR:   External File
Interface 338   *   DATE:      18APR14 339   *   DESC:      Generated SAS
Datastep Code 340   *   TEMPLATE SOURCE:(None Specified.)341
************************************************************************/
342      data WORK.STATES    ; 343      %let _EFIERR_ = 0; /* set the ERROR
detection macro variable */ 344      infile STDATA delimiter = ' ' MISSOVER DSD
lrecl=32767 firstobs=2 ; 345         informat Region $7.; 346         informat
State $16.; 347         informat Capital $11.; 348         informat Bird $20.;
349         format Region $7.; 350         format State $16.; 351         format
Capital $11.; 352         format Bird $20.; 353      input 354
Region $ 355                 State $ 356                 Capital
$ 357                  Bird $ 358      ; 359      if _ERROR_ then call
symputx('_EFIERR_',1);  /* set ERROR detection 359! macro variable */ 360
run; NOTE:The infile STDATA is:Filename=c:\myfiles\state_data.txt,
RECFM=V,LRECL=32767,File Size (bytes)=225, Last Modified=15Apr2014:11:27:14,
Create Time=15Apr2014:11:25:38 NOTE:5 records were read from the infile
STDATA.The minimum record length was 32.The maximum record length was
44.NOTE:The data set WORK.STATES has 5 observations and 4 variables.NOTE:DATA
statement used (Total process time): real time           0.03 seconds cpu
time            0.03 seconds 5 rows created in WORK.STATES from
STDATA.NOTE:WORK.STATES data set was successfully created.
```

出力例

アウトプット 33.2 *Work.States データセット*

The SAS System

Obs	Region	State	Capital	Bird
1	South	Georgia	Atlanta	Brown Thrasher
2	South	North Carolina	Raleigh	Cardinal
3	North	Connecticut	Hartford	Robin
4	West	Washington	Olympia	American Goldfinch
5	Midwest	Illinois	Springfield	Cardinal

例 3: タブ区切り文字で区切られたファイルのインポート

要素: PROC OPTIONS ステートメントオプション
- DATAFILE=
- DATAROW=
- DBMS=
- OUT=
- REPLACE

他の要素: DELIMITER=ステートメント

PRINT プロシジャ

詳細

この例では、次のタブ区切り文字で区切られたファイルをインポートし、Work.Class という名前の一時 SAS データセットを作成します。

入力データ 33.1 入力

```
Name     Gender   Age Joyce    F        11 Thomas   M        11 Jane     F        12 Lou
```

プログラム

```
proc import datafile='c:\temp\tab.txt'
            out=class
            dbms=dlm
            replace;

       datarow=5;

     delimiter='09'x;
run;

proc print data=class;
run;
```

プログラムの説明

入力ファイルを指定します。 GETNAMES=オプションはデフォルトの'yes'になります。入力ファイルには区切り文字で区切られたファイルを指定します。既存のデータセットが存在する場合、置き換えられます。出力データセットを指定します。

```
proc import datafile='c:\temp\tab.txt'
            out=class
            dbms=dlm
            replace;
```

最初の行の読み込みは、DATAROW=オプションの仕様により行 5 になります。

```
       datarow=5;
```

区切り文字を指定します。 ASCII プラットフォームではタブの 16 進表現は **'09'x** になります。EBCDIC プラットフォームではタブの 16 進表現は **'05'x** になります。

```
     delimiter='09'x;
run;
```

出力データセットを印刷します。

```
proc print data=class;
run;
```

ログの例

SAS ログには、正常なインポートに関する情報が表示されます。この例の場合、次の一部のログにあるように、IMPORT プロシジャが SAS DATA ステップを生成します。

ログ 33.3 タブ区切り文字で区切られたファイルのインポート

```
374  proc import datafile='c:\myfiles\tab.txt' 375          out=class
376          dbms=dlm 377          replace; 378      datarow=5; 379
delimiter='09'x; 380  run; 381   /
****************************************************************** 382   *
PRODUCT:   SAS 383   *   VERSION:   9.4 384   *   CREATOR:   External File
Interface 385   *   DATE:      18APR14 386   *   DESC:      Generated SAS
Datastep Code 387   *   TEMPLATE SOURCE:(None Specified.)388
******************************************************************/
389      data WORK.CLASS    ; 390      %let _EFIERR_ = 0; /* set the ERROR
detection macro variable */ 391      infile 'c:\myfiles\tab.txt' delimiter='09'x
MISSOVER DSD lrecl=32767 391! firstobs=5 ; 392         informat
Name_____Gender___Age $20.; 393         format Name_____Gender___Age $20.;
394      input 395                  Name_____Gender___Age $ 396      ; 397
if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection 397! macro
variable */ 398      run; NOTE:The infile 'c:\myfiles\tab.txt' is:Filename=c:
\myfiles\tab.txt, RECFM=V,LRECL=32767,File Size (bytes)=439, Last
Modified=18Apr2014:10:17:54, Create Time=18Apr2014:10:17:54 NOTE:16 records were
read from the infile 'c:\myfiles\tab.txt'.The minimum record length was 20.The
maximum record length was 20.NOTE:The data set WORK.CLASS has 16 observations
and 1 variables.NOTE:DATA statement used (Total process time): real
time           0.01 seconds cpu time            0.01 seconds 16 rows created in
WORK.CLASS from c:\myfiles\tab.txt.NOTE:WORK.CLASS data set was successfully
created.NOTE:The data set WORK.CLASS has 16 observations and 1
variables.NOTE:PROCEDURE IMPORT used (Total process time): real time
0.09 seconds cpu time            0.07 seconds 399
```

出力例

アウトプット 33.3 *Work.Class データセット*

The SAS System

Obs	Name	Gender	Age
1	Louise	F	12
2	James	M	12
3	John	M	12
4	Robert	M	12
5	Alice	F	13
6	Barbara	F	13
7	Jeffery	M	13
8	Carol	F	14
9	Judy	F	14
10	Alfred	M	14
11	Henry	M	14
12	Jenet	F	15
13	Mary	F	15
14	Ronald	M	15
15	William	M	15
16	Philip	M	16

例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート

要素: PROC OPTIONS ステートメントオプション
DATAFILE=
DBMS=
GETNAMES=
OUT=
REPLACE

他の要素: PRINT プロシジャ

詳細

この例では、次のカンマ区切り文字で区切られたファイルをインポートし、Work.Shoes という名前の一時 SAS データセットを作成します。

```
"Africa","Boot","Addis Ababa","12","$29,761","$191,821","$769" "Asia","Boot","Bangkok","
```

プログラム

```
proc import datafile="C:\temp\test.csv"
        out=shoes
        dbms=csv
        replace;


     getnames=no;
run;

proc print data=work.shoes;
run;
```

プログラムの説明

入力データファイルを指定します。 既存のデータセットが存在する場合、置き換えられます。出力データセットを指定します。

```
proc import datafile="C:\temp\test.csv"
        out=shoes
        dbms=csv
        replace;
```

GETNAMES=オプションを'no'に設定すると、レコード 1 にある変数名は使用されなくなります。

```
     getnames=no;
run;
```

データセットを出力します。

```
proc print data=work.shoes;
run;
```

ログの例

SAS ログには、正常なインポートに関する情報が表示されます。この例の場合、次の一部のログにあるように、IMPORT プロシジャが SAS DATA ステップを生成します。

ログ 33.4 カンマ区切り文字で区切られたファイルのインポート

```
457  proc import datafile="C:\myfiles\test.csv" 458       dbms=csv 459
out=shoes 460       replace; 461       getnames=no; 462  run; 463   /
****************************************************************** 464   *
PRODUCT:   SAS 465   *   VERSION:   9.4 466   *   CREATOR:   External File
Interface 467   *   DATE:      18APR14 468   *   DESC:      Generated SAS
Datastep Code 469   *   TEMPLATE SOURCE:(None Specified.)470
******************************************************************/
471      data WORK.SHOES    ; 472      %let _EFIERR_ = 0; /* set the ERROR
detection macro variable */ 473      infile 'C:\myfiles\test.csv' delimiter =
',' MISSOVER DSD lrecl=32767 ; 474         informat VAR1 $27.; 475
informat VAR2 $6.; 476         informat VAR3 $13.; 477         informat VAR4
$4.; 478         informat VAR5 $10.; 479         informat VAR6 $10.; 480
informat VAR7 $8.; 481         format VAR1 $27.; 482         format VAR2 $6.;
483         format VAR3 $13.; 484         format VAR4 $4.; 485         format
VAR5 $10.; 486         format VAR6 $10.; 487         format VAR7 $8.; 488
input 489                 VAR1 $ 490                 VAR2
$ 491                 VAR3 $ 492                 VAR4 $ 493
VAR5 $ 494                 VAR6 $ 495                 VAR7 $ 496      ;
497      if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection
497! macro variable */ 498      run; NOTE:The infile 'C:\myfiles\test.csv'
is:Filename=C:\myfiles\test.csv, RECFM=V,LRECL=32767,File Size (bytes)=657, Last
Modified=18Apr2014:10:34:47, Create Time=18Apr2014:10:33:23 NOTE:10 records were
read from the infile 'C:\myfiles\test.csv'.The minimum record length was 51.The
maximum record length was 81.NOTE:The data set WORK.SHOES has 10 observations
and 7 variables.
```

出力例

アウトプット 33.4 *Work.Shoes データセット*

The SAS System

Obs	VAR1	VAR2	VAR3	VAR4	VAR5	VAR6	VAR7
1	Africa	Boot	Addis Ababa	12	$29,761	$191,821	$769
2	Asia	Boot	Bangkok	1	$1,996	$9,576	$80
3	Canada	Boot	Calgary	8	$17,720	$63,280	$472
4	Central America/Caribbean	Boot	Kingston	33	$102,372	$393,376	$4,454
5	Eastern Europe	Boot	Budapest	22	$74,102	$317,515	$3,341
6	Middle East	Boot	Al-Khobar	10	$15,062	$44,658	$765
7	Pacific	Boot	Auckland	12	$20,141	$97,919	$962
8	South America	Boot	Bogota	19	$15,312	$35,805	$1,229
9	United States	Boot	Chicago	16	$82,483	$305,061	$3,735
10	Western Europe	Boot	Copenhagen	2	$1,663	$4,657	$129

34 章

JAVAINFO プロシジャ

概要: JAVAINFO プロシジャ

JAVAINFO プロシジャは、SAS が使用している Java 環境に関する診断情報をユーザーに通知します。診断情報は、SAS Java 環境が正しく設定されていることを確認するために使用でき、SAS テクニカルサポートに問題を報告するときに役立ちます。また、PROC JAVAINFO は Java を使用してその診断を報告するため、SAS Java 環境が正しく動作していることを検証するために頻繁に使用されます。

構文: JAVAINFO プロシジャ

PROC JAVAINFO <*option(s)*>;

ステートメント	タスク
“PROC JAVAINFO ステートメント”	SAS Java 環境に関する診断情報を表示します

PROC JAVAINFO ステートメント

SAS Java 環境に関する診断情報を表示します。

操作: SAS サーバーがロック状態のときは、JAVAINFO プロシジャは実行されません。詳細については、“SAS Processing Restrictions for Servers in a Locked-Down State” (*SAS Language Reference: Concepts*)を参照してください。

構文

PROC JAVAINFO <*option(s)*>;

オプション引数

ALL
SAS Java 環境に関する最新情報を示します。

CLASSPATHS
Java が使用しているクラスパスに関する情報を示します。

HELP
JAVAINFO プロシジャの使用アシスタンスを示します。

JREOPTIONS
JREOPTIONS 構成オプションの指定時に設定される Java プロパティを指定します。

- JREOPTIONS は PROC JAVAINFO での使用時に、Java の起動時に設定される JREOPTIONS Java プロパティを指定します。
- JREOPTIONS は PROC OPTIONS での使用時に、SAS の起動時に構成ファイルにある Java オプションを指定します。

注: SAS.cfg はインストール時に指定される構成ファイルですが、その他の構成ファイルが指定可能です。

OS
SAS が実行中のオペレーティングシステムに関する情報を指定します。

version
SAS が使用している Java ランタイム環境(JRE)を指定します。

35 章

JSON プロシジャ

概要: JSON プロシジャ

JSON プロシジャは、データを SAS データセットから読み込み、外部データソースに書き込みます [1] 表現。複数のオプションを使用してエクスポートしたデータを制御できます。各オプションによってコンテンツは削除され、形式に影響が及ぼされます。データを SAS データセットからエクスポートするほか、PROC JSON はステートメントを提供し、このステートメントを使用して追加データを外部ファイルに書き込んだり、JSON コンテナを制御したりできます。

[1] Java Script Object Notation (JSON)は、テキストベースのオープンスタンダードデータ形式で、人間が読み取り可能なデータを交換できるように設計されています。JSON は JavaScript プログラミング言語のサブセットを基本として、データオブジェクトの記述には JavaScript 構文を使用します。

概念:JSON プロシジャ

JSON 出力ファイルコンテナ

JSON 出力ファイルの構造

JSON 出力ファイルは、最低でも 1 つのコンテナから構成され、これが最上位コンテナと呼ばれます。すべてのデータ、メタデータ、および追加コンテナは最上位コンテナ内に格納されます。最上位コンテナの種類は、PROC JSON ステートメントの直前のステートメントおよびそのステートメントで有効化されるオプションによって決定されます。

JSON コンテナ

JSON 出力は、データ構造コンテナの 2 つの種類から構成されます。

JSON オブジェクトコンテナ({ })
: 左中括弧({)で始まり、右中括弧(})で終わります。オブジェクトコンテナは名前-値ペアを収集し、名前と値のペアとして書き込まれます。値には、サポートされる JSON データ型、オブジェクト、または配列のいずれかを指定できます。各名前の後にはコロン、値の順に続きます。名前-値のペアはカンマで区切ります。

JSON 配列コンテナ([ ])
: 左大括弧({)で始まり、右大括弧(})で終わります。配列コンテナは、名前を指定せずに値のリストとして書き込まれる値のリストを収集します。値には、サポートされる JSON データ型、オブジェクト、または配列のいずれかを指定できます。値はカンマで区切ります。

オブジェクトコンテナの作成

次のステートメントによって、オブジェクトコンテナが作成されます。

- デフォルトオプションを有効にした状態の EXPORT ステートメントは、オブジェクトコンテナを最上位コンテナとして黙示的に開きます。黙示的に開くと、最上位コンテナが自動的に閉じられます。次の EXPORT ステートメントでは、デフォルトの SASTAGS オプションはオブジェクトとして最上位コンテナになり、エクスポートされた SAS データセットオブザベーションが配列コンテナで収集され、デフォルトの KEY オプションは、名前-値ペアとして子オブジェクトコンテナ内で収集される各オブザベーションのデータになります。

```
proc json out="example.json";
   export sashelp.class (where=(age=11));
run;
```

```
{"SASJSONExport":"1.0","SASTableData+CLASS":[{"Name":"Joyce","Sex":"F","Age":11,"Height"
```

- WRITE VALUES ステートメントは、PROC JSON ステートメントの後の最初のステートメントである場合、オブジェクトコンテナを最上位コンテナとして黙示的に開きます。黙示的に開くと、最上位コンテナが自動的に閉じられます。次のステートメントによって、名前-値ペアを収集するオブジェクトコンテナが作成されます。

```
proc json out="example.json";
   write values "container" "object";
   write values "created" "implicitly";
```

```
run;
```

```
{"container":"object","created":"implicitly"}
```

- WRITE OPEN OBJECT ステートメントは、オブジェクトコンテナを明示的に開きます。このコンテナは、WRITE CLOSE ステートメントを使用して明示的に閉じる必要があります。次のステートメントによって、名前-値ペアを収集するオブジェクトコンテナが作成されます。

```
proc json out="example.json"
   write open object;
   write values "container" "object";
   write values "created" "explicitly";
   write close;
run;
```

```
{"container":"object","created":"explicitly"}
```

配列コンテナの作成

次のステートメントによって、配列コンテナが作成されます。

- NOSASTAGS オプションを有効にした状態の EXPORT ステートメントは、配列コンテナを最上位コンテナとして黙示的に開きます。黙示的に開くと、最上位コンテナが自動的に閉じられます。次の EXPORT ステートメントでは、NOSASTAGS オプションは配列として最上位コンテナになり、NOKEYS オプションは値のリストとして配列コンテナで収集される各オブザベーションのデータになります。

```
proc json out="example.json";
   export sashelp.class (where=(age=11)) / nosastags nokeys;
run;
```

```
[["Joyce","F",11,51.3,50.5],["Thomas","M",11,57.5,85]]
```

- WRITE OPEN ARRAY ステートメントは、配列コンテナを明示的に開きます。このコンテナは、WRITE CLOSE ステートメントを使用して明示的に閉じる必要があります。次のステートメントによって、値のリストを収集する配列コンテナが作成されます。

```
proc json out="example.out";
   write open array;
   write values "container" "array";
   write values "created" "explictly";
   write close;
run;
```

```
["container","array","created","explictly"]
```

コンテナの入れ子構造

最上位コンテナには、任意の数のコンテナを含めることができます。同様に、コンテナは任意の深さまでコンテナを入れ子構造にできます。コンテナを入れ子構造にするときには、現在のコンテナのデータ構造要件を順守するよう注意してください。

- オブジェクトには名前-値のペアのリストが必要になり、ここでは値自体をオブジェクトまたは配列に指定できます。
- 配列には名前-値のペアの構造要件が含まれないため、値、オブジェクト、または配列のリストのみです。

- WRITE VALUES ステートメントまたはオブジェクトコンテナのステートメントは、値の数にする必要があり、名前-値ペアの名前部分は文字列にする必要があります。

この例では、黙示的に開いたオブジェクト内で配列は入れ子構造になります。

```
proc json out="example.json";
   write values "level" 1;              /* implicit open of object         */
   write values "container" "object";   /* write data to object             */
   write values "created" "implicitly"; /* write data to object             */
   write values "nest";                 /* required string to start object */
   write open array;                    /* explicit open of array          */
   write values "level" 2;              /* write data to array             */
   write values "container" "array";    /* write data to array             */
   write values "created" "explicitly"; /* write data to array             */
   write close;                         /* close explicit open             */
run;
```

```
{"level":1,"container":"object","created":"implicitly","nest":["level",2, "container","a
```

この例では、エクスポートしたデータは 3 つの配列コンテナ内で入れ子構造になります。

```
proc json out="example.json";
   write open array;                                 /* explicit open of array */
   write values "level" 1;                           /* write data to array    */
   write values "container" "array";                 /* write data to array    */
   write values "created" "explicitly";              /* write data to array    */
   write open array;                                 /* explicit open of array */
   write values "level" 2;                           /* write data to array    */
   write values "container" "array";                 /* write data to array    */
   write values "created" "explicitly";              /* write data to array    */
   export sashelp.class (where=(age=11)) / nokeys;  /* export SAS data        */
   write close;                                      /* close explicit open    */
   write close;                                      /* close explicit open    */
run;
```

```
["level",1,"container","array","created","explicitly",["level",2,"container", "array","c
```

欠損値

SAS 内の欠損値は、特定のオブザベーションまたは変数のデータを含まない変数の値タイプです。デフォルトでは、SAS は 1 つの期間として欠損数値を表し、空白スペースとして欠損文字値を表します。

また、JSON にはヌル値という欠損値の概念も含まれ、これは情報が存在しないことを示す特別な値です。

PROC JSON は、次が TRUE のときに JSON ヌル値を JSON 出力ファイルに書き込みます。

- SAS データセットの数値変数には欠損値が含まれます
- WRITE VALUES ステートメントは NULL キーワードを値として書き込みます

デフォルトでは、SAS データセット文字変数に欠損値が含まれるとき、SAS は空の文字列("")を JSON 出力ファイルに書き込みます。ただし、NOTRIMBLANKS オプションを指定する場合、ブランクの文字列全体が JSON 出力ファイルに書き込まれます。

入力文字列のスキャン

JSON 文字列は、二重引用符で囲まれたゼロ以上の文字シーケンスです。JSON 文字列には任意の Unicode 文字を指定できますが、二重引用符(")、バックスラッシュ(\)、制御文字は指定できません。JSON 文字列にはエスケープ文字としてバックスラッシュを含めることができ、このエスケープ文字によって文字シーケンスの次の文字に関する代替翻訳が呼び出されます。たとえば、JSON 文字列に二重引用符を含めることができるようにするには、"の前に\文字を置くと"はエスケープされます。

デフォルトでは、JSON 文字列として受け入れ可能な文字のみが含まれていることを保証するため、PROC JSON は入力文字列をスキャンします。入力文字列内の受け入れられない文字は、適切なエスケープシーケンスに置換されます。

NOSCAN オプションを使用すると、入力文字列に受け入れ可能な文字が含まれるか、またはすでにスキャン済みと認識されるように指定できます。NOSCAN がサポートされるのは、PROC JSON ステートメント、EXPORT ステートメント、および WRITE VALUES ステートメントです。

JSON 出力ファイルのエンコード

デフォルトでは、結果の JSON 出力ファイルでは Unicode エンコーディングの UTF-8 が使用されます。JSON 出力ファイルのエンコードを上書きするには、FILENAME ステートメント内で ENCODING=オプションを使用します。ただし、これを実行できるのは次の状況に限られます。

- 現在の SAS セッションエンコードが UTF-8 の場合、次の Unicode エンコーディング形式のいずれかを指定できます。UTF-8 および次のエンコーディング形式のみが JSON スタンダードに準拠しています。他のエンコーディング形式は、スタンダードに準拠する JSON パーサーによって認識されない可能性があります。

名前	説明
utf-16be	Unicode (UTF-16BE)
utf-32be	Unicode (UTF-32BE)
utf-32le	Unicode (UTF-32LE)

- 現在の SAS セッションのエンコーディングが UTF-8 ではない場合、JSON 出力ファイルのエンコーディングを上書きできるのは、現在の SAS セッションエンコーディングに US-ASCII との互換性があり、JSON 出力ファイルに書き込まれるすべての文字列に含まれるのが小範囲の Latin1 文字(コードポイント 0-127)のみである場合に限られます。

たとえば、次のコードによって、Unicode UTF-16BE 文字セットエンコーディングを使用する JSON 出力ファイルがエクスポートされます。現在の SAS セッションエンコーディングは Wlatin1 です。FILENAME ステートメント内の ENCODING=オプションは、外部ファイルへの書き込み時に PROC JSON がデータを Wlatin1 から指定された Unicode UTF-16BE へトランスコードするよう命令します。

```
filename jsonout "C:\JsonOutput.json" encoding="utf-16be";

proc json out=jsonout;
   export sashelp.class;
```

```
quit;
```

構文: JSON プロシジャ

PROC JSON OUT=*fileref* | "*external-file*" <*option(s)*>;
EXPORT <*libref.*>*SAS-data-set* <(*SAS-data-set-option(s)*)> </*option(s)*>;
WRITE VALUES *value(s)* </*option(s)*>;
WRITE OPEN *type*;
WRITE CLOSE;

ステートメント	タスク	例
“PROC JSON ステートメント”	JSON 出力ファイルを指定し、結果出力を制御します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6
“EXPORT ステートメント”	エクスポートする SAS データセットを識別し、結果出力を制御します	Ex. 1, Ex. 2, Ex. 4, Ex. 5, Ex. 6
“WRITE VALUES ステートメント”	1 つ以上の値を出力ファイルに書き込みます	Ex. 3, Ex. 4, Ex. 6
“WRITE OPEN ステートメント”	出力ファイル内で JSON コンテナを開いて入れ子構造にします	Ex. 3, Ex. 4, Ex. 6
“WRITE CLOSE ステートメント”	出力ファイル内で開いている JSON コンテナを閉じます	Ex. 3, Ex. 4, Ex. 6

PROC JSON ステートメント

JSON 出力ファイルを指定し、結果出力を制御します。

例:
“例 1: デフォルトオプションを使用した JSON ファイルのエクスポート” (1044 ページ)
“例 2: JSON 出力を制御するためのオプションの使用” (1046 ページ)
“例 3: SAS データセットをエクスポートせずに JSON 出力を書き込む” (1047 ページ)
“例 4: JSON コンテナの制御と値の書き込み” (1049 ページ)
“例 5: 結果出力への SAS 形式の適用” (1052 ページ)
“例 6: 複数の SAS データセットの JSON ファイルへのエクスポート” (1053 ページ)

構文

PROC JSON OUT=*fileref* | "*external-file*" <*option(s)*>;

オプション引数の要約

FMTCHARACTER | NOFMTCHARACTER

文字 SAS 形式が SAS データセット変数に関連付けられている場合、文字 SAS 形式を結果出力に適用するかどうかを決定します。

FMTDATETIME | NOFMTDATETIME
日付、時刻、または日時 SAS 形式が SAS データセット変数に関連付けられている場合、日付、時刻、または日時 SAS 形式を結果出力に適用するかどうかを決定します。

FMTNUMERIC | NOFMTNUMERIC
数値 SAS 形式が SAS データセット変数に関連付けられている場合、数値 SAS 形式を結果出力に適用するかどうかを決定します。

KEYS | NOKEYS
JSON 出力ファイルに SAS 変数名を含めるか抑制するかを決定します。

PRETTY | NOPRETTY
JSON 出力のフォーマット方法を決定します。

SASTAGS | NOSASTAGS
JSON 出力ファイルの最上段に SAS メタデータを含めるか抑制するかを決定します。

SCAN | NOSCAN
受け入れ可能な文字のみが JSON 出力ファイルにエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

TRIMBLANKS | NOTRIMBLANKS
JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。

必須引数

OUT=*fileref* | "*external-file*"
JSON 出力ファイルを識別します。

fileref
JSON 出力ファイルに割り当てられる SAS ファイル参照名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

"*external-file*"
JSON 出力ファイルの物理的な場所です。完全パス名とファイル名を含みます。物理名を一重引用符または二重引用符で囲みます。最大長は、200 文字です。

オプション引数

FMTCHARACTER | NOFMTCHARACTER
文字 SAS 形式が SAS データセット変数に関連付けられている場合、文字 SAS 形式を結果出力に適用するかどうかを決定します。

別名	FMTCHAR | NOFMTCHAR
デフォルト	NOFMTCHARACTER
操作	PROC JSON ステートメント、EXPORT ステートメントまたはその両方で FMTCHARACTER | NOFMTCHARACTER を指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。
注	ユーザー定義形式もサポートされます。

FMTDATETIME | NOFMTDATETIME
日付、時刻、または日時 SAS 形式が SAS データセット変数に関連付けられている場合、日付、時刻、または日時 SAS 形式を結果出力に適用するかどうかを決定します。SAS 形式を適用すると、結果の JSON 出力内の日付と時刻の値がより読みやすくなります。

別名 FMTDT | NOFMTDT

デフォルト FMTDATETIME

操作 You can specify FMTDATETIME | NOFMTDATETIME は、PROC JSON ステートメント、EXPORT ステートメント、またはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

ヒント ユーザー定義形式もサポートされます。

FMTNUMERIC | NOFMTNUMERIC
数値 SAS 形式が SAS データセット変数に関連付けられている場合、数値 SAS 形式を結果出力に適用するかどうかを決定します。

別名 FMTNUM | NOFMTNUM

デフォルト NOFMTNUMERIC

制限事項 SAS 形式 BEST*w*、E*w*、および *w* のみ。*d* は出力ファイルに JSON 番号を書き込みます。その他すべての SAS 形式は、JSON 文字列となります。

要件 FMTNUMERIC は数値の SAS 形式を適用します。日付、時刻、および日時 SAS 形式の場合、FMTDATETIME オプションを使用してください。

操作 NOFMTNUMERIC を使用するか、または数値変数に関連 SAS 形式が含まれない場合、最大 12 ケタの数値が出力がいるに書き込まれます。

FMTNUMERIC | NOFMTNUMERIC は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

ヒント ユーザー定義形式もサポートされます。

KEYS | NOKEYS
JSON 出力ファイルに SAS 変数名を含めるか抑制するかを決定します。

デフォルト KEYS

操作 KEYS | NOKEYS は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

PRETTY | NOPRETTY
JSON 出力のフォーマット方法を決定します。PRETTY は、インデントを使用して JSON コンテナ構造を表すより人間が読みやすい形式を作成します。NOPRETTY は、出力を 1 行で書き込みます。

デフォルト NOPRETTY

制限事項 PRETTY | NOPRETTY を指定できるのは、PROC JSON ステートメントのみです。

SASTAGS | NOSASTAGS
JSON 出力ファイルの最上段に SAS メタデータを含めるか抑制するかを決定します。メタデータは、SAS エクスポートバージョン、エクスポートした SAS データセット名、および任意のデフォルトオプション指定(PRETTY など)から構成されます。

デフォルト SASTAGS

操作 SASTAGS | NOSASTAGS は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

SCAN | NOSCAN
受け入れ可能な文字のみが JSON 出力ファイルにエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

デフォルト SCAN

操作 NOSCAN は、入力文字列に受け入れ可能な JSON テキストが含まれるか、またはすでにスキャン済みと認識されるように指定します。NOSCAN が有効な場合、入力文字列または文字値はそのまま取得され、出力 JSON 文字列は 引用符で囲まれます。

SCAN | NOSCAN は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つ全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

参照項目 “入力文字列のスキャン” (1031 ページ)

TRIMBLANKS | NOTRIMBLANKS
JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。スペース文字のみ削除されます。

デフォルト TRIMBLANKS

操作 TRIMBLANKS | NOTRIMBLANKS は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つ全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

EXPORT ステートメント

エクスポートする SAS データセットを識別し、結果出力を制御します。

別名: EX

操作: EXPORT ステートメントが PROC JSON ステートメントの後の最初のステートメントである場合、最上位コンテナは JSON オブジェクトです。ただし、NOSASTAGS オプションが PROC JSON または EXPORT ステートメントで指定される場合、最上位コンテナは JSON 配列です。PROC JSON は、黙示的に開かれた最上位コンテナを自動的に閉じます。

注: 複数の EXPORT ステートメントを送信すると、複数の SAS データセットを JSON 出力ファイルにエクスポートできます。

結果の JSON 出力では、Unicode エンコーディング形式 UTF-8 が使用されます。EXPORT ステートメントで ENCODING=データオプションを指定しても、出力ファイル内のエンコーディングを上書きできません。エンコーディングの上書きに関する詳細は、“JSON 出力ファイルのエンコード” (1031 ページ)を参照してください。

例:
“例 1: デフォルトオプションを使用した JSON ファイルのエクスポート” (1044 ページ)
“例 2: JSON 出力を制御するためのオプションの使用” (1046 ページ)
“例 4: JSON コンテナの制御と値の書き込み” (1049 ページ)
“例 5: 結果出力への SAS 形式の適用” (1052 ページ)
“例 6: 複数の SAS データセットの JSON ファイルへのエクスポート” (1053 ページ)

構文

EXPORT *<libref.>SAS-data-set <(SAS-data-set-option(s))> </option(s)>*;

オプション引数の要約

(SAS-data-set-option(s))
入力 SAS データセットに適用される SAS データセットオプションを指定します。

FMTCHARACTER | NOFMTCHARACTER
文字 SAS 形式が SAS データセット変数に関連付けられている場合、文字 SAS 形式を結果出力に適用するかどうかを決定します。

FMTDATETIME | NOFMTDATETIME
日付、時刻、または日時 SAS 形式が SAS データセット変数に関連付けられている場合、日付、時刻、または日時 SAS 形式を結果出力に適用するかどうかを決定します。

FMTNUMERIC | NOFMTNUMERIC
数値 SAS 形式が SAS データセット変数に関連付けられている場合、数値 SAS 形式を結果出力に適用するかどうかを決定します。

KEYS | NOKEYS
JSON 出力ファイルに SAS 変数名を含めるか抑制するかを決定します。

SASTAGS | NOSASTAGS
JSON 出力ファイルの最上段に SAS メタデータを含めるか抑制するかを決定します。

SCAN | NOSCAN
受け入れ可能な文字のみが JSON 出力にエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

TABLENAME="*name*"
エクスポートした SAS データセットの名前を指定します。

TRIMBLANKS | NOTRIMBLANKS

JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。

必須引数

<libref.>SAS-data-set
エクスポートする SAS データセットを 1 または 2 レベルの SAS 名(ライブラリとメンバ名)で識別します。1 レベルの名前を指定した場合、JSON プロシジャはデフォルトで USER ライブラリ(割り当てられている場合)または WORK ライブラリのどちらかを使用します。

オプション引数

(*SAS-data-set-option(s)*)
入力 SAS データセットに適用される SAS データセットオプションを指定します。たとえば、エクスポートするデータセットにパスワードが割り当てられている場合、ALTER=、PW=、READ=、または WRITE=データセットオプションを使用できます。指定の条件を満たすサブセットのデータをエクスポートするには、WHERE オプション=を使用します。SAS データセットオプションの詳細については、*SAS データセットオプション: リファレンス*を参照してください。

FMTCHARACTER | NOFMTCHARACTER
文字 SAS 形式が SAS データセット変数に関連付けられている場合、文字 SAS 形式を結果出力に適用するかどうかを決定します。

別名 FMTCHAR | NOFMTCHAR

デフォルト NOFMTCHARACTER

操作 PROC JSON ステートメント、EXPORT ステートメントまたはその両方で FMTCHARACTER | NOFMTCHARACTER を指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

FMTDATETIME | NOFMTDATETIME
日付、時刻、または日時 SAS 形式が SAS データセット変数に関連付けられている場合、日付、時刻、または日時 SAS 形式を結果出力に適用するかどうかを決定します。SAS 形式を適用すると、結果の JSON 出力内の日付と時刻の値がより読みやすくなります。

別名 FMTDT | NOFMTDT

デフォルト FMTDATETIME

操作 You can specify FMTDATETIME | NOFMTDATETIME は、PROC JSON ステートメント、EXPORT ステートメント、またはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

FMTNUMERIC | NOFMTNUMERIC
数値 SAS 形式が SAS データセット変数に関連付けられている場合、数値 SAS 形式を結果出力に適用するかどうかを決定します。

別名 FMTNUM | NOFMTNUM

デフォルト NOFMTNUMERIC

制限事項 SAS 形式 BEST*w*、E*w*、および *w* のみ。*d* は出力ファイルに JSON 番号を書き込みます。その他すべての SAS 形式は、JSON 文字列となります。

要件 FMTNUMERIC は数値の SAS 形式を適用します。日付、時刻、および日時 SAS 形式の場合、FMTDATETIME オプションを使用してください。

操作 NOFMTNUMERIC を使用するか、または数値変数に関連 SAS 形式が含まれない場合、最大 12 ケタの数値が出力がいるに書き込まれます。

FMTNUMERIC | NOFMTNUMERIC は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

KEYS | NOKEYS
JSON 出力ファイルに SAS 変数名を含めるか抑制するかを決定します。

デフォルト KEYS

操作 KEYS | NOKEYS は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

EXPORT ステートメントに NOKEYS オプションが含まれる場合、データコンテナは JSON 配列です。

SASTAGS | NOSASTAGS
JSON 出力ファイルの最上段に SAS メタデータを含めるか抑制するかを決定します。メタデータは、SAS エクスポートバージョン、エクスポートした SAS データセット名、および任意のデフォルトオプション指定(PRETTY など)から構成されます。

デフォルト SASTAGS

操作 SASTAGS | NOSASTAGS は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

EXPORT ステートメントが PROC JSON ステートメントの後の最初のステートメントである場合、最上位コンテナは JSON オブジェクトです。ただし、NOSASTAGS オプションが PROC JSON または EXPORT ステートメントで指定される場合、最上位コンテナは JSON 配列です。PROC JSON は、黙示的に開かれた最上位コンテナを自動的に閉じます。

SCAN | NOSCAN
受け入れ可能な文字のみが JSON 出力にエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

デフォルト SCAN

操作 NOSCAN は、入力文字列に受け入れ可能な JSON テキストが含まれるか、またはすでにスキャン済みと認識されるように指定します。NOSCAN が有効な場合、入力文字列または文字値はそのまま取得され、出力 JSON 文字列は 引用符で囲まれます。

SCAN | NOSCAN は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つのステートメント全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

参照項目 “入力文字列のスキャン” (1031 ページ)

TABLENAME="*name*"
エクスポートした SAS データセットの名前を指定します。名前は JSON 出力ファイル内の SAS メタデータとしてエクスポートされます。名前を一重引用符または二重引用符で囲みます。

デフォルト デフォルトは、SAS データセットメンバ名です。

要件 TABLENAME=オプションでは、SASTAGS オプションに JSON 出力内の SAS メタデータを含める必要があります。

TRIMBLANKS | NOTRIMBLANKS
JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。スペース文字のみ削除されます。

デフォルト TRIMBLANKS

操作 TRIMBLANKS | NOTRIMBLANKS は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つのステートメント全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

WRITE VALUES ステートメント

1 つ以上の値を JSON 出力ファイルに書き込みます。

別名: W V

操作: WRITE VALUES ステートメントが PROC JSON ステートメントの後の最初のステートメントである場合、PROC JSON によって JSON オブジェクトが最上位コンテナとして開かれます。PROC JSON は、黙示的に開かれた最上位コンテナを自動的に閉じます。

1 つの WRITE VALUES ステートメント内に複数の値を指定することは、複数の WRITE VALUES ステートメントをそれぞれ 1 つの値のみを指定して送信することに相当します。値の順番のみが重要です。

例: “例 4: JSON コンテナの制御と値の書き込み” (1049 ページ)
“例 6: 複数の SAS データセットの JSON ファイルへのエクスポート” (1053 ページ)

構文

WRITE VALUES *value(s)* </*option(s)*>;

オプション引数の要約

SCAN | NOSCAN
: 受け入れ可能な文字のみが JSON 出力にエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

TRIMBLANKS | NOTRIMBLANKS
: JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。

必須引数

value(s)
1 つ以上の値を JSON 出力ファイルに書き込むよう指定します。値は空白スペースで区切ります。値は次のうちいずれかになります。

- *string* は一重引用符または二重引用符で囲みます。文字列が引用符で囲まれる場合、コンテンツまたは長さに関して制約はありません。ただし、文字列が引用符で囲まれない場合、次の規則が適用されます。
  - 文字列の最大長は 256 バイトです。
  - 最初の文字はラテンアルファベット(A–Z、a–z)またはアンダースコアで始める必要があります。後続の文字にはラテンアルファベット、数値、またはアンダースコアを指定できます。
  - 文字列にはアンダースコア以外のブランクや特殊文字を含めることができません。文字列には大文字と小文字を両方使用できます。
- *number* は整数、浮動小数、または指数関数形式で表されます。
- ブール値 TRUE | T または FALSE | F.
- NULL | N.

たとえば、`write values "success" true;`というステートメントによって、次の結果が JSON 出力ファイルに書き込まれます。

- `"success":true` 現在のコンテナが JSON オブジェクトの場合
- `"success", true` 現在のコンテナが JSON 配列の場合

要件
: 値を JSON オブジェクトコンテナに書き込む際、名前-値ペアの名前部分は文字列にする必要があります。たとえば、`write values "abcd" "1";`というステートメントが適切です。ただし、`write values 1 "abcd";`というステートメントはエラーを生成します。

オプション引数

SCAN | NOSCAN
受け入れ可能な文字のみが JSON 出力にエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

デフォルト
: SCAN

操作 NOSCAN は、入力文字列に受け入れ可能な JSON テキストが含まれるか、またはすでにスキャン済みと認識されるように指定します。NOSCAN が有効な場合、入力文字列または文字値はそのまま取得され、出力 JSON 文字列は 引用符で囲まれます。

SCAN | NOSCAN は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つのステートメント全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

参照項目 “入力文字列のスキャン” (1031 ページ)

TRIMBLANKS | NOTRIMBLANKS
JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。スペース文字のみ削除されます。

デフォルト TRIMBLANKS

操作 TRIMBLANKS | NOTRIMBLANKS は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つのステートメント全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

WRITE OPEN ステートメント

出力ファイル内で JSON コンテナを開いて入れ子構造にします。

別名: W O

操作: WRITE OPEN ステートメントが PROC JSON ステートメントの後の最初のステートメントである場合、WRITE OPEN ステートメントは最上位コンテナを確立します。

WRITE OPEN ステートメントを使用して明示的に開いたコンテナの WRITE CLOSE ステートメントを送信します。

例: “例 4: JSON コンテナの制御と値の書き込み” (1049 ページ)
“例 6: 複数の SAS データセットの JSON ファイルへのエクスポート” (1053 ページ)

構文

WRITE OPEN *type*;

必須引数

type
JSON コンテナの種類を指定します。

ARRAY
JSON コンテナが配列であるよう指定します。この配列によって値のリストが収集されます。例としては、`write open array;`というステートメントが挙げられます。

別名 A

OBJECT
JSON コンテナがオブジェクトであるよう指定します。このオブジェクトによって名前-値のペアが収集されます。例としては、`write open object;`というステートメントが挙げられます。

別名 O

WRITE CLOSE ステートメント

JSON 出力ファイル内で開いている JSON コンテナを閉じます。

別名: W C

制限事項: WRITE CLOSE ステートメントは、PROC JSON ステートメントの後の最初のステートメントにすることはできません。

操作: The WRITE CLOSE ステートメントは、WRITE OPEN ステートメントを使用して明示的に開かれたいずれかの種類の最近開いたコンテナを閉じます。WRITE OPEN ステートメントを使用してコンテナを明示的に開いた場合に限り、コンテナの WRITE CLOSE ステートメントを送信する必要があります。

例: “例 4: JSON コンテナの制御と値の書き込み” (1049 ページ)
“例 6: 複数の SAS データセットの JSON ファイルへのエクスポート” (1053 ページ)

構文

WRITE CLOSE;

JSON プロシジャの使用

データのエクスポート

JSON プロシジャを使用すると、1 つ以上の SAS データセットを JSON 出力ファイルにエクスポートできます。プロシジャステートメントでは、エクスポートする JSON 出力ファイル名と SAS データセット名を指定します。たとえば、次の PROC JSON コードは Sashelp.Class という名前の SAS データセット内のデータを Output.json という名前の JSON 出力ファイルに書き込みます。

```
proc json out="C:\Users\sasabc\JSON\Output.json";
   export sashelp.class;
run;
```

例については、“例 1: デフォルトオプションを使用した JSON ファイルのエクスポート” (1044 ページ)を参照してください。

JSON 出力ファイルへの値の書き込み

データを SAS データセットからエクスポートするほか、PROC JSON は WRITE VALUES ステートメントを提供し、このステートメントを使用して追加データを外部ファ

イルに書き込むことができます。WRITE VALUES ステートメントは 1 つ以上の値を JSON 出力ファイルに書き込みます。値には、文字列、数字、ブール値 TRUE/FALSE、または NULL キーワードを指定できます。例については、“例 4: JSON コンテナの制御と値の書き込み” (1049 ページ)を参照してください。

また、PROC JSON を使用すると、SAS データセットをエクスポートせずに独自の JSON 出力を書き込むこともできます。例については、“例 3: SAS データセットをエクスポートせずに JSON 出力を書き込む” (1047 ページ)を参照してください。

オプションを使用した JSON 出力の制御

PROC JSON は、結果の JSON 出力を制御する複数のオプションをサポートします。PROC JSON、EXPORT、および WRITE VALUES ステートメントでオプションを指定できます。

次の表では、オプションと、それがステートメントでサポートされるかどうかをリストで示します。

注: デフォルトのオプションキーワードは太字で表示されます。

表 35.1 *ステートメントオプションを使用できるかどうか*

オプション	関数	PROC JSON ステートメント	EXPORT ステートメント	WRITE VALUES ステートメント
FMTCHARACTER | **NOFMTCHARACTER**	関連文字の SAS 形式を適用するかどうかを決定します。	はい	はい	いいえ
FMTDATETIME | NOFMTDATETIME	関連する日付、時刻、または日時の SAS 形式を適用するかどうかを決定します。	はい	はい	いいえ
FMTNUMERIC | **NOFMTNUMERIC**	関連する数値の SAS 形式(日付、時刻、および日時の SAS 形式を除く)を適用するかどうかを決定します。	はい	はい	いいえ
KEYS | NOKEYS	SAS 変数名を含めるか抑制するかを決定します。	はい	はい	いいえ
PRETTY | **NOPRETTY**	JSON 出力のフォーマット方法を決定します。	はい	いいえ	いいえ
SASTAGS | NOSASTAGS	SAS メタデータを含めるか抑制するかを決定します。	はい	はい	いいえ
SAS-data-set-option(s)	SAS データセットに適用するアクションを指定します。	いいえ	はい	いいえ

オプション	関数	PROC JSON ステートメント	EXPORT ステートメント	WRITE VALUES ステートメント
SCAN | NOSCAN	PROC JSON が入力文字列をスキャンしてエンコードするかどうかを決定します。	はい	はい	はい
TABLENAME="*name*"	エクスポートした SAS データセットの SAS メタデータで名前を指定します。	いいえ	はい	いいえ
TRIMBLANKS | NOTRIMBLANKS	末尾のブランクを削除または保持するかどうかを決定します。	はい	はい	はい

注: PROC JSON ステートメントで指定されるオプションは、プロシジャの持続時間に適用されます。EXPORT および WRITE VALUES ステートメントで指定されるオプションが適用されるのは、そのステートメントのみです。オプションが複数のステートメントで指定される場合、EXPORT ステートメントまたは WRITE VALUES ステートメントで指定されるオプションの方が優先されます。

例については、“例 2: JSON 出力を制御するためのオプションの使用” (1046 ページ) を参照してください。

PROC JSON ビデオ

PROC JSON の使い方のデモを示すビデオは、SAS から JSON 出力を書き込む方法を参照してください。このビデオでは、SAS データセットを JSON 出力ファイルにエクスポートする方法、データのサブセットをエクスポートして JSON 出力を制御する方法、自由な形式の JSON 出力を書き込む方法、JSON コンテナを制御および入れ子にする方法、および、複数の SAS データセットのエクスポート方法を示します。

例: JSON プロシジャ

例 1: デフォルトオプションを使用した JSON ファイルのエクスポート

要素: PROC JSON ステートメント
EXPORT ステートメント

詳細

この PROC JSON の例では、Sashelp.Class データセットが JSON 出力ファイルにエクスポートされます。この例では、出力を制御するためのオプションは指定されません。つまり、すべてのデフォルトオプションは有効化されています。

結果の JSON 出力ファイルには、次のコンテンツが含まれます。

- PROC JSON は、出力ファイルを最上位コンテナとして開かれた JSON オブジェクトコンテナ({)で開始します。
- 開かれたオブジェクトコンテナの直後では、SAS エクスポートバージョンとエクスポートされた SAS データセット名は、出力ファイルの最初に含まれるデフォルトの SAS メタデータです。
- PROC JSON は JSON 配列コンテナ([)を開き、値のリストとして SAS データセットオブザベーションを収集します。
- SAS データセット内の各オブザベーションは、開かれた配列コンテナ内の入れ子構造の JSON オブジェクト({ })としてエクスポートされます。
  - デフォルトでは、PROC JSON はデータを 1 行で記述します。
  - 各オブザベーションは、変数の名前と値の名前-値ペアから構成されます。
  - デフォルトでは、末尾のブランクは文字データの最後から削除されます。
- PROC JSON は、配列コンテナ(])と最上位コンテナを閉じて出力ファイルを終了します。

プログラム

```
proc json out="C:\Users\sasabc\JSON\DefaultOutput.json";

   export sashelp.class;
run;
```

プログラムの説明

JSON 出力ファイルを指定します。 PROC JSON ステートメントは、完全パス名とファイル名を使用して JSON オプションの物理的な場所を指定します。ステートメントでは、出力を制御するためのオプションは指定されません。

```
proc json out="C:\Users\sasabc\JSON\DefaultOutput.json";
```

エクスポートする SAS データセットを識別します。 EXPORT ステートメントは、2 レベル SAS 名を指定します。ステートメントでは、出力を制御するためのオプションは指定されません。

```
   export sashelp.class;
run;
```

出力:デフォルトオプションを使用した JSON ファイルのエクスポート

この例の出力は、エクスポートされた各オブザベーションについて改行が入った状態で表示されますのでご注意ください。実際の JSON 出力ファイルは 1 行のテキストです。

アウトプット 35.1 *PROC JSON 出力ファイル DefaultOutput.json*


```
{"SASJSONExport":"1.0","SASTableData+CLASS":[{"Name":"Alfred","Sex":"M","Age":
14,"Height":69,"Weight":112.5}, {"Name":"Alice","Sex":"F","Age":13,"Height":
56.5,"Weight":84}, {"Name":"Barbara","Sex":"F","Age":13,"Height":65.3,"Weight":
98}, {"Name":"Carol","Sex":"F","Age":14,"Height":62.8,"Weight":102.5},
{"Name":"Henry","Sex":"M","Age":14,"Height":63.5,"Weight":102.5},
{"Name":"James","Sex":"M","Age":12,"Height":57.3,"Weight":83},
{"Name":"Jane","Sex":"F","Age":12,"Height":59.8,"Weight":84.5},
{"Name":"Janet","Sex":"F","Age":15,"Height":62.5,"Weight":112.5},
{"Name":"Jeffrey","Sex":"M","Age":13,"Height":62.5,"Weight":84},
{"Name":"John","Sex":"M","Age":12,"Height":59,"Weight":99.5},
{"Name":"Joyce","Sex":"F","Age":11,"Height":51.3,"Weight":50.5},
{"Name":"Judy","Sex":"F","Age":14,"Height":64.3,"Weight":90},
{"Name":"Louise","Sex":"F","Age":12,"Height":56.3,"Weight":77},
{"Name":"Mary","Sex":"F","Age":15,"Height":66.5,"Weight":112},
{"Name":"Philip","Sex":"M","Age":16,"Height":72,"Weight":150},
{"Name":"Robert","Sex":"M","Age":12,"Height":64.8,"Weight":128},
{"Name":"Ronald","Sex":"M","Age":15,"Height":67,"Weight":133},
{"Name":"Thomas","Sex":"M","Age":11,"Height":57.5,"Weight":85},
{"Name":"William","Sex":"M","Age":15,"Height":66.5,"Weight":112}]}
```


例 2: JSON 出力を制御するためのオプションの使用

要素: PROC JSON ステートメントオプション
- PRETTY

EXPORT ステートメントオプション
- WHERE=データセットオプション
- NOKEYS
- NOSASTAGS

詳細

この PROC JSON の例では、Sashelp.Class データセットのサブセットが JSON 出力ファイルにエクスポートされ、JSON 出力を制御するためのオプションが指定されます。

結果の JSON 出力ファイルには、次のコンテンツが含まれます。

- PROC JSON は、出力ファイルを最上位コンテナとして開かれた JSON 配列コンテナ([)で開始します。
- 出力ファイルの最初には SAS メタデータはありません。
- SAS データセットで選択された各オブザベーションは、最上位コンテナ内で入れ子構造になった配列コンテナ([ ])の値のリストとしてエクスポートされます。
  - この出力は、インデントを使用して JSON コンテナ構造を表すより人間が読みやすい形式です。
  - SAS 変数名は出力に含まれません。
  - デフォルトでは、末尾のブランクは文字データの最後から削除されます。
- PROC JSON は、最上位コンテナを閉じて出力ファイルを終了します。

プログラム

```
proc json out="C:\Users\sasabc\JSON\ControlOutput.json" pretty;
```

```
   export sashelp.class (where=(age=12)) / nokeys nosastags;
run;
```

プログラムの説明

JSON 出力ファイルを指定し、結果出力ファイルを制御します PROC JSON ステートメントは、完全パス名とファイル名を使用して JSON オプションの物理的な場所を指定します。PRETTY オプションは、より読みやすい形式を作成します。

```
proc json out="C:\Users\sasabc\JSON\ControlOutput.json" pretty;
```

エクスポートする SAS データセットを識別し、結果出力ファイルを制御します。 EXPORT ステートメントは、2 レベル SAS 名を指定します。WHERE=データセットオプションは、SAS データセットからオブザベーションを選択する条件を指定します。NOKEYS オプションは SAS 変数名を削除し、EXPORT ステートメントに NOKEYS を含めることで、選択された各オブザベーションは、JSON オブジェクト内の名前-値ペアのかわりに配列コンテナ([ ])内の値のリストとしてエクスポートされます。NOSASTAGS オプションは SAS メタデータを削除し、EXPORT ステートメントに NOSASTAGS を含めることで、最上位コンテナは JSON オブジェクトではなく JSON 配列([ ])になります。

```
   export sashelp.class (where=(age=12)) / nokeys nosastags;
run;
```

出力:JSON 出力を制御するためのオプションの使用

アウトプット 35.2 *PROC JSON 出力ファイル ControlOutput.json*

```
[ [ "James", "M", 12, 57.3, 83 ], [ "Jane", "F", 12, 59.8, 84.5 ], [ "John",
"M", 12, 59, 99.5 ], [ "Louise", "F", 12, 56.3, 77 ], [ "Robert", "M", 12,
64.8, 128 ] ]
```

例 3: SAS データセットをエクスポートせずに JSON 出力を書き込む

要素: PROC JSON ステートメントオプション
PRETTY
WRITE OPEN ステートメント
WRITE VALUES ステートメント
WRITE CLOSE ステートメント

詳細

この PROC JSON の例では、データを SAS データセットからエクスポートせずに JSON 出力を書き込む方法を説明します。これによって、JSON 出力のコンテンツ全体を完全に制御し、任意の JSON 出力を生成できます。

プログラム

```
proc json out="C:\Users\sasabc\JSON\Output.json" pretty;

   write open object;
   write values "Nested object sample";
```

```
      write open object;
      write values "Comment" "In a nested object";
      write close;

      write values "Nested array sample";
      write open array;
         write open array;
            write values "In a nested array";
            write values 1 true null;
         write close;
      write close;

      write values "Finished" "End of samples";

      write close;
   run;
```

プログラムの説明

JSON 出力ファイルを指定し、結果出力を制御します。 PROC JSON ステートメントは、完全パス名とファイル名を使用して JSON オプションの物理的な場所を指定します。PRETTY オプションは、より読み取りやすい形式を作成することで、結果出力を制御します。

```
proc json out="C:\Users\sasabc\JSON\Output.json" pretty;
```

JSON オブジェクトコンテナを開き、値を JSON 出力ファイルに書き込みます。 WRITE OPEN OBJECT ステートメントは、オブジェクトコンテナ({})を最上位コンテナとして開きます。WRITE VALUES ステートメントは、最上位コンテナにある出力ファイルに文字列を書き込みます。現在のコンテナはオブジェクトであるため、文字列は名前-値ペアとしての出力です。

```
      write open object;
      write values "Nested object sample";
```

値を使用してオブジェクトコンテナを入れ子構造にして、入れ子構造のオブジェクトコンテナを閉じます。 WRITE OPEN OBJECT ステートメントは、最上位コンテナ内で開いているオブジェクトコンテナを入れ子構造にします。WRITE VALUES ステートメントは、第 2 レベルオブジェクトコンテナ内に含まれる 2 つの文字列(名前-値ペアとしての出力)を書き込みます。WRITE CLOSE ステートメントは、第 2 レベルオブジェクトコンテナである最近開いたコンテナを閉じます。

```
      write open object;
      write values "Comment" "In a nested object";
      write close;
```

値を JSON 出力ファイルに書き込み、配列コンテナを入れ子構造にし、値を使用して追加配列コンテナを入れ子構造にし、2 つの配列コンテナを閉じます。 WRITE VALUES ステートメントは、第 2 レベルコンテナ内にある出力ファイル(オブジェクト)に文字列を書き込み、名前-値ペアを出力します。WRITE OPEN ARRAY ステートメントは、最上位コンテナ内の開いているコンテナを入れ子構造にして、前に開いた配列コンテナ内で開いている配列コンテナを入れ子構造にします。WRITE VALUES ステートメントは、文字列、数値、ブール値 TRUE、NULL キーワードの順に文字列を出力ファイルに書き込みます。両方の値が第 2 レベル配列コンテナ内に書き込まれ、値のリストとしての出力になります。WRITE CLOSE ステートメントは、最近開いたコンテナ(第 2 レベル配列コンテナ)を閉じ、次に第 1 レベル配列コンテナを閉じます。

```
      write values "Nested array sample";
```

```
   write open array;
      write open array;
         write values "In a nested array";
         write values 1 true null;
      write close;
   write close;
```

最終値を書き込みます。 WRITE VALUES ステートメントは、第 2 レベルコンテナ内にある出力ファイル(オブジェクト)に 2 つの文字列を書き込み、名前-値ペアとして出力します。

```
   write values "Finished" "End of samples";
```

最上位コンテナを閉じます。 WRITE CLOSE ステートメントは、残りの開いているコンテナ(最上位コンテナ)を閉じます。

```
   write close;
run;
```

出力:SAS データセットをエクスポートせずに JSON 出力を書き込む

アウトプット 35.3 *PROC JSON 出力ファイル Output.json*

```
{ "Nested object sample":{ "Comment":"In a nested object" }, "Nested array
sample":[ [ "In a nested array", 1, true, null ] ], "Finished":"End of
samples" }
```

例 4: JSON コンテナの制御と値の書き込み

要素: PROC JSON ステートメントオプション
- NOSASTAGS
- PRETTY

WRITE OPEN ステートメント

WRITE VALUES ステートメント

EXPORT ステートメント

WRITE CLOSE ステートメント

詳細

この PROC JSON の例では、追加値を JSON 出力ファイルに書き込み、JSON コンテナを制御して入れ子構造にする方法について説明します。この例では、Sashelp.Cars データセットのサブセットが JSON 出力ファイルにエクスポートされます。

プログラム

```
%let vehicleType=Truck;
%let minCost=26000;

proc json out="C:\Users\sasabc\JSON\WriteOutput.json" nosastags pretty;

   write open array;
   write values "Vehicles";
```

```
   write open array;
   write values "&vehicleType";
   write open array;
   write values "Greater than $&minCost";
   /*********** Asian **********************/
   %let originator=Asia;
   write open object;
   write values "&originator";
   write open array;
   export sashelp.cars(where=((origin = "&originator")  and
                              (type   = "&vehicleType") and
                              (MSRP   > &minCost)           )
                       keep=make model type origin MSRP);
   write close; /* data values */
   write close; /* Asia */
   /*********** European **********************/
   %let originator=Europe;
   write open object;
   write values "&originator";
   write open array;
   export sashelp.cars(where=((origin = "&originator")  and
                              (type   = "&vehicleType") and
                              (MSRP   > &minCost)           )
                       keep=make model type origin MSRP);
   write close; /* data values */
   write close; /* Europe */
   /*********** American **********************/
   %let originator=USA;
   write open object;
   write values "&originator";
   write open array;
   export sashelp.cars(where=((origin = "&originator")  and
                              (type   = "&vehicleType") and
                              (MSRP   > &minCost)           )
                       keep=make model type origin MSRP);
   write close; /* data values */
   write close; /* USA */
   write close; /* expensive */
   write close; /* vehicleType */
   write close; /* cars */
run;
```

プログラムの説明

マクロ変数を割り当てます。 %LET ステートメントは、マクロ変数を作成し、値がコード全体で使用されるように割り当てます。

```
%let vehicleType=Truck;
%let minCost=26000;
```

JSON 出力ファイルを指定し、結果出力を制御します。 PROC JSON ステートメントは、完全パス名とファイル名を使用して JSON オプションの物理的な場所を指定します。NOSASTAGS オプションは SAS メタデータを抑制し、PRETTY オプションはより読み取りやすい形式を作成します。

```
proc json out="C:\Users\sasabc\JSON\WriteOutput.json" nosastags pretty;
```

PROC JSON ステートメントを送信します。 このステートメントは、各カータイプの入れ子構造になった一連のコンテナを開き、値をコンテナのラベルとして書き込みます。EXPORT ステートメントは、エクスポートする SAS データセットを指定し、特定のオブザベーションを選択して、所定の SAS 変数のみを要求します。

```
   write open array;
   write values "Vehicles";
   write open array;
   write values "&vehicleType";
   write open array;
   write values "Greater than $&minCost";
   /*********** Asian **********************/
   %let originator=Asia;
   write open object;
   write values "&originator";
   write open array;
   export sashelp.cars(where=((origin = "&originator")  and
                              (type   = "&vehicleType") and
                              (MSRP   > &minCost)           )
                       keep=make model type origin MSRP);
   write close; /* data values */
   write close; /* Asia */
   /*********** European **********************/
   %let originator=Europe;
   write open object;
   write values "&originator";
   write open array;
   export sashelp.cars(where=((origin = "&originator")  and
                              (type   = "&vehicleType") and
                              (MSRP   > &minCost)           )
                       keep=make model type origin MSRP);
   write close; /* data values */
   write close; /* Europe */
   /*********** American **********************/
   %let originator=USA;
   write open object;
   write values "&originator";
   write open array;
   export sashelp.cars(where=((origin = "&originator")  and
                              (type   = "&vehicleType") and
                              (MSRP   > &minCost)           )
                       keep=make model type origin MSRP);
   write close; /* data values */
   write close; /* USA */
   write close; /* expensive */
   write close; /* vehicleType */
   write close; /* cars */
run;
```

出力:JSON コンテナの制御と値の書き込み

アウトプット 35.4 *PROC JSON 出力ファイル WriteOutput.json*

```
[ "Vehicles", [ "Truck", [ "Greater than $26000", { "Asia":[ { "Make":"Nissan",
"Model":Titan King Cab XE", "Type":"Truck", "Origin":"Asia", "MSRP":26650 } ] },
{ "Europe":[ ] }, { "USA":[ { "Make":"Cadillac", "Model":" Escalade EXT",
"Type":"Truck", "Origin":"USA", "MSRP":52975 }, { "Make":"Chevrolet",
"Model":Avalanche 1500", "Type":"Truck", "Origin":"USA", "MSRP":36100 },
{ "Make":"Chevrolet", "Model":" Silverado SS", "Type":"Truck", "Origin":"USA",
"MSRP":40340 }, { "Make":"Chevrolet", "Model":" SSR", "Type":"Truck",
"Origin":"USA", "MSRP":41995 },
```

```
         { "Make":"Ford", "Model":" F-150 Supercab Lariat", "Type":"Truck",
"Origin":"USA", "MSRP":33540 }, { "Make":"GMC", "Model":" Sierra HD 2500",
"Type":"Truck", "Origin":"USA", "MSRP":29322 } ] } ] ] ]
```

例 5: 結果出力への SAS 形式の適用

要素: PROC JSON ステートメントオプション
PRETTY
EXPORT ステートメントオプション
FMTDATETIME
FMTNUMERIC

他の要素: DATA ステップ

詳細

この PROC JSON の例では、SAS 形式に関連付けられた変数を含む Work.Formats という名前の SAS データセットがエクスポートされます。結果の JSON 出力ファイルは SAS 形式に適用され、出力値がより読み取りやすくなります。

プログラム

```
data formats;
   input name $ idnumber $ salary hiredate mmddyy10.;
   format salary dollar7. hiredate date9.;
   datalines;
Brad 0755 21163 9/24/2012
Lindzey 0767 34321 9/04/2012
;

proc json out="C:\Users\sasabc\JSON\FormatsOutput.json" pretty;

   export work.formats / fmtnumeric;
run;
```

プログラムの説明

SAS 形式に関連付けられた変数を含む SAS データセットを作成します。 DATA ステップによって、4 つの変数を含む Work.Formats という名前の SAS データセットが作成されま

す。FORMAT ステートメントによって、DOLLAR7 が関連付けられます。変数 Salary を含む SAS 数値形式および変数 HireDate を含む SAS 日付形式 DATE9。

```
data formats;
   input name $ idnumber $ salary hiredate mmddyy10.;
   format salary dollar7. hiredate date9.;
   datalines;
Brad 0755 21163 9/24/2012
Lindzey 0767 34321 9/04/2012
;
```

JSON 出力ファイルを指定し、結果出力ファイルを制御します PROC JSON ステートメントは、完全なパス名とファイル名を持つ JSON 出力ファイルの物理的場所を指定し、より読み取りやすい形式を作成するための PRETTY オプションを含めます。

```
proc json out="C:\Users\sasabc\JSON\FormatsOutput.json" pretty;
```

エクスポートする SAS データセットを識別します。 EXPORT ステートメントは、SAS データセット名を指定します。FMTDATETIME オプションは、変数 HireDate に関連付けられた日付 SAS 形式を適用するためにデフォルトで使用できます。FMTNUMERIC オプションは、変数 Salary に関連付けられた数値 SAS 形式 DOLLAR7.を適用するために指定されます。

```
   export work.formats / fmtnumeric;
run;
```

出力:結果出力への SAS 形式の適用

アウトプット 35.5 *FMTDATETIME および FMTNUMERIC を含む PROC JSON 出力ファイル FormatsOutput.json*

```
{ "SASJSONExport":"1.0 PRETTY FMTNUMERIC", "SASTableData+FORMATS":
[ { "name":"Brad", "idnumber":"0755", "salary":"$21,163",
"hiredate":"24SEP2012" }, { "name":"Lindzey", "idnumber":"0767",
"salary":"$34,321", "hiredate":"04SEP2012" } ] }
```

EXPORT ステートメントが NOFMTDATETIME オプションを指定し、FMTNUMERIC オプションは指定しなかった場合、結果の JSON 出力ファイルには、読み取りやすさが低い Salary および HireDate の値が含まれます。

アウトプット 35.6 *NOFMTDATETIME および NOFMTNUMERIC を含む PROC JSON 出力ファイル FormatsOutput.json*

```
{ "SASJSONExport":"1.0 PRETTY NOFMTDATETIME", "SASTableData+FORMATS":
[ { "name":"Brad", "idnumber":"0755", "salary":21163, "hiredate":19260 },
{ "name":"Lindzey", "idnumber":"0767", "salary":34321, "hiredate":19240 } ] }
```

例 6: 複数の SAS データセットの JSON ファイルへのエクスポート

要素: PROC JSON ステートメントオプション
- PRETTY
- NOKEYS
- NOSASTAGS

WRITE OPEN ステートメント
WRITE VALUES ステートメント
EXPORT ステートメントオプション
  WHERE=データセットオプション
  DROP=データセットオプション
  KEYS オプション
WRITE CLOSE ステートメント

詳細

この PROC JSON の例では、2 つの SAS データセットが JSON 出力ファイルにエクスポートされます。SasHelp.Class データセットには、学生の名前、年齢、性別などの情報が含まれています。MyFiles.Fitness データセットには、学生が実行できる腹筋や腕立て伏せの回数などの運動記録が含まれます。この例ではまた、JSON コンテナを制御およびネストする方法と、JSON 出力ファイルに追加の値を書き込む方法についても説明します。

結果の JSON 出力ファイルには、次のコンテンツが含まれます。

- オブジェクトコンテナ({})を最上位コンテナとして開くと、出力が開始されます。これは、PROC JSON ステートメントの後の最初のステートメントが WRITE OPEN OBJECT であるためです。
- 出力ファイルの最初には SAS メタデータはありません。
- この出力は、インデントを使用して JSON コンテナ構造を表すより人間が読みやすい形式です。
- ユーザー定義の文字列は、コンテナのラベルとして書き込まれます。
- 一連の JSON コンテナが開かれ、ネストされます。
- 2 つの SAS データセットのサブセットがエクスポートされます。最初の SAS データセットの値は入れ子構造になった配列コンテナとしてエクスポートされ、値のリストで構成されます。2 番目の SAS データセットの値は入れ子構造になったオブジェクトコンテナとしてエクスポートされ、名前-値のペアで構成されます。
- 明示的に開かれた JSON コンテナは閉じられます。

プログラム

```
proc json out='C:\JSON\MultipleDataSets.json' pretty nokeys nosastags;

   write open object; /* top-level object */
     write value "Fitness";
     write open array; /* fitness array */
       write value "Class List";
       write open array; /* class list array */

         export sashelp.class (where=(age eq 11) drop=height weight);


       write close; /* class list array */
     write close;  /* fitness array */


     write value "Results";
     write open array; /* results array */
```

```
      export myfiles.fitness (where=(age eq 11) drop=name)/ keys;

    write close; /* results array */
  write close; /* top-level object */
run;
```

プログラムの説明

JSON 出力ファイルを指定し、結果出力を制御します。 PROC JSON ステートメントは、完全パス名とファイル名を使用して JSON オプションの物理的な場所を指定します。PRETTY オプションは、より読みやすい形式を作成します。NOKEYS オプションは、SAS 変数名を非表示にします。NOSASTAGS オプションは、SAS メタデータを非表示にします。

```
proc json out='C:\JSON\MultipleDataSets.json' pretty nokeys nosastags;
```

ラベル付けされているコンテナを開いてネストします。 このステートメントは、入れ子構造になった一連のコンテナを開き、値をコンテナのラベルとして書き込みます。

```
write open object; /* top-level object */
  write value "Fitness";
  write open array; /* fitness array */
    write value "Class List";
    write open array; /* class list array */
```

最初にエクスポートする SAS データセットを識別します。 EXPORT ステートメントは、2 レベル SAS 名を指定します。WHERE=データセットオプションは、オブザベーションの選択に必要な条件を指定します。DROP=データセットオプションは、指定した変数が出力ファイルに書き込まれないようにします。選択したオブザベーションは入れ子構造になった配列コンテナとしてエクスポートされ、値のリストで構成されます。

```
      export sashelp.class (where=(age eq 11) drop=height weight);
```

2 つの配列コンテナを閉じます。 2 つの WRITE CLOSE ステートメントにより、2 つの配列コンテナがを閉じられます。コンテナを明示的に開いた場合は明示的に閉じる必要があるので、注意が必要です。

```
    write close; /* class list array */
  write close;  /* fitness array */
```

ネストされた配列コンテナにラベルを付けて開きます。 WRITE VALUE ステートメントは、最上位コンテナ内のユーザー定義の文字列結果を入れ子構造にします。WRITE OPEN ARRAY ステートメントは、ネストされた配列コンテナを開きます。

```
  write value "Results";
  write open array; /* results array */
```

2 番目にエクスポートする SAS データセットを識別します。 EXPORT ステートメントは、2 レベル SAS 名を指定します。WHERE=データセットオプションは、オブザベーションの選択に必要な条件を指定します。DROP=データセットオプションは、指定した変数が出力ファイルに書き込まれないようにします。KEYS オプションでは、選択されたオブザベーションが入れ子構造になったオブジェクトコンテナとしてエクスポートされ、名前-値

のペアで構成されます。EXPORT ステートメントでは、KEYS オプションは、PROC JSON ステートメントで指定された NOKEYS オプションよりも優先されます。

```
export myfiles.fitness (where=(age eq 11) drop=name)/ keys;
```

2 つの開いている配列コンテナを閉じます。 2 つの WRITE CLOSE ステートメントは、配列コンテナと最上位コンテナのオブジェクトコンテナを明示的に閉じます。

```
    write close; /* results array */
  write close; /* top-level object */
run;
```

出力:複数の SAS データセットの JSON ファイルへのエクスポート

アウトプット 35.7 *PROC JSON 出力ファイル MultipleDataSets.json*


```
{ "Fitness":[ "Class List", [ [ "Joyce", "F", 11 ], [ "Thomas", "M", 11 ] ] ],
"Results":[ { "Age":11, "Push-ups":15, "Crunches":20 }, { "Age":11, "Push-ups":
22, "Crunches":33 } ] }
```

36 章

LUA プロシジャ

概要: LUA プロシジャ

Lua プログラミング言語は、埋め込み可能なスクリプト言語で、標準の C コンパイラを持つすべてのプラットフォームで実行します。Lua には、UNIX、Windows およびモバイルオペレーティングシステム(Android、iOS など)のすべてのバージョンが含まれます。Lua ではシンプルな構文が使用され、計算処理が高速です。また、メモリ割り当てが自動的に管理されます。

LUA プロシジャを使用すると、SAS コード内の Lua5.2 プログラミング言語からステートメントを実行できます。Lua ステートメントを外部 Lua スクリプトからサブミットしたり、SAS コードに直接入力したりできます。

注: LUA プロシジャのサポートは、SAS 9.4 メンテナンスリリース 3 で追加されました。

PROC LUA を使用すると、次のタスクを実行できます。

- SAS セッションで Lua コードを実行する
- Lua ステートメントでほとんどの SAS 関数を呼び出す
- Lua ステートメントで PROC FCMP 関数を呼び出す
- Lua ステートメントで SAS データセットを操作する
- Lua から SAS コードをサブミットする

概念: LUA プロシジャ

Lua スクリプトの入力場所の指定

通常、長い Lua コードブロックについては、個別のスクリプトファイルに保存します。これにより、Lua コードをより簡単に管理できます。本章の例では、SUBMIT ステートメントと ENDSUBMIT ステートメントを使用して、Lua コードを特定します。ただし、Lua コードはすべて、個別の Lua スクリプトに保存できます。

スクリプトからコードを実行するには、次の情報を指定します。

- Lua スクリプトの場所
- 実行する Lua スクリプトの名前

Lua スクリプトの場所を指定するには、FILENAME ステートメントを使用して、LUAPATH ファイル参照名を定義します。1 つの場所を定義するには、次のような SAS ステートメントを入力します。

```
filename LUAPATH "/usr/local/scripts/lua";
```

Lua スクリプトに対して複数の場所を指定できます。入力された Lua スクリプトは、リストされた場所から、ユーザーが指定した順序で検索されます。これらの場所のいずれかに、Lua システムスクリプトが含まれている必要があります。複数の場所を指定するには、リストをかっこで囲み、各値をカンマで区切ります。

```
filename LUAPATH ("/usr/local/scripts/lua","/user/my_name/my_lua_scripts");
```

注: SAS セッション中に LUAPATH の値を変更する場合は、LUA プロシジャを呼び出して、RESTART オプションを指定します。RESTART オプションは、Lua のステータスをリセットし、LUAPATH の最後の値を選択します。

実行する Lua スクリプトの名前を指定するには、PROC LUA ステートメントで INFILE=オプションを使用します。詳細については、"例 1: 外部 Lua スクリプトからの入力の指定" (1069 ページ)を参照してください。

外部 Lua ファイルの実行

Lua プログラミング言語で記述された外部スクリプトは、SAS コマンド行から実行できます。未コンパイルの Lua スクリプト(*.lua ファイル)またはプリコンパイル済み Lua スクリプト(*.luc ファイル)の両方を実行できます。SAS 起動で外部 Lua ファイルを直接実行する機能のサポートは、SAS 9.4 メンテナンスリリース 3 で追加されました。

注: プリコンパイル済み*.luc ファイルを作成するには、Lua 5.2 以降が必要です。Lua コンパイラの使用方法の詳細については、Lua のドキュメントを参照してください。

起動時に Lua ファイルを実行するには、-SYSIN オプションを使用します。たとえば、abc.lua ファイルを実行するには、次のコマンドをサブミットします。

```
sas -sysin abc.lua
```

また、SAS セッションで%INCLUDE ステートメントを使用して、外部 Lua スクリプト(*.lua ファイルまたは*.luc ファイル)を実行することもできます。たとえば、Lua スクリプト abc.luc を実行するには、SAS プログラムで次の行を入力します。

```
%include "./tmp/abc.luc";
```

Lua ステートメントでの標準 SAS 関数の呼び出し

Lua ステートメントで標準 SAS 関数を実行するには、関数の前に接頭辞"SAS."を付けます。これにより、標準 SAS 関数のほとんどを実行できます。たとえば、SAS MAX(<*arguments*>)関数を呼び出すには、次のように関数の前に"SAS."を付けます。

```
local max = sas.max(a,b,c)
```

注: ADDRLONG など、DATA ステップでのみ実行される関数は、Lua コードで実行しません。

SAS 関数で必要な引数に対して値を指定しないと、その引数は、SAS 関数によって欠損値に変換されます。

Lua ステートメントでの SAS コードのサブミット

SAS コードをサブミットする関数

Lua は SAS 内のスクリプト言語であるため、SAS コードをサブミットし、Lua ステートメント内で SAS 値をかわりに使うことができます。SAS をサブミットするには、次の関数を使用します。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

SAS.SUBMIT(*[[SAS code]]*, {*substitution(s)*})
Lua ステートメント内で関数が発行されるときに、指定された SAS コードがサブミットされ、実行されます。SAS コードは、角かっこ([[および]])で囲みます。

代入値が指定されている場合、その値は最初に割り当てられます。残りの代入値はすべて、ローカル変数によって指定されます。

代入値がサブミットされていない場合は、呼び出し側の関数環境からローカル変数が使用されます。代入用の変数は@記号で囲みます。たとえば、userID という変数の値を、可変ユーザーに割り当てるには、次の割り当てを発行します。

```
user = @userID@
```

SAS.SUBMIT への呼び出しによって代入を使用する例を次に示します。

```
sas.submit([[
   data @name@;
   x = @x@;
run;
]],{name="foo"})
```

SAS.SUBMIT からの戻り値は、SYSERR 自動マクロ変数の値に割り当てられます。

SAS.SUBMIT_(*SAS code*, *substitution(s)*)
SAS コードと代入値をサブミットしますが、コードは実行しません。サブミットされたすべての SAS コマンドを実行し、SAS.SUBMIT_への前の呼び出しを終了するには、SAS.SUBMIT 関数(アンダースコアなし)を発行する必要があります。

代入値がサブミットされていない場合は、呼び出し側の関数環境からローカル変数が使用されます。代入用の変数は@記号で囲みます。

詳細については、“Lua コード内で SAS データセットを開く” (1069 ページ)を参照してください。

SAS ステートメントでの Lua 変数代入について

変数代入を実行するには、キーと値のペアが含まれるハッシュテーブルを使用します。キーと値のペアは、サブミットされた Lua ステートメント内で行われた Lua 変数割り当てによって定義されます。値を代入するには、キーを@記号で囲みます(例:@key@)。@key@の値は、それに対応する値で置き換えられます。代入では大文字と小文字が区別されます。つまり、@key@と@Key@のキー値は異なります。

Lua によるブール値の解釈

Lua には、ブールコンテキストで使用されるデータ型が 2 つあります。ニル型は、1 つの値 **nil** をとることができます。ブール型は、値として **true** または **false** をとることができます。Lua では、**nil** 値または **false** 値が false と見なされます。0 を含め、その他の値はすべて、true と見なされます。

SAS には、条件が true の場合に値 1 を返し、条件が false の場合に値 0 を返す数値関数が多数あります。たとえば、SYMEXIST、SYMLOCAL、SYMGLOBAL、MISSING が、こうした関数に該当します。ANY*文字関数など、その他の SAS 関数は、指定された文字を文字列内で検索し、最初に見つかった文字の位置を返します。その文字が見つからない場合、これらの関数は 0 を返します。

Lua コードで任意の SAS 関数を呼び出す場合は、意図したとおりに Lua コードが結果を解釈することを確認します。

たとえば、マクロ変数 FOOBAR が定義されていないとします。このため、この変数はローカルシンボルテーブルに存在しません。次のコードは、変数が存在すること、およびローカルであることを誤って記述しています。

```
/* INCORRECT use of SAS Boolean functions */
proc lua;
   submit;
```

```
      if sas.symexist("foobar") then
         if sas.symlocal("foobar") then
            print("In Proc LUA, foobar exists and is LOCAL.")
         else
            print("In Proc LUA, foobar exists but is not LOCAL.")
         end
      else
         print(In Proc LUA, foobar does not exist.")
      end
   endsubmit;
run;
```

前のコードは、FOOBAR が存在すること、およびローカルであることを誤って記述しています。SAS 関数 SYMEXIST および SYMLOCAL によって返された 0 値が、Lua で true として解釈されているからです。

かわりに、SAS ブール関数を呼び出すときに、必要な戻り値を Lua コードで明示的にテストします。次のコードは、マクロ変数 FOOBAR が存在するかどうか、また、ローカルかどうかを確認する適切なテストを示しています。

```
/* CORRECT use of SAS Boolean functions */
proc lua;
   submit;
      if sas.symexist("foobar") == 1 then
         if sas.symlocal("foobar") == 1 then
            print("In Proc LUA, foobar exists and is LOCAL.")
         else
            print("In Proc LUA, foobar exists but is not LOCAL.")
         end
      else
         print("In Proc LUA, foobar does not exist.")
      end
   endsubmit;
run;
```

前のコードは、マクロ変数 FOOBAR(未定義)が存在しないこと、およびローカルでないことを正しく記述しています。SAS 関数 SYMEXIST からの戻り値 0 は、明示的に値 1 と比較されています。

LUA プロシジャのデータセット関数

次の関数は、LUA プロシジャで実行されるように定義されています。これらの関数は、SAS システムまたはデータセットと対話します。

小さなデータセットについては、SAS データセット全体を Lua テーブルに読み込むことができます。その後、Lua テーブルを変更するか、テーブルに対してクエリを実行できます。また、新しい SAS データセットに変更を書き込むことができます。ただし、大きなデータセットの場合は、DATA ステップでの処理と同じように、一度に 1 つのオブザベーションでデータを処理する方が効率的です。この理由から、大きなデータセットについては、Lua コードで、DATA ステップをサブミットすることをお勧めします。

LUA プロシジャ内で実行されるデータセット関数を次に示します。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

SAS.ADD_VARS(*data-set-ID*, *variable-definitions*)
指定された Lua 変数をデータセットに追加します。変数を定義するための出力形式は次のとおりです。

```
{ {name="varname", type="N|C",
format="format.", length="value",
label="varlabel", informat="informat."},
{additional variable definition}, ...
}
```

変数名のみが必須です。デフォルトの変数の種類は数値(N)です(同じ名前の Lua 変数も数値の場合)。デフォルトの文字変数の長さは 200 文字です。

完全な SAS 出力形式(`format="best12.3"`など)を指定できます。ピリオド(.)が含まれる出力形式を指定すると、完全な出力形式と見なされ、追加の出力形式の属性は無視されます。完全な SAS 出力形式を指定しない場合は、これらの変数属性の組み合わせを使用して、変数の出力形式を指定できます。

- FORMAT では、長さも小数点も指定しません(例:`format="best"`)
- FORMAT_WIDTH では、文字数または桁数を指定します(例:`format_width=12`)
- FORMAT_DEC では、小数点以下の桁数を指定します(例:`format_dec=3`)

詳細については、“例 10: 変数の定義と、データセットへの変数の追加” (1085 ページ)を参照してください。

SAS.ATTR(*data-set-ID*, *attribute-name*)
データセットの *attribute-name* の値を返します。たとえば、`sas.attr(data-set-ID, 'label')`は、指定されたデータセットのラベルを返します。

SAS.CLOSE(*data-set-ID*)
開いているデータセットを閉じます。この関数は、データセット ID が無効の場合は終了します。

SAS.EXISTS(*SAS-data-set-name*)
データセットが存在する場合はブール値 `true` を、存在しない場合は `false` を返します。

注: この関数は、0 または 1 の値を返す標準 SAS 関数 EXIST(末尾に's'がありません)とは異なります。Lua コードでは、0 および 1 は true として解釈されます。別の方法として、条件`(sas.exist("work.test") > 0)`が true かどうかをテストします。SAS.EXISTS 関数は、Lua コードブロック内でのみ使用できます。

SAS.LOAD_DS(*SAS-data-set-name*)
指定された SAS データセットのデータを含む Lua テーブルを返します。データセットが存在しない場合、関数は `nil` を返します。したがって、SAS.EXISTS 関数は、SAS.LOAD_DS 関数を呼び出す前に使用します。

最良の方法として、小さなデータセットについては、SAS.LOAD_DS 関数のみを使用してください。大きなデータセットの場合は、個別のオブザベーションを処理する関数を使用してオブザベーションを反復します。詳細については、“オブザベーションを処理する関数” (1063 ページ)を参照してください。

Alias:SAS.READ_DS(*SAS-data-set-name*)

SAS.NOBS(*data-set-ID*)
データセット内のオブザベーションの数を返します。

SAS.NVARS(*data-set-ID*)
データセット内の変数の数を返します。

SAS.OPEN(*SAS-data-set-name<, mode>*)
SAS データセットを開き、データセットが適切に開いた場合は、データセット ID を返します。データセットが開かない場合、関数は **nil** を返します。したがって、SAS.EXISTS 関数は、SAS.OPEN 関数を呼び出す前に使用します。

有効なデータセットモードは、*I*(読み込み)、*O*(作成)および *U*(更新)です。データセットモードを指定しないと、デフォルト値 I が使用されます。

SAS.SET_ATTR(*data-set-ID*, *attribute-name*, *value*)
値をデータセットの属性に割り当てます。

SAS.WHERE(*data-set-ID*, *where-clause*)
WHERE 句をデータセットに適用します。データセットにすでに WHERE 句が存在する場合、指定された WHERE 句は、前の WHERE 句に追加されます。ただし、オプションの *replace-where-clause* 引数のブール値が true の場合は、指定された WHERE 句によって既存の WHERE 句が置き換えられます。

この関数のリターンコードはブール値です。戻り値が false の場合は、オプションのメッセージが SAS ログに出力されます。

where("also <new condition>")、**where("undo")**、**where("clear")**などの操作がサポートされています。詳細については、*SAS Component Language:Reference* を参照してください。

次のような Lua ステートメントをサブミットして、WHERE 句をデータセットに適用できます。

```
sas.submit("data work.air; set sashelp.air; run;")
dsid = sas.open("work.air")
rc, msg = sas.where(dsid,"air=222 or air=999")
print(rc, msg)
rc=sas.close(dsid)
```

SAS.WRITE_DS(*Lua-table*, *SAS-data-set-name*)
Lua テーブルから SAS データセットを作成します。SAS データセットには、Work.Random などの 2 レベルの名前を指定できます。Lua テーブルは、SAS.LOAD_DS 関数で返される構造に従っている必要があります。

データセットオブザベーションの処理

オブザベーションを処理する関数

次の Lua 関数を使用して、データセット内の個別のオブザベーションを処理できます。まず、更新モード('u')でデータセットを開く必要があります。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

SAS.APPEND(*data-set-ID*)
新しく作成されたオブザベーションをデータセットに追加します。

SAS.DELOBS(*data-set-ID*)
データセット内の現在のオブザベーションを削除します。

SAS.GET_VALUE(*data-set-ID*, *variable-number* | *variable-name*)
現在のオブザベーションで指定された変数の値を返します。データセット内の位置(数値)または名前で変数を特定します。

SAS.NEXT(*data-set-ID*)
処理のためにデータセット内の次のオブザベーションに移動します。データセットの処理を開始していない場合、SAS.NEXT 関数は、データセット内の最初のオブザベーションに移動します。SAS.NEXT 関数を使用すると、SAS データセットで直接操作できます。最初にデータを Lua テーブルに読み込む必要がありません。この関数は、大きなデータセットで有用です。

SAS.PUT(*data-set-ID*, *variable-number*, *value*)
データセットの指定された変数に値をロードします。データセット内の変数を位置(数値)で特定します。

Alias:SAS.PUT_VALUE(*data-set-ID*, *variable-number*, *value*)

SAS.ROWS(*data-set-ID*)
データセット内のオブザベーションを反復し、各行を処理のために Lua テーブルにロードして、処理の終了時に Lua nil を追加します。この関数は、変数が比較的少ないデータセットで有用です。

SAS.UPDATE(*data-set-ID*)
SAS.PUT 関数を呼び出して追加された値でオブザベーションを更新します。

SAS.VARS(*data-set-ID*)
データセットの変数を反復します。

オブザベーションを追加する順序

新しいオブザベーションを追加するには、オブザベーション処理関数を次の順序で呼び出します。

1. SAS.APPEND。
2. SAS.PUT。オブザベーションの必要な変数すべてに値が設定されるまで、次の関数呼び出しを繰り返します。
3. SAS.UPDATE。

範囲外の警告

データセット ID(ハンドル)が範囲外になると、関連する SAS データセットは自動的に閉じます(まだ閉じていない場合)。SAS ログには、次のような警告が表示されます。

```
WARNING: Closing SASHELP.CLASS - handle has gone out of scope.
```

プログラムがアクセスできなくなると、そのデータセット ID は範囲外になります。たとえば、データセット ID が、ユーザー定義関数でローカル変数として定義されている場合、データセット ID は、その関数定義の最後で範囲外になります。最良の方法として、データセット ID が範囲外になる前にデータセットを閉じてください。

LUA プロシジャのシステム関数

次の関数は、サブミットされた SAS コードの動作を LUA プロシジャ内から制御します。詳細については、“Lua ステートメントでの SAS コードのサブミット” (1059 ページ)を参照してください。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

SAS.GET_MAX_SYSERR()
SAS ログでエラーをトリガせずに、SAS.SUBMIT への呼び出しから返すことができる現在の SYSERR の最大値を返します。

SAS.IS_QUIET()
SAS.SET_QUIET 関数を使用して設定された値に応じて、**true** または **false** を返します。このブール値は、SAS.SUBMIT 関数にサブミットされた SAS 言語ステートメントが、SAS ログに書き込まれたかどうかを示します。

SAS.SET_MAX_ERR(*value*)
SAS.SUBMIT 呼び出しによって返すことができる SYSERR の最大許容値を指定します。SAS が、この値よりも大きい SYSERR 値を返した場合は、エラーが SAS ログに出力されます。デフォルト値はゼロで、可能な値には正の整数が含まれます。

SAS.SET_QUIET(*value*)
SAS.SUBMIT 関数にサブミットされた SAS 言語ステートメントが、SAS ログに書き込まれるかどうかを指定します。

可能な引数値は **true** または **false** です。デフォルト値は **false** です。

Lua 拡張機能:テーブルライブラリ

LUA プロシジャがアクセスできる、テーブル関数がいくつかあります。こうした関数は、通常、Lua テーブルライブラリにあります。このテーブル関数を呼び出すには、**table.size(t)** のように、'TABLE'を前に付けます。LUA プロシジャからアクセスできるテーブル関数を次に示します。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

TABLE.CONTAINS(*Lua-table-name*,*v*)
指定されたテーブルに値 *v* が含まれる場合は、ブール値 **true** を返します。文字値を一重引用符または二重引用符で囲みます。

TABLE.SIZE(*Lua-table-name*)
指定されたテーブルの要素の数を返します。

TABLE.TOSTRING(*Lua-table-name*)
指定されたテーブルの出力形式が定義された文字列表現を返します。

詳細については、“例 2: SAS データセットのロードと、結果として生成される Lua テーブルの表示” (1070 ページ) および“例 4: TABLE ライブラリ関数の使用” (1074 ページ)を参照してください。

Lua ステートメントでの PROC FCMP 関数の呼び出し

Lua コード内で FCMP プロシジャを使用して作成された関数をサブミットできます。PROC FCMP 関数を呼び出す場合は、関数が保存されているパッケージを、SAS OPTIONS ステートメントで指定する必要があります。

PROC FCMP 関数によって引数のいずれかが変更された場合、その引数は、OUTARGS ステートメントで指定されています。OUTARGS ステートメントの引数に対する変更を Lua コードで取得するには、その引数は、配列として定義されている必要があります。

詳細については、次の情報を参照してください。

- 22 章, “FCMP プロシジャ,” (694 ページ)
- “例 11: PROC FCMP 関数の実行” (1087 ページ)

Lua ステートメントのオブジェクト構文

データセット ID など、一部のオブジェクトについては、Lua コードによってオブジェクト構文がサポートされており、関数の前で、その関数の処理対象となるオブジェクトを指定します。たとえば、次の Lua ステートメントは同等です。

```
local luavar = sas.get_value(dsid, 'some_var')

local luavar = dsid:get_value('some_var')
```

関数の処理対象のオブジェクトの名前を、その関数の前に配置する場合は、必ずコロン(:)を使用してください。

注: LUA プロシジャでは、**os.date()**、**os.clock()**など、Lua OS ライブラリ内の関数を参照するオブジェクト構文はサポートされません。ただし、ほとんどの場合、対応する SAS 関数を呼び出すことができます。

同様に、“例 9: 大きなテーブルにイテレータ関数を使用” (1083 ページ)の次のコードブロックは、2 つの方法で書き込まれる可能性があります。

```
-- Iterate over the rows of the data set
local i=0
while sas.next(dsid) do
   i=i+1
   print("OBS=" .. i)
   for vname,var in pairs(vars) do
      print(vname, '=', sas.get_value(dsid, vname) )
   end
end
```

SAS.NEXT 関数と SAS.GET_VALUE 関数は両方とも、オブジェクト構文で表すことができます。

```
-- Iterate over the rows of the data set
local i=0
while dsid:next() do
   i=i+1
   print("OBS=" .. i)
   for vname,var in pairs(vars) do
      print(vname, '=', dsid:get_value(vname) )
   end
end
```

構文: LUA プロシジャ

PROC LUA <INFILE='*filename*'> <RESTART> <TERMINATE>;
<SUBMIT <"*assignment(s);*">;>
Lua statements
<ENDSUBMIT;>
run;

ステートメント	タスク
“PROC LUA ステートメント”	SAS コード内で Lua ステートメントを実行するか、実行する Lua ステートメントが含まれるファイルを指定します
“SUBMIT ステートメント”	Lua ステートメントのブロックの先頭を特定します
“ENDSUBMIT ステートメント”	Lua ステートメントのブロックの末尾を特定します

PROC LUA ステートメント

SAS セッションで Lua ステートメントを実行します。

構文

PROC LUA <INFILE='*filename*'> <RESTART> <TERMINATE>;

オプション引数

INFILE= '*filename*'
SAS セッションで実行する Lua ステートメントが含まれるソースファイルを特定します。SAS では、このファイル名の末尾には `.lua` ファイル拡張子が付いているものとされますが、この拡張子は、指定するファイル名には含まれていません。

要件 ファイル名を一重引用符または二重引用符で囲みます。

INFILE=オプションを使用する場合は、Lua スクリプトへのパスを指定する必要があります。PROC LUA ステートメントの前で LUAPATH ファイル名の値を指定して、Lua スクリプトへのパスを定義します。詳細については、“例 1: 外部 Lua スクリプトからの入力の指定” (1069 ページ)を参照してください。

例 INFILE='*open_data*'を指定して、open_data.lua Lua スクリプトでコードを使用します。

RESTART
SAS セッションの Lua コードサブミットの状態をリセットします。LUA プロシジャは、LUA プロシジャへの呼び出し間で Lua コードの状態が維持されるリエントラントプロシジャです。つまり、RESTART オプションまたは TERMINATE オプションを発行するか、SAS セッションを終了するまで、Lua グローバル変数割り当てまたは関数定義がメモリに残ります。RESTART は、Lua コードの新しいブロックの先頭で指定できます。

例
```
proc lua restart;
submit;
  <lua statements...>
endsubmit;
run;
```

TERMINATE
LUA プロシジャの完了時に、メモリへの Lua コード状態の保持を停止し、Lua 状態を終了します。以降、LUA プロシジャを呼び出すと、Lua コード状態の新しいインスタンスが開始されます。

SUBMIT ステートメント

Lua コードのブロックの先頭を特定します。SUBMIT ステートメントと ENDSUBMIT ステートメントの間に、Lua ステートメントを入力します。

要件 各 SUBMIT ステートメントには、対応する ENDSUBMIT ステートメントが必要です。

構文

SUBMIT <'*assignment(s);*'>;

オプション引数

assignment(s)
Lua ステートメントのブロックに渡された 1 つ以上のマクロ変数割り当てを特定します。割り当てが 1 つだけの場合は、引用符内のセミコロン(;)は不要です。SAS では、Lua ステートメントのブロック内ではマクロ変数が拡張されません。したがって、SUBMIT ステートメントの割り当てリスト内でマクロ値を渡す必要があります。

例 マクロ変数の値 N を Lua 変数 Name に割り当てるには、次の SUBMIT ステートメントを入力します。

```
SUBMIT "name=&n";
```

ENDSUBMIT ステートメント

Lua ステートメントのブロックの末尾を特定します。

構文

ENDSUBMIT;

SAS プログラム内での Lua ステートメントのサブミット

PROC LUA 起動内の、SUBMIT ステートメントと ENDSUBMIT ステートメントの間で Lua ステートメントをサブミットできます。次のコードは、1 つの Lua print ステートメントを実行します。

```
proc lua;
submit;
   print("Hello from Lua")
endsubmit;
run;
```

Lua コード内で SAS データセットを開く

SAS.SUBMIT 関数を呼び出すと、SAS コードのブロックをサブミットできます。SAS コードは、角かっこ([[および]])で囲みます。このサンプルでサブミットされたコードは、最初に、SAS データセットが存在するかどうかを確認します。サブミットされた DATA ステップを介して、Base SAS コードで行うようにデータを変更できます。

```
/* Test whether a data set exists */
proc lua;
submit;
   if sas.exists("sashelp.air") then
      print("The data set SASHELP.AIR exists.")
   else
      print("The data set SASHELP.AIR does not exist.")
   end
endsubmit;
run;
```

データセットが存在する場合は、データセットを開いて、それを新しいデータセット WORK.AIR に読み込むことができます。サブミットされた DATA ステップを介して、標準の DATA ステップで行うようにデータを変更できます。

注: 通常、より長い SAS コードブロックが Lua 変数に割り当てられます。

```
proc lua;
submit;
   sas.submit( [[ data work.air; set sashelp.air; run; ]] )
endsubmit;
run;
```

また、SAS コード内で Lua 変数値を代入することもできます。次のコードは、シンプルな代入を示しています。詳細については、“例 6: Lua 変数代入による SAS コードのサブミット” (1077 ページ)を参照してください。

```
proc lua;
submit;
   local dest = 'work.class'
   local source = 'sashelp.class'

   sas.submit( [[ data @dest@; set @source@; run; ]] )
endsubmit;
run;
```

例: LUA プロシジャ

例 1: 外部 Lua スクリプトからの入力の指定

要素: FILENAME ステートメント
PROC LUA ステートメント、INFILE=オプション

詳細

この例では、FILENAME ステートメントと、PROC LUA ステートメントの INFILE=オプションを使用して、外部 Lua スクリプトと、そのスクリプトへの考えられるパスを 1 つ以上指定します。指定するディレクトリが 1 つの場合は、FILENAME ステートメントのかっことカンマは省略できます。ディレクトリパスは必ず一重引用符または二重引用符で囲みます。

プログラム

```
filename LUAPATH ('/usr/local/scripts/lua','/home/user/myname/my_scripts');

proc lua infile='my_script';
run;
```

プログラムの説明

入力 Lua スクリプトを検索するディレクトリを指定します。 FILENAME ステートメントは、Lua スクリプトが保存されているディレクトリを定義し、それを LUAPATH ファイル参照名に割り当てます。SAS は、ユーザーが指定した順序でパスを検索します。指定するディレクトリが 1 つの場合は、かっことカンマを省略できます。ディレクトリパスは一重引用符または二重引用符で囲みます。

```
filename LUAPATH ('/usr/local/scripts/lua','/home/user/myname/my_scripts');
```

PROC LUA ステートメントを実行し、Lua スクリプト名を指定します。 PROC LUA ステートメントを使用すると、SAS セッションで Lua コードを呼び出すことができます。INFILE=オプションは、Lua ステートメントが含まれる Lua スクリプトの名前を指定します。この例では、SAS が my_script.lua ファイルの Lua ステートメントを実行します。'.lua'または'.luc'のファイル拡張子は使用しないでください。システムの package.path 変数を確認して、最初に LUA ファイルまたは LUC ファイルが開いているかどうかを確かめます。

```
proc lua infile='my_script';
run;
```

例 2: SAS データセットのロードと、結果として生成される Lua テーブルの表示

要素: SAS.LOAD_DS 関数
TABLE.TOSTRING 関数

詳細

この例では、Sashelp.Fish データセットをロードして出力します。この例は、SAS データセットを読み込んで、データを Lua テーブルとして出力しています。個別のオブザベーションを、配列のエントリとして処理できます。各配列のエントリには、元の SAS データセットの変数から派生する、関連付けられた属性が含まれます。たとえば、出力セクションでは、Fish[3].weight が 340 であることを確認できます。

プログラム

```
proc lua;
submit;
   local fish=sas.load_ds("sashelp.fish")
   print("fish=", table.tostring(fish))
```

```
endsubmit;
run;
```

プログラムの説明

Sashelp ライブラリの SAS データセット Fish を開き、そのデータを Lua テーブルとして出力します。

```
proc lua;
submit;
   local fish=sas.load_ds("sashelp.fish")
   print("fish=", table.tostring(fish))
endsubmit;
run;
```

出力:Lua テーブル

アウトプット 36.1 *サンプル Lua テーブル*

```
1    proc lua;
2    submit;
3       local fish=sas.load_ds("sashelp.fish")
4       print("fish=", table.tostring(fish))
5    endsubmit;
6    run;

NOTE: Lua initialized.
fish=    table: 0BE3DA58=
{
  [1]=table: 0BE3DAA8=
  {
    ["height"]=11.52
    ["length1"]=23.2
    ["weight"]=242
    ["length2"]=25.4
    ["species"]="Bream"
    ["length3"]=30
    ["width"]=4.02
  }
  [2]=table: 0BE3DE98=
  {
    ["height"]=12.48
    ["length1"]=24
    ["weight"]=290
    ["length2"]=26.3
    ["species"]="Bream"
    ["length3"]=31.2
    ["width"]=4.3056
  }
  [3]=table: 0BE3DF28=
  {
    ["height"]=12.3778
    ["length1"]=23.9
    ["weight"]=340
    ["length2"]=26.5
    ["species"]="Bream"
    ["length3"]=31.1
    ["width"]=4.6961
  }
  [4]=table: 0BE3DF78=
  {
    ["height"]=12.73
    ["length1"]=26.3
    ["weight"]=363
    ["length2"]=29
    ["species"]="Bream"
    ["length3"]=33.5
```

例 3: Lua テーブルからの SAS データセットの書き込み

要素: PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント
SAS.WRITE_DS 関数
TABLE.TOSTRING 関数

詳細

この例では、Lua テーブルを作成し、そのテーブルを SAS データセットに書き込みます。Lua テーブルの値は、SAS RANNOR および RANUNI 乱数ジェネレータ関数を呼び出すことで生成されます。このコードは、配列を処理するための Lua 規則も使用しています。

プログラム

```
proc lua;
submit;
   local tbl = {}


   for i=1,10 do
       vars = {}
       vars.seed = 1234 * i;
       vars.randnor = sas.rannor( vars.seed )
       vars.randuni = sas.ranuni( vars.seed )
       vars.color = "purple"
       tbl[#tbl+1] = vars
   end


   print("Lua table:", table.tostring(tbl))


   sas.write_ds(tbl, "work.random")
endsubmit;
run;


proc print data=random;run;
```

プログラムの説明

PROC LUA ステートメントを実行して、Lua コードのブロックを開始します。 **TBL** と呼ばれるローカル Lua 配列を宣言します。

```
proc lua;
submit;
   local tbl = {}
```

Lua テーブルのコンテンツを生成します。 この例では、配列 **TBL** のコンテンツを生成します。変数 Seed、Randnor、Randuni、Color が作成され、FOR ループの 10 の反復にわたって値が割り当てられます。

```
   for i=1,10 do
       vars = {}
       vars.seed = 1234 * i;
       vars.randnor = sas.rannor( vars.seed )
       vars.randuni = sas.ranuni( vars.seed )
       vars.color = "purple"
       tbl[#tbl+1] = vars
   end
```

生成された Lua テーブルを出力します。 結果として生成される Lua テーブルは、SAS ログに出力されます。

```
print("Lua table:", table.tostring(tbl))
```

Lua テーブルを SAS データセットに書き込みます。 この例では、Lua テーブルを、Work ライブラリの Random と呼ばれる SAS データセットに書き込みます。Work ライブラリに保存したデータセットには、同じ SAS セッションでのみアクセスできます。データセットを永久に保存するには、そのデータセットを、Sasuser などのライブラリに保存します。

```
sas.write_ds(tbl, "work.random")
endsubmit;
run;
```

SAS データセットを出力します。 SAS データセットを書き込んだ後、その SAS データセットには、LUA プロシジャ外でアクセスできます。データセットを Sasuser などのライブラリに保存すると、そのデータセットには、後の SAS セッションでもアクセスできます。

```
proc print data=random;run;
```

出力:Lua テーブルの SAS テーブル

アウトプット 36.2 *SAS ログの Lua テーブル*

```
1    proc lua;
2    submit;
3       local tbl = {}
4
5       for i=1,10 do
6           vars = {}
7           vars.seed = 1234 * i;
8           vars.randnor = sas.rannor( vars.seed )
9           vars.randuni = sas.ranuni( vars.seed )
10          vars.color = "purple"
11          tbl[#tbl+1] = vars
12      end
13
14      print("Lua table:", table.tostring(tbl))
15
16      sas.write_ds(tbl, "work.random")
17   endsubmit;
18   run;

NOTE: Lua initialized.
Lua table:    table: 0BEEA038=
{
  [1]=table: 0BEEA5F8=
  {
    ["randuni"]=0.3831937143
    ["color"]="purple"
    ["randnor"]=1.4215132075
    ["seed"]=1234
  }
  [2]=table: 0BEEB208=
  {
    ["randuni"]=0.088249594
    ["color"]="purple"
    ["randnor"]=-0.102644377
    ["seed"]=2468
  }
  [3]=table: 0BEEAFD8=
  {
    ["randuni"]=0.4458283407
    ["color"]="purple"
    ["randnor"]=2.0089305781
    ["seed"]=3702
  }
  [4]=table: 0BEEB258=
  {
    ["randuni"]=0.4562234927
    ["color"]="purple"
    ["randnor"]=1.9125666228
```

アウトプット 36.3 生成された SAS テーブル

The SAS System

Obs	color	randuni	randnor	seed
1	purple	0.38319	1.42151	1234
2	purple	0.08825	-0.10264	2468
3	purple	0.44583	2.00893	3702
4	purple	0.45622	1.91257	4936
5	purple	0.96744	1.82697	6170
6	purple	0.37316	-0.63993	7404
7	purple	0.28390	-1.49036	8638
8	purple	0.43640	-0.05252	9872
9	purple	0.00276	-0.45912	11106
10	purple	0.23450	0.96227	12340

例 4: TABLE ライブラリ関数の使用

要素: PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント
TABLE.SIZE 関数
TABLE.TOSTRING 関数

詳細

この例では、単純なテーブルを作成し、要素の数を出力して、テーブルを SAS ログに出力します。

プログラム

```
proc lua;
submit;

  local t={"a", "b", "c", f="foo",b="bar"}

  print(table.size(t))
  print(table.tostring(t))

endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントを実行します。 PROC LUA ステートメントを使用すると、SAS セッションで Lua コードを呼び出すことができます。

```
proc lua;
```

Lua ステートメントのブロックの先頭を特定します。 SUBMIT ステートメントは、Lua ステートメントのブロックの先頭を特定します。

```
submit;
```

単純なテーブルを作成します。 要素を単純な 1 行テーブル t に割り当てます。

```
local t={"a", "b", "c", f="foo",b="bar"}
```

テーブルに関する情報を出力します。 テーブル内の要素の数を出力し、TABLE ライブラリの関数を使用してテーブルを出力します。

```
print(table.size(t))
print(table.tostring(t))
```

ENDSUBMIT ステートメントを使用して、Lua ステートメントのブロックの末尾を特定します

```
endsubmit;
run;
```

出力:テーブルの詳細が表示された SAS ログ

アウトプット 36.4 TABLE 関数の結果

```
19   proc lua;
20   submit;
21
22     local t={"a", "b", "c", f="foo",b="bar"}
23
24     print(table.size(t))
25     print(table.tostring(t))
26   endsubmit;
27   run;

NOTE: Resuming Lua state from previous PROC LUA invocation.
5
table: 0878D1F8=
{
  [1]="a"
  [2]="b"
  [3]="c"
  ["b"]="bar"
  ["f"]="foo"
}
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

例 5: Lua ステートメントでの SAS マクロ変数の使用

要素: PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント

マクロ変数割り当て
連結演算子(..)

詳細

SUBMIT ステートメントの末尾にあるセミコロン(;)と ENDSUBMIT ステートメントの先頭の間では、マクロ代入が発生しません。マクロ値の割り当てを SUBMIT ステートメント内で指定する必要があります。この例では、Lua コード内で使用するためにマクロ変数値を割り当てる方法を示します。

複数の割り当てがある場合はそれぞれをセミコロン(;)で区切り、すべての割り当てを 1 つの引用符で囲みます。この例では、次のテキストを SAS ログに出力します。

```
Hello, George.Have a nice day.
```

プログラム

```
%let g='George';
%let h='Have a nice day.';

proc lua;

submit "name=&g; msg=&h";

   print('Hello, ' .. name .. '. ' .. msg)

endsubmit;
run;
```

プログラムの説明

マクロ変数値を定義します。 マクロ変数 G および H は、文字値に割り当てられます。Lua では、マクロ変数割り当てに一重引用符を使用する必要があります。SAS のマクロ変数割り当ての場合、通常、一重引用符は不要です。

```
%let g='George';
%let h='Have a nice day.';
```

PROC LUA ステートメントを実行します。 PROC LUA ステートメントを使用すると、SAS セッションで Lua コードを呼び出すことができます。

```
proc lua;
```

Lua ステートメントのブロックの先頭を特定し、任意のマクロ変数を割り当てます。 SUBMIT ステートメントは、Lua ステートメントのブロックの先頭を特定します。この例では、マクロ変数 **g** および **h** の値を、ローカル Lua 変数 **name** および **msg** にそれぞれ割り当てます。SUBMIT ステートメントの末尾と ENDSUBMIT ステートメントの先頭の間では、マクロの展開が発生しません。Lua コードは、ローカル Lua 変数を参照します。

```
submit "name=&g; msg=&h";
```

Lua ステートメントを実行します。 SUBMIT ステートメントと ENDSUBMIT ステートメントの間に、Lua ステートメントを入力します。Lua ステートメントはセミコロン(;)で終了する必要はありません。この例では、テキスト値を SAS ログに出力します。'..'演算子を使用して文字列を連結します。ログでは、使用する print ステートメントはそれぞれ改行されて、表示されます。

```
   print('Hello, ' .. name .. '. ' .. msg)
```

ENDSUBMIT ステートメントを使用して、Lua ステートメントのブロックの末尾を特定します

```
endsubmit;
run;
```

例 6: Lua 変数代入による SAS コードのサブミット

要素: PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント
SAS.SUBMIT 関数
変数代入

詳細

この例では、Lua ステートメントのブロックをサブミットします。Lua ステートメント内で、SAS コードのブロックをローカル変数に割り当てます。その後、SAS コードを SAS.SUBMIT 関数で呼び出して、変数値を代入します。

この例では、Work.Answer データセットを、ローカル Lua 変数コードへの入力として使用します。

図 36.1 *Work.Answer データセット*

The SAS System

Obs	CCUID	Value
1	60	12
2	61	27
3	62	5
4	62	8
5	67	9
6	67	4

プログラム

```
proc lua;

submit;

   local rc

   local code = [[
         data sample; set answer;
         where CCUID = @ccuid@;
         y = @subValue@;
         run;
   ]]

   rc = sas.submit(code, {ccuid="67", subValue=72})
```

```
endsubmit;
run;


proc print data=sample;
run;
```

プログラムの説明

LUA プロシジャを実行します。

```
proc lua;
```

実行する Lua ステートメントのブロックの先頭を特定します。 SUBMIT ステートメントを使用して、Lua ステートメントを特定します。ローカル変数 Rc を宣言します。Lua ステートメントの末尾にセミコロン(;)は不要です。

```
submit;

   local rc
```

SAS コードのブロックをローカル Lua 変数に割り当てます。 ローカル Lua 変数コードに、角かっこ([[および]])で示されている SAS コードのブロックが割り当てられています。SAS コードは、WORK.ANSWER データセットを開き、CCUID の値が、@ccuid@によって指定されている値と一致するレコードのみを保持します。SAS コードが実行されると、値がキー**@ccuid@**に割り当てられます。新しい変数 Y は、**@subValue@**によって指定されている値に割り当てられます。結果データセットは、WORK.SAMPLE データセットに保存されます。

```
local code = [[
      data sample; set answer;
      where CCUID = @ccuid@;
      y = @subValue@;
      run;
]]
```

SAS コードを実行し、結果を Lua 変数 Rc に割り当てます。 SAS.SUBMIT 関数は、SAS コードを実行するようシステムに指示します。これにより、Code 変数に割り当てられたコードブロックが実行されます。SAS.SUBMIT 関数への呼び出しには 2 つの変数代入が含まれます。SAS コードが実行されたら、文字値*"67"*は**@ccuid@**に代入され、値 *72* は**@subValue@**に代入されます。

```
rc = sas.submit(code, {ccuid="67", subValue=72})
```

ENDSUBMIT ステートメントは、Lua ステートメントのブロックの末尾を特定します。 RUN ステートメントによって、PROC LUA への呼び出しが完了します。

```
endsubmit;
run;
```

結果データセット Work.Sample を出力します。

```
proc print data=sample;
run;
```

出力:代入された Lua 値の結果 SAS テーブル

アウトプット 36.5 *生成された Work.Sample テーブル*

The SAS System

Obs	CCUID	Value	y
1	67	9	72
2	67	4	72

例 7: Lua コードの配列の操作

要素: 配列操作
Lua の DO ループ
連結演算子(..)

詳細

この例では、Lua コード内の DO ループを使用して、SAS で配列を作成し、その配列にアクセスします。また、SAS コードを生成し、そのコードを、Lua ステートメントのブロックの末尾でサブミットします。

プログラム

```
data temp(drop=i);
   array x(*) var1-var10;
   do i=1 to dim(x);
      x(i)=i;
   end;
   output;
run;


proc lua;
submit;
   sascode="proc print data=temp; var "
   for i=1,5 do
      sascode=sascode .. "var" .. i .. " "
   end
   sascode=sascode .. "; run;"
   sas.submit(sascode);
endsubmit;
run;
```

プログラムの説明

一時データセットに配列を生成します。 配列 X は、10 の変数(Var1～Var10)で構成され、その変数はそれぞれ、1～10 の連続する値に設定されます。

```
data temp(drop=i);
   array x(*) var1-var10;
   do i=1 to dim(x);
      x(i)=i;
   end;
   output;
run;
```

PROC LUA ステートメントを実行して、SAS コードを生成します。 Lua 変数 Sascode に、SAS コードが割り当てられます。配列 X に割り当てられた変数に基づいて、追加コードが既存のコードに連結されます。DO ループの完了後は、さらに SAS コードが既存の Sascode 変数に追加されます。最後に、生成されたコードが SAS にサブミットされます。

```
proc lua;
submit;
   sascode="proc print data=temp; var "
   for i=1,5 do
      sascode=sascode .. "var" .. i .. " "
   end
   sascode=sascode .. "; run;"
   sas.submit(sascode);
endsubmit;
run;
```

出力:配列の操作

アウトプット 36.6 *SAS ログの生成された Lua コード*

```
174  data temp(drop=i);
175     array x(*) var1-var10;
176     do i=1 to dim(x);
177        x(i)=i;
178     end;
179     output;
180  run;

NOTE: The data set WORK.TEMP has 1 observations and 10 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.03 seconds


181
182  proc lua;
183  submit;
184     sascode="proc print data=temp; var "
185     for i=1,5 do
186        sascode=sascode .. "var" .. i .. " "
187     end
188     sascode=sascode .. "; run;"
189     sas.submit(sascode);
190  endsubmit;
191  run;

NOTE: Resuming Lua state from previous PROC LUA invocation.
proc print data=temp; var var1 var2 var3 var4 var5 ; run;
NOTE: There were 1 observations read from the data set WORK.TEMP.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds


NOTE: PROCEDURE LUA used (Total process time):
      real time           0.04 seconds
      cpu time            0.03 seconds
```

アウトプット 36.7 *Work.Temp からの最初の 5 つの変数出力*

The SAS System

Obs	var1	var2	var3	var4	var5
1	1	2	3	4	5

例 8: 小さなテーブルにイテレータ関数を使用

要素: SAS.OPEN 関数
SAS.ROWS 関数
SAS.CLOSE 関数

詳細

この例では、データセットを走査し、変数名と対応する値をオブザベーションごとに出力します。SAS.ROWS 関数は、変数が比較的少ない場合に使用します。

プログラム

```
proc lua;
submit;

   local dsid = sas.open("sashelp.class")

   for row in sas.rows(dsid) do
      for n,v in pairs(row) do
         if type(n)=="string" then
            print(n,'=', v)
         end
      end
   end

   sas.close(dsid)
endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントと SUBMIT ステートメントを実行して、Lua コードのブロックを開始します。

```
proc lua;
submit;
```

Sashelp.Class SAS データセットを開いて、コンテンツを Lua 変数 Dsid にロードします。 デフォルトでは、データセットは読み込みのために開かれます。

```
   local dsid = sas.open("sashelp.class")
```

データセットの行を処理します。 外部 FOR ループは、データセット内の各行を処理します。PAIRS 関数は、変数名と値のペアを Sashelp.Class データセットの各行から取り出します。その後、変数名と値のペアは SAS ログに出力されます。Lua では、IF 条件で 2 つの値が等しいかどうかを評価するときに、2 つの等号(==)が使用されることに注意してください。

```
   for row in sas.rows(dsid) do
      for n,v in pairs(row) do
         if type(n)=="string" then
            print(n,'=', v)
         end
      end
   end
```

データセットを閉じて、Lua コードのブロックを終了し、LUA プロシジャへの呼び出しを完了します。

```
   sas.close(dsid)
endsubmit;
run;
```

出力:小さなデータセット用の ROWS イテレータ

アウトプット 36.8 *ROWS 関数で小さなデータセットを反復(部分出力)*

```
208  proc lua;
209  submit;
210     local dsid = sas.open("sashelp.class")
211        for row in sas.rows(dsid)
212           do for n,v in pairs(row) do
213              if type(n)=="string" then
214                 print(n,'=', v)
215              end
216           end
217        end
218
219     sas.close(dsid)
220  endsubmit;
221  run;

NOTE: Resuming Lua state from previous PROC LUA invocation.
name     =    Alfred
sex      =    M
height   =    69
age      =    14
weight   =    112.5
name     =    Alice
sex      =    F
height   =    56.5
age      =    13
weight   =    84
name     =    Barbara
sex      =    F
height   =    65.3
age      =    13
weight   =    98
name     =    Carol
sex      =    F
height   =    62.8
age      =    14
weight   =    102.5
name     =    Henry
sex      =    M
height   =    63.5
age      =    14
weight   =    102.5
name     =    James
sex      =    M
height   =    57.3
age      =    12
weight   =    83
name     =    Jane
sex      =    F
height   =    59.8
age      =    12
weight   =    84.5
name     =    Janet
sex      =    F
height   =    62.5
```

例 9: 大きなテーブルにイテレータ関数を使用

要素: SAS.OPEN 関数
SAS.VARS 関数
SAS.NEXT 関数
SAS.GET_VALUE 関数
SAS.CLOSE 関数

詳細

この例では、SAS.VARS 関数と SAS.NEXT 関数を使用して、SAS データセットを走査し、そのコンテンツを出力します。

プログラム

```
proc lua;
submit;
```

```
   local dsid = sas.open("sashelp.company") -- open for input

   local vars = {}
   -- Iterate over the variables in the data set
   for var in sas.vars(dsid) do
      vars[var.name:lower()] = var
   end

   -- Iterate over the rows of the data set
   local i=0
   while sas.next(dsid) do
      i=i+1
      print("OBS=" .. i)
      for vname,var in pairs(vars) do
         print(vname, '=', sas.get_value(dsid, vname) )
      end
   end

   sas.close(dsid)
endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントと SUBMIT ステートメントを実行して、Lua コードのブロックを開始します。

```
proc lua;
submit;
```

Sashelp.Company データセットを開きます。

```
   local dsid = sas.open("sashelp.company") -- open for input
```

ローカル Lua 配列 VARS を宣言し、その配列に、データセットの変数の値を入力します。 波かっこ({ })は、Lua の配列を特定します。SAS.VARS 関数を使用して、データセット内のすべての変数を反復します。この例では、各変数の値を Vars 配列に割り当てます。ここで、配列キーは小文字の変数名です。

```
   local vars = {}
   -- Iterate over the variables in the data set
   for var in sas.vars(dsid) do
      vars[var.name:lower()] = var
   end
```

各データセットオブザベーションを反復し、変数ごとに値を出力します。 この例では、イテレータ変数 I を 0 に初期化します。その後、SAS.NEXT イテレータ関数は、データセットのオブザベーションを繰り返します。この例は、オブザベーションごとに、変数名と各変数に対応する値を出力しています。

```
   -- Iterate over the rows of the data set
   local i=0
   while sas.next(dsid) do
      i=i+1
      print("OBS=" .. i)
      for vname,var in pairs(vars) do
         print(vname, '=', sas.get_value(dsid, vname) )
      end
```

```
    end
```

データセットを閉じて、LUA プロシジャへの呼び出しを終了します。

```
    sas.close(dsid)
endsubmit;
run;
```

出力:大きなデータセットでのイテレータ

アウトプット 36.9 *大きなデータセットでイテレータ関数を使用*

```
222  proc lua;
223  submit;
224     local dsid = sas.open("sashelp.company") -- open for input
225     local vars = {}
226
227     -- Iterate over the variables in the data set
228     for var in sas.vars(dsid) do
229        vars[var.name:lower()] = var
230     end
231
232     -- Iterate over the rows of the data set
233     local i=0
234     while sas.next(dsid) do
235        i=i+1
236        print("OBS=" .. i)
237        for vname,var in pairs(vars) do
238           print(vname, '=', sas.get_value(dsid, vname) )
239        end
240     end
241
242     sas.close(dsid)
243  endsubmit;
244  run;

NOTE: Resuming Lua state from previous PROC LUA invocation.
OBS=1
n     =     1
level2     =     TOKYO
level1     =     International Ai
job1     =     MANAGER
level3     =     ADMIN
level5     =     So Suumi
level4     =     CONTRACTS
depthead     =     1
OBS=2
n     =     1
level2     =     TOKYO
level1     =     International Ai
job1     =     ASSISTANT
level3     =     ADMIN
level5     =     Steffen Graff
level4     =     CONTRACTS
depthead     =     2
OBS=3
n     =     1
level2     =     TOKYO
level1     =     International Ai
job1     =     ACCOUNTANT
level3     =     ADMIN
level5     =     Karin Schmidt
level4     =     FINANCE
depthead     =     2
OBS=4
n     =     1
level2     =     LONDON
level1     =     International Ai
job1     =     MANAGER
level3     =     ADMIN
```

例 10: 変数の定義と、データセットへの変数の追加

要素: PROC LUA ステートメント
SAS.OPEN 関数と'o'オプション
SAS.ADD_VARS 関数
SAS.CLOSE 関数

詳細

この例では、Lua コード内で新しい変数を定義し、その変数をデータセットに追加します。この例は、SAS.ADD_VARS 関数を使用して変数を定義し、出力データセットに追加しています。

プログラム

```
proc lua;
submit;

   local dsid = sas.open("work.sample","o")

   sas.add_vars(dsid, {  {name="var1", type="N", format="BEST12.2"},
                         {name="var2", type="C", length=500, format="$char80."},
                         {name="var3", type="C", label="Character Data"}
                      })

   sas.close(dsid)
endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントを実行して、Lua ステートメントのブロックを開始します。 PROC LUA ステートメントを使用すると、SAS セッションで Lua コードを呼び出すことができます。SUBMIT ステートメントは、Lua ステートメントのブロックの先頭を特定します。

```
proc lua;
submit;
```

データセットを開いて、そのコンテンツをローカル Lua 変数に割り当てます。 SAS.OPEN 関数を呼び出して、WORK.SAMPLE データセットを開きます。データセットが存在しない場合は、'o'モードを使用して作成します。

```
   local dsid = sas.open("work.sample","o")
```

変数を定義して、データセットに追加します。 この例では、SAS.ADD_VARS 関数を呼び出して、3 つの変数(Var1、Var2、Var3)をデータセットに追加します。種類(数値(N)または文字(C))は、変数ごとに指定されています。変数には、さまざまな追加属性が割り当てられます。名前属性のみが必須です。詳細については、“LUA プロシジャのデータセット関数” (1061 ページ)を参照してください。

```
   sas.add_vars(dsid, {  {name="var1", type="N", format="BEST12.2"},
                         {name="var2", type="C", length=500, format="$char80."},
                         {name="var3", type="C", label="Character Data"}
                      })
```

データセットを閉じて、LUA プロシジャへの呼び出しを終了します。 この例では、SAS.CLOSE 関数を実行して、Lua 変数 Dsid に関連付けられているデータセットを閉じます。

```
   sas.close(dsid)
endsubmit;
run;
```

出力:データセットへの変数の追加

アウトプット 36.10 *Work.Sample の PROC CONTENTS からの部分出力*

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
1	var1	Num	8	BEST12.2	
2	var2	Char	500	$CHAR80.	
3	var3	Char	200		Character Data

例 11: PROC FCMP 関数の実行

要素: PROC FCMP ステートメント
PROC LUA ステートメント
FILENAME ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント

制限事項: FCMP プロシジャで作成された関数を呼び出すときは、配列である OUTARG 引数のみを変更できます。

詳細

この例では、FCMP プロシジャを使用して定義された関数を呼び出します。これは標準 SAS 関数を呼び出すのと似ています。関数の定義は、同じ SAS プログラムに含める必要はありません。PROC FCMP 関数を関数ライブラリに保存すると、CMPLIB システムオプションが定義されている任意のプログラムで使用できます。この例では、SUMX 関数と ADD_SCALAR 関数を定義して、ArrayFuncs パッケージの Sasuser.myFuncs データセットに保存します。*package* は、ユーザーによって指定された関連ルーチンの集合です。パッケージ名は、PROC FCMP 関数が含まれるデータセットで関連する関数をグループ化します。

プログラム

```
proc fcmp outlib=sasuser.myFuncs.ArrayFuncs;
function sumx(x[*]);
  sum = 0;
  do i = 1 to dim(x);
     sum = sum + x[i];
  end;
  return(sum );
endsub;

function add_scalar( scalar, x[*] );
   outargs x;
   do i = 1 to dim(x);
      x[i] = x[i] + scalar;
   end;
   return( dim(x) );
```

```
endsub;
run;


options cmplib=sasuser.myFuncs;


proc lua;
submit;
   array = { 1, 2, 3, 4, 5 }
   sum = sas.sumx(array)
   print(sum)
endsubmit;
run;


proc lua;
submit;
   array = { 1, 2, 3, 4, 5 }
   dim = sas.add_scalar(5, array)
   a1 = array[1]
   a2 = array[2]
   print(a1)
   print(a2)
endsubmit;
run;
```

プログラムの説明

FCMP プロシジャを使用して関数を定義します。 SUMX 関数は、配列の値を合計し、その値を返します。ADD_SCALAR 関数は、スカラー値を配列の各メンバに追加します。ADD_SCALAR の OUTARGS 引数は配列を指定するため、PROC LUA を使用してこの関数を呼び出すことで、サブミットされた配列の値を変更できます。ADD_SCALAR 関数は、配列の要素の数を返します。

```
proc fcmp outlib=sasuser.myFuncs.ArrayFuncs;
function sumx(x[*]);
  sum = 0;
  do i = 1 to dim(x);
     sum = sum + x[i];
  end;
  return(sum );
endsub;

function add_scalar( scalar, x[*] );
   outargs x;
   do i = 1 to dim(x);
      x[i] = x[i] + scalar;
   end;
   return( dim(x) );
endsub;
run;
```

コンパイルされた関数の場所を指定します。 この例では、Sasuser.myFuncs データセットで、前に定義された関数を検索します。関数が保存されている場合、FCMP プロシジャへの呼び出しは、LUA プロシジャと同じプログラムに含める必要はありません。

LUA プロシジャ内で関数を呼び出す前に、OPTIONS ステートメントを使用して関数の場所を指定します。

```
options cmplib=sasuser.myFuncs;
```

LUA プロシジャ内で SUMX 関数を呼び出します。 SUMX 関数を呼び出すには、呼び出しの前に"SAS."を付けます。これは、標準 SAS 関数を呼び出すのと似ています。

```
proc lua;
submit;
   array = { 1, 2, 3, 4, 5 }
   sum = sas.sumx(array)
   print(sum)
endsubmit;
run;
```

LUA プロシジャ内で ADD_SCALAR 関数を呼び出します。 ADD_SCALAR 関数を呼び出すには、呼び出しの前に"SAS."を付けます。これは、標準 SAS 関数を呼び出すのと似ています。SUMX と ADD_SCALAR への呼び出しは、同じ LUA プロシジャ内で実行できます。

```
proc lua;
submit;
   array = { 1, 2, 3, 4, 5 }
   dim = sas.add_scalar(5, array)
   a1 = array[1]
   a2 = array[2]
   print(a1)
   print(a2)
endsubmit;
run;
```

出力:ユーザー定義の配列関数の操作

アウトプット 36.11 *PROC FCMP を使用して作成された関数の呼び出し*

```
265  proc fcmp outlib=sasuser.myFuncs.ArrayFuncs;
266  function sumx(x[*]);
267    sum = 0;
268    do i = 1 to dim(x);
269      sum = sum + x[i];
270    end;
271    return(sum );
272  endsub;
273
274  function add_scalar( scalar, x[*] );
275    outargs x;
276    do i = 1 to dim(x);
277      x[i] = x[i] + scalar;
278    end;
279    return( dim(x) );
280  endsub;
281  run;

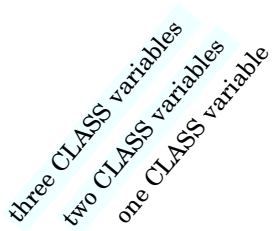
NOTE: Function add_scalar saved to sasuser.myFuncs.ArrayFuncs.
NOTE: Function sumx saved to sasuser.myFuncs.ArrayFuncs.
NOTE: PROCEDURE FCMP used (Total process time):
      real time           0.37 seconds
      cpu time            0.17 seconds


282  options cmplib=sasuser.myFuncs;
283
284  proc lua;
285  submit;
286     array = { 1, 2, 3, 4, 5 }
287     sum = sas.sumx(array)
288     print(sum)
289  endsubmit;
290  run;

NOTE: Resuming Lua state from previous PROC LUA invocation.
15
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.13 seconds
      cpu time            0.06 seconds


291
292  proc lua;
293  submit;
294     array = { 1, 2, 3, 4, 5 }
295     dim = sas.add_scalar(5, array)
296     a1 = array[1]
297     a2 = array[2]
298     print(a1)
299     print(a2)
300  endsubmit;
301  run;

NOTE: Resuming Lua state from previous PROC LUA invocation.
6
7
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.04 seconds
      cpu time            0.06 seconds
```

37 章

MEANS プロシジャ

概要: MEANS プロシジャ

MEANS プロシジャの動作について

MEANS プロシジャは、すべてのオブザベーションにわたる変数とオブザベーショングループ内の変数に対する記述統計量を計算するためのデータ要約ツールを提供します。たとえば、PROC MEANS では次を行います。

- モーメントを基準として記述統計量を計算する
- 中央値を含む分位点を推定する
- 平均値の信頼限界を計算する
- 極値を識別する
- *t* 検定を実行する

デフォルトでは、PROC MEANS は出力を表示します。OUTPUT ステートメントを使用して、統計量を SAS データセットに保存することもできます。

>PROC MEANS と PROCSUMMARY は類似しています。63 章, “SUMMARY プロシジャ,” (1861 ページ)相違点の説明についてを参照してください。

PROC MEANS が作成可能な出力の種類について

PROC MEANS デフォルト出力

出力 1.1 に、PROC MEANS が表示するデフォルトの出力を示します。PROC MEANS が分析するデータセットには、1 から 10 までの整数が含まれています。出力には、オブザベーション数、平均値、標準偏差、最小値、最大値がレポートされます。出力を生成するステートメントは次のとおりです。

```
proc means data=OnetoTen;
run;
```

アウトプット 37.1 *デフォルト記述統計量*

```
The SAS System                              1 The MEANS Procedure Analysis Variable:Integer
N            Mean         Std Dev         Minimum         Maximum
-------------------------------------------------------------------------------- 10
5.5000000       3.0276504       1.0000000       10.0000000
--------------------------------------------------------------------------------
```

PROC MEANS カスタマイズ出力

次の出力に、2 つの変数、MoneyRaised と HoursVolunteered の複数の広範分析の結果を示します。分析データセットには、地元の慈善団体に対し高校生たちが集めた金

額とボランティア活動に充てた時間数に関する情報が含まれています。PROC MEANS は、2 つのカテゴリ変数の 6 つの組み合わせを使用して、オブザベーション数、平均値、範囲を計算します。最初の変数 School には 2 つの値があり、もう 1 つの変数 Year には 3 つの値があります。出力を生成するプログラムの説明については、“例 11: 出力統計量を使用し、極値を識別する” (1161 ページ)を参照してください。

アウトプット 37.2 *分類水準と最大値の ID に対して指定された統計量*

```
Summary of Volunteer Work by School and Year                1 The MEANS Procedure N School
Year  Obs  Variable            N          Mean         Range
-------------------------------------------------------------------------- Kennedy          1992
15  MoneyRaised        15    29.0800000    39.7500000 HoursVolunteered   15    22.1333333    30.0000000
1993   20  MoneyRaised        20    28.5660000    23.5600000 HoursVolunteered   20    19.2000000
20.0000000 1994   18  MoneyRaised        18    31.5794444    65.4400000 HoursVolunteered   18
24.2777778    15.0000000 Monroe           1992   16  MoneyRaised        16    28.5450000    48.2700000
HoursVolunteered   16    18.8125000    38.0000000 1993   12  MoneyRaised        12    28.0500000
52.4600000 HoursVolunteered   12    15.8333333    21.0000000 1994   28  MoneyRaised        28
29.4100000    73.5300000 HoursVolunteered   28    19.1428571    26.0000000
-------------------------------------------------------------------------
```

```
Best Results:             .Most Money Raised and Most Hours Worked             2 Most     Most
Money     Hours Obs  School   Year  _TYPE_  _FREQ_  Cash     Time     Raised  Volunteered 1     0
109   Willard  Tonya     78.65       40 2           1992     1       31   Tonya    Tonya
55.16       40 3           1993     1       32   Cameron  Amy       65.44       31 4           1994
1       46   Willard  L.T.           . L.T.      78.65       33 5  Kennedy     .2       53   Luther
Jay       72.22       35 6  Monroe     2       56   Willard  Tonya     78.65       40 7  Kennedy
1992     3       15   Thelma   Jay       52.63       35 8  Kennedy  1993     3       20   Bill
Amy       42.23       31 9  Kennedy  1994     3       18   Luther   Che-Min   72.22       33 10
Monroe   1992     3       16   Tonya    Tonya     55.16       40 11  Monroe   1993     3       12
Cameron  Myrtle    65.44       26 12  Monroe   1994     3       28   Willard      78.65       33
```

プログラムではレポートだけでなく、最も多くのお金を集めた学生とボランティア活動にほとんどの時間を充てた学生のうちすべてのオブザベーションを超えるものと School と Year の組み合わせ内のものを識別する出力データセット(出力の 2 ページ目)も作成します。

- データセットの最初のオブザベーションには、MoneyRaised と HoursVolunteered の値が全体で最大の生徒が示されます。
- オブザベーション 2 から 4 には、学校に関係なく、年度ごとの値が最大の学生が示されます。
- オブザベーション 5 と 6 には、年度に関係なく、学校ごとの値が最大の学生が示されます。
- オブザベーション 7 から 12 には、学校と年度の組み合わせごとの値が最大の生徒が示されます。

PROC MEANS ステートメントと ODS OUTPUT ステートメント

MEANS および SUMMARY プロシジャのテンプレートによって、ODS はデフォルトの出力形式および出力形式属性を出力に適用し、入力データセットの変数に関連付けられ、適用されている任意の出力形式を上書きします。これは、オプション USE_FORMAT_DEFAULTS がテンプレート内で指定されているためです。このオプションは、ODS 形式の出力の幅および配置に影響するため、これらの幅および配置は、入力データセットまたは FORMAT ステートメントや ATTRIB ステートメントに見られる出力形式の幅および配置とは異なる場合があります。デフォルトの出力形式は ODS 出力のみに影響し、内部の数値表現や計算には影響しません。

たとえば、デフォルト出力形式の幅がデータセットまたはステートメントで指定した幅よりも広い場合、数値形式の出力の幅が予想よりも広くなったり、より多くの文字を含んだりすることがあります。

ODS OUTPUT ステートメントではなく OUTPUT ステートメントを使用して出力データセットを作成することにより、デフォルト出力形式の出力データセットへの適用を防止できます。

デフォルトの出力形式を ODS 出力に適用したくない場合は、次のコードを使用して Base.Summary ODS テーブルテンプレートを変更できます。

```
proc template;
   edit base.summary;
   use_format_defaults=off;
end;
run;
```

概念:MEANS プロシジャ

分類変数の使用

TYPES ステートメントと WAYS ステートメントの使用

TYPES ステートメントは、PROC MEANS がデータのサブグループ化に使用する利用可能な分類変数を制御します。入力データセットの 1 つのオブザベーションで発生するこれらのアクティブな分類変数値の一意の組み合わせにより、データサブグループが決定されます。指定した種類に対して PROC MEANS が生成するサブグループはそれぞれ、その種類の水準と呼ばれます。すべての種類に対し、非アクティブな分類変数が欠損値を含むオブザベーションのリジェクトのオブザベーション合計数に影響する可能性があります。

WAYS ステートメントを使用すると、PROC MEANS は、分類変数の完全セットから選択された *n* 分類変数のすべての可能な一意の組み合わせに対応する種類を生成します。たとえば、次のようになります。

```
proc means;
 class a b c d e;
 ways 2 3;
 run;
```

これは、次と同じです。

```
proc means;
 class a b c d e;
 types a*b a*c a*d a*e b*c b*d b*e c*d c*e d*e
       a*b*c a*b*d a*b*e a*c*d a*c*e a*d*e
       b*c*d b*c*e c*d*e;
 run;
```

TYPES ステートメントと WAYS ステートメントを省略すると、PROC MEANS はすべての分類変数を使用して表示されている出力に対しデータ(種類 NWAY)をサブグループ化し、出力データセットに対しすべての種類(2^k)を計算します。

分類値の並べ替え

PROC MEANS は、対応する 1 次元の種類での分類変数の順序を検証して、すべての種類での各分類変数の順序を決定します。オプション ORDER=DATA または ORDER=FREQ でのこの動作の影響を確認します。DBMS テーブルの入力データで ORDER=DATA オプションを指定すると、PROC MEANS の計算によって同じ分析の実行ごとに異なる結果が算出される可能性があります。ORDER=オプションは、クロス集計テーブルで変数水準の順序を決定するのに加え、PROC MEANS が計算する数多くの検定統計量と測定にも影響を与える可能性があります。PROC MEANS が入力データセットをサブセットに分割する場合、分類処理はサブグループごとにオプション ORDER=DATA または ORDER=FREQ にそれぞれ適用されません。代わりに、1 つの度数とデータ順序が、すべての出力に対し全体のデータセットの未分割ビューに基づいて作成されます。たとえば、次のステートメントについて考えます。

```
data pets;
 input Pet $ Gender $;
 datalines;
dog  m
dog  f
dog  f
dog  f
cat  m
cat  m
cat  f
;

proc means data=pets order=freq;
   class pet gender;
run;
```

ステートメントにより、次の出力が生成されます。

アウトプット 37.3 *分類値の並べ替え*

```
The SAS System                                1 The MEANS Procedure N Pet          Gender      Obs
--------------------------- dog         f             3 m             1 cat         f             1
m             2 ---------------------------
```

例では、PROC MEANS は雄猫を雌猫より前にリストしません。代わりに、データセット全体のすべての種類に対する性別の順序を決定します。PROC MEANS によって、雌のペットのオブザベーションの方が多いことがわかります(f=4、m=3)。ORDER のデフォルト値は ORDER=INTERNAL です。

コンピュータリソース

PROC MEANS は、すべての動作環境にわたり同一のメモリ割り当てスキームを使用します。分類変数が含まれると、PROC MEANS はメモリに各分類変数の一意の値ごとのコピーを保持する必要があります。次を計算して、分類変数をグループ化するためのメモリ要件を推定できます。

$$Nc_1(Lc_1 + K) + Nc_2(Lc_2 + K) + ... + Nc_n(Lc_n + K)$$

この場合、

Nc_i

分類変数に対する一意の値の数です。

Lc_i
c_iのフォーマットされていない長さとフォーマットされた長さの組み合わせです。

K
約 32 バイト(64 ビットアーキテクチャの場合は 64 バイト)の一部の定数です。

CLASS ステートメントで GROUPINTERNAL オプションを使用する場合、Lc_iはc_iの未フォーマットの長さです。

指定されている種類の分類変数の一意の各組み合わせ$c_{1_i}\ c_{2_j}$によって、その種類の水準が形成されます。“TYPES ステートメント” (1125 ページ)を参照してください。次を計算して、すべての組み合わせが実際にデータ(完全な種類)に存在する場合に、指定された種類のすべての水準に対し可能性のある最大必要メモリを推定できます。

$$W * Nc_1 * Nc_2 * \ ... \ * Nc_n$$

ここでは次のようになります。

W
分析された変数の数と計算された統計量の数に基づく定数です(分位点の計算をするための QMETHOD=OS を要求しない限り)。

$Nc_1 ... Nc_n$
指定された種類のアクティブな分類変数に対する一意の水準の数です。

明らかに、水準の必要メモリは分類変数の水準に負担をかけます。このため、PROC MEANS は 1 つ以上のユーティリティファイルを開き、1 つ以上の種類の水準をディスクに書き込むことができます。これらの種類は、PROC MEANS が入力データスキャン時に作成したプライマリの種類、または派生の種類です。

PROC MEANS がすべてのプライマリの種類を入力データの処理時にディスクに一部書き込む必要がある場合、1 つ以上のマージパスがメモリの種類の水準とディスクの水準を組み合わせるために必要になる可能性があります。また、分類変数に DATA 以外の順序を使用する場合、PROC MEANS はディスクで完了した種類をグループ化します。このため、最大のディスク必要量は、指定されている種類の必要メモリの 2 倍になることがあります。

PROC MEANS が一時作業ファイルを使用する場合、次の記載が SAS ログに表示されます。

```
Processing on disk occurred during summarization.
Peak disk usage was approximately nnn
Mbytes.
Adjusting MEMSIZE or REALMEMSIZE may improve performance.
```

ほとんどの場合、処理は正常に終了します。

CLASS ステートメントで分類変数を指定する場合、PROC MEANS がユーティリティファイルに書き込む前に使用するデータ依存メモリ量が、SAS システムオプション REALMEMSIZE=によって制御されます。REALMEMSIZE=の値は、SAS で割り当て可能な仮想メモリとは対照的な実メモリ量を示します。PROC MEANS は、これらの 2 つの値のうち小さい方を計算することによってユーティリティファイルに書き込む前に使用するデータ依存メモリ量を決定します。

- REALMEMSIZE=の値
- 0.8*(M-U) (M は MEMSIZE=の値で、U はすでに使用中のメモリ量)

REALMEMSIZE は、PROC SORT など、その他のメモリ集中型 PROC の動作にも影響します。

代替として、PROC オプション SUMSIZE=を使用できます。PROC オプション SORTSIZE=と同様に、SUMSIZE=は、ディスク基準操作が開始するメモリいき値を設定します。最高の結果を得るには、SUMSIZE=をタスクに利用可能になる可能性がある実メモリ量よりも少ない値に指定します。効率的にするため、PROC MEANS は SUMSIZE=の値の端数を内部的に切り上げることができます。SUMSIZE=は、分類変数を指定しない限り影響しません。

動作環境の情報
REALMEMSIZE= SAS システムオプションは、すべての動作環境で使用できるわけではありません。詳細については、使用している動作環境に関する SAS コンパニオンを参照してください。

PROC MEANS によりメモリが不十分であることがレポートされる場合、SUMSIZE= (または REALMEMSIZE=)の値を増やします。MEMSIZE=より大きい SUMSIZE= (または REALMEMSIZE=)値は影響しません。そのため、MEMSIZE=の値も増やす必要があることがあります。PROC MEANS によりディスク容量が不十分であることがレポートされる場合、WORK スペース割り当てを増やします。計算リソースパラメータの調整方法については、使用している動作環境に関する SAS ドキュメントを参照してください。

パフォーマンスを向上させるもう 1 つの方法として、TYPES ステートメントまたは WAYS ステートメントを注意して適用し、計算を関連する分類変数の組み合わせにのみ制限します。特に、すべての分類変数の組み合わせを要求しないことによって重要なリソースの節約が可能になります。

PROC MEANS の In-Database 処理

大きなデータセットが外部データベースに格納される場合、SAS を実行するコンピュータへのデータセットの転送がパフォーマンス、セキュリティ、リソース管理問題によって影響を受ける可能性があります。SAS In-Database 処理でデータベースで初期データ集計が実行されるようにすることにより、データ転送を大幅に減らすことができます。

PROC MEANS の In-Database 処理では、次のデータベース管理システムがサポートされています。

- Aster
- DB2
- Greenplum
- HADOOP
- Impala
- Netezza
- Oracle
- SAP HANA
- Teradata

適切な条件下で、PROC MEANS は使用されるステートメントと、PROC ステップで指定される出力統計量に基づいて SQL クエリを生成します。分類変数が指定されると、プロシジャは N 次元の種類を表す SQL GROUP BY 句を作成します。データベースでの集計クエリの実行時に作成される結果セットが内部 PROC MEANS データ構造に読み込まれ、すべての後続の種類が元の N 次元の種類から派生し、最終分析結果を形成します。SAS 出力形式定義がデータベースで配置される際に、分類変数のフォーマットがデータベースで発生します。SAS 出力形式定義がデータベースで配置されていない場合、未加工値に対し In-Database 集計が実行され、結果のセットが PROC MEANS 内部構造と結合されるときに関連する出力形式が適用されます。マルチラベ

ルフォーマットは常に、データベースによって返される最初に集計された結果セットを使用して実行されます。ステートメント CLASS、TYPES、WAYS、VAR、BY、FORMAT、WHERE は、PROC MEANS がデータセット内で処理される場合にサポートされます。FREQ、ID、IDMIN、IDMAX、IDGROUPS は、サポートされていません。次の統計量が、データベース内処理に対しサポートされています。N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN、CSS、USS、VAR、STD、STDERR、PRET、UCLM、LCLM、CLM、CV

データベース内処理の重みづけは、N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN に対してのみサポートされています。

次の統計量は現在、In-Database 処理に対しサポートされていません。SKEW、KURT、P1、P5、P10、P20、P25/Q1、P30、P40、P50/MEDIAN、P60、P70、P75/Q3、P80、P90、P95、P99、MODE

SQLGENERATION システムオプションまたは LIBNAME ステートメントオプションは、データベース内プロシジャがデータベース内で実行されるかどうか、およびその実行方法を制御します。デフォルトで、データベース内プロシジャは可能な場合はデータベース内で実行されます。データベース処理を行わないようにする、OBS=、FIRSTOBS=、RENAME=、DBCONDITION=などの多くのデータセットオプションがあります。完全なリストについては、*SAS/ACCESS for Relational Databases: Reference* の “In-Database Procedures in Teradata”を参照してください。

In-Database 処理は、分類変数がない(1 行が返される)、または選択されている分類変数に少数の一意の値が含まれている場合に、プロシジャに転送されるデータ量を大幅に減らすことができます。ただし、PROC MEANS は結果セットをその内部構造に読み込むため、SAS プロセスの必要メモリはデータベース処理が実行されない場合に要求されたものと同じになります。データ要約のほとんどがデータベース内で実行される場合、SAS プロセスの必要 CPU を大幅に削減する必要があります。要約に必要な実時間を大幅に削減する必要があります。多くのデータベースプロセスクエリが並列処理されるためです。

データベース処理の詳細については、*SAS/ACCESS for Relational Databases: Reference* を参照してください。

入力データセットのスレッド処理

THREADS オプションは、入力データセットの並列処理を有効化または無効化します。スレッド処理により、処理操作における一定の並列処理が達成されます。この並列処理の目的は、所定の操作を完了するための処理時間を削減し、それによって追加 CPU リソースのコストを抑えることです。詳細については、*SAS 言語リファレンス: 解説編*の“Support for Parallel Processing"を参照してください。

SAS システムオプション CPUCOUNT=の値はスレッド化された並べ替えのパフォーマンスに影響します。CPUCOUNT=は、スレッド化されたプロシジャで使用できるシステム CPU の数を示します。

詳細については、*SAS システムオプション: リファレンス*の“THREADS System Option” および“CPUCOUNT= System Option”を参照してください。

構文: MEANS プロシジャ

ヒント: ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。グローバルステートメントを使用することもでき

ます。リストについては、*SAS ステートメント: リファレンス*の“グローバルステートメント”を参照してください。

PROC MEANS <*option(s)*> <*statistic-keyword(s)*>;
BY<DESCENDING> *variable-1* <<DESCENDING> *variable-2* …>< NOTSORTED;>
CLASS *variable(s)*</ *option(s)*>;
FREQ *variable*;
ID *variable(s)*;
OUTPUT <OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
<*id-group-specification(s)*> <*maximum-id-specification(s)*>
<*minimum-id-specification(s)*> </ *option(s)*>;
TYPES *request(s)*;
VAR *variable(s)* </ WEIGHT=*weight-variable*>;
WAYS *list*;
WEIGHT *variable*;

ステートメント	タスク	例
“PROC MEANS ステートメント”	変数の記述統計量を計算します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10, Ex. 12, Ex. 13
“BY ステートメント”	BY グループごとに別の統計量を計算します。	Ex. 3
“CLASS ステートメント”	値が分析対象のサブグループを定義する変数を識別します。	Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10, Ex. 11, Ex. 12
“FREQ ステートメント”	値が各オブザベーションの度数を表す変数を識別します。	
“ID ステートメント”	追加の ID 変数を出力データセットに含めます。	
“OUTPUT ステートメント”	指定されている統計量と ID 変数を含む出力データを作成します。	Ex. 8, Ex. 9, Ex. 10, Ex. 11, Ex. 12
“TYPES ステートメント”	データの分割に使用する分類変数の特定の組み合わせを識別します。	Ex. 2, Ex. 5, Ex. 12
“VAR ステートメント”	分析変数と結果での順序を識別します。	Ex. 1
“WAYS ステートメント”	分類変数の一意の組み合わせを作成するための方法の数を指定します。	Ex. 6

ステートメント	タスク	例
“WEIGHT ステートメント”	値が統計量計算で各オブザベーションを重み付ける変数を識別します。	Ex. 6

PROC MEANS ステートメント

変数の記述統計量を計算します。

参照項目: 63 章, “SUMMARY プロシジャ,” (1861 ページ)

例:
“例 1: 特定の記述統計量を計算する” (1134 ページ)
“例 2: 分類変数を使用し、記述統計量を計算する” (1136 ページ)
“例 3: BY ステートメントを分類変数とともに使用する” (1139 ページ)
“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1141 ページ)
“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1144 ページ)
“例 6: すでに読み込まれている出力形式を分類変数とともに使用する” (1149 ページ)
“例 7: 平均の信頼限界の計算” (1153 ページ)
“例 8: 出力統計量を計算する” (1155 ページ)
“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1157 ページ)
“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1159 ページ)
“例 11: 出力統計量を使用し、極値を識別する” (1161 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)
“例 13: STACKODSOUTPUT オプションによるデータの制御” (1168 ページ)

構文

PROC MEANS <*option(s)*> <*statistic-keyword(s)*>;

オプション引数の要約

DATA=*SAS-data-set*
入力データセットを指定します。
NOTHREADS
SAS システムオプション THREADS | NOTHREADS を無効にします。
NOTRAP
浮動小数点例外の復旧を無効化します。
PCTLDEF=
分位点の計算に使用される数学的定義を指定します。
SUMSIZE=*value*
分類変数によるデータ要約に使用するメモリ量を指定します。
THREADS | NOTHREADS
SAS システムオプション THREADS | NOTHREADS を無効にします。

Control the classification levels

CLASSDATA=*SAS-data-set*
分析する分類変数の組み合わせを含む 2 次データセットを指定します。
COMPLETETYPES

分類変数値の可能な組み合わせをすべて作成します。

EXCLUSIVE
CLASSDATA=データセットにない分類変数値の組み合わせをすべて分析から除外します。

MISSING
欠損値を有効値として使用し、分類変数の組み合わせを作成します。

Control the output

FW=*field-width*
統計量のフィールド幅を指定します。

MAXDEC=*number*
統計量に対し小数点以下の桁数を指定します。

NONOBS
分類変数の一意の各組み合わせに対するオブザベーション合計数のレポートを行いません。

NOPRINT
表示されているすべての出力を非表示にします。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
分類変数の値を指定されている順序に従って並べ替えます。

PRINT | NOPRINT
出力を表示します。

PRINTALLTYPES
分類変数の要求されたすべての組み合わせに対する分析を表示します。

PRINTIDVARS
ID 変数の値を表示します。

STACKODSOUTPUT
ODS 出力オブジェクトを生成します

Control the output data set

CHARTYPE
_TYPE_変数に文字値が含まれるように指定します。

DESCENDTYPES
_TYPE_値の降順で出力データセットを並べ替えます。

IDMIN
ID 変数を最小値に基づいて選択します。

NWAY
出力統計量を_TYPE_の値が最高のオブザベーションに制限します。

Control the statistical analysis

ALPHA=*value*
信頼限界に対する信頼水準を指定します。

EXCLNPWGTS
非正の重みが付いたオブザベーションを分析から除外します。

QMARKERS=*number*
P2 分位点推定方法に使用するサンプルサイズを指定します。

QMETHOD=OS|P2
分位点推定方法を指定します。

QNTLDEF=1 | 2 | 3 | 4 | 5
分位点の計算に使用される数学的定義を指定します。

statistic-keyword(s)
統計量を選択します。

VARDEF=*divisor*
分散の計算のための分母を指定します。

オプション引数

ALPHA=*value*
平均値に対する信頼限界を計算する信頼水準を指定します。信頼限界のパーセントは、(1−*value*)×100 です。たとえば、ALPHA=.05 は、95%の信頼水準となります。

デフォルト .05

範囲 0 から 1

操作 信頼限界を計算するには、*statistic-keyword* CLM、LCLM または UCLM を指定します。

参照項目 “信頼限界” (1130 ページ)

“例 7: 平均の信頼限界の計算” (1153 ページ)

CHARTYPE
出力データセットの_TYPE_変数が_TYPE_のバイナリ値の文字表現になるように指定します。変数の長さは分類変数の数と同じです。

操作 32 を超える分類変数を指定すると、_TYPE_は自動的に文字変数になります。

参照項目 “出力データセット” (1133 ページ)

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1159 ページ)

CLASSDATA=*SAS-data-set*
出力に表示が必要な分類変数の値の組み合わせを含むデータセットを指定します。入力データセットではなくCLASSDATA=データセットで発生する分類変数の値の組み合わせが出力に表示され、度数はゼロとなります。

制限事項 CLASSDATA=データセットにすべての分類変数を含める必要があります。このデータの種類と出力形式は、入力データセットの対応する分類変数と一致する必要があります。

操作 EXCLUSIVE オプションを使用する場合、PROC MEANS は分類変数の組み合わせが CLASSDATA=データセットにない入力データセットのオブザベーションを除外します。

ヒント CLASSDATA=データセットを使用して、入力データセットをフィルタまたは補足します。

参照項目 “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1141 ページ)

COMPLETETYPES
組み合わせが入力データセットで発生しない場合でも、分類変数の可能な組み合わせをすべて作成します。

操作 CLASS ステートメントの PRELOADFMT オプションにより、度数がゼロの場合でも PROC MEANS が分類変数の組み合わせに対するユーザー定義出力形式範囲または値を出力に書き込むようになります。

ヒント COMPLETETYPES を使用しても、必要メモリは増えません。

参照項目 “例 6: すでに読み込まれている出力形式を分類変数とともに使用する” (1149 ページ)

DATA=*SAS-data-set*
入力 SAS データセットを識別します。

参照項目 “入力データセット” (25 ページ)

DESCENDTYPES
_TYPE_値の降順で出力データセットのオブザベーションを並べ替えます。

別名 DESCENDING | DESCEND

操作 NWAY を指定すると、降順は無効です。

ヒント DESCENDTYPES を使用して、全体合計(_TYPE_=0)を各 BY グループの最終オブザベーションにします。

参照項目 “出力データセット” (1133 ページ)

“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1157 ページ)

EXCLNPWGTS
非正(ゼロまたは負)の重みがついたオブザベーションを分析から除外します。デフォルトでは、PROC MEANS は非正の重みが付いたオブザベーションを重みがゼロのオブザベーションのように扱い、オブザベーションの合計数にカウントします。

別名 EXCLNPWGTS

参照項目 “WEIGHT ステートメント” (1128 ページ)

VAR ステートメントの WEIGHT=オプション (1127 ページ)

EXCLUSIVE
CLASSDATA=データセットにない分類変数の組み合わせをすべて分析から除外します。

要件 CLASSDATA=データセットが指定されていない場合、このオプションは無視されます。

参照項目 “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1141 ページ)

FW=*field-width*
統計量を印刷出力または表示出力に表示するフィールド幅を指定します。FW=は、出力データセットに保存される統計量に影響しません。

デフォルト 12

ヒント PROC MEANS が出力の列ラベルを切り捨てる場合、フィールド幅を増やします。

参照項目 “例 1: 特定の記述統計量を計算する” (1134 ページ)

“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1141 ページ)

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1144 ページ)

IDMIN
出力データセットに ID 変数の最小値が含まれるように指定します。

操作 出力に ID 変数の値を表示する PRINTIDVARS を指定します。

参照項目 “ID ステートメント” (1117 ページ)

MAXDEC=*number*
印刷出力または表示出力に統計量を表示する小数点以下の桁の最大数を指定します。MAXDEC=は、出力データセットで保存される統計量に影響しません。

デフォルト BEST.(柱状出力形式の幅。通常は約 7)。

範囲 0-8

参照項目 “例 2: 分類変数を使用し、記述統計量を計算する” (1136 ページ)

“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1141 ページ)

MISSING
欠損値を有効値とみなし、分類変数の組み合わせを作成します。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING を省略すると、PROC MEANS は欠損分類変数値を含むオブザベーションを分析から除外します。

参照項目 *SAS 言語リファレンス: 解説編* 特別な意味を持つ欠損値の説明

“例 6: すでに読み込まれている出力形式を分類変数とともに使用する” (1149 ページ)

NONOBS
分類変数の値の一意の各組み合わせに対するオブザベーション合計数を表示する列を非表示にします。この列は、出力データセットの_FREQ_変数に対応しています。

参照項目 “N Obs 統計量” (1132 ページ)

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1144 ページ)

“例 6: すでに読み込まれている出力形式を分類変数とともに使用する” (1149 ページ)

NOPRINT
`“PRINT | NOPRINT” (1106 ページ)オプションを参照してください。

NOTHREADS
“THREADS | NOTHREADS” (1109 ページ)オプションを参照してください。

NOTRAP
データ処理時の浮動小数点例外(Floating Point Exception (FPE))の復旧を無効化します。デフォルトでは、PROC MEANS はこれらのエラーをトラップし、統計量を欠損に設定します。

FPE 回復のオーバーヘッドが重要な動作環境では、NOTRAP はパフォーマンスを向上させることができます。算術例外の場合に PROC MEANS が強制終了するように、通常の SAS FPE 処理は有効です。

NWAY
出力データセットに_TYPE_と_WAY_の値が最高のオブザベーションの統計量のみ含まれるように指定します。分類変数の指定時に、NWAY はすべての分類変数の組み合わせに対応します。

操作 TYPES ステートメントまたは WAYS ステートメントを指定すると、PROC MEANS はこのオプションを無視します。

参照項目 “出力データセット” (1133 ページ)

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1159 ページ)

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
並べ替え順序を指定して、出力の分類変数の値の一意の組み合わせを作成します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 CLASS ステートメントで PRELOADFMT を使用すると、各分類変数の値の順序が、PROC FORMAT が関連するユーザー定義出力形式の値の格納に使用する順序と一致します。CLASSDATA=オプションを使用すると、PROC MEANS は CLASSDATA=データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC MEANS はまずユーザー定義出力形式を使用して、出力を並べ替えます。EXCLUSIVE を省略すると、PROC MEANS はユーザー定義出力形式と CLASSDATA=値の後に入力データセットの分類変数の一意の値を発生順に基づいて追加します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。この順序は、使用している動作環境によって異なります。

別名 FMT | EXTERNAL

FREQ
最も多くのオブザベーションを含む水準が最初に表示されるように、値を度数カウントの降順で並べ替えます。

操作 分類変数の多重組み合わせに対し、PROC MEANS は個別の分類変数度数から分類変数組み合わせの順序を決定します。

CLASS ステートメントで ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

UNFORMATTED
値をフォーマットされていない値別に並べ替えます。これにより、PROC SORT と同じ順序が作成されます。この順序は、使用している動作環境によって異なります。

別名 UNFMT | INTERNAL

デフォルト UNFORMATTED

参照項目 “分類値の並べ替え” (1095 ページ)

PCTLDEF=
PCTLDEF は、QNTLDEF=の別名です。

参照項目 QNTLDEF=オプション

PRINT | NOPRINT
PROC MEANS が統計量分析を表示するかどうかを指定します。NOPRINT は、すべての出力を行いません。

デフォルト PRINT

ヒント OUT=出力データセットのみ作成する場合は、NOPRINT を使用します。

参照項目 “例 8: 出力統計量を計算する” (1155 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)

PRINTALLTYPES
印刷または表示出力の分類変数の要求された組み合わせをすべて(すべての _TYPE_値)表示します。通常、PROC MEANS は種類 NWAY のみ表示します。

別名 PRINTALL

操作 NWAY オプション、TYPES ステートメントまたは WAY ステートメントを使用すると、PROC MEANS はこのオプションを無視します。

参照項目 “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1141 ページ)

PRINTIDVARS
ID 変数の値を印刷または表示出力に表示します。

別名 PRINTIDS

操作 ID 変数の最小値を表示する IDMIN を指定します。

参照項目 “ID ステートメント” (1117 ページ)

QMARKERS=*number*
P2 分位点推定方法に使用するマーカーのデフォルト数を指定します。マーカー数によって固定メモリ空間のサイズを制御します。

デフォルト デフォルト値は、要求する分位点によって異なります。中央値(P50)の場合、*number* は 7 です。分位点(P25 と P50)の場合、*number* は 25 です。分位点 P1、P5、P10、P75 P90、P95 または P99 の場合、*number* は 105 です。複数の分位点を要求すると、PROC MEANS は *number* の最大値を使用します。

範囲 3 より大きい奇数の整数

ヒント マーカー数をデフォルト設定よりも増やして推定の正確性を向上させます。マーカー数を減らして、メモリと計算時間を節約します。

参照項目 “分位点” (1131 ページ)

QMETHOD=OS|P2
PROC MEANS が分位点の計算時に入力データを処理するために使用する方法を指定します。オブザベーション数が QMARKERS=値および QNTLDEF=5 以下の場合、両方の方法による結果は同じになります。

OS
順序統計量を使用します。この方法は、PROC UNIVARIATE が使用する方法と同じです。

注: この方法は、非常に多くのメモリを必要とする可能性があります。

P2
P2 分位点推定方法を使用して、分位点を概算します。

デフォルト OS

制限事項 QMETHOD=P2 の場合、PROC MEANS では MODE または重み付き分位点は計算されません。

ヒント QMETHOD=P2 の場合、一部の分位点(P、P5、P95、P99)の信頼性のある推定は、一部のデータセットに対し可能でないことがあります。

参照項目 “分位点” (1131 ページ)

QNTLDEF=1 | 2 | 3 | 4 | 5
QMETHOD=OS の場合に PROC MEANS が分位点の計算に使用する数学的定義を指定します。QMETHOD=P2 を使用するには、QNTLDEF=5 を使用する必要があります。

別名 PCTLDEF=

デフォルト 5

参照項目 “分位数と関連統計量” (2083 ページ)

statistic-keyword(s)
計算する統計量と、それを出力に表示する順序を指定します。PROC ステートメントで利用可能なキーワードは、次のとおりです。

記述統計量キーワード	
CLM	NMISS
CSS	RANGE
CV	SKEWNESS | SKEW
KURTOSIS | KURT	STDDEV | STD
LCLM	STDERR
MAX	SUM
MEAN	SUMWGT
MIN	UCLM
MODE	USS
N	VAR
四分位範囲統計量キーワード	
MEDIAN | P50	Q3 | P75
P1	P90
P5	P95
P10	P99
P20	P30
P40	P60
P70	P80
Q1 | P25	QRANGE
仮説検定キーワード	
PROBT | PRT	T

デフォルト N、MEAN、STD、MIN、MAX

要件 平均値の標準誤差、信頼限界、スチューデントの *t* 検定を計算するには、VARDEF=オプションのデフォルト値、DF を使用する必要があります。歪度または尖度を計算するには、VARDEF=N または VARDEF=DF を使用する必要があります。

ヒント CLM または LCLM と UCLM の両方を使用して、平均値の両側信頼限界を計算します。LCLM のみ、または UCLM を使用して、片側信頼限界を計算します。

参照項目 関連する統計量のキーワードと式の定義については、“キーワードと式” (2078 ページ)を参照してください。

“例 1: 特定の記述統計量を計算する” (1134 ページ)

“例 3: BY ステートメントを分類変数とともに使用する” (1139 ページ)

STACKODSOUTPUT
データセットが印刷出力に類似している ODS 出力オブジェクトを生成します。

STACKODSOUTPUT オプションは、PROC MEANS OUTPUT ステートメントではなく、ODS OUTPUT ステートメントで作成された出力データセットに影響します。

別名 STACKODS

参照項目 “例 13: STACKODSOUTPUT オプションによるデータの制御” (1168 ページ)

SUMSIZE=*value*
分類変数の使用時にデータ要約に使用できるメモリ量を指定します。*value* は、次のうちいずれかになります。

***n*|*n*K| *n*M| *n*G**
利用可能なメモリ量をバイト、キロバイト、メガバイト、ギガバイトでそれぞれ指定します。*n* が 0 の場合、PROC MEANS は SAS システムオプション SUMSIZE=の値を使用します。

MAXIMUM|MAX
利用可能なメモリの最大量を指定します。

デフォルト SUMSIZE=システムオプションの値。

注 SUMSIZE=0 を指定すると、PROC MEANS は優先するグローバル REALMEMSIZE オプションを使用できるようになります。

ヒント 最高の結果を得るには、SUMSIZE=に PROC ステップで利用可能な物理メモリ量より大きい値を指定しないでください。追加容量が必要な場合、PROC MEANS はユーティリティファイルを使用します。

参照項目 SAS システムオプション SUMSIZE=*SAS システムオプション: リファレンス*.

THREADS | NOTHREADS
入力データセットの並列処理を有効化または無効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします。(制約を参照)。詳細については、*SAS 言語リファレンス: 解説編*の“Support for Parallel Processing”を参照してください。

デフォルト SAS システムオプション THREADS | NOTHREADS の値。

制限事項 サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

操作 PROC MEANS は、BY ステートメントが指定されている、または SAS システムオプション CPUCOUNT の値が 2 未満の場合を除いて SAS システムオプション THREADS を有効にします。PROC MEANS ステートメントで THREADS を使用して、PROC MEANS がこれらの状況で並列処理を使用するように強制できます。

注 THREADS が指定され(SAS システムオプションとして、または PROC MEANS ステートメントで)、もう 1 つのプログラムが入力データセットを読み込み、書き込み、更新のために開いておくと、PROC MEANS が入力データセットを開けない場合があります。この場合、PROC MEANS は処理を停止し、メッセージを SAS ログに書き込みます。

VARDEF=*divisor*

分散と標準偏差の計算に使用する分母を指定します。次の表に、*divisor* に可能な値と、関連する分母を示します。

表 37.1 *VARDEF=に可能な値*

値	分母	分母の式
DF	自由度	$n - 1$
N	オブザベーション数	n
WDF	重みの合計 - 1	$(\Sigma_i w_i) - 1$
WEIGHT | WGT	重みの合計	$\Sigma_i w_i$

プロシジャは分散を*CSS*/*divisor*として計算します。*CSS*は修正平方和で、$\Sigma (x_i - \bar{x})^2$と等しくなります。分析変数に重み付けをする場合、*CSS*は$\Sigma w_i (x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト DF

要件 平均の標準誤差、平均の信頼限界またはスチューデントの *t*-検定を計算するには、VARDEF=のデフォルト値を使用します。

ヒント WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。*i* 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$であり、w_iは *i* 番目のオブザベーションの重みです。この方法により、単位重みを含むオブザベーションの分散の推定が生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな n に対し)$\sigma^2/\overline{w}$の推定になります。$\overline{w}$は平均重みです。この方法により、平均重みを含むオブザベーションの分散の漸近的な推定が生成されます。

参照項目
“キーワードと式” (2078 ページ)

“重み付き統計量の例” (76 ページ)

BY ステートメント

BY グループごとに別の統計量を生成します。

参照項目:
“BY” (68 ページ)

“BY ステートメントと CLASS ステートメントの比較” (1116 ページ)

例:
“例 3: BY ステートメントを分類変数とともに使用する” (1139 ページ)

構文

BY<DESCENDING> *variable-1* <<DESCENDING> *variable-2* …>< NOTSORTED;>

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを省略する場合、データセットのオブザベーションを指定するすべての変数別に並べ替えるか、適切にインデックス付けする必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
オブザベーションが BY ステートメントの文字 DESCENDING の直後に続く変数で降順に並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。オブザベーションは、別の方法(日付順など)で並べ替えられます。

BY 変数の値に応じたオブザベーションの順序付けまたはインデックスの要件は、NOTSORTED オプションを使用すると、BY グループの処理を行うために保留されます。プロシジャは、NOTSORTED を指定する場合はインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じであるオブザベーションが連続していない場合、プロシジャでは連続するセットを個別の BY グループとして処理されます。

詳細

BY ステートメントと SAS システムオプション NOBYLINE の使用

BY ステートメントを、BY グループ処理によって生成される出力に通常表示される BY 行を非表示にする SAS システムオプション NOBYLINE と使用する場合、PROC MEANS は常に BY グループごとに新しいページを開始します。この動作により、タイトルに BY グループ情報を入力し、NOBYLINE でデフォルトの BY 行を非表示にしてカスタマイズした BY 行を作成した場合、タイトルの情報がページのレポートと一致するようになります。“BY グループの情報を含むタイトルの作成 ” (53 ページ)を参照してください。また、“デフォルトの BY 行を表示しない” (53 ページ)も参照してください。

CLASS ステートメント

値が分析対象のサブグループの組み合わせを定義する変数を指定します。

注: オプションのない CLASS ステートメントは ORDER=INTERNAL(デフォルト)、または PROC MEANS ステートメントの ORDER=オプションで指定された値を使用します。たとえば、次のコードでは、変数 c と d は ORDER=INTERNAL を使用します。PROC MEANS ステートメントで ORDER=オプションが指定されている場合、変数 c と d は PROC MEANS ステートメントの ORDER=オプションで指定された値を使用します。

```
class a b / order=data;
 class c d;
```

ヒント: 複数の CLASS ステートメントを使用できます。

一部の CLASS ステートメントオプションも PROC MEANS ステートメントで使用できます。これらは、すべての CLASS 変数に影響します。CLASS ステートメントで指定するオプションは、CLASS ステートメントの変数にのみ適用されます。

参照項目: CLASS ステートメントによるフォーマットされた値のグループ化方法については、“フォーマットされた値” (58 ページ)を参照してください。

例:
“例 2: 分類変数を使用し、記述統計量を計算する” (1136 ページ)
“例 3: BY ステートメントを分類変数とともに使用する” (1139 ページ)
“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1141 ページ)
“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1144 ページ)
“例 6: すでに読み込まれている出力形式を分類変数とともに使用する” (1149 ページ)
“例 7: 平均の信頼限界の計算” (1153 ページ)
“例 8: 出力統計量を計算する” (1155 ページ)
“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1157 ページ)
“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1159 ページ)
“例 11: 出力統計量を使用し、極値を識別する” (1161 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)

構文

CLASS *variable(s)* </ *options*>;

必須引数

variable(s)
プロシジャがデータのグループ化に使用する 1 つ以上の変数を指定します。CLASS ステートメントの変数は、*分類変数*といいます。分類変数は、数値または

文字です。分類変数には連続値を含めることができますが、通常は変数の水準を定義するいくつかの不連続値が含まれます。データを分類変数別に並べ替える必要はありません。

操作 TYPES ステートメントまたは WAYS ステートメントを使用して、PROC MEANS がデータのグループ化に使用する分類変数を制御します。

ヒント 分類変数水準の数を減らすには、FORMAT ステートメントを使用して、変数値をまとめます。出力形式が複数の内部値を 1 つのフォーマットされた値にまとめると、PROC MEANS は最小内部値を出力します。

参照項目 “分類変数の使用 ” (1094 ページ)

オプション引数

ASCENDING
分類変数水準を昇順に並べ替えるように指定します。

別名 ASCEND

操作 ASCENDING と DESCENDING の両方を指定すると、PROC MEANS は警告メッセージを発行し、両方のオプションを無視します。

参照項目 “例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1159 ページ)

DESCENDING
分類変数水準を降順で並べ替えるように指定します。

別名 DESCEND

操作 ASCENDING と DESCENDING の両方を指定すると、PROC MEANS は警告メッセージを発行し、両方のオプションを無視します。

EXCLUSIVE
ユーザー定義出力形式のすでに読み込まれた範囲にない分類変数のすべての組み合わせを分析から除外します。

要件 PRELOADFMT を指定して、分類変数出力形式を事前に読み込む必要があります。

参照項目 “例 6: すでに読み込まれている出力形式を分類変数とともに使用する” (1149 ページ)

GROUPINTERNAL
PROC MEANS が値をグループ化して分類変数の組み合わせを作成する場合は、出力形式を分類変数に適用しないように指定します。

操作 PRELOADFMT オプションを指定すると、PROC MEANS は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ORDER=FORMATTED オプションを指定すると、PROC MEANS は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ヒント このオプションは、数値分類変数に不連続値が含まれている場合にコンピュータリソースを節約します。

参照項目 “コンピュータリソース” (1116 ページ)

MISSING
欠損値を分類変数水準に有効な値とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING を省略すると、PROC MEANS は欠損分類変数値を含むオブザベーションを分析から除外します。

参照項目 *SAS 言語リファレンス: 解説編* 特別な意味を持つ欠損値の説明

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1159 ページ)

MLF
PROC MEANS により、マルチラベル出力形式が分類変数に割り当てられている場合に指定の範囲や重複する範囲の 1 番目と 2 番目の出力形式ラベルを使用して、サブグループの組み合わせを作成できるようになります。

要件 VALUE ステートメントで PROC FORMAT と MULTILABEL オプションを使用して、マルチラベル出力形式を作成する必要があります。

操作 OUTPUT ステートメントと MLF を使用すると、分類変数にはフォーマットされた値に対応する文字列が含まれます。フォーマットされた値が内部値になるため、この変数の長さは最長の出力形式ラベルの文字数です。

MLF と ORDER=FREQ を使用すると、フォーマットされた値に使用する順序が生成されない場合があります。MLF を CLASSDATA および EXCLUSIVE と使用した場合に期待する結果が得られない場合があります。MLF 処理で各 TYPE が別個に計算されるように要求されるためです。NWAY 以外の種類には、予想よりも多くの水準が含まれています。

注 フォーマットされた値が重複する場合、1 つの内部分類変数値が複数の分類変数サブグループの組み合わせにマップします。そのため、すべてのサブグループの N 統計量の合計がデータセットのオブザベーション数(N 統計量全体)を超えます。

ヒント MLF を省略すると、PROC MEANS は 1 番目の出力形式ラベルを使用します。このアクションは、最初の外部出力形式値を使用したサブグループの組み合わせの決定に一致します。

参照項目 FORMAT プロシジャの VALUE ステートメントにおける MULTILABEL オプション“オプション引数” (850 ページ).

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1144 ページ)

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
出力の分類変数の水準をグループ化する順序を指定します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 PRELOADFMT を使用すると、各分類変数の値の順序が、PROC FORMAT が関連するユーザー定義出力形式の値の格納に使用する順序と一致します。PROC ステートメントで CLASSDATA=オプションを使用すると、PROC MEANS は CLASSDATA=データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC MEANS はまずユーザー定義出力形式を使用して、出力を並べ替えます。PROC ステートメントで EXCLUSIVE を省略すると、PROC MEANS はユーザー定義出力形式と CLASSDATA=値の後に入力データセットの分類変数の一意の値を発生順に基づいて追加します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

参照項目 “例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1159 ページ)

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。この順序は、使用している動作環境によって異なります。出力形式がクラス変数に割り当てられていない場合、デフォルトの出力形式 BEST12 が使用されます。

別名 FMT | EXTERNAL

参照項目 “例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1144 ページ)

FREQ
最も多くのオブザベーションを含む水準が最初に表示されるように、値を度数カウントの降順で並べ替えます。

操作 分類変数の多重組み合わせに対し、PROC MEANS は個別の分類変数度数から水準の順序を決定します。

ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

参照項目 “例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1144 ページ)

UNFORMATTED
値をフォーマットされていない値別に並べ替えます。これにより、PROC SORT と同じ順序が作成されます。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を時系列で表示する場合に特に便利です。

別名 UNFMT | INTERNAL

デフォルト UNFORMATTED

ヒント デフォルトでは、FREQ 以外のすべての順序は昇順です。降順の順序の場合は、DESCENDING オプションを使用します。

参照項目 “分類値の並べ替え” (1095 ページ)

PRELOADFMT
すべての出力形式が分類変数に対して事前にロードされることを示します。

要件 COMPLETETYPES、EXCLUSIVE、または ORDER=DATA を指定し、出力形式を分類変数に割り当てない限り、PRELOADFMT は影響しません。

操作 PROC TABULATE 出力を入力データセットに存在するフォーマットされた分類変数値の組み合わせに制限するには、CLASS ステートメントで EXCLUSIVE オプションを使用します。

度数がゼロの場合でもユーザー定義出力形式のすべての範囲と値を出力に含めるには、PROC ステートメントで COMPLETETYPES を使用します。

参照項目 “例 6: すでに読み込まれている出力形式を分類変数とともに使用する” (1149 ページ)

詳細

BY ステートメントと CLASS ステートメントの比較

BY ステートメントを使用することは、PROC MEANS が各 BY グループを入力データの独立したサブセットとして要約する、CLASS ステートメントと NWAY オプションを使用することと似ています。そのため、入力データの全体の要約は使用できません。ただし、CLASS ステートメントとは異なり、BY ステートメントでは BY ステートメントの変数でデータを事前に並べ替えるかインデックス付けする必要があります。

NWAY オプションを使用すると、PROC MEANS がすべての分類変数を要約するにはメモリが不足することがあります。一部の分類変数を BY ステートメントに移動できます。最大活用するには、すでに並べ替えられた、または最も多い一意の値を含む BY ステートメントに分類変数を移動します。

CLASS ステートメントと BY ステートメントを使用して、データを BY グループ内の分類変数の水準別に分析できます。“例 3: BY ステートメントを分類変数とともに使用する” (1139 ページ)を参照してください。

分類変数の欠損値の PROC MEANS による処理方法

デフォルトでは、オブザベーションに分類変数の欠損値が含まれている場合、PROC MEANS はそのオブザベーションを分析から除外します。PROC ステートメントで MISSING オプションを指定すると、プロシジャは欠損値を分類変数の組み合わせに有効な水準とみなします。

CLASS ステートメントで MISSING オプションを指定すると、個別の分類変数に対する欠損値の受け入れを制御できます。

コンピュータリソース

PROC MEANS が使用できる一意の分類値の合計は、利用可能なコンピュータメモリの量によって異なります。詳細については、“コンピュータリソース” (1095 ページ)を参照してください。

GROUPINTERNAL オプションは、コンピュータパフォーマンスを向上させることができます。グループ化プロセスが分類変数の内部値に基づいているためです。数値分類変数に出力形式が割り当てられておらず、GROUPINTERNAL を指定しない場合、PROC MEANS はデフォルト出力形式、BEST12.を使用して文字列として数値をフォーマットします。PROC MEANS はこれらの数値変数を文字値別にグループ化します。これにはより多くの時間とコンピュータメモリが必要になります。

FREQ ステートメント

各オブザベーションの度数を含む数値変数を指定します。

参照項目: “FREQ” (72 ページ)

構文

FREQ *variable*;

必須引数

variable
値がオブザベーションの度数を表す数値変数を指定します。FREQ ステートメントを使用する場合、プロシジャは各オブザベーションが *n* オブザベーションを表すとみなします。*n* は *variable* の値です。*n* は整数でない場合、切り捨てられます。*n* が 1 未満または欠損値の場合、プロシジャはそのオブザベーションを統計量の計算に使用しません。

度数変数の合計は、オブザベーションの合計数を表します。

注: FREQ 変数は、OUTPUT ステートメントで IDGROUP 構文を使用するときの PROC MEANS による複数の極値の識別方法に影響しません。

ID ステートメント

追加の変数を出力データセットに含めます。

参照項目: ID グループ仕様の説明については、“OUTPUT ステートメント” (1118 ページ)を参照してください。

構文

ID *variable(s)*;

必須引数

variable(s)
PROC MEANS がそのオブザベーショングループの最大値を出力データセットに含める 1 つ以上の変数を入力データセットから識別します。

操作 PROC ステートメントで IDMIN を使用して、ID 変数の最小値を出力データセットに含めます。

ヒント PROC ステートメントで PRINTIDVARS オプションを使用して、ID 変数の値を表示出力に含めます。

詳細

ID 変数の値の選択

ID ステートメントで変数を 1 つだけ指定する場合、指定されているオブザベーションに対する ID 変数の値が、入力データセットの対応するオブザベーショングループで検出される最大(最小)値となります。ID ステートメントで複数の変数を指定すると、PROC MEANS は ID ステートメントの変数を表示されている順序で処理し、最大値を選択します。PROC MEANS は最初の ID 変数の値を比較して、使用するオブザベーションをすべての ID 変数から決定します。複数のオブザベーションに同一の最大(最小) ID 値が含まれている場合、PROC MEANS は 2 番目と後続の ID 変数値を“タイブレーカー”として使用します。いかなる場合でも、すべての ID 値は、指定されている BY グループまたは種類内の分類水準に対する同一のオブザベーションから取得されます。

PROC MEANS による最大値を決定するための文字値の比較方法については、“文字変数の並べ替え順序 ” (1788 ページ)を参照してください。

OUTPUT ステートメント

統計量を新しい SAS データセットに書き込みます。

ヒント: 複数の OUTPUT ステートメントを使用して、複数の OUT=データセットを作成できます。

例:
“例 8: 出力統計量を計算する” (1155 ページ)
“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1157 ページ)
“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1159 ページ)
“例 11: 出力統計量を使用し、極値を識別する” (1161 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)

構文

OUTPUT <OUT=*SAS-data-set*> <*output-statistic-specification(s)*> <*id-group-specification(s)*> <*maximum-id-specification(s)*> <*minimum-id-specification(s)*> </ *option(s)*>;

オプション引数

OUT=*SAS-data-set*
新しい出力データセットに名前を付けます。*SAS-data-set* が存在しない場合、PROC MEANS が作成します。OUT=を省略すると、データセットの名前は DATA*n* となります。*n* は、名前を一意のものにする最小整数です。

デフォルト DATA*n*

ヒント データセットオプションを OUT=オプションと使用できます。

output-statistic-specification(s)
OUT=データセットに格納する統計量を指定し、その統計量を含む 1 つ以上の変数に名前を付けます。*output-statistic-specification* の形式は、次のとおりです。

statistic-keyword<(*variable-list*)>=<*name(s)*>
この場合、

statistic-keyword
出力データセットに格納する統計量を指定します。利用可能な統計量キーワードは、次のとおりです。

記述統計量キーワード	
CSS	RANGE
CV	SKEWNESS | SKEW
KURTOSIS | KURT	STDDEV | STD
LCLM	STDERR
MAX	SUM
MEAN	SUMWGT
MIN	UCLM
MODE	USS
N	VAR
NMISS	
四分位範囲統計量キーワード	
MEDIAN | P50	Q3 | P75
P1	P90
P5	P95
P10	P99
P20	P30
P40	P60
P70	P80
Q1 | P25	QRANGE
仮説検定キーワード	
PROBT | PRT	T

デフォルトでは、出力データセットの統計量は分析変数の出力形式、入力形式、ラベルを自動的に継承します。ただし、N、NMISS、SUMWGT、USS、CSS、VAR、CV、T、PROBT、PRT、SKEWNESS、KURTOSIS に対して計算された統計量は、分析変数の出力形式を継承しません。この出力形式がこれらの統計量に無効な場合があるためです(ドル出力形式、日時出力形式など)。

制限事項 *variable* と *name(s)*を省略すると、AUTONAME オプションも使用しない限り、PROC MEANS は *statistic-keyword* を単一の OUTPUT ステートメントで一度だけ使用できるようにします。

参照項目 “例 8: 出力統計量を計算する” (1155 ページ)

“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1157 ページ)

“例 11: 出力統計量を使用し、極値を識別する” (1161 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)

variable-list
統計量を出力データセットに格納する 1 つ以上の数値分析変数の名前を指定します。

デフォルト すべての数値分析変数

name(s)
分析変数統計量を含む出力データセットの変数に対し 1 つ以上の名前を指定します。たとえば、最初の名前には最初の分析変数に対する統計量が含まれます。2 番目の名前には 2 番目の分析変数の統計量が含まれます。

デフォルト 分析変数名。AUTONAME を指定する場合、デフォルトは分析変数名と *statistic-keyword* の組み合わせです。*output-statistic-specification* を使用せずに CLASS ステートメントと OUTPUT ステートメントを使用すると、出力データセットには分類変数の各組み合わせに対し、N、MIN、MAX、MEAN および STD の値という 5 つのオブザベーションが含まれます。WEIGHT ステートメント、または VAR ステートメントで WEIGHT オプションを使用すると、出力データセットには分類変数の各組み合わせに対する重みの合計(SUMWGT)を含むオブザベーションも含まれます。

操作 *variable-list* を指定する場合、PROC MEANS は分析変数を指定する順序を使用して、統計量を出力データセット変数に格納します。

ヒント AUTONAME オプションを使用すると、PROC MEANS は複数の変数と統計量に対し、一意の名前を生成します。

参照項目 “例 8: 出力統計量を計算する” (1155 ページ)

id-group-specification
機能を組み合わせ、ID ステートメント、PROC ステートメントの IDMIN オプション、OUTPUT ステートメントの MAXID オプションと MINID オプションを拡張して、複数の極値を識別する OUT=データセットを作成します。*id-group-specification* の形式は、次のとおりです。

IDGROUP (<MIN | MAX (*variable-list-1*) <…MIN | MAX (*variable-list-n*)>> <<MISSING> <OBS> <LAST>> OUT <[*n*]>
(*id-variable-list*)=<*name(s)*>)

MIN|MAX(*variable-list*)
選択基準を指定し、*variable-list* で指定された 1 つ以上の入力データセット変数の極値を決定します。MIN を使用して最小極値を決定し、MAX を使用して最大極値を決定します。

複数の選択変数を指定すると、*n* 極値の選択に対するオブザベーションの並べ替えが、PROC SORT が複数の BY 変数を含むデータを並べ替えるのと同じ方法で行われます。PROC MEANS は、変数値を単一のキーにまとめます。MAX (*variable-list*)選択基準は、PROC SORT と、BY ステートメントで DESCENDING オプションを選択することに類似しています。

デフォルト MIN または MAX を指定しない場合、PROC MEANS はオブザベーション番号をオブザベーションを出力するための選択基準として使用します。

制限事項 矛盾した基準を指定すると、PROC MEANS は最初の選択基準のみ使用します。

操作 複数のオブザベーションですべての MIN 変数または MAX 変数に同じ極値が含まれている場合、PROC MEANS はオブザベーション番号を使用して、出力に書き込むオブザベーションを解決します。デフォルトでは、PROC MEANS は最初のオブザベーションを使用して、関係を解決します。ただし、LAST オプションを指定すると、PROC MEANS は最後のオブザベーションを使用して関係を解決します。

LAST
OUT=データセットに最後のオブザベーション(または *n* が指定されている場合は最後の *n* オブザベーション)からの値が含まれるように指定します。LAST を指定しない場合、OUT=データセットには、最初のオブザベーション(または *n* が指定されている場合は最初の *n* オブザベーション)からの値が含まれます。OUT=データセットには、複数のオブザベーションが含まれている場合があります。これは、最後の(最初の)オブザベーションの値に加え、OUT=データセット値には分類変数値の組み合わせによって定義される各サブグループ水準の最後の(最初の)オブザベーションからの値が含まれるためです。

操作 MIN または MAX を指定し、複数のオブザベーションに同じ極値が含まれている場合、PROC MEANS はオブザベーション番号を使用して、OUT=データセットに保存するオブザベーションを解決します。LAST を指定すると、PROC MEANS は関係の解決に後のオブザベーションを使用します。LAST を使用しない場合、PROC MEANS は関係の解決に前のオブザベーションを使用します。

MISSING
欠損値が選択基準で使用されるように指定します。

別名 MISS

OBS
極値が見つかった入力データセットにオブザベーション数を含む OUT=データセットに_OBS_変数を含めます。

操作 WHERE 処理を使用すると、_OBS_の値は、入力データセットのオブザベーションの場所に一致しないことがあります。

n]を使用して複数の極値を出力に書き込む場合、PROC MEANS は *n*_OBS_変数を作成し、接尾辞 *n* を使用して変数名を作成します。*n* は、1 から *n* までの連続した整数です。

[n]

OUT=データセットに含める *id-variable-list* の各変数に対する極値の数を指定します。PROC MEANS は *n* の新規変数を作成し、接尾辞 *_n* を使用して変数名を作成します。*n* は、1 から *n* までの連続した整数です。

デフォルトでは、PROC MEANS は、要求された各種類の水準ごとに 1 つの極値を決定します。*n* が 1 より大きい場合、*n* 極値が各種類の水準ごとに出力されます。*n* が 1 より大きく、極値選択を要求する場合、時間計算量は $O(T * N \log_2 n)$になります。*T*は要求された種類の数であり、*N*は入力データセットのオブザベーションの数です。比較すると、データセット全体をグループ化するための時間計算量は $O(N \log_2 N)$になります。

デフォルト 1

範囲 1 から 100 までの整数

例 たとえば、変数ごとに 2 つの最小極値を出力するには、次を使用します。

```
idgroup(min(x) out[2](x y z)=MinX MinY MinZ);
```

OUT=データセットには、変数 MinX_1、MinX_2、MinY_1、MinY_2、MinZ_1、MinZ_2 が含まれます。

(id-variable-list)

PROC MEANS がその値を OUT=データセットに含める 1 つ以上の入力データセット変数を識別します。PROC MEANS は、指定する選択基準(MIN、MAX および LAST)によって出力するオブザベーションを決定します。

別名 IDGRP

要件 まず MIN|MAX 選択基準を選択し、OUT(*id-variable-list*)=をサブオプション MISSING、OBS、LAST の後に指定する必要があります。

ヒント ID ステートメントの動作を模倣するには、*id-group-specification* を使用し、また OUTPUT ステートメントで *maximum-id-specification* または *minimum-id-specification* を使用することができます。

出力データセットに極値とその他の ID 変数を含める場合、それらを *id-variable-list* に含めると個別の統計量を求めるよりも効率的です。たとえば、ステートメント **`output idgrp(max(x) out(x a b)= );`**は、ステートメント **`output idgrp(max(x) out(a b)= ) max(x)=;`**より効率的です。

参照項目 “例 8: 出力統計量を計算する” (1155 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)

name(s)

OUT=データセットの変数に対し 1 つ以上の名前を指定します。

デフォルト *name* を省略すると、PROC MEANS は *id-variable-list* の変数名を使用します。

ヒント AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注意:

IDGROUP 構文を使用して、同じ名前の出力変数を作成できます。 このアクションが発生するのは、出力データセットに最初の変数が表示される場合だけです。AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注: 指定する新しい変数名が分析変数と ID 変数の組み合わせよりも少ない場合、その他の出力変数は、PROC MEANS が新しい変数名のリストを使い果たすとすぐに ID 変数の対応する名前を使用します。

maximum-id-specification(s)

1 つ以上の ID 変数が分析変数の最大値と関連付けられるように指定します。*maximum-id-specification* の形式は、次のとおりです。

MAXID <(*variable-1* <(*id-variable-list-1*)> <…*variable-n* <(*id-variable-list-n*)>>)> = *name(s)*

variable

PROC MEANS が最大値を決定する数値分析変数を識別します。PROC MEANS は、1 つの変数に対して複数の最大値を決定できます。これは、全体の最大値に加え、分類変数値の組み合わせによって定義されるサブグループ水準にも最大値が含まれているためです。

ヒント ID ステートメントを使用し、*variable* を省略すると、PROC MEANS はすべての分析変数を使用します。

id-variable-list

値が分析変数の最大値を含むオブザベーションを識別する 1 つ以上の変数を識別します。

デフォルト ID ステートメント変数

name(s)

各分析変数の最大値と関連付けられている ID 変数の値を含む新しい変数の名前を指定します。

注 複数のオブザベーションに分類水準内の最大値が含まれている場合、PROC MEANS は出力データセットのこれらのオブザベーションのうち最初のものだけに対する ID 変数の値を保存します。

ヒント ID ステートメントを使用し、*variable* と *id-variable* を省略すると、PROC MEANS はすべての ID ステートメント変数と各分類変数を関連付けます。このため、各分析変数に対し、出力データセットで作成される変数の数は ID ステートメントで指定する変数の数と同じです。

AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

参照項目 “例 11: 出力統計量を使用し、極値を識別する” (1161 ページ)

注意:

MAXID 構文を使用して、同じ名前の出力変数を作成できます。 このアクションが発生するのは、出力データセットに最初の変数が表示される場合だけです。AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注: 指定する新しい変数名が分析変数と ID 変数の組み合わせよりも少ない場合、その他の出力変数は、PROC MEANS が新しい変数名のリストを使い果たすとすぐに ID 変数の対応する名前を使用します。

minimum-id-specification

maximum-id-specification の説明を参照してください。このオプションは、PROC MEANS が最大値ではなく最小値を決定することを除けば、まったく同じように作動します。*minid-specification* の形式は、次のとおりです。

MINID<(*variable-1* <(*id-variable-list-1*)> <…*variable-n* <(*id-variable-list-n*)>>)> = *name(s)*

MINID は明示的な変数リストなしで使用される場合、次のさらに詳細な IDGROUP 構文例と類似しています。

IDGRP(min(x) missing out(id_variable)=idminx) idgrp(min(y) missing out(id_variable)=idminy)

1 つ以上の分類変数に欠損値が含まれている場合、id_variable 値は、MIN 統計量に対する値を含むオブザベーションではなく、欠損値を含むオブザベーションに対応します。

オプション

は、次の項目のうちいずれかになります。

AUTOLABEL

PROC MEANS が統計量名を変数ラベルの終わりに追加することを指定します。分析変数にラベルがない場合、PROC MEANS は統計量名を分析変数名に追加してラベルを作成します。

参照項目 “例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)

AUTONAME

OUTPUT ステートメントで変数名を割り当てない場合、PROC MEANS は出力統計量に対して一意の変数名を作成することを指定します。このアクションは、統計量の派生元入力変数名の終わりに *statistic-keyword* を追加して実行されます。

たとえば、ステートメント `output min(x)=/autoname;` では、出力データセットに x_Min 変数が生成されます。

AUTONAME は、SAS の内部メカニズムを使用して出力データセットの変数名の競合問題を自動的に解決します。変数が重複しても、エラーは発生しません。

その結果、ステートメント `output min(x)= min(x)=/autoname;` では、出力データセットに 2 つの変数 x_Min と x_Min2 が生成されます。

新しい変数名が 32 文字を超えると、variable-name 部分が切り捨てられます。

参照項目 “例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)

KEEPLEN

出力データセットの統計量は、それらの派生に使用される分析変数の長さを継承することを指定します。

注意:

分析変数の長さが原因で PROC MEANS が統計量の値を切り捨てるか丸めると、数値精度を永久に失います。ただし、統計量の精度は入力の精度と一致します。

LEVELS
出力データセットに_LEVEL_という名前の変数を含めます。この変数には、分類変数の値(_TYPE_変数の値)の一意の組み合わせを示す 1 から *n* までの値が含まれます。

参照項目 “出力データセット” (1133 ページ)

“例 8: 出力統計量を計算する” (1155 ページ)

NOINHERIT
統計量を含む出力データセットの変数は、それらを派生するときに使用される分析変数の属性(ラベルと出力形式)を継承しないことを指定します。

操作 オプションが使用されない場合(暗黙の INHERIT)、統計量は入力分析変数の属性、ラベル、出力形式を継承します。OUTPUT ステートメントで INHERIT オプションが使用されると、統計量は入力分析変数の長さ、ラベルと出力形式を継承します。

ヒント デフォルトでは、出力データセットに、各分析変数と、N、MIN、MAX、MEAN、STDDEV を含む 5 つのオブザベーションに対する 1 つの出力変数が含まれます。NOINHERIT を指定しない場合、この変数は分析変数の出力形式を継承します。この出力形式は N 統計量に対して無効である場合があります(日時出力形式など)。

WAYS
出力データセットに_WAY_という名前の変数を含めます。この変数には、PROC MEANS が TYPE 値を作成するために組み合わせる分類変数の数を指定する、1 から分類変数の最大数までの値が 1 つ含まれます。

参照項目 “出力データセット” (1133 ページ)

“WAYS ステートメント” (1128 ページ)

“例 8: 出力統計量を計算する” (1155 ページ)

TYPES ステートメント

生成する分類変数の可能な組み合わせを決定します。

要件 CLASS ステートメント

参照項目: “出力データセット” (1133 ページ)

例: “例 2: 分類変数を使用し、記述統計量を計算する” (1136 ページ)
“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1144 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1164 ページ)

構文

TYPES *request(s)*;

必須引数

request(s)

PROC MEANS が種類の作成に使用する分類変数の2^kの組み合わせを指定します。この*k*は分類変数の数です。要求には、1 つの分類変数名、アスタリスクまたは()で区切られた複数の分類変数名が含まれます。

分類変数の組み合わせをすばやく要求するには、複数の変数をかっこで囲み、他の変数または変数の組み合わせを結合してグループ化構文を使用します。たとえば、次のステートメントはグループ化構文を示しています。

要求	等しい
`types A*(B C);`	`types A*B A*C;`
`types (A B)*(C D);`	`types A*C A*D B*C B*D;`
`types (A B C)*D;`	`types A*D B*D C*D;`

操作 CLASSDATA=オプションは、NWAY 種類に制約を設定します。PROC MEANS は、他のすべての種類を結果の NWAY 種類から派生したように生成します。

ヒント ( )を使用して全体合計を要求します(_TYPE_=0)。

出力データセットですべての種類が必要でない場合、WHERE 句をデータセットに適用するのではなく、TYPES ステートメントを使用して特定のサブタイプを指定します。これを行うことにより、時間とコンピュータメモリが節約されます。

詳細

出力における分析の順序

分析は、PROC MEANS によって計算される、_TYPE_変数の値が増える順序で出力に書き込まれます。_TYPE_変数には、分類変数の各組み合わせに対し一意の値が含まれています。値は TYPES ステートメントではなく、CLASS ステートメントの指定方法によって決定されます。そのため、

```
class A B C;
types (A B)*C;
```

を指定すると、B*C 分析(_TYPE_=3)が最初に書き込まれ、A*C 分析(_TYPE_=5)が後に続きます。ただし、

```
class B A C;
types (A B)*C;
```

を指定すると、A*C 分析が最初に実行されます。

_TYPE_変数は、出力データセットが要求されなくても計算されます。_TYPE_変数の詳細については、“出力データセット” (1133 ページ)を参照してください。

VAR ステートメント

分析変数を識別し、出力における順序を識別します。

デフォルト: VAR ステートメントを省略すると、PROC MEANS はその他のステートメントでリストされないすべての数値変数を標準化します。すべての変数が文字変数の場合、PROC MEANS はオブザベーションの単純なカウントを生成します。

ヒント: 複数の VAR ステートメントを使用できます。

参照項目: 63 章, “SUMMARY プロシジャ,” (1861 ページ)

例: “例 1: 特定の記述統計量を計算する” (1134 ページ)

構文

VAR *variable(s)*< / WEIGHT=*weight-variable*>;

必須引数

variable(s)
分析変数を識別し、結果での順序を識別します。

オプション引数

WEIGHT=*weight-variable*
その値が VAR ステートメントで指定される変数の値に重みを付ける数値変数を指定します。重み変数が整数である必要がないのは、次の表に、PROC MEANS による WEIGHT 変数のさまざまな値の処理方法を説明します。

重み値	PROC MEANS 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 未満	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

重み変数は、プロシジャによる範囲、極値、欠損値数の決定方法を変更しません。

制限事項 重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。

歪度と尖度は、WEIGHT オプションと使用できません。

注 バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

ヒント WEIGHT オプションを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。VARDEF=の説明を参照してください。

複数の VAR ステートメントで WEIGHT オプションを使用して、分析変数に対し異なる重みを指定します。

WAYS ステートメント

分類変数の一意の組み合わせを作成するための方法の数を指定します。

ヒント: TYPES ステートメントを使用して、分類変数の追加の組み合わせを指定します。

例: “例 6: すでに読み込まれている出力形式を分類変数とともに使用する” (1149 ページ)

構文

WAYS *list*;

必須引数

list

分類変数のすべての一意の組み合わせを作成するために組み合わせる分類変数の数を定義する 1 つ以上の整数を指定します。たとえば、可能なすべてのペアに対して 2 を指定し、可能なすべてのトリプルに対して 3 を指定できます。*list* は次の方法で指定できます。

- *m*
- *m1 m2 ... mn*
- *m1,m2,...,mn*
- *m* TO *n* <BY *increment*>
- *m1,m2,* TO *m3* <BY *increment*>, *m4*

範囲 0 から分類変数の最大数

参照項目 WAYS オプション

例 次のコードは、分類変数 A、B、C に対する 2 次元の種類を作成する例です。この WAYS ステートメントは、TYPES ステートメントで a*b、a*c、b*c を指定する場合と同じです。

```
class A B C ; ways 2;
```

WEIGHT ステートメント

統計量計算でのオブザベーションの重みを指定します。

参照項目: 重み付き統計量の計算方法および WEIGHT ステートメントを使用した例については、“WEIGHT” (74 ページ)を参照してください。

構文

WEIGHT *variable*;

必須引数

variable
分析変数の値に重みをつける数値変数を指定します。変数の値は、整数である必要はありません。次の表に、PROC MEANS による WEIGHT 変数のさまざまな値の処理方法を説明します。

重み値	PROC MEANS 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 未満	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

注意:
単一の極値重み値により、不正確な結果が生成される可能性があります。 1 つの(唯一の)重み値がその他の重み値より桁違いに大きい場合(49 重み値が 1、1 重み値が 1×10^{14} など)、特定の統計量は可能な正しい制限内にないことがあります。影響を受ける統計量は、2 次モーメント(標準偏差、修正平方和、分散、平均値の標準誤差など)に基づいています。特定の状況下では、警告は SAS ログに書き込まれません。

制限事項
重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。

歪度と尖度は、WEIGHT ステートメントと使用できません。

PROC MEANS は、重み変数がアクティブな場合、MODE を計算しません。代わりに、MODE の計算が必要で、重み変数がアクティブな場合、UNIVARIATE プロシジャを使用しようとします。

操作
VAR ステートメントで WEIGHT=オプションを使用して重み変数を指定すると、代わりに PROC MEANS はこの変数を使用してこれらの VAR ステートメント変数を重み付けます。

注
バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

ヒント
WEIGHT ステートメントを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。詳細については、“キーワードと式” (2078 ページ)で VARDEF=と、重み付き統計量の計算の説明を参照してください。

統計量の計算:MEANS プロシジャ

モーメント統計量の計算

PROC MEANS は、モーメント統計量(平均値、分散、歪度、尖度など)の計算にシングルパスアルゴリズムを使用します。統計式については、“キーワードと式” (2078 ページ)を参照してください。

信頼限界、仮説検定統計量、四分位範囲統計量に関する計算詳細について、次に示します。

信頼限界

キーワード CLM、LCLM、UCLM を使用して、平均値に対する信頼限界を計算できます。信頼限界はサンプル統計量の値の周囲に構成される範囲で、反復サンプリング時に指定される確率(ALPHA=)を含む対応する真の母集団値が含まれます。

平均値に対する両側$100(1-\alpha)$ %信頼区間には、上限と下限があります。

$$\overline{x} \pm t_{(1-\alpha/2;n-1)}\frac{s}{\sqrt{n}}$$

ここで、sは$\sqrt{\frac{1}{n-1}\Sigma\left(x_i-\overline{x}\right)^2}$であり、$t_{(1-\alpha/2;n-1)}$は自由度が$n-1$であるスチューデントの t 統計量の$(1-\alpha/2)$限界値です。

片側$100(1-\alpha)$%信頼区間は、次のように計算されます。

$$\overline{x} + t_{(1-\alpha;n-1)}\frac{s}{\sqrt{n}} \quad \text{(upper)}$$

$$\overline{x} - t_{(1-\alpha;n-1)}\frac{s}{\sqrt{n}} \quad \text{(lower)}$$

WEIGHT ステートメント、または VAR ステートメントで WEIGHT=を使用し、VARDEF=のデフォルト値、DF を使用する場合、重み付き平均に対する$100(1-\alpha)$ %信頼区間には、上限と下限があります。

$$\overline{y}_w \pm t_{(1-\alpha/2)}\frac{s_w}{\sqrt{\sum_{i=1}^{n} w_i}}$$

ここで、$\overline{y}_w$は重み付き平均、s_wは重み付き標準偏差、w_iは$i$$th$オブザベーションに対する重み、$t_{(1-\alpha/2)}$は自由度が$n-1$のスチューデントの t 分布の$(1-\alpha/2)$限界値です。

スチューデントの t 検定

PROC MEANS は、t 統計量を次のように計算します。

$$t = \frac{\overline{x}-\mu_0}{s/\sqrt{n}}$$

ここで、$\bar{x}$はサンプル平均、nは変数の非欠損値数、sはサンプル標準偏差です。帰無仮説では、母集団の平均値はμ_0に等しくなります。データ値がほぼ正常に分布されると、観測値(p-value)と同じ、またはそれ以上の極値である t 統計量の帰無仮説での確率が、t 分布(自由度$n - 1$)から取得されます。大きなnの場合、t 統計量は漸近的に z 検定と等しくなります。

WEIGHT ステートメント、または VAR ステートメントで WEIGHT=を使用し、VARDEF=のデフォルト値、DF を使用する場合、スチューデントの t 統計量は次のように計算されます。

$$t_w = \frac{\bar{y}_w - \mu_0}{s_w / \sqrt{\sum_{i=1}^{n} w_i}}$$

ここで、$\bar{y}_w$は重み付き平均、s_wは重み付き標準偏差、w_iは$i$$th$オブザベーションに対する重みになります。$t_w$統計量は、スチューデントの t 分布と$n - 1$自由度を含むものとして扱われます。PROC ステートメントで EXCLNPWGT オプションを使用すると、WEIGHT 変数の値が正のときはnが非欠損オブザベーション数となります。デフォルトでは、nが、WEIGHT 変数に対する非欠損オブザベーション数です。

分位点

オプション QMETHOD=、QNTLDEF=、QMARKERS=により、PROC MEANS の分位点計算方法が決定されます。QNTLDEF=は、分位点の数学的定義を扱います。“分位数と関連統計量” (2083 ページ)を参照してください。QMETHOD=は、PROC MEANS の入力データ処理方法の手法を扱います。次の 2 つの方法があります。

OS
: すべてのデータをメモリに読み込み、一意の値別に並べ替えます。

P2
: すべてのデータを分位点の概算に使用される固定サンプルサイズに累積します。

データセット A に数値変数 X に対する一意の値が 100、データセット B に数値変数 X に対する一意の値が 1000 含まれている場合、データセット B の QMETHOD=OS には、データセット A のものよりも 10 倍のメモリが必要になります。QMETHOD=P2 の場合、A と B の両方のデータセットには、分位点の生成に同じメモリ空間が必要です。

QMETHOD=P2 の手法は、Jain と Chlamtac (1985)によって考案された piecewise-parabolic (P^2)アルゴリズムに基づいています。P^2は、大きなデータセットに対し分位点を決定するためのワンパスアルゴリズムです。種類内の各水準の各変数に対し、一定量のメモリが必要です。ただし、シミュレーションスタディを使用した、一部の分位点(P1、P5、P95、P99)の信頼性のある推定は、裾の重い分布または歪度が高い分布を含むデータセットなど、一部のデータセットには可能でないことがあります。

オブザベーション数が QMARKERS=値未満である場合、QMETHOD=P2 の結果は QNTLDEF=5 のときの QMETHOD=OS と同じになります。重み付き分位点を計算するには、QMETHOD=OS を使用する必要があります。

結果:MEANS プロシジャ

欠損値

PROC MEANS は、統計量の計算前に分析変数に対する欠損値を除外します。分析変数はそれぞれ個別に処理されます。1 つの変数のオブザベーションに対する欠損値はその他の変数の計算に影響しません。ステートメントは、欠損値を次のように処理します。

- 分類変数にオブザベーションに対する欠損値が含まれている場合、PROC ステートメントまたは CLASS ステートメントで MISSING オプションを使用しない限り、PROC MEANS はそのオブザベーションを分析から除外します。
- BY 変数値または ID 変数値が欠損している場合、PROC MEANS はそれをその他の BY 変数値または ID 変数値と同様に処理します。欠損値により、別の BY グループが形成されます。
- FREQ 変数値が欠損している、または非正の場合、PROC MEANS はそのオブザベーションを分析から除外します。
- WEIGHT 変数値が欠損している場合、PROC MEANS はそのオブザベーションを分析から除外します。

PROC MEANS は、欠損値の数を表形式で示します。欠損値の数を表形式で示す前に、PROC MEANS は FREQ ステートメントの使用時に度数が非正のオブザベーションと、WEIGHT ステートメントの使用時(EXCLNPWGT オプションの使用時)に重みが欠損している、または非正のオブザベーションを除外します。この情報をプロシジャ出力にレポートするには、PROC ステートメントで NMISS 統計量キーワードを使用します。

出力の列幅

PROC ステートメントで FW=オプションを使用して、表示される統計量の列幅を制御します。出力形式を数値分類変数または ID 変数に割り当てない限り、PROC MEANS は FW=オプションの値を使用します。出力形式を数値分類変数または ID 変数に割り当てると、PROC MEANS は列幅を出力形式から直接決定します。CLASS ステートメントで PRELOADFMT オプションを使用すると、PROC MEANS は割り当てられている出力形式から分類変数の列幅を決定します。

N Obs 統計量

デフォルトで CLASS ステートメントを使用すると、PROC MEANS は N Obs という追加の統計量を表示します。この統計量は、オブザベーションの合計数、または PROC MEANS が分類水準ごとに処理する FREQ 変数のオブザベーションの合計をレポートします。PROC MEANS は、オブザベーションをこの合計から省略することがあります。1 つ以上の分類変数に欠損値があるため、あるいは EXCLUSIVE オプションを PRELOADFMT オプションまたは CLASSDATA=オプションと使用する場合の EXCLUSIVE オプションの影響のためです。このアクションと、WEIGHT 変数に欠損値が含まれている場合のオブザベーションの除外により、N Obs、N および NMISS の間に常に直接関係があるわけではありません。

出力データセットでは、N Obs の値は_FREQ_変数に格納されます。PROC ステートメントで NONOBS オプションを使用して、表示出力でこの情報を非表示にします。

出力データセット

PROC MEANS は、1 つ以上の出力データセットを作成できます。プロシジャは出力データセットを印刷しません。PROC PRINT、PROC REPORT または別の SAS レポートツールを使用して、出力データセットを表示します。

注: デフォルトでは、出力データセットの統計量は、分析変数の出力形式とラベルを自動的に継承します。ただし、N、NMISS、SUMWGT、USS、CSS、VAR、CV、T、PROBT、PRT、SKEWNESS および KURTOSIS に対して計算される統計量は、分析変数の出力形式を継承しません。この出力形式がこれらの統計量に無効である可能性があるためです。OUTPUT ステートメントで NOINHERIT オプションを使用して、その他の統計量が出力形式とラベル属性を継承しないようにします。

出力データセットには、次の変数が含まれている可能性があります。

- BY ステートメントで指定された変数。
- ID ステートメントで指定された変数。
- CLASS ステートメントで指定された変数。
- 分類変数に関する情報が含まれている変数_TYPE_.デフォルトでは、_TYPE_は数値変数です。PROC ステートメントで CHARTYPE を指定する場合、_TYPE_は文字変数になります。32 を超える分類変数を使用する場合、_TYPE_は自動的に文字変数になります。
- 指定出力水準が表すオブザベーション数を含む変数_FREQ_。
- 出力統計量と極値が含まれている OUTPUT ステートメントで要求される変数。
- 統計量キーワードを省略する場合にデフォルトの統計量の名前が含まれている変数_STAT_。
- LEVEL オプションを指定する場合の変数_LEVEL_。
- WAYS オプションを指定する場合の変数_WAY_。

_TYPE_の値は、PROC MEANS が統計量の計算に使用する分類変数の組み合わせを示します。_TYPE_の文字値は一連のゼロと 1 です。1 の値はそれぞれ種類のアクティブな分類変数を示します。たとえば、3 つの分類変数を使用して、PROC MEANS は種類 1 を 001、種類 5 を 101 として表します。

通常、出力データセットには、種類別の水準ごとに 1 つのオブザベーションが含まれています。ただし、OUTPUT ステートメントで統計量キーワードを省略すると、出力データセットには水準ごとに 5 つ(WEIGHT 変数を指定する場合は 6 つ)のオブザベーションが含まれます。そのため、出力データセットのオブザベーションの合計数は、適用可能な場合に 1、5 または 6 を乗算して求めるすべての種類の水準の合計に等しくなります。

CLASS ステートメント(_TYPE_ = 0)を省略すると、BY グループごとに出力水準が 1 つだけ常に存在します。CLASS ステートメントを使用すると、要求する各種類の水準数の上限は入力データセットのオブザベーション数と同じになります。デフォルトでは、PROC MEANS はすべての可能な種類を生成します。この場合、各 BY グループの水準の合計数の上限は、次と同じになります。

$$m \cdot \left(2^k - 1\right) \cdot n + 1$$

ここで、kは分類変数の数、nは入力データセットの指定 BY グループに対するオブザベーション数、mは 1、5 または 6 です。

PROC MEANS は、アクティブな各分類変数の一意の組み合わせ数から、指定種類に対する水準の実際の数を決定します。単一の水準は、フォーマットされた分類値が一致するすべての入力オブザベーションから構成されます。

次の図に、分類変数を 1 つ、2 つ、3 つ指定したときの_TYPE_の値とデータセットのオブザベーション数を示します。

図 37.1 *OUTPUT データセットにおける分類変数の影響*



C	B	A	_WAY_	_TYPE_	Subgroup defined by	Number of observations of this _TYPE_ and _WAY_ in the data set	Total number of observations in the data set
0	0	0	0	0	Total	1	
0	0	1	1	1	A	a	1+a
0	1	0	1	2	B	b	
0	1	1	2	3	A*B	a*b	1+a+b+a*b
1	0	0	1	4	C	c	
1	0	1	2	5	A*C	a*c	
1	1	0	2	6	B*C	b*c	1+a+b+a*b+c
1	1	1	3	7	A*B*C	a*b*c	+a*c+b*c+a*b*c
Character binary equivalent of _TYPE_ (CHARTYPE option)					A ,B ,C=CLASS variables	a, b, c,=number of levels of A, B, C, respectively	

例: MEANS プロシジャ

例 1: 特定の記述統計量を計算する

要素: PROC MEANS ステートメントオプション
統計量キーワード
FW=

VAR ステートメント

データセット: CAKE

詳細

この例では、次を行います。

- 分析変数を指定します。
- 指定キーワードに対する統計量を計算し、順番に表示します。
- 統計量のフィールド幅を指定します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data cake;
   input LastName $ 1-12 Age 13-14 PresentScore 16-17
         TasteScore 19-20 Flavor $ 23-32 Layers 34 ;
   datalines;
Orlando     27 93 80  Vanilla    1
Ramey       32 84 72  Rum        2
Goldston    46 68 75  Vanilla    1
Roe         38 79 73  Vanilla    2
Larsen      23 77 84  Chocolate  .
Davis       51 86 91  Spice      3
Strickland  19 82 79  Chocolate  1
Nguyen      57 77 84  Vanilla    .
Hildenbrand 33 81 83  Chocolate  1
Byron       62 72 87  Vanilla    2
Sanders     26 56 79  Chocolate  1
Jaeger      43 66 74             1
Davis       28 69 75  Chocolate  2
Conrad      69 85 94  Vanilla    1
Walters     55 67 72  Chocolate  2
Rossburger  28 78 81  Spice      2
Matthew     42 81 92  Chocolate  2
Becker      36 62 83  Spice      2
Anderson    27 87 85  Chocolate  1
Merritt     62 73 84  Chocolate  1
;

proc means data=cake n mean max min range std fw=8;

   var PresentScore TasteScore;

   title 'Summary of Presentation and Taste Scores';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

CAKE データセットを作成します。 CAKE には、ケーキ作りコンテストの次のデータが含まれています。各参加者の姓、年齢、出来ばえのスコア、味のスコア、ケーキの風味、ケーキの層の数。ケーキの層の数が 2 つのオブザベーションにありません。ケーキの風味が別のオブザベーションにありません。

```
data cake;
   input LastName $ 1-12 Age 13-14 PresentScore 16-17
         TasteScore 19-20 Flavor $ 23-32 Layers 34 ;
   datalines;
Orlando     27 93 80  Vanilla    1
Ramey       32 84 72  Rum        2
Goldston    46 68 75  Vanilla    1
Roe         38 79 73  Vanilla    2
Larsen      23 77 84  Chocolate  .
Davis       51 86 91  Spice      3
Strickland  19 82 79  Chocolate  1
Nguyen      57 77 84  Vanilla    .
Hildenbrand 33 81 83  Chocolate  1
Byron       62 72 87  Vanilla    2
Sanders     26 56 79  Chocolate  1
Jaeger      43 66 74             1
Davis       28 69 75  Chocolate  2
Conrad      69 85 94  Vanilla    1
Walters     55 67 72  Chocolate  2
Rossburger  28 78 81  Spice      2
Matthew     42 81 92  Chocolate  2
Becker      36 62 83  Spice      2
Anderson    27 87 85  Chocolate  1
Merritt     62 73 84  Chocolate  1
;
```

分析と分析オプションを指定します。 統計量キーワードは、統計量と出力での順序を指定します。FW=は、フィールド幅 8 を使用して統計量を表示します。

```
proc means data=cake n mean max min range std fw=8;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が PresentScore 変数と TasteScore 変数の統計量を計算するように指定します。

```
   var PresentScore TasteScore;
```

タイトルを指定します。

```
   title 'Summary of Presentation and Taste Scores';
run;
```

例 2: 分類変数を使用し、記述統計量を計算する

要素: PROC MEANS ステートメントオプション
MAXDEC=
CLASS ステートメント
TYPES ステートメント

データセット: GRADE

詳細

この例では、次を行います。

- 分類変数の 2 次元組み合わせのデータとすべてのオブザベーションにわたるデータを分析します。
- 表示される統計量に対する小数点以下桁数を制限します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data grade;
   input Name $ 1-8 Gender $ 11 Status $13 Year $ 15-16
         Section $ 18 Score 20-21 FinalGrade 23-24;
   datalines;
Abbott    F 2 97 A 90 87
Branford  M 1 98 A 92 97
Crandell  M 2 98 B 81 71
Dennison  M 1 97 A 85 72
Edgar     F 1 98 B 89 80
Faust     M 1 97 B 78 73
Greeley   F 2 97 A 82 91
Hart      F 1 98 B 84 80
Isley     M 2 97 A 88 86
Jasper    M 1 97 B 91 93
;

proc means data=grade maxdec=3;

   var Score;

   class Status Year;

   types () status*year;

   title 'Final Exam Grades for Student Status and Year of Graduation';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

GRADE データセットを作成します。 GRADE には、各生徒の姓、性別、大学生(1)または大学院生(2)のいずれかの区分、卒業見込年度、クラスセクション(A または B)、期末試験のスコア、クラスの最終成績が含まれています。

```
data grade;
   input Name $ 1-8 Gender $ 11 Status $13 Year $ 15-16
         Section $ 18 Score 20-21 FinalGrade 23-24;
   datalines;
Abbott    F 2 97 A 90 87
Branford  M 1 98 A 92 97
Crandell  M 2 98 B 81 71
Dennison  M 1 97 A 85 72
Edgar     F 1 98 B 89 80
Faust     M 1 97 B 78 73
Greeley   F 2 97 A 82 91
```

```
Hart      F 1 98 B 84 80
Isley     M 2 97 A 88 86
Jasper    M 1 97 B 91 93
;
```

デフォルトの統計量を生成して、分析オプションを指定します。 PROC MEANS ステートメントで統計量が指定されていないため、すべてのデフォルトの統計量(N、MEAN、STD、MIN、MAX)が生成されます。MAXDEC=は、表示される統計量を小数点以下 3 桁に制限します。

```
proc means data=grade maxdec=3;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が Score 変数の統計量を計算するように指定します。

```
var Score;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは分析をサブグループに分割します。Status と Year の一意の値の組み合わせはそれぞれサブグループを表します。

```
class Status Year;
```

分析するサブグループを指定します。 TYPES ステートメントは、GRADE データセットのすべてのオブザベーション、および結果が 4 つのサブグループとなる(Status と Year にはそれぞれ 2 つの一意の値が含まれているため)Status と Year の 2 次元組み合わせで分析が実行されるように要求します。

```
types () status*year;
```

タイトルを指定します。

```
title 'Final Exam Grades for Student Status and Year of Graduation';
run;
```

出力

PROC MEANS は、すべてのオブザベーション(_TYPE_=0)のデフォルト統計量および Status と Year の組み合わせの 4 つの分類水準(Status=1, Year=97; Status=1, Year=98; Status=2, Year=97; Status=2, Year=98)を表示します。

アウトプット 37.4 期末試験の成績

Final Exam Grades for Student Status and Year of Graduation

The MEANS Procedure

Analysis Variable : Score					
N Obs	N	Mean	Std Dev	Minimum	Maximum
10	10	86.000	4.714	78.000	92.000

Analysis Variable : Score							
Status	Year	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	97	3	3	84.667	6.506	78.000	91.000
	98	3	3	88.333	4.041	84.000	92.000
2	97	3	3	86.667	4.163	82.000	90.000
	98	1	1	81.000	.	81.000	81.000

例 3: BY ステートメントを分類変数とともに使用する

要素: PROC MEANS ステートメントオプション
統計量キーワード
BY ステートメント
CLASS ステートメント

他の要素: SORT プロシジャ

データセット: GRADE

詳細

この例では、次を行います。

- BY 値内の分類変数の組み合わせに対する分析を分割する
- BY ステートメントに対する並べ替え順序要件を表示する
- 最小値、最大値、中央値を計算する

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc sort data=Grade out=GradeBySection;
   by section;
run;

proc means data=GradeBySection min max median;
```

```
   by Section;

   var Score;

   class Status Year;

   title1 'Final Exam Scores for Student Status and Year of Graduation';
   title2 ' Within Each Section';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

GRADE データセットを並べ替えます。 PROC SORT はオブザベーションを変数 Section 別に並べ替えます。並べ替えは、Section を BY 変数として PROC MEANS ステップで使用するために必要です。

```
proc sort data=Grade out=GradeBySection;
   by section;
run;
```

分析を指定します。 統計量キーワードは、統計量と出力での順序を指定します。

```
proc means data=GradeBySection min max median;
```

データセットを BY グループに分割します。 BY ステートメントは、Section の各値に対し別の分析を生成します。

```
   by Section;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が Score 変数の統計量を計算するように指定します。

```
   var Score;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Status と Year の値別に分割します。このプログラムには TYPES ステートメントがないため、分析は各 BY グループ内の各サブグループに対して実行されます。

```
   class Status Year;
```

タイトルを指定します。

```
   title1 'Final Exam Scores for Student Status and Year of Graduation';
   title2 ' Within Each Section';
run;
```

出力

アウトプット 37.5 *期末試験のスコア*

Final Exam Scores for Student Status and Year of Graduation Within Each Section

The MEANS Procedure

Section=A

Analysis Variable : Score					
Status	Year	N Obs	Minimum	Maximum	Median
1	97	1	85.0000000	85.0000000	85.0000000
	98	1	92.0000000	92.0000000	92.0000000
2	97	3	82.0000000	90.0000000	88.0000000

Section=B

Analysis Variable : Score					
Status	Year	N Obs	Minimum	Maximum	Median
1	97	2	78.0000000	91.0000000	84.5000000
	98	2	84.0000000	89.0000000	86.5000000
2	98	1	81.0000000	81.0000000	81.0000000

例 4: CLASSDATA=データセットを分類変数とともに使用する

要素: PROC MEANS ステートメントオプション
- CLASSDATA=
- EXCLUSIVE
- FW=
- MAXDEC=
- PRINTALLTYPES

CLASS ステートメント

データセット: CAKE

CAKETYPE

詳細

この例では、次を行います。

- 表示される統計量のフィールド幅と小数点以下桁数を指定します。
- CLASSDATA=データセットの値のみを分類変数の組み合わせの水準として使用します。
- 範囲、中間値、最小値、最大値を計算します。
- 分析に分類変数のすべての組み合わせを表示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data caketype;
   input Flavor $ 1-10  Layers 12;
   datalines;
Vanilla    1
Vanilla    2
Vanilla    3
Chocolate  1
Chocolate  2
Chocolate  3
;

proc means data=cake range median min max fw=7 maxdec=0
           classdata=caketype exclusive printalltypes;

   var TasteScore;


   class flavor layers;

   title 'Taste Score For Number of Layers and Cake Flavor';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

CAKETYPE データセットを作成します。 CAKETYPE には、PROC MEANS 出力に表示する必要があるケーキの風味と層の数が含まれています。

```
data caketype;
   input Flavor $ 1-10  Layers 12;
   datalines;
Vanilla    1
Vanilla    2
Vanilla    3
Chocolate  1
Chocolate  2
Chocolate  3
;
```

分析と分析オプションを指定します。 FW=オプションはフィールド幅 7、MAXDEC=オプションは小数点以下桁数ゼロをそれぞれ使用して統計量を表示します。CLASSDATA= と EXCLUSIVE は、分類水準を CAKETYPE データセットにある値に制限します。PRINTALLTYPES は、出力に分類変数のすべての組み合わせを表示します。

```
proc means data=cake range median min max fw=7 maxdec=0
           classdata=caketype exclusive printalltypes;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が TasteScore 変数の統計量を計算するように指定します。

```
    var TasteScore;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Flavor と Layers の値別に分割します。これらの変数、およびこれらの変数のみが CAKETYPE データセットに表示される必要があります。

```
    class flavor layers;
```

タイトルを指定します。

```
    title 'Taste Score For Number of Layers and Cake Flavor';
run;
```

出力

PROC MEANS は、13 個のチョコレートケーキとバニラケーキに対する統計量を計算します。CLASSDATA=データセットに Layers の値として 3 が含まれているため、PROC MEANS は度数がゼロの場合でも 3 を分類値として使用します。

アウトプット 37.6 味のスコア

Taste Score For Number of Layers and Cake Flavor

The MEANS Procedure

Analysis Variable : TasteScore				
N Obs	Range	Median	Minimum	Maximum
13	22	80	72	94

Analysis Variable : TasteScore					
Layers	N Obs	Range	Median	Minimum	Maximum
1	8	19	82	75	94
2	5	20	75	72	92
3	0	.	.	.	.

Analysis Variable : TasteScore					
Flavor	N Obs	Range	Median	Minimum	Maximum
Chocolate	8	20	81	72	92
Vanilla	5	21	80	73	94

Analysis Variable : TasteScore						
Flavor	Layers	N Obs	Range	Median	Minimum	Maximum
Chocolate	1	5	6	83	79	85
	2	3	20	75	72	92
	3	0	.	.	.	.
Vanilla	1	3	19	80	75	94
	2	2	14	80	73	87
	3	0	.	.	.	.

例 5: 値のマルチラベル出力形式を分類変数とともに使用する

要素: PROC MEANS ステートメントオプション
統計量キーワード
FW=

NONOBS

CLASS ステートメントオプション

MLF

ORDER=

TYPES ステートメント

他の要素: FORMAT プロシジャ

FORMAT ステートメント

データセット: CAKE

詳細

この例では、次を行います。

- 指定キーワードに対する統計量を計算し、順番に表示します。
- 統計量のフィールド幅を指定します。
- オブザベーションの合計数を含む列を非表示にします。
- ケーキの風味の 1 次元組み合わせと、ケーキの風味と参加者の年齢の 2 次元組み合わせに対しデータを分析します。
- ユーザー定義出力形式を分類変数に割り当てます。
- マルチラベル出力形式を分類変数の水準として使用します。
- ケーキの風味の水準を度数カウントの降順で並べ替え、年齢の水準をフォーマットされた値の昇順で並べ替えます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=64;

proc format;
   value $flvrfmt
                'Chocolate'='Chocolate'
                'Vanilla'='Vanilla'
                'Rum','Spice'='Other Flavor';
   value agefmt (multilabel)
                  15 - 29='below 30 years'
                  30 - 50='between 30 and 50'
                  51 - high='over 50 years'
                  15 - 19='15 to 19'
                  20 - 25='20 to 25'
                  25 - 39='25 to 39'
                  40 - 55='40 to 55'
                  56 - high='56 and above';
run;

proc means data=cake fw=6 n min max median nonobs;

   class flavor/order=data;
   class  age /mlf order=fmt;

    types flavor flavor*age;

   var TasteScore;

   format age agefmt. flavor $flvrfmt.;
```

```
   title 'Taste Score for Cake Flavors and Participant''s Age';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=64;
```

$FLVRFMT.出力形式と AGEFMT.出力形式を作成します。 PROC FORMAT は、ユーザー定義出力形式を作成して、ケーキの風味と参加者の年齢を分類します。MULTILABEL は、Age に対しマルチラベル出力形式を作成します。マルチラベル出力形式は、複数のラベルを同じ値に割り当てることができる出力形式です。この場合は、範囲が重複するためです。各値は、発生する範囲ごとに出力に表示されます。

```
proc format;
   value $flvrfmt
               'Chocolate'='Chocolate'
               'Vanilla'='Vanilla'
               'Rum','Spice'='Other Flavor';
   value agefmt (multilabel)
                 15 - 29='below 30 years'
                 30 - 50='between 30 and 50'
                 51 - high='over 50 years'
                 15 - 19='15 to 19'
                 20 - 25='20 to 25'
                 25 - 39='25 to 39'
                 40 - 55='40 to 55'
                 56 - high='56 and above';
run;
```

分析と分析オプションを指定します。 FW=は、フィールド幅 6 を使用して統計量を表示します。統計量キーワードは、統計量と出力での順序を指定します。NONOBS は、N Obs 列を非表示にします。

```
proc means data=cake fw=6 n min max median nonobs;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Flavor と Age の値別に分割します。ORDER=DATA は、値を入力データセットの順序に従って並べ替えます。ORDER=FMT は、Age の水準をフォーマットされた値の昇順で並べ替えます。MLF は、マルチラベル値出力形式が Age に使用されるように指定します。

```
   class flavor/order=data;
   class  age /mlf order=fmt;
```

分析するサブグループを指定します。 TYPES ステートメントは、Flavor の 1 次元の組み合わせと、Flavor と Age の 2 次元の組み合わせに対し分析を要求します。

```
    types flavor flavor*age;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が TasteScore 変数の統計量を計算するように指定します。

```
   var TasteScore;
```

出力をフォーマットします。 FORMAT ステートメントは、ユーザー定義出力形式をこの分析の対象となる Age 変数と Flavor 変数に割り当てます。

```
format age agefmt. flavor $flvrfmt.;
```

タイトルを指定します。

```
    title 'Taste Score for Cake Flavors and Participant''s Age';
run;
```

出力

分類変数の 1 次元の組み合わせは、2 次元の組み合わせの前に表示されます。フィールド幅 6 は、統計量を小数点以下 4 桁に切り捨てます。Age と Flavor の 2 次元の組み合わせの場合、オブザベーションの合計数は Flavor の 1 次元の組み合わせよりも大きくなります。このような状況が発生するのは、1 つの内部値を複数のフォーマットされた値にマップする Age のマルチラベル出力形式のためです。Flavor の水準の順序は、各水準の度数カウントに基づいています。Age の水準の順序は、ユーザー定義出力形式の順序に基づいています。

アウトプット 37.7 ケーキの風味に対する味のスコア

Taste Score for Cake Flavors and Participant's Age

The MEANS Procedure

Analysis Variable : TasteScore				
Flavor	N	Minimum	Maximum	Median
Vanilla	6	73.00	94.00	82.00
Other Flavor	4	72.00	91.00	82.00
Chocolate	9	72.00	92.00	83.00

Analysis Variable : TasteScore					
Flavor	Age	N	Minimum	Maximum	Median
Vanilla	25 to 39	2	73.00	80.00	76.50
	40 to 55	1	75.00	75.00	75.00
	56 and above	3	84.00	94.00	87.00
	below 30 years	1	80.00	80.00	80.00
	between 30 and 50	2	73.00	75.00	74.00
	over 50 years	3	84.00	94.00	87.00
Other Flavor	25 to 39	3	72.00	83.00	81.00
	40 to 55	1	91.00	91.00	91.00
	below 30 years	1	81.00	81.00	81.00
	between 30 and 50	2	72.00	83.00	77.50
	over 50 years	1	91.00	91.00	91.00
Chocolate	15 to 19	1	79.00	79.00	79.00
	20 to 25	1	84.00	84.00	84.00
	25 to 39	4	75.00	85.00	81.00
	40 to 55	2	72.00	92.00	82.00
	56 and above	1	84.00	84.00	84.00
	below 30 years	5	75.00	85.00	79.00
	between 30 and 50	2	83.00	92.00	87.50
	over 50 years	2	72.00	84.00	78.00

例 6: すでに読み込まれている出力形式を分類変数とともに使用する

要素: PROC MEANS ステートメントオプション
- COMPLETETYPES
- FW=
- MISSING
- NONOBS

CLASS ステートメントオプション
- EXCLUSIVE
- ORDER=
- PRELOADFMT

WAYS ステートメント

他の要素: FORMAT プロシジャ

FORMAT ステートメント

データセット: CAKE

詳細

この例では、次を行います。

- 統計量のフィールド幅を指定します。
- オブザベーションの合計数を含む列を非表示にします。
- 度数がゼロの場合でも、分類変数値のすべての可能な組み合わせを分析に含めます。
- 欠損値を有効な分類水準とします。
- 分類変数の 1 次元および 2 次元の組み合わせを分析します。
- ユーザー定義出力形式を分類変数に割り当てます。
- ユーザー定義出力形式のすでに読み込まれている範囲のみ分類変数の水準として使用します。
- 結果をフォーマットされたデータの値別に並べ替えます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=64;

proc format;
   value layerfmt 1='single layer'
                  2-3='multi-layer'
                  .='unknown';
   value $flvrfmt (notsorted)
                  'Vanilla'='Vanilla'
                  'Orange','Lemon'='Citrus'
                  'Spice'='Spice'
                  'Rum','Mint','Almond'='Other Flavor';
run;

proc means data=cake fw=7 completetypes missing nonobs;

   class flavor layers/preloadfmt exclusive order=data;
```

```
    ways 1 2;

    var TasteScore;

    format layers layerfmt. flavor $flvrfmt.;

    title 'Taste Score For Number of Layers and Cake Flavors';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=64;
```

**LAYERFMT.出力形式と$FLVRFMT.出力形式を作成します。** PROC FORMAT は、ユーザー定義出力形式を作成して、ケーキの層の数とケーキの風味を分類します。NOTSORTED は$FLVRFMT を並べ替えず、出力形式値の元の順序を維持します。

```
proc format;
   value layerfmt 1='single layer'
                  2-3='multi-layer'
                  .='unknown';
   value $flvrfmt (notsorted)
                  'Vanilla'='Vanilla'
                  'Orange','Lemon'='Citrus'
                  'Spice'='Spice'
                  'Rum','Mint','Almond'='Other Flavor';
run;
```

デフォルトの統計量を生成して、分析オプションを指定します。 FW=は、フィールド幅 7 を使用して統計量を表示します。COMPLETETYPES には、度数がゼロの分類水準が含まれています。MISSING は、欠損値をすべての分類変数に有効な値とみなします。NONOBS は、N Obs 列を非表示にします。特定の分析が要求されないため、すべてのデフォルトの分析がすべて実行されます。

```
proc means data=cake fw=7 completetypes missing nonobs;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Flavor と Layers の値別に分割します。PRELOADFMT と EXCLUSIVE は、水準をユーザー定義出力形式のすでに読み込まれた値に制限します。ORDER=DATA は、Flavor と Layer の水準をフォーマットされたデータ値別に並べ替えます。

```
    class flavor layers/preloadfmt exclusive order=data;
```

分析するサブグループを指定します。 WAYS ステートメントは、分析変数の 1 次元および 2 次元の組み合わせを要求します。

```
    ways 1 2;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が TasteScore 変数の統計量を計算するように指定します。

```
    var TasteScore;
```

出力をフォーマットします。 FORMAT ステートメントは、ユーザー定義出力形式をこの分析の対象となる Flavor 変数と Layers 変数に割り当てます。

```
    format layers layerfmt. flavor $flvrfmt.;
```

タイトルを指定します。

```
    title 'Taste Score For Number of Layers and Cake Flavors';
run;
```

出力

分類変数の 1 次元の組み合わせは、2 次元の組み合わせの前に表示されます。PROC MEANS は、オブザベーションの度数がゼロ(この場合はかんきつ類)の場合でも、ユーザー定義出力形式のすでに読み込まれた範囲に表示される水準値のみレポートします。PROC MEANS は、指定オブザベーションの単一の分類値の除外に基づいてオブザベーション全体をリジェクトします。そのため、層の数が不明な場合、統計量は 1 つのオブザベーションについてのみ計算されます。チョコレート風味が Flavor 用のすでに読み込まれているユーザー定義出力形式に含まれていないため、その他のオブザベーションは除外されます。水準の順序は、ユーザー定義出力形式の順序に基づいています。PROC FORMAT は自動的に Layers 出力形式を並べ替え、Flavor 出力形式を並べ替えませんでした。

アウトプット 37.8 層の数に対する味のスコア

Taste Score For Number of Layers and Cake Flavors

The MEANS Procedure

Analysis Variable : TasteScore					
Layers	N	Mean	Std Dev	Minimum	Maximum
unknown	1	84.000	.	84.000	84.000
single layer	3	83.000	9.849	75.000	94.000
multi-layer	6	81.167	7.548	72.000	91.000

Analysis Variable : TasteScore					
Flavor	N	Mean	Std Dev	Minimum	Maximum
Vanilla	6	82.167	7.834	73.000	94.000
Citrus	0	.	.	.	.
Spice	3	85.000	5.292	81.000	91.000
Other Flavor	1	72.000	.	72.000	72.000

Analysis Variable : TasteScore						
Flavor	Layers	N	Mean	Std Dev	Minimum	Maximum
Vanilla	unknown	1	84.000	.	84.000	84.000
	single layer	3	83.000	9.849	75.000	94.000
	multi-layer	2	80.000	9.899	73.000	87.000
Citrus	unknown	0	.	.	.	.
	single layer	0	.	.	.	.
	multi-layer	0	.	.	.	.
Spice	unknown	0	.	.	.	.
	single layer	0	.	.	.	.
	multi-layer	3	85.000	5.292	81.000	91.000
Other Flavor	unknown	0	.	.	.	.
	single layer	0	.	.	.	.
	multi-layer	1	72.000	.	72.000	72.000

例 7: 平均の信頼限界の計算

要素: PROC MEANS ステートメントオプション
ALPHA=
FW=
MAXDEC=
CLASS ステートメント

データセット: Charity

詳細

この例では、次を行います。

- 統計量のフィールド幅と小数点以下桁数を指定します。
- 3 年間のデータの MoneyRaised と HoursVolunteered の平均値に対する両側 90% 信頼限界を計算します。

このデータがより大きいボランティアの母集団を表す場合、信頼限界は真の母平均に対し可能性のある値の範囲を提供します。

プログラム

```
data charity;
   input School $ 1-7 Year 9-12 Name $ 14-20 MoneyRaised 22-26
         HoursVolunteered 28-29;
   datalines;
Monroe  2007 Allison 31.65 19
Monroe  2007 Barry   23.76 16
Monroe  2007 Candace 21.11  5


     . . . more data lines . . .

Kennedy 2009 Sid     27.45 25
Kennedy 2009 Will    28.88 21
Kennedy 2009 Morty   34.44 25
;

proc means data=charity fw=8 maxdec=2 alpha=0.1 clm mean std;

   class Year;

   var MoneyRaised HoursVolunteered;

   title 'Confidence Limits for Fund Raising Statistics';
   title2 '2007-09';
run;
```

プログラムの説明

CHARITY データセットを作成します。 CHARITY には、高校生の慈善事業のボランティア活動に関する情報が含まれています。変数は、高校名、募金活動の年度、各生徒

の名前、各学生が集めた金額、各学生がボランティア活動をした時間数を提供します。DATA ステップは、次のデータセットを作成します。

```
data charity;
   input School $ 1-7 Year 9-12 Name $ 14-20 MoneyRaised 22-26
         HoursVolunteered 28-29;
   datalines;
Monroe  2007 Allison 31.65 19
Monroe  2007 Barry   23.76 16
Monroe  2007 Candace 21.11  5


     . . . more data lines . . .

Kennedy 2009 Sid     27.45 25
Kennedy 2009 Will    28.88 21
Kennedy 2009 Morty   34.44 25
;
```

分析と分析オプションを指定します。 FW=はフィールド幅 8 を、MAXDEC=は小数点以下 2 桁をそれぞれ使用して、統計量を表示します。ALPHA=0.1 は 90%の信頼限界を指定し、CLM キーワードは両側信頼限界を要求します。MEAN と STD は、平均値と標準偏差をそれぞれ要求します。

```
proc means data=charity fw=8 maxdec=2 alpha=0.1 clm mean std;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Year の値別に分割します。

```
   class Year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が MoneyRaised 変数と HoursVolunteered 変数の統計量を計算するように指定します。

```
   var MoneyRaised HoursVolunteered;
```

タイトルを指定します。

```
   title 'Confidence Limits for Fund Raising Statistics';
   title2 '2007-09';
run;
```

出力

PROC MEANS は、各年度の両方の変数に対し、下限信頼限界と上限信頼限界を表示します。

アウトプット 37.9 信頼限界

Confidence Limits for Fund Raising Statistics
2007-09

The MEANS Procedure

Year	N Obs	Variable	Lower 90% CL for Mean	Upper 90% CL for Mean	Mean	Std Dev
2007	34	MoneyRaised	25.69	32.29	28.99	11.37
		HoursVolunteered	17.68	22.73	20.21	8.71
2008	29	MoneyRaised	24.61	31.61	28.11	11.09
		HoursVolunteered	15.67	20.19	17.93	7.17
2009	46	MoneyRaised	26.73	33.78	30.26	14.23
		HoursVolunteered	19.68	22.63	21.15	5.96

例 8: 出力統計量を計算する

要素: PROC MEANS ステートメントオプション
NOPRINT
CLASS ステートメント
OUTPUT ステートメントオプション
統計量キーワード
IDGROUP
LEVELS
WAYS

他の要素: PRINT プロシジャ

データセット: GRADE

詳細

この例では、次を行います。

- PROC MEANS 出力の表示を非表示にします。
- 平均の最終成績を新しい変数に格納します。
- 期末試験のスコアが最高の生徒名を新しい変数に格納します。
- 組み合わせる分類変数の数を_WAY_変数に格納します。
- 分類水準の値を_LEVEL_変数に格納します。
- 出力データセットを表示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc means data=Grade noprint;
```

```
   class Status Year;

   var FinalGrade;

   output out=sumstat mean=AverageGrade
          idgroup (max(score) obs out (name)=BestScore)
          / ways levels;
run;

proc print data=sumstat noobs;
   title1 'Average Undergraduate and Graduate Course Grades';
   title2 'For Two Years';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

分析オプションを指定します。 NOPRINT は、すべての PROC MEANS 出力の表示を非表示にします。

```
proc means data=Grade noprint;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Status と Year の値別に分割します。

```
   class Status Year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が FinalGrade 変数の統計量を計算するように指定します。

```
   var FinalGrade;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは SUMSTAT データセットを作成し、最終成績の平均値を新しい変数 AverageGrade に書き込みます。IDGROUP は、試験のスコアが最上位の生徒の名前を変数 BestScore と、上位のスコアが含まれたオブザベーション番号に書き込みます。WAYS と LEVELS は、分類変数の組み合わせ方法に関する情報を書き込みます。

```
   output out=sumstat mean=AverageGrade
          idgroup (max(score) obs out (name)=BestScore)
          / ways levels;
run;
```

出力データセット WORK.SUMSTAT を印刷します。 NOOBS オプションは、オブザベーション番号を非表示にします。

```
proc print data=sumstat noobs;
   title1 'Average Undergraduate and Graduate Course Grades';
   title2 'For Two Years';
run;
```

出力

最初のオブザベーションには、2 年間にわたるクラスの平均成績と最高スコアの学生の名前が含まれます。次の 4 つのオブザベーションには、各分類変数値に対する値が含まれています。その他の 4 つのオブザベーションには、Year と Status の組み合わせに対する値が含まれています。変数_WAY_、_TYPE_、および_LEVEL_は、PROC MEANS が分類変数の組み合わせを作成した方法を示します。変数_OBS_には、試験の最高スコアが含まれた GRADE データセットのオブザベーション番号が含まれています。

アウトプット 37.10 *クラスの平均成績*

Average Undergraduate and Graduate Course Grades For Two Years

Status	Year	_WAY_	_TYPE_	_LEVEL_	_FREQ_	AverageGrade	BestScore	_OBS_
		0	0	1	10	83.0000	Branford	2
	97	1	1	1	6	83.6667	Jasper	10
	98	1	1	2	4	82.0000	Branford	2
1		1	2	1	6	82.5000	Branford	2
2		1	2	2	4	83.7500	Abbott	1
1	97	2	3	1	3	79.3333	Jasper	10
1	98	2	3	2	3	85.6667	Branford	2
2	97	2	3	3	3	88.0000	Abbott	1
2	98	2	3	4	1	71.0000	Crandell	3

例 9: 複数の変数に対して、異なる出力統計量を計算する

要素: PROC MEANS ステートメントオプション
- DESCEND
- NOPRINT

CLASS ステートメント

OUTPUT ステートメントオプション
- 統計量キーワード

他の要素: PRINT プロシジャ

WHERE=データセットオプション

データセット: GRADE

詳細

この例では、次を行います。

- PROC MEANS 出力の表示を非表示にします。
- 分類水準と、WHERE=によって指定される分類変数の組み合わせを出力データセットに格納します。
- 出力データセットのオブザベーションを_TYPE_値の降順で並べ替えます。
- 新しい変数名を割り当てずに、試験スコアの平均値と最終成績の平均値を格納します。

- 最終成績の中央値を新しい変数に格納します。
- 出力データセットを表示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc means data=Grade noprint descend;

   class Status Year;

   var Score FinalGrade;

   output out=Sumdata (where=(status='1' or _type_=0))
          mean= median(finalgrade)=MedianGrade;
run;

proc print data=Sumdata;
   title 'Exam and Course Grades for Undergraduates Only';
   title2 'and for All Students';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

分析オプションを指定します。 NOPRINT は、すべての PROC MEANS 出力の表示を非表示にします。DESCEND は、_TYPE_値の降順で OUT=データセットのオブザベーションを並べ替えます。

```
proc means data=Grade noprint descend;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Status と Year の値別に分割します。

```
   class Status Year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が Score 変数と FinalGrade 変数の統計量を計算するように指定します。

```
   var Score FinalGrade;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは、Score と FinalGrade の平均値を同じ名前の変数に書き込みます。最終成績の中間値が、変数 MedianGrade に書き込まれます。WHERE=データセットオプションは、SUMDATA のオブザベーションを制限します。1 つのオブザベーションには、全体の統計量 (_TYPE_=0)が含まれています。その他の区分は、1 である必要があります。

```
   output out=Sumdata (where=(status='1' or _type_=0))
          mean= median(finalgrade)=MedianGrade;
run;
```

出力データセット WORK.SUMDATA を印刷します。

```
proc print data=Sumdata;
   title 'Exam and Course Grades for Undergraduates Only';
```

```
   title2 'and for All Students';
run;
```

出力

最初の 3 つのオブザベーションには、区分 1 の分類変数水準に対する統計量が含まれています。最後のオブザベーションには、すべてのオブザベーションに対する統計量が含まれています(サブグループなし)。Score にはテストスコアの平均値が、FinalGrade には最終成績の平均値がそれぞれ含まれています。

アウトプット 37.11 *試験とクラスの成績*

Exam and Course Grades for Undergraduates Only and for All Students

Obs	Status	Year	_TYPE_	_FREQ_	Score	FinalGrade	MedianGrade
1	1	97	3	3	84.6667	79.3333	73
2	1	98	3	3	88.3333	85.6667	80
3	1		2	6	86.5000	82.5000	80
4			0	10	86.0000	83.0000	83

例 10: 分類変数の欠損値を使用し、出力統計量を計算する

要素: PROC MEANS ステートメントオプション
- CHARTYPE
- NOPRINT
- NWAY

CLASS ステートメントオプション
- ASCENDING
- MISSING
- ORDER=

OUTPUT ステートメント

他の要素: PRINT プロシジャ

データセット: CAKE

詳細

この例では、次を行います。

- PROC MEANS 出力の表示を非表示にします。
- 欠損値を 1 つの分類変数にのみ有効な水準値とみなします。
- 出力データセットのオブザベーションを 1 つの分類変数に対する度数の昇順で並べ替えます。
- _TYPE_の値が最高のオブザベーションを格納します。
- _TYPE_をバイナリ文字値として格納します。
- 味のスコアの最大値を新しい変数に格納します。

- 出力データセットを表示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc means data=cake chartype nway noprint;

   class flavor /order=freq ascending;
   class layers /missing;

   var TasteScore;

   output out=cakestat max=HighScore;
run;

proc print data=cakestat;
   title 'Maximum Taste Score for Flavor and Cake Layers';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

分析オプションを指定します。 NWAY は、_TYPE_値が最高のオブザベーションを印刷します。NOPRINT は、すべての PROC MEANS 出力の表示を非表示にします。

```
proc means data=cake chartype nway noprint;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Flavor と Layers の値別に分割します。ORDER=FREQ と ASCENDING は、度数の昇順で Flavor の水準を並べ替えます。MISSING は、Layers の欠損値を有効な分類水準値として使用します。

```
   class flavor /order=freq ascending;
   class layers /missing;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が TasteScore 変数の統計量を計算するように指定します。

```
   var TasteScore;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは CAKESTAT データセットを作成し、味のスコアの最大値を新しい変数 HighScore に出力します。

```
   output out=cakestat max=HighScore;
run;
```

出力データセット WORK.CAKESTAT を印刷します。

```
proc print data=cakestat;
   title 'Maximum Taste Score for Flavor and Cake Layers';
run;
```

出力

CAKESTAT 出力データセットには、2 つの分類変数、Flavor と Layers の組み合わせに対するオブザベーションのみ含まれています。そのため、_TYPE_の値は、すべてのオブザベーションの場合は 11 となります。オブザベーションは、Flavor の度数の昇順で並べ替えられます。Layers の欠損値は、この分類変数に有効な値です。PROC MEANS は、風味が欠損しているオブザベーションを除外します。これは Flavor に無効な値であるためです。

アウトプット 37.12 *味の最大スコア*

Maximum Taste Score for Flavor and Cake Layers

Obs	Flavor	Layers	_TYPE_	_FREQ_	HighScore
1	Rum	2	11	1	72
2	Spice	2	11	2	83
3	Spice	3	11	1	91
4	Vanilla	.	11	1	84
5	Vanilla	1	11	3	94
6	Vanilla	2	11	2	87
7	Chocolate	.	11	1	84
8	Chocolate	1	11	5	85
9	Chocolate	2	11	3	92

例 11: 出力統計量を使用し、極値を識別する

要素: CLASS ステートメント
OUTPUT ステートメントオプション
統計量キーワード
MAXID

他の要素: PRINT プロシジャ

データセット: Charity

詳細

この例では、次を行います。

- 2 つの変数に対する最大値を含むオブザベーションを識別します。
- 最大値に対し新しい変数を作成します。
- 出力データセットを表示します。

プログラム

```
proc means data=Charity n mean range chartype;
   class School Year;
   var MoneyRaised HoursVolunteered;
   output out=Prize maxid(MoneyRaised(name)
          HoursVolunteered(name))= MostCash MostTime
          max= ;

   title 'Summary of Volunteer Work by School and Year';
run;
proc print data=Prize;
   title 'Best Results: Most Money Raised and Most Hours Worked';
run;
```

プログラムの説明

分析を指定します。 統計量キーワードは、統計量と出力での順序を指定します。CHARTYPE は、_TYPE_値をバイナリ文字として出力データセットに書き込みます。

```
proc means data=Charity n mean range chartype;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を School と Year 別に分割します。

```
   class School Year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が MoneyRaised 変数と HoursVolunteered 変数の統計量を計算するように指定します。

```
   var MoneyRaised HoursVolunteered;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは、最も多くの金額を集めた生徒と最も多くの時間をボランティアに充てた生徒の名前がそれぞれ含まれている新しい変数、MostCash と MostTime を PRIZE データセットに書き込みます。

```
   output out=Prize maxid(MoneyRaised(name)
          HoursVolunteered(name))= MostCash MostTime
          max= ;
```

タイトルを指定します。

```
   title 'Summary of Volunteer Work by School and Year';
run;
```

WORK.PRIZE 出力データセットを印刷します。

```
proc print data=Prize;
   title 'Best Results: Most Money Raised and Most Hours Worked';
run;
```

出力

出力の最初のページには、PROC MEANS からの出力が 6 つの分類水準に対する統計量とともに表示されます。これは、2007 年、2008 年、2009 年の Monroe High に対するものと、同じ 3 年間の Kennedy High に対するものです。

アウトプット 37.13 *ボランティア活動の概要*

Summary of Volunteer Work by School and Year

The MEANS Procedure

School	Year	N Obs	Variable	N	Mean	Range
Kennedy	2007	18	MoneyRaised	18	29.3811111	39.7500000
			HoursVolunteered	18	21.4444444	30.0000000
	2008	17	MoneyRaised	17	28.1564706	23.5600000
			HoursVolunteered	17	19.4117647	20.0000000
	2009	18	MoneyRaised	18	31.5794444	65.4400000
			HoursVolunteered	18	24.2777778	15.0000000
Monroe	2007	16	MoneyRaised	16	28.5450000	48.2700000
			HoursVolunteered	16	18.8125000	38.0000000
	2008	12	MoneyRaised	12	28.0500000	52.4600000
			HoursVolunteered	12	15.8333333	21.0000000
	2009	28	MoneyRaised	28	29.4100000	73.5300000
			HoursVolunteered	28	19.1428571	26.0000000

PROC PRINT からの出力には、MoneyRaised と HoursVolunteered の最大値と、担当した学生の名前が表示されます。最初のオブザベーションには全体の結果が含まれ、次の 3 つには年度別の結果、次の 2 つには学校別の結果、最後の 6 つには School と Year 別の結果がそれぞれ含まれています。

アウトプット 37.14 最高結果

Best Results: Most Money Raised and Most Hours Worked

Obs	School	Year	_TYPE_	_FREQ_	MostCash	MostTime	MoneyRaised	HoursVolunteered
1		.	00	109	Willard	Tonya	78.65	40
2		2007	01	34	Tonya	Tonya	55.16	40
3		2008	01	29	Cameron	Amy	65.44	31
4		2009	01	46	Willard	L.T.	78.65	33
5	Kennedy	.	10	53	Luther	Jay	72.22	35
6	Monroe	.	10	56	Willard	Tonya	78.65	40
7	Kennedy	2007	11	18	Thelma	Jay	52.63	35
8	Kennedy	2008	11	17	Bill	Amy	42.23	31
9	Kennedy	2009	11	18	Luther	Che-Min	72.22	33
10	Monroe	2007	11	16	Tonya	Tonya	55.16	40
11	Monroe	2008	11	12	Cameron	Myrtle	65.44	26
12	Monroe	2009	11	28	Willard	L.T.	78.65	33

例 12: 出力統計量を使用し、上位 3 位の極値を識別する

要素: PROC MEANS ステートメントオプション
- NOPRINT

CLASS ステートメント

OUTPUT ステートメントオプション
- 統計量キーワード
- AUTOLABEL
- AUTONAME
- IDGROUP

TYPES ステートメント

他の要素: FORMAT プロシジャ

FORMAT ステートメント

PRINT プロシジャ

RENAME=データセットオプション

データセット: Charity

詳細

この例では、次を行います。

- PROC MEANS 出力の表示を非表示にします。
- 分類変数の 1 次元組み合わせのデータとすべてのオブザベーションにわたるデータを分析します。
- 募金金額の合計と平均を新しい変数に格納します。
- 新しい変数に上位 3 位の募金金額、それを集めた 3 人の学生の名前、その年度、学生が通った学校を格納します。

- 名前が出力データセットの新しい変数に割り当てられているときの変数名の競合問題を自動的に解決します。
- 分析変数に対して計算された統計量を含む出力データセットの変数のラベルに統計量名を追加します。
- この変数から計算される統計量が出力データセットの属性を継承するように、出力形式を分析変数に割り当てます。
- 出力データセットの_FREQ_変数の名前を変更します。
- 出力データセットとその内容を表示します。

プログラム

```
proc format;
   value yrFmt . = " All";
   value $schFmt ' ' = "All    ";
run;

proc means data=Charity noprint;

   class School Year;

   types () school year;

   var MoneyRaised;

   output out=top3list(rename=(_freq_=NumberStudents))sum= mean=
          idgroup( max(moneyraised) out[3] (moneyraised name
            school year)=)/autolabel autoname;

   label MoneyRaised='Amount Raised';
   format year yrfmt. school $schfmt.
          moneyraised dollar8.2;
  run;

proc print data=top3list;
   title1 'School Fund Raising Report';
   title2 'Top Three Students';
run;

proc datasets library=work nolist;
   contents data=top3list;
   title1 'Contents of the PROC MEANS Output Data Set';
run;
```

プログラムの説明

YRFMT.出力形式と$SCHFMT.出力形式を作成します。 PROC FORMAT は、**All** の値を分類変数の欠損水準に割り当てるユーザー定義出力形式を作成します。

```
proc format;
   value yrFmt . = " All";
   value $schFmt ' ' = "All    ";
run;
```

デフォルトの統計量を生成して、分析オプションを指定します。 NOPRINT は、すべての PROC MEANS 出力の表示を非表示にします。

```
proc means data=Charity noprint;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を School と Year の値別に分割します。

```
class School Year;
```

分析するサブグループを指定します。 TYPES ステートメントは、すべてのオブザベーションにわたる分析と、School と Year の各 1 次元の組み合わせに対する分析を要求します。

```
types () school year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が MoneyRaised 変数の統計量を計算するように指定します。

```
var MoneyRaised;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは、TOP3LIST データセットを作成します。RENAME=は、各分類水準に対する度数カウントを含む_FREQ_変数の名前を変更します。SUM=と MEAN=は、分析変数(MoneyRaised)の合計と平均値が出力データセットに書き込まれるように指定します。IDGROUP は、上位 3 位の募金金額、対応する 3 人の生徒、学校および年度を含む 12 の変数を書き込みます。AUTOLABEL は、分析変数の名前を合計と平均値を含む出力変数のラベルに追加します。AUTONAME は、これらの変数の名前の競合問題を解決します。

```
output out=top3list(rename=(_freq_=NumberStudents))sum= mean=
       idgroup( max(moneyraised) out[3] (moneyraised name
         school year)=)/autolabel autoname;
```

出力をフォーマットします。 LABEL ステートメントは、ラベルを分析変数 MoneyRaised に割り当てます。FORMAT ステートメントは、ユーザー定義出力形式を Year 変数と School 変数に、SAS ドル出力形式を MoneyRaised 変数にそれぞれ割り当てます。

```
label MoneyRaised='Amount Raised';
format year yrfmt. school $schfmt.
       moneyraised dollar8.2;
run;
```

出力データセット WORK.TOP3LIST を印刷します。

```
proc print data=top3list;
   title1 'School Fund Raising Report';
   title2 'Top Three Students';
run;
```

TOP3LIST データセットに関する情報を表示します。 PROC DATASETS は、TOP3LIST データセットの内容を表示します。NOLIST は、WORK データライブラリのディレクトリリストの表示を非表示にします。

```
proc datasets library=work nolist;
   contents data=top3list;
   title1 'Contents of the PROC MEANS Output Data Set';
run;
```

出力

PROC PRINT からの出力には、MoneyRaised の上位 3 位の値、これらの金額を集めた学生の名前、その学生が通う学校、募金が行われた年度が表示されます。最初のオブザベーションには全体の結果が含まれ、次の 3 つには年度別の結果、最後の 2

つには学校別の結果が含まれています。School と Year の欠損分類水準は、値 ALL と置き換えられます。MoneyRaised から計算された統計量を含む変数のラベルには、ラベルの最後に統計量名が含まれます。

アウトプット 37.15 *学校基金調達*

Results Viewer - SAS Output

School Fund Raising Report
Top Three Students

MoneyRaised_Sum	MoneyRaised_Mean	MoneyRaised_1	MoneyRaised_2	MoneyRaised_3	Name_1	Name_2	Name_3	School_1
$3192.75	$29.29	$78.65	$72.22	$65.44	Willard	Luther	Cameron	Monroe
$892.92	$28.80	$55.16	$53.76	$52.63	Tonya	Edward	Thelma	Monroe
$907.92	$28.37	$65.44	$47.33	$42.23	Cameron	Myrtle	Bill	Monroe
$1391.91	$30.26	$78.65	$72.22	$56.87	Willard	Luther	L.T.	Monroe
$1575.95	$29.73	$72.22	$52.63	$43.89	Luther	Thelma	Jenny	Kennedy
$1616.80	$28.87	$78.65	$65.44	$56.87	Willard	Cameron	L.T.	Monroe

この出力データセットに対するカスタムテーブルテンプレートの作成方法の例については、*SAS Output Delivery System: ユーザーガイド*の TEMPLATE プロシジャを参照してください。

アウトプット 37.16 *PROC MEANS 出力データセット*

School Fund Raising Report
Top Three Students

Obs	School	Year	_TYPE_	NumberStudents	MoneyRaised_Sum	MoneyRaised_Mean	MoneyRaised_1	MoneyRaised_2	MoneyRaised_3	Name_1
1	All	All	0	109	$3192.75	$29.29	$78.65	$72.22	$65.44	Willard
2	All	2007	1	34	$985.58	$28.99	$55.16	$53.76	$52.63	Tonya
3	All	2008	1	29	$815.26	$28.11	$65.44	$47.33	$42.23	Cameron
4	All	2009	1	46	$1391.91	$30.26	$78.65	$72.22	$56.87	Willard
5	Kenn	All	2	53	$1575.95	$29.73	$72.22	$52.63	$43.89	Luther
6	Monr	All	2	56	$1616.80	$28.87	$78.65	$65.44	$56.87	Willard

Contents of the PROC MEANS Output Data Set

The DATASETS Procedure

Data Set Name	WORK.TOP3LIST	**Observations**	6
Member Type	DATA	**Variables**	18
Engine	V9	**Indexes**	0
Created	Tuesday, July 05, 2011 10:24:55 AM	**Observation Length**	144
Last Modified	Tuesday, July 05, 2011 10:24:55 AM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	12288
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	85
Obs in First Data Page	6
Number of Data Set Repairs	0
Filename	C:\DOCUME~1\lirezn\LOCALS~1\Temp\SAS Temporary Files_TD3456_d21560_\top3list.sas7bdat
Release Created	9.0301M0
Host Created	XP_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
7	MoneyRaised_1	Num	8	DOLLAR8.2	Amount Raised
8	MoneyRaised_2	Num	8	DOLLAR8.2	Amount Raised
9	MoneyRaised_3	Num	8	DOLLAR8.2	Amount Raised
6	MoneyRaised_Mean	Num	8	DOLLAR8.2	Amount Raised_Mean
5	MoneyRaised_Sum	Num	8	DOLLAR8.2	Amount Raised_Sum
10	Name_1	Char	7		
11	Name_2	Char	7		
12	Name_3	Char	7		
4	NumberStudents	Num	8		
1	School	Char	7	$SCHFMT.	
13	School_1	Char	7	$SCHFMT.	
14	School_2	Char	7	$SCHFMT.	
15	School_3	Char	7	$SCHFMT.	
2	Year	Num	8	YRFMT.	
16	Year_1	Num	8	YRFMT.	
17	Year_2	Num	8	YRFMT.	
18	Year_3	Num	8	YRFMT.	
3	_TYPE_	Num	8		

例 13: STACKODSOUTPUT オプションによるデータの制御

要素: PROC PRINT
PROC CONTENTS

最初の例では、STACKODSOUTPUT オプションは使用しません。2 つ目の例では、STACKODSOUTPUT オプションを使用します。

プログラム

```
proc means data=sashelp.class;
class sex;
var weight height;
ods output summary=default;
run;

proc print data=default; run;
proc contents data=default; run;
```

プログラムの説明

このコードは、STACKODSOUTPUT オプションを使用せずにデータを処理します。

```
proc means data=sashelp.class;
```

```
class sex;
var weight height;
ods output summary=default;
run;
```

PROC PRINT を使用してデータを印刷します。PROC CONTENTS を使用してプロシジャの内容を印刷します。

```
proc print data=default; run;
proc contents data=default; run;
```

OUTPUT

次の出力に、STACKODSOUTPUT オプションを使用した場合と使用しない場合のデータ処理の違いを示します。

この出力は、STACKODSOUTPUT オプションを使用せずに生成されます。

アウトプット 37.17 *STACKODSOUTPUT オプションを使用しない出力*

The SAS System

The MEANS Procedure

Sex	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
F	9	Weight	9	90.1111111	19.3839137	50.5000000	112.5000000
		Height	9	60.5888889	5.0183275	51.3000000	66.5000000
M	10	Weight	10	108.9500000	22.7271864	83.0000000	150.0000000
		Height	10	63.9100000	4.9379370	57.3000000	72.0000000

The SAS System

Obs	Sex	NObs	VName_Weight	Weight_N	Weight_Mean	Weight_StdDev	Weight_Min	Weight_Max	VName_H
1	F	9	Weight	9	90.111111111	19.38391372	50.5	112.5	Height
2	M	10	Weight	10	108.95	22.727186363	83	150	Height

The SAS System

The CONTENTS Procedure

Data Set Name	WORK.DEFAULT	Observations	2
Member Type	DATA	Variables	14
Engine	V9	Indexes	0
Created	Sunday, January 23, 2011 02:48:10 PM	Observation Length	104
Last Modified	Sunday, January 23, 2011 02:48:10 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Summary statistics		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	12288
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	117
Obs in First Data Page	2
Number of Data Set Repairs	0
Filename	C:\DOCUME~1\lirezn\LOCALS~1\Temp\SAS Temporary Files_TD3828_d21560_\default.sas7bdat
Release Created	9.0301B0
Host Created	XP_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
14	Height_Max	Num	8	BEST12.	Maximum
11	Height_Mean	Num	8	BEST12.	Mean
13	Height_Min	Num	8	BEST12.	Minimum
10	Height_N	Num	8	BEST2.	N
12	Height_StdDev	Num	8	BEST12.	Std Dev
2	NObs	Num	8	BEST2.	N Obs
1	Sex	Char	1		
9	VName_Height	Char	6		Variable
3	VName_Weight	Char	6		Variable
8	Weight_Max	Num	8	BEST12.	Maximum
5	Weight_Mean	Num	8	BEST12.	Mean
7	Weight_Min	Num	8	BEST12.	Minimum
4	Weight_N	Num	8	BEST2.	N
6	Weight_StdDev	Num	8	BEST12.	Std Dev

このコードは、STACKODSOUTPUT オプションを使用してデータを処理します。

```
proc means data=sashelp.class STACKODSOUTPUT;
class sex;
var weight height;
ods output summary=stacked;
run;
```

PROC PRINT を使用してデータを印刷します。PROC CONTENTS を使用してプロシジャの内容を印刷します。

```
proc print data=stacked; run;
proc contents data=stacked; run;
```

この出力は、STACKODSOUTPUT オプションを使用して生成されます。

アウトプット 37.18 STACKODSOUTPUT オプションを使用した出力

The SAS System

The MEANS Procedure

Sex	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
F	9	Weight	9	90.111111	19.383914	50.500000	112.500000
		Height	9	60.588889	5.018328	51.300000	66.500000
M	10	Weight	10	108.950000	22.727186	83.000000	150.000000
		Height	10	63.910000	4.937937	57.300000	72.000000

The SAS System

Obs	Sex	NObs	_control_	Variable	N	Mean	StdDev	Min	Max
1	F	9		Weight	9	90.111111	19.383914	50.500000	112.500000
2	F	9		Height	9	60.588889	5.018328	51.300000	66.500000
3	M	10	1	Weight	10	108.950000	22.727186	83.000000	150.000000
4	M	10		Height	10	63.910000	4.937937	57.300000	72.000000

The SAS System

The CONTENTS Procedure

Data Set Name	WORK.STACKED	Observations	4
Member Type	DATA	Variables	9
Engine	V9	Indexes	0
Created	Sunday, January 23, 2011 03:19:07 PM	Observation Length	56
Last Modified	Sunday, January 23, 2011 03:19:07 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Summary statistics		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	145
Obs in First Data Page	4
Number of Data Set Repairs	0
Filename	C:\DOCUME~1\lirezn\LOCALS~1\Temp\SAS Temporary Files_TD5560_d21560_\stacked.sas7bdat
Release Created	9.0301B0
Host Created	XP_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
9	Max	Num	8	D12.3	Maximum
6	Mean	Num	8	D12.3	
8	Min	Num	8	D12.3	Minimum
5	N	Num	8	BEST2.	
2	NObs	Num	8	BEST2.	N Obs
1	Sex	Char	1		
7	StdDev	Num	8	D12.3	Std Dev
4	Variable	Char	6		
3	_control_	Char	1		

参考文献

Jain R., and I. Chlamtac. 1985. “The P² Algorithm for Dynamic Calculation of Quantiles and Histograms without Sorting Observations.” *Communications of the Association of Computing Machinery* 28 (10): 1076-0185.

38 章

MIGRATE プロシジャ

概要: MIGRATE プロシジャ

MIGRATE プロシジャの動作について

MIGRATE プロシジャは、SAS ライブラリのメンバを現在の SAS バージョンに移行します。

このプロシジャは、大部分の SAS 6、SAS 7、SAS 8、SAS®9 動作環境のライブラリを現在の SAS のリリースに移行します。移行は同じエンジンファミリ内で行う必要があります。たとえば、V6、V7 または V8 は V9 に移行できますが、V6TAPE は V9TAPE に移行する必要があります。

このプロシジャは、次のライブラリメンバを移行します。

- 代替照合順序、監査証跡、圧縮、作成日時と変更日時、削除されたオブザベーション、暗号化(AES 暗号化を除く)、拡張属性、世代、インデックス、一貫性制約、パスワードを含むデータセット
- 多くの場合、ビュー、カタログ、アイテムストア、多次元データベース(MDDB)(“メンバの種類ごとの考慮事項” (1176 ページ)を参照)

このプロシジャは保存されたコンパイル済み DATA ステッププログラムや保存されたコンパイル済みマクロをサポートしていません。そこではコンパイルと保存が可能です)。このプロシジャは SAS プログラムファイルをサポートしていません。このプロシジャは、スケーラブルパフォーマンスデータ(SPD)エンジンデータセットをサポートしていません。(*SAS Scalable Performance Data Engine: リファレンス*を参照してください)。このプロシジャは、拡張オブザベーションカウント属性をサポートしていません。

ベストプラクティス

まずは SAS のウェブサイト上の互換性カリキュレータを使用します。アドレスは次のとおりです。です。support.sas.com/migration/planning/files/calculator これにより、ライブラリを移行する必要があるかを調べます。次に、SAS のウェブサイト上の PROC MIGRATE カリキュレータを使用します。アドレスは次のとおりです support.sas.com/migration/planning/files/migratecalc これにより、移行用の PROC MIGRATE 構文が得られます。

ライブラリの移行をドキュメント化し検証するには、MIGRATE プロシジャの検証ツールを使用します。検証ツールは、SAS ウェブサイトにあります。詳細については、“検証ツールを使用したライブラリの移行” (1186 ページ)を参照してください。

メンバの種類ごとの考慮事項

メンバの種類ごとの考慮事項を次に示します。SAS データセットがデータファイルやデータビューの場合もあることに注意してください。また、構文セクションの最初にある複数の重要な制限事項も参照してください。

データファイル

PROC MIGRATE は、代替照合順序、圧縮、作成日時と変更日時、削除されたオブザベーション、暗号化、拡張属性、インデックス、一貫性制約、パスワードを保持します。監査証跡と世代も移行されます。インデックスと一貫性制約は、ターゲットライブラリのメンバで再作成されます。移行されたデータファイルでは、ターゲットライブラリのデータ表現とエンコーディング属性が採用されます。“データファイルの移行(監査証跡、世代、インデックス、または一貫性制約)” (1182 ページ)を参照してください。

より多くのバイトで文字が表されるエンコーディングにデータファイルを移行すると、文字データの切り捨てが問題になる場合があります。たとえば、ある文字が Wlatin1 エンコーディングでは 1 バイトとして表示されるが、UTF-8 エンコーディングでは 2 バイトとして表示される場合があります。文字データをより大きいバイトサイズに移行すると、より長くなった文字サイズに列の長さが対応できなければ、切り捨てられる場合があります。最良の解決法は、移行前に CVP エンジンを使用して列の長さを拡張することです。通常は、CVPMULTIPLIER=2.5 の値を指定すれば切り捨ては回避されます。データにアジアの言語が含まれている場合は、CVPMULTIPLIER=4 を使用することをお勧めします(PROC MIGRATE では、ASCII-OEM の英語以外の文字は移行されません。“英語以外の文字を含むデータセットの移行” (1184 ページ)を参照してください)。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*で CVPMULTIPLIER=オプションと文字データ切り捨ての回避に関する説明を参照してください。また、support.sas.com の資料“Multilingual Computing with SAS® 9.4”も参照してください。

ビュー

データファイルと同様に、移行されたデータビューでは、ターゲットライブラリのデータ表現とエンコーディング属性が採用されます。DATA ステップビューを含むライブラリを異なる動作環境に移行する際、そのビューが SAS 9.2 より前に作成されていた場合は、正しいエンコーディングの設定が必要になることがあります。SAS 9.2 より前のリリースでは、DATA ステップビューにはエンコーディング情報が保存されていませんでした。このため、ビューにターゲットセッションと異なるエンコーディングが含まれる場合は、ソースライブラリの LIBNAME ステートメントに対して INENCODING=オプションを指定する必要がありました。次に例を示します。

```
libname Srclib 'source-library-pathname' inencoding="OPEN_ED-1047"; libname Lib1 'target-li
```

ノートがログに印刷されます。ここでは、検証ツールを使用して DATA ステップビューを再コンパイルし、移行が正常終了したかどうかを判断することが勧められます。“検証ツールを使用したライブラリの移行” (1186 ページ)を参照してください。

また、ビューに関連付けられている埋め込まれたライブラリ参照名は移行時に更新されません。次の例で、この問題について説明します。この例では、Lib1.MyView にデータファイル Lib1.MyData のビューが含まれます。

```
data Lib1.MyData;x=1; run; proc sql;    create view Lib1.MyView as select * from Lib1.MyData
```

Lib1 を Lib2 に移行すると、Lib2.MyView と Lib2.MyData が得られます。ただし Lib2.MyView は、当初の作成時に埋め込まれた参照名が Lib1 であったため、現在もデータファイル Lib2.MyData ではなく、Lib1.MyData を参照します。次の例では、処理が失敗して、Lib1 が見つからないというエラーメッセージが出ます。

```
proc print data=Lib2.MyView; run;
```

PROC MIGRATE では3つのタイプのビューをサポートします。PROC MIGRATE では、DATA ステップビュー、SQL ビュー、SAS/ACCESS ビューの 3 種類のビューがサポートされています。

DATA ステップビュー
DATA ステップビューを作成する際に、SOURCE=オプションを指定すると、DATA ステップコードをビューと一緒に保存できます。PROC MIGRATE では、コードの保存された DEAT ステップビューがサポートされています。保存されたコードは、DATA ステップビューがターゲット環境で初めて SAS にアクセスされる際に、再コンパイルされます。PROC MIGRATE では、SAS 8 より前に作成された DATA ステップビューや、コードが保存されていない DATA ステップビューはサポートされていません。コードの保存されていない DATA ステップビューでは、ソースセッションで DESCRIBE ステートメントを使用して DATA ステップコードを復元します。次に、ターゲットセッションでその DATA ステップコードをサブミットして、再コンパイルします。

PROC SQL ビュー
PROC MIGRATE では、既知の問題がない PROC SQL ビューがサポートされています。

SAS/ACCESS ビュー
PROC MIGRATE では、Oracle エンジン、Sybase エンジンまたは DB2 エンジンによって書き込まれた SAS/ACCESS ビューがサポートされています。PROC MIGRATE が自動的に使用する CV2VIEW プロシジャによって、SAS/ACCESS ビューが SQL ビューに変換されます。SAS/ACCESS ビューの異なる動作環境への移行はサポートされていません。変換の詳細については、*SAS/ACCESS for Relational Databases: Reference* の CV2VIEW プロシジャの概要を参照してください。

カタログ

カタログを移行するために、PROC MIGRATE は PROC CPORT と PROC CIMPORT を呼び出します。移行時に、CPORT と CIMPORT のノートが SAS ログに書き込まれる場合があります。PROC CPORT と CIMPORT の制限が適用されます。たとえば、順次ライブラリのカタログは移行されません。カタログに保存されている保存済みでコンパイル済みのマクロはサポートされません(かわりに、ソースコードをターゲットに移動します。そこではコンパイルと保存が可能です)。移行後にカタログエントリの更新が必要になる可能性があります(たとえば、ハードコードされたパス名を含んでいるコード)。

制限　PROC MIGRATE では、SAS 6 AIX カタログはサポートされていません。代わりに PROC CPORT と PROC CIMPORT を使用します。“サポートされないカタログへの追加操作”を参照してください

要件　クロス環境データアクセス(CEDA)処理を起動する場合、および IN=ソースライブラリにカタログが含まれる場合は、IN=オプションまたは SLIBREF=オプションで SAS/CONNECT または SAS/SHARE ライブラリ参照名を指定する必要があります。ライブラリ参照名の指定を IN=引数で行うか、SLIBREF=引数行うかを決めるには、“SLIBREF=オプションの使用”を参照してください。カタログが SAS 6 または SAS 8 で作成された場合、SLIBREF=は SAS 8 サーバー経由で割り当てる必要があります。

MDDB

PROC MIGRATE では、既知の問題がない MDDB がサポートされています。

アイテムストア

PROC MIGRATE では、32 ビット環境から 64 ビット環境への移行でなければ、アイテムストアがサポートされます。32 ビット環境から 64 ビット環境への移行ではリモートラ

イブラリサービス(RLS)が使用されますが、ここではアイテムストアはサポートされていません。この場合、SAS ログにエラーメッセージが書き込まれますが、アイテムストアがターゲットライブラリでは正しく機能しない場合があります。

サポート対象外

PROC MIGRATE は保存されたコンパイル済み DATA ステッププログラムや保存されたコンパイル済みマクロをサポートしていません。そこではコンパイルと保存が可能です)。PROC MIGRATE では SAS プログラムファイルはサポートされていません。PROC MIGRATE では SPD エンジンデータセットはサポートされていません。(*SAS Scalable Performance Data Engine: リファレンス*を参照してください)。また、各メンバタイプについては前述の制限事項を参照してください。

構文: MIGRATE プロシジャ

制限事項: SAS データセットオプションは、PROC MIGRATE ではサポートされていません。

ソースライブラリとターゲットライブラリの物理的な場所は別にする必要があります。

PROC MIGRATE がターゲットライブラリでメンバを作成した場合、そのメンバではソースライブラリの権限は保持されません。PROC MIGRATE 実行者のユーザー ID に関連付けられた権限が含まれます。

以下の動作環境にある SAS8.2 ソースライブラリはサポートされていません。CMS、OS/2、OpenVMS VAX または 64 ビット AIX“PROC MIGRATE プロシジャの代替” (1188 ページ)を参照してください。

Tru64 UNIX ソース環境で作成されたカタログでは、Linux (x64)や Solaris (x64)ターゲット環境はサポートされていません。これらの状況下でカタログを移行するには、“サポートされないカタログへの追加操作” (1187 ページ)を参照してください。

SAS 6.12 (SAS 6.09E for z/OS)より前に作成された SAS ファイルは、現在の SAS のリリースに移行する前に、SAS 6.12 に変換しておく必要があります。サポートされていない SAS 6 ソース環境もあります。“SAS 6 ライブラリの移行” (1183 ページ)を参照してください

操作: 言語照合順序があるデータセットをソートするには、ユニコード用国際化コンポーネント(ICU)バージョンを使用します。言語ソートされたデータセットに現在の SAS セッションの ICU バージョン番号とは異なる番号が含まれている場合、PROC MIGRATE は OUT=の出力先ライブラリでもそのデータセットの並べ替え順序を維持します。ただし、そのデータセットからソート済みのマーキングがなくなり、SAS ログに警告メッセージが書き込まれます。言語ソートの詳細については、59 章, “SORT プロシジャ,” (1785 ページ)を参照してください。

ヒント: OUT=ターゲットライブラリを空の場所に割り当てます。ターゲットライブラリに、名前とメンバの種類がソースライブラリのメンバと同じメンバがすでに存在する場合、そのメンバは移行されません。SAS ログにエラーメッセージが書き込まれ、PROC MIGRATE は次のメンバの処理を続行します。順次ライブラリのメンバは例外であることに注意してください。これは、PROC MIGRATE がテープ全体を読み取って存在するかどうか判断することがないからです。

エンコードとトランスコードの問題については、http://support.sas.com/rnd/migration/の移行トピック内の各国語サポートについてのトピックを参照するか、あるいは *SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

移行に関するその他のトピックについては、http://support.sas.com/rnd/migration/の移行フォーカスエリアを参照してください。

PROC MIGRATE IN=*libref-1* OUT=*libref-2* <*option(s)*>;

ステートメント	タスク	例
“PROC MIGRATE ステートメント”	SAS ライブラリのメンバを現在の SAS バージョンに移行	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5

PROC MIGRATE ステートメント

SAS ライブラリをより新しい現在の SAS のリリースに移行します。

構文

PROC MIGRATE IN=*libref-1* OUT=*libref-2*
<BUFSIZE=KEEPSIZE | *n* | *n*K | *n*M | *n*G>
<MOVE>
<SLIBREF=*libref*>
<KEEPNODUPKEY>;

必須引数

IN=*libref-1*
メンバの移行元のソース SAS ライブラリの名前を指定します。

要件 SAS/CONNECT や SAS/SHARE などのサーバーを使用する場合は、SAS 9.1.3 以降のサーバーにする必要があります。

クロス環境データアクセス(CEDA)処理を起動する場合、および IN=ソースライブラリにカタログが含まれる場合は、IN=オプションまたは SLIBREF=オプションで SAS/CONNECT ライブラリ参照名または SAS/SHARE ライブラリ参照名を指定する必要があります。ライブラリ参照名の指定を IN=引数で行うか、SLIBREF=引数で行うかを決めるには、“SLIBREF=オプションの使用” (1186 ページ)を参照してください。

ライブラリ内のメンバの種類が CATALOG だけのときに SLIBREF=オプションを使用する場合は、IN=引数を省略します。

OUT=*libref-2*
移行したメンバを含めるターゲット SAS ライブラリの名前を指定します。

制限事項 OUT=ターゲットライブラリを SAS/CONNECT や SAS/SHARE などのサーバーに割り当てないて
ンジンでは、ターゲットライブラリはサポートされていません。次の例では、エラーが引き起こされ

```
libname Lib1 'source-library-pathname'; libname Lib2 server=server id; proc migrate
```

要件 OUT=ターゲットライブラリは、IN=ソースライブラリとは別の物理的な場所に割り当てます。

操作 PROC MIGRATE では、DATA、VIEW、ACCESS、MDDB、DMDB というメンバの種類に対して
OUTREP=を使用できます。OUTREP=オプションを指定する場合は、EXTENDOBSCOUNTER=
とがあります。LIBNAME ステートメントでは、これらのオプションは OUT=ライブラリに適してい
*リファレンス*を参照してください。

ヒント ターゲットライブラリを空の場所に割り当てます。ターゲットライブラリに、名前とメンバの種類がソ
と同じメンバがすでに存在する場合、そのメンバは移行されません。SAS ログにエラーメッセージ
MIGRATE は次のメンバの処理を続行します。順次ライブラリのメンバは例外であることに注意し
PROC MIGRATE がテープ全体を読み取って存在するかどうか判断することがないからです。

オプション引数

BUFSIZE=KEEPSIZE | *n* | *n*K | *n*M | *n*G
ターゲットライブラリに書き込まれるメンバのバッファページサイズを指定します。たとえば、`10000` の値では 10,000 バイトのページサイズ、`4k` の値では 4096 バイトのページサイズが指定されます。デフォルト値は `0` です。ページサイズを設定すると、SAS のパフォーマンスを最適化できます。SAS 9.4 メンテナンスリリース 3 では、BUFSIZE のデフォルトが変更されています。新しいデフォルト値は、現在のセッションのバッファページのサイズです。ソースライブラリのメンバのページサイズをコピーするという以前の動作をそのまま使用し続けるには、BUFSIZE=KEEPSIZE を指定します。

BUFSIZE=データセットオプションまたはシステムオプションの詳細については、動作環境に関する資料を参照してください。

KEEPSIZE
ソースライブラリのメンバのページサイズを保持(コピー)します。

n
バイト数を指定します。

***n*K**
キロバイト数を指定します。

***n*M**
メガバイト数を指定します。

***n*G**
ギガバイト数を指定します。

デフォルト 現在のセッションのバッファページサイズ

MOVE
ソースライブラリから元のメンバを削除します。ターゲットライブラリにすでにメンバが存在する場合、メンバはソースライブラリから削除されず、SAS ログにメッセージが送信されます。ターゲットライブラリにすでにカタログが存在する場合、カタログはソースライブラリから削除されず、ログにメッセージが送信されます。データセットに参照一貫性制約がある場合、このデータセットがソースライブラリから削除され、ログにエラーが送信されます。MOVE を指定すると、検証ツールのスコープが減ります。“検証ツールを使用したライブラリの移行” (1186 ページ)を参照してください。

制限事項 IN=ソースライブラリに関連付けられるエンジンでは、テーブルの削除がサポートされている必要があります。順次エンジンでは、テーブルの削除はサポートされていません。

ヒント システムのスペースが制限されている場合に限り、MOVE オプションを使用します。メンバを削除する前に、移行の確認をすることをお勧めします。

SLIBREF=*libref*
SAS/CONNECT または SAS/SHARE サーバー経由で割り当てられるライブラリ参照名を指定します。クロス環境データアクセス(CEDA)処理を起動する場合、および IN=ソースライブラリにカタログが含まれる場合は、IN=オプションまたは SLIBREF=オプションで SAS/CONNECT または SAS/SHARE ライブラリ参照名を

指定する必要があります。ライブラリ参照名の指定を IN=引数で行うか、SLIBREF=引数行うかを決めるには、“SLIBREF=オプションの使用” (1186 ページ)を参照してください。

要件 カタログが SAS 6 または SAS 8 で作成された場合、SLIBREF=は SAS 8 サーバー経由で割り当てる必要があります。

SLIBREF=サーバーが稼働する動作環境の種類は、ソースライブラリと同じにする必要があります。たとえば、ソースセッションが UNIX で稼働している場合、サーバーも UNIX で稼働している必要があります。

操作 ライブラリ内のメンバの種類が CATALOG だけのときに SLIBREF=オプションを使用する場合は、IN=引数を省略します。

SAS®9 以降から移行する場合(SAS 9.1.3 から SAS 9.4 への移行など)、SLIBREF=は必要ありません。カタログに互換性がない場合は、IN=引数で SAS/CONNECT または SAS/SHARE サーバーのライブラリ参照名を指定して、SLIBREF=引数は省略します。IN=引数では、サーバーを SAS 9.1.3 以降にする必要があります。

KEEPNODUPKEY
指定すると、NODUPKEY 並べ替え順序が保持されます。“NODUPKEY ソートインジケータを含む SAS データセットの移行” (1183 ページ)を参照してください。

データファイルの移行(監査証跡、世代、インデックス、または一貫性制約)

すべての場合において、まずデータファイルが移行してから、監査証跡、世代、インデックス、一貫性制約のいずれかに処理が適用されます。エラーは次の方法で処理されます。

- 移行したデータファイルのインデックス作成時にエラーが発生した場合、データファイルがインデックスなしで移行されるか、処理が停止することがあります。メッセージが SAS ログに書き込まれます。インデックスの移行が失敗した場合は、エラーを解決して、ターゲットライブラリにインデックスを再作成します。

  ソースライブラリにインデックスがない場合は、環境とシステムオプションに応じて、インデックスの再作成による移行時のデータセットの修復が試行されます。データセットに、セッションエンコーディングやデータ表現との互換性がない場合は、インデックスを再作成するとエラーが発生します。エラーを解決するには、元のオペレーティングシステムを使用してソースライブラリにインデックスを再作成するか、元の場所からインデックスを移動して、PROC MIGRATE を再度サブミットします。カスタマが OS を使用してデータセットを移動し、移動時にインデックスを含めるのに失敗すると、エラーが発生する場合があります。
- 移行したデータファイルに一貫性制約が適用されるか、または監査証跡や世代が移行されたときに、エラーが発生すると、データファイルがターゲットライブラリから削除されます。ノートが SAS ログに書き込まれます。MOVE オプションを指定した場合、ソースライブラリのデータファイルは削除されません。
- 参照一貫性制約を含むデータファイルの場合、MOVE オプションは、移行が正常終了しても、ソースライブラリのメンバを削除しません。メンバを削除するには、参照一貫性制約を削除しておく必要があります。エラーメッセージが SAS ログに書き込まれます。

NODUPKEY ソートインジケータを含む SAS データセットの移行

NODUPKEY オプションを使用して並べ替えられた SAS データセットを移行する際は、デフォルトの動作を使用することも、KEEPNODUPKEY オプションを指定することもできます。

デフォルトの動作(KEEPNODUPKEY オプションなし)では、ターゲットライブラリで SAS データセットのソートインジケータが保持されます。ただし、NODUPKEY 属性は削除され、SAS ログに警告メッセージが書き込まれます。これがデフォルトの動作となるのは、前のリリースで NODUPKEY オプションを使用して並べ替えられた SAS データセットに、重複キーを含むオブザベーションが保持されている可能性があるためです。PROC SORT のキー変数を基準にして移行した SAS データセットをもう一度並べ替えると、キーの重複したオブザベーションを削除し、正しい属性を記録できます。

KEEPNODUPKEY オプションを指定する場合は、移行したデータを検証して、キーの重複したオブザベーションが存在するかどうかを判断する必要があります。存在する場合は、SAS データセットをもう一度並べ替えて、データと NODUPKEY 並べ替え属性をマッチさせる必要があります。

SAS 6 ライブラリの移行

SAS 6 ソースライブラリでは、次の動作環境がサポートされています。使用している動作環境を 1 列目から探してください。その行に、サポートされているターゲット動作環境についての情報が記載されています。

使用しているライブラリがサポートされていない場合は、“PROC MIGRATE プロシジャの代替” (1188 ページ)を参照してください。カタログだけがサポートされていない場合は、“サポートされないカタログへの追加操作” (1187 ページ)を参照してください。SLIBREF=オプションの詳細については、“SLIBREF=オプションの使用” (1186 ページ)を参照してください。

表 38.1 *サポートされている SAS 6 ライブラリ*

ソースライブラリ	ターゲットライブラリ	カタログのあるライブラリの使用手順
SAS 6.12 AIX または HP-UX または Solaris SPARC	AIX または HP-UX または Solaris SPARC	PROC MIGRATE では、SAS 6 AIX のカタログはサポートされていません。 カタログを含む HP-UX ライブラリまたは Solaris ライブラリに対しては、SLIBREF=オプションを指定します。
SAS 6.12 Windows	Windows(32 ビット版)	カタログはサポートされています。SLIBREF=オプションは使用しないでください。

ソースライブラリ	ターゲットライブラリ	カタログのあるライブラリの使用手順
SAS 6.12 Windows	Windows(64 ビット版)	カタログを含むライブラリに対しては、SLIBREF=オプションを指定します。
SAS 6.09E z/OS	z/OS	カタログはサポートされています。SLIBREF=オプションは使用しないでください。

SAS 6.12 (SAS 6.09E for z/OS)より前に作成された SAS ファイルは、現在の SAS のリリースに移行する前に、SAS 6.12 に変換しておく必要があります。“PROC MIGRATE プロシジャの代替” (1188 ページ)も参照してください。

英語以外の文字を含むデータセットの移行

ASCII-OEM 文字セットを使用する SAS データセットでは、PROC MIGRATE は英語以外の文字は変換しません。ASCII-OEM 文字を含む SAS データセットを現在の SAS のリリースに移行するには、CPORT プロシジャと CIMPORT プロシジャで TRANTAB オプションを使用します。ソースおよびターゲットのデータセットに適切な TRANTAB 値を指定します。(15 章, “CPORT プロシジャ,” (417 ページ)および 11 章, “CIMPORT プロシジャ,” (321 ページ)を参照してください)。

PC 動作環境での短い拡張子のファイルの移行

概要

SAS 7 と SAS 8 では、SHORTFILEEXT オプションは、PC 動作環境でのみ、短縮された 3 文字の拡張子が付いたファイルを作成します。FAT (ファイルアロケーションテーブル)ファイルシステムを使用するオペレーティングシステムには、この機能が必要です。ファイル名に 8 文字まで、ファイル拡張子に 3 文字まで含められるので、FAT ファイルシステムは 8.3 としても参照されます。これらのファイルは PC 環境で作成されます。これらは他の環境の SAS では使用できません。

注: SAS 6 ファイルにはすべて 3 文字の拡張子がついていますが、この問題には影響しません。SAS 6 ファイルは、拡張子に数字の 7 が含まれていないので、区別できます。

次の表は、SAS 7 と SAS 8 のファイルの短い拡張子と標準拡張子です。ライブラリに短い拡張子のファイルが含まれているかどうかを判断するには、SAS エクスプローラでファイル名を参照するか、使用している動作環境のファイル管理ツールを使用します。

表 38.2 SAS 7 と SAS 8 のファイルの短い拡張子と標準拡張子

メンバの種類	短い拡張子	標準拡張子
ACCESS	sa7	sas7bacs
AUDIT	st7	sas7baud
CATALOG	sc7	sas7bcat
DATA	sd7	sas7bdat
DMDB	s7m	sas7bdmd
FDB	sf7	sas7bfdb
INDEX	si7	sas7bndx
ITEMSTOR	sr7	sas7bitm
MDDB	sm7	sas7bmdb
PROGRAM	ss7	sas7bpgm
PUTILITY	sp7	sas7bput
UTILITY	su7	sas7butl
VIEW	sv7	sas7bvew

SAS®9 の短い拡張子のファイルとの互換性

以前のリリースで作成された短い拡張子のファイルについては、読み取り専用アクセスに対応しています。PROC MIGRATE を使用すると、ライブラリを現在の SAS のリリースに移行できます。ソースライブラリに対して LIBNAME ステートメントで SHORTFILEEXT オプションを指定する必要があります。標準拡張子の付いたファイルがターゲットライブラリに書き込まれます。

ファイルは、フルアクセスに対応しています。たとえば、MyLib という名前のライブラリには、短い拡張子をもつ、MyData.sd7 という名前の SAS データセットファイルと、MyCat.sc7 という名前のカタログファイルの 2 つが含まれています。ライブラリを SAS®9 に移行するには、次のコードを使用します。

```
libname MyLib v8 'source-library-pathname' shortfileext; libname NewLib v9 'target-library-p
```

移行作業後、ターゲットライブラリの NewLib には標準拡張子のついた MyData.sas7bdat という名前の SAS データセットファイルと、MyCat.sas7bcat という名前のカタログファイルの 2 つが含まれました。

ライブラリに標準拡張子のファイルも含まれている場合、そのファイルを移行するには、LIBNAME ステートメントで SHORTFILEEXT オプションを指定せずに移行を追加実行します。標準拡張子のファイルと同じ名前の短い拡張子のファイルがないことを確認してください。ターゲットライブラリでは、すべてのファイルに標準拡張子が付いて

います。ターゲットライブラリで、短い拡張子のファイルと標準拡張子のファイルの名前とメンバの種類が同じ場合、2 番目のファイルは移行に失敗します。

検証ツールを使用したライブラリの移行

検証ツールを使用して PROC MIGRATE を実行すると、正常な移行を検証するアウトプットデリバリシステム(ODS)レポートが生成されます。検証ツールは、SAS ウェブサイトの http://support.sas.com/rnd/migration/procmigrate/validtools.html にあります。検証ツールは、基本的な移行、または SLIBREF オプションを使用した移行(RLS を使用)をサポートします。

検証ツールは、サンプルコードのマクロとテンプレートのセットで構成されています。一部のマクロは移行前に実行され、PROC MIGRATE の予測動作を記録します。また別のグループのマクロは移行後に実行され、実際の動作を記録します。最後のグループのマクロは、予測した動作と実際の動作を比較します。

PROC MIGRATE で MOVE オプションを使用すると、検証ツールで、移行されたメンバについてのみ検証出力を作成できます。MOVE オプションは、ターゲットライブラリへの移動後に、ソースライブラリを削除します。そのため、MOVE オプションを使用すると、検証ツールが著しく制限されます。サンプルコードについては、SAS ウェブサイト(前述の URL)の検証ツールのトピックを参照してください。

SLIBREF=オプションの使用

SLIBREF=オプションを使用する場合

次の 2 つの条件が両方とも満たされた場合、SLIBREF=オプションが必須になります。その条件は、SAS 9.1.3 より前に作成されたカタログがソースライブラリに含まれていることと、処理によってターゲットセッションで CEDA が呼び出されることです。

通常、CEDA は互換性のない動作環境に移行する際に呼び出されます。CEDA 処理が PROC MIGRATE で使用されるかどうかを判断するために、検証作業を行うことができます。ターゲットセッションで、ソースセッションで作成されたデータセットの処理を試します。CONTENTS プロシジャのような単純なコードをサブミットします。SAS ログを確認します。CEDA が使用されると、SAS はメッセージをログに書き込みます。CEDA 処理が使用された場合に、ソースライブラリに SAS 9.1.3 より前のリリースで作成されたカタログが含まれている場合は、SLIBREF=オプションを指定する必要があります。

SAS 6 ファイルの場合は、カタログを含む SAS 6 HP-UX ライブラリか Solaris ライブラリに対してのみ SLIBREF=オプションを使用します。詳細については、“SAS 6 ライブラリの移行” (1183 ページ)を参照してください。

SLIBREF=の指定が必要かどうかが不明な場合は、次の SAS ウェブサイトの PROC MIGRATE カルキュレータを使用します。support.sas.com/migration/planning/files/migratecalc

コードの例は“例 2: カタログの互換性がない場合のコンピュータ間の移行” (1190 ページ) および“例 4: カタログの互換性がない場合の同一コンピュータ上の移行 ” (1192 ページ)を参照してください。CEDA の詳細については、次を参照してください。 *SAS 言語リファレンス: 解説編*.

SLIBREF=オプションを使用しない場合

- SAS®9 以降から移行する場合(SAS 9.1.3 から SAS 9.4 への移行など)、SLIBREF=は必要ありません。カタログに互換性がない場合は、IN=引数で SAS/CONNECT または SAS/SHARE サーバーのライブラリ参照名を指定して、SLIBREF=引数は省略します。
- ライブラリにカタログが含まれていない場合、SLIBREF=オプションは使用しないでください。
- ライブラリにカタログが含まれている場合、PROC MIGRATE で CEDA 処理が使用されないのであれば、SLIBREF=オプションは使用しないでください。

SLIBREF=の指定が必要かどうかが不明な場合は、次の SAS ウェブサイトの PROC MIGRATE カルキュレータを使用します。support.sas.com/migration/planning/files/migratecalc

SAS/CONNECT サーバーまたは SAS/SHARE サーバーの必要条件

SLIBREF=オプションを使用するには、ソースライブラリと同じ種類の動作環境で稼働する SAS/CONNECT または SAS/SHARE サーバーへのアクセス権限が必要です。たとえば、ソースセッションが UNIX で稼働している場合、サーバーも UNIX で稼働している必要があります。

カタログが SAS 6 または SAS 8 で作成された場合、SLIBREF=は SAS 8 サーバー経由で割り当てる必要があります。(注:これは、IN=引数で割り当てたサーバーとは異なります)。IN=引数でサーバーを割り当てる場合、IN=サーバーは SAS 9.1.3 以降にする必要があります)。

これらの要件を満たせない場合は、“サポートされないカタログへの追加操作” (1187 ページ)で説明している代替方法を使用してください。

SLIBREF=オプションの制限

ライブラリ内のメンバの種類が CATALOG だけのときに SLIBREF=オプションを使用する場合は、IN=引数を省略します。

SAS 8.2 ライブラリに対して SLIBREF=オプションを使用する場合、マルチラベル出力形式はサポートされません。カタログにマルチラベル出力形式が含まれている場合、ターゲットではその出力形式は作成されず、ログにエラーが表示されます。SAS Note 20052 を参照してください。SAS カスタマーサポートから入手できます。

サポートされないカタログへの追加操作

追加操作を行うとき

PROC MIGRATE では、次の状況でのカタログの移行はサポートされていません。

- ターゲットライブラリへの SAS 6 AIX カタログの移行。
- Linux (x64)または Solaris (x64)ターゲットライブラリへの Tru64 UNIX カタログの移行。

- SLIBREF を使用する必要があり、かつ SAS/CONNECT ソフトウェアまたは SAS/SHARE ソフトウェアのいずれにもアクセス権がない状況での任意のカタログの移行。(SLIBREF=が必須かどうかを確かめるには、"SLIBREF=オプションの使用" (1186 ページ)を参照してください)。

このプロセスはカタログを変換する最良の方法なので、ターゲット環境および現在の SAS のリリースでネイティブです。

この処理は、カタログだけに使用することも、ライブラリのすべてのメンバに使用することもできます。次の理由で、この手法をカタログに限定することをお勧めします。

- 大きなライブラリでは、FTP に時間がかかる可能性があります。
- ライブラリの他のメンバに PROC MIGRATE を使用すると、そのメンバの属性を保持して、検証ツールを使用できます。

詳細については、*SAS ファイルの移動とアクセス*を参照してください。

プロセス

1. ソースセッションで、PROC CPORT を使用して移送ファイルを作成します。(15 章, "CPORT プロシジャ," (417 ページ)を参照してください)。
2. 移送ファイルをターゲット環境に移動します。カタログの移動に RLS (SAS/CONNECT および SAS/SHARE ソフトウェアの機能)を使用しないでください。使用するとエラーが発生します。バイナリ FTP、DOWNLOAD プロシジャ、NFS (Network File System)など、ファイルに直接アクセスする手法を使用する必要があります。
3. ターゲットセッションで、CIMPORT を使用して移送ファイルをインポートします。(11 章, "CIMPORT プロシジャ," (321 ページ)を参照してください)。

PROC MIGRATE プロシジャの代替

代替を使用する場合

PROC MIGRATE では、次のソース環境で作成されたライブラリはサポートされていません。

- CMS、OS/2、OpenVMS VAX、64 ビット版 AIX の SAS 8.2 ライブラリ。
- 未サポートの SAS 6 動作環境すべて。サポートされている SAS 6 動作環境のリストについては、"SAS 6 ライブラリの移行" (1183 ページ)を参照してください。

プロセス

PROC MIGRATE を使用できない複数の動作環境にわたる移行を行うには、次の処理を実行します。

1. SAS のソースインストールでは、変換プロシジャである PROC COPY、PROC CPORT、PROC UPLOAD のいずれかを使用して、ソースライブラリを含む移送ファイルを作成します。*SAS ファイルの移動とアクセス*を参照してください。
2. SAS のターゲットインストールでは、PROC COPY、PROC CIMPORT または PROC DOWNLOAD を使用して、移送ファイルを SAS®9 ライブラリに変換します。

例: MIGRATE プロシジャ

例 1: コンピュータ間の移行

要素: PROC MIGRATE ステートメントのオプション
IN=
OUT=

注: 検証ツールを使用して PROC MIGRATE を実行することをお勧めします。“検証ツールを使用したライブラリの移行” (1186 ページ)を参照してください。
これが移行に対して正しい PROC MIGRATE 例かどうかを調べるには、次の SAS ウェブサイトの PROC MIGRATE カリキュレータを使用してください。support.sas.com/migration/planning/files/migratecalc

詳細

この例では、次のような場合について説明します。

- ソースライブラリとターゲットライブラリが異なるコンピュータ上にある。
- SLIBREF=引数が使用されていない。(SLIBREF=が必須かどうかを確かめるには、“SLIBREF=オプションの使用” (1186 ページ)を参照してください)。
- IN=引数は、ソースライブラリでサポートされているファイルの種類すべてにアクセスします。次のどちらかの方法で、ソースライブラリを IN=引数に割り当てることができます。
  - ネットワークファイルシステム(NFS)経由の直接割り当て
  - SAS®9 SAS/CONNECT または SAS/SHARE サーバー経由の割り当て

  直接手法は NFS 経由でライブラリにアクセスできる場合に限り可能です。この NFS とは UNIX 作業環境の標準プロトコルです。その手法を使用する場合は、NFS と動作環境のドキュメントを参照してください。

  この例では、SAS/CONNECT ソフトウェアを使用します。IN=引数に割り当てる SAS/CONNECT サーバーまたは SAS/SHARE サーバーは、SAS 9.1.3 以降にする必要があります。

プログラム

```
signon serv-ID sascmd='my-sas-invocation-command';

rsubmit; libname Source <engine> 'source-library-pathname'; endrsubmit;

libname Source <engine> server=serv-ID;

libname Target <engine> 'target-library-pathname';

proc migrate in=Source out=Target <options>; run;
```

プログラムの説明

現在の SAS のリリースのセッションから SIGNON コマンドをサブミットして SAS/CONNECT サーバーセッションを起動します。 複数のコンピュータにわたる作業なので、サーバー ID にマシン名を指定してください。

```
signon serv-ID sascmd='my-sas-invocation-command';
```

このリモートセッション内で、移行対象のライブラリメンバを含むソースライブラリにライブラリ参照名を割り当てます。 SAS/CONNECT に対して RSUBMIT および ENDRSUBMIT コマンドを使用します。

```
rsubmit; libname Source <engine> 'source-library-pathname'; endrsubmit;
```

現在のリリースのローカル(クライアント)セッションで、ステップ 2 と同じソースライブラリ参照名を割り当てます。 ただし、物理的な場所にライブラリ参照名を割り当てないでください。その代わり、ステップ 1 で SIGNON コマンドに割り当てたサーバー ID(この例では serv-ID)を SERVER=オプションに指定します。

```
libname Source <engine> server=serv-ID;
```

ターゲットライブラリを割り当てます。

```
libname Target <engine> 'target-library-pathname';
```

PROC MIGRATE を使用します。 ライブラリにカタログが含まれている場合は、そのかわりに“例 2: カタログの互換性がない場合のコンピュータ間の移行” (1190 ページ)を参照してください。

```
proc migrate in=Source out=Target <options>; run;
```

例 2: カタログの互換性がない場合のコンピュータ間の移行

要素: PROC MIGRATE ステートメントのオプション
IN=
OUT=
SLIBREF=

注: 検証ツールを使用して PROC MIGRATE を実行することをお勧めします。“検証ツールを使用したライブラリの移行” (1186 ページ)を参照してください。

これが移行に対して正しい PROC MIGRATE 例かどうかを調べるには、次の SAS ウェブサイトの PROC MIGRATE カリキュレータを使用してください。support.sas.com/migration/planning/files/migratecalc

詳細

この例では、次のような場合について説明します。

- ソースライブラリとターゲットライブラリが異なるコンピュータ上にある。
- SLIBREF=引数が、ソースライブラリのカタログにアクセスする。(SLIBREF=が必須かどうかを確かめるには、“SLIBREF=オプションの使用” (1186 ページ)を参照してください)。SLIBREF=引数は、カタログにアクセス可能な SAS セッションで稼働している SAS/CONNECT サーバーまたは SAS/SHARE サーバーに割り当てる必要があります。たとえば、ソースライブラリに 32 ビット版 Solaris で作成された SAS 8.2 カタログが含まれている場合、SLIBREF=は SAS 8.2 32 ビット版 Solaris サーバーに割り当てる必要があります。カタログが SAS 6 で作成された場合、

SLIBREF=は、SAS 6 カタログのデータ表現と互換性のある SAS 8.2 サーバー経由で割り当てる必要があります。

- IN=引数は、ソースライブラリでサポートされている残りのファイルの種類にアクセスします。次のどちらかの方法で、ソースライブラリを IN=引数に割り当てることができます。
  - ネットワークファイルシステム(NFS)経由の直接割り当て
  - SAS/CONNECT または SAS/SHARE サーバー経由の割り当て

  この例では、UNIX 作業環境の標準プロトコルである NFS を使用します。NFS と動作環境に関するドキュメントを参照してください。

プログラム

```
signon v8srv sascmd='my-v8-sas-invocation-command';

rsubmit; libname Srclib <engine> 'source-library-pathname'; endrsubmit;

libname Source <engine> '/nfs/v8machine-name/source-library-pathname';

libname Srclib <engine> server=v8srv;

libname Target <engine> 'target-library-pathname';

proc migrate in=Source out=Target slibref=Srclib <options>; run;

proc migrate out=Target slibref=Srclib <options>; run;
```

プログラムの説明

現在の SAS のリリースのセッションから SIGNON コマンドをサブミットして SAS/CONNECT サーバーセッションを起動します。 複数のコンピュータにわたる作業なので、サーバー ID にマシン名を指定してください。

```
signon v8srv sascmd='my-v8-sas-invocation-command';
```

このリモート SAS 8.2 セッション内で、移行対象のライブラリメンバを含むソースライブラリにライブラリ参照名を割り当てます。 SAS/CONNECT に対して RSUBMIT および ENDRSUBMIT コマンドを使用します。

```
rsubmit; libname Srclib <engine> 'source-library-pathname'; endrsubmit;
```

現在のリリースのローカル(クライアント)セッションで、NFS 経由で同じソースライブラリに割り当てます。

```
libname Source <engine> '/nfs/v8machine-name/source-library-pathname';
```

同じライブラリ参照名を、ステップ 2 と同じソースライブラリ参照名(この例では、Srclib)に割り当てます。 ただし、物理的な場所にライブラリ参照名を割り当てないでください。その代わり、ステップ 1 で SIGNON コマンドに割り当てたサーバー ID(この例では V8SRV)を SERVER=オプションに指定します。

```
libname Srclib <engine> server=v8srv;
```

ターゲットライブラリを割り当てます。

```
libname Target <engine> 'target-library-pathname';
```

PROC MIGRATE で SLIBREF=オプションを使用します。 IN=オプションと OUT=オプションについては、通常のソースライブラリ参照名とターゲットライブラリ参照名(この例で

は、それぞれ Source と Target)を指定します。SERVER=オプションを使用するライブラリ参照名(この例では、Srclib)に SLIBREF=を設定します。

```
proc migrate in=Source out=Target slibref=Srclib <options>; run;
```

あるいは、ライブラリ内のメンバの種類が CATALOG だけのときに SLIBREF=オプションを使用する場合は、IN=引数を省略します。

```
proc migrate out=Target slibref=Srclib <options>; run;
```

例 3: 同一コンピュータ上の移行

要素: PROC MIGRATE ステートメントのオプション
IN=
OUT=

注: 検証ツールを使用して PROC MIGRATE を実行することをお勧めします。“検証ツールを使用したライブラリの移行” (1186 ページ)を参照してください。

これが移行に対して正しい PROC MIGRATE 例かどうかを調べるには、次の SAS ウェブサイトの PROC MIGRATE カリキュレータを使用してください。support.sas.com/migration/planning/files/migratecalc

詳細

この例では、次のような場合について説明します。

- ソースライブラリとターゲットライブラリが 1 つのコンピュータ上にある。
- SLIBREF=引数が使用されていない。(SLIBREF=が必須かどうかを確かめるには、“SLIBREF=オプションの使用” (1186 ページ)を参照してください)。
- IN=引数は、ソースライブラリでサポートされているファイルの種類すべてにアクセスします。ソースとターゲットが 1 つのコンピュータ上に存在するので、NFS は使用されません。

プログラム

```
libname Source <engine> 'source-library-pathname'; libname Target base 'target-library-p
```

プログラムの説明

現在の SAS のリリースのセッションから、次をサブミットします。

```
libname Source <engine> 'source-library-pathname'; libname Target base 'target-library-p
```

例 4: カタログの互換性がない場合の同一コンピュータ上の移行

要素: PROC MIGRATE ステートメントのオプション
IN=
OUT=
SLIBREF=

注: 検証ツールを使用して PROC MIGRATE を実行することをお勧めします。“検証ツールを使用したライブラリの移行” (1186 ページ)を参照してください。

これが移行に対して正しい PROC MIGRATE 例かどうかを調べるには、次の SAS ウェブサイトの PROC MIGRATE カリキュレータを使用してください。support.sas.com/migration/planning/files/migratecalc

詳細

この例では、次のような場合について説明します。

- ソースライブラリとターゲットライブラリが 1 つのコンピュータ上にある。
- SLIBREF=引数が、ソースライブラリのカタログにアクセスする。(SLIBREF=が必須かどうかを確かめるには、“SLIBREF=オプションの使用” (1186 ページ)を参照してください)。SLIBREF=引数は、カタログにアクセス可能な SAS セッションで稼働している SAS/CONNECT サーバーまたは SAS/SHARE サーバーに割り当てる必要があります。たとえば、ソースライブラリに 32 ビット版 Solaris で作成された SAS 8.2 カタログが含まれている場合、SLIBREF=は SAS 8.2 32 ビット版 Solaris サーバーに割り当てる必要があります。カタログが SAS 6 で作成された場合、SLIBREF=は、SAS 6 カタログのデータ表現と互換性のある SAS 8.2 サーバー経由で割り当てる必要があります。
- IN=引数は、ソースライブラリでサポートされている残りのファイルの種類にアクセスします。ソースライブラリとターゲットライブラリが 1 つのコンピュータ上に存在するので、NFS は使用されません。

プログラム

```
signon v8srv sascmd='my-v8-sas-invocation-command';

rsubmit; libname Srclib <engine> 'source-library-pathname'; endrsubmit;

libname Srclib <engine> server=v8srv;

libname Source <engine> 'source-library-pathname'; libname Target <engine> 'target-libra

proc migrate in=Source out=Target slibref=Srclib <options>; run;

proc migrate out=Target slibref=Srclib <options>; run;
```

プログラムの説明

現在の SAS のリリースのセッションから SIGNON コマンドをサブミットして SAS/CONNECT サーバーセッションを起動します。

```
signon v8srv sascmd='my-v8-sas-invocation-command';
```

このリモート SAS 8.2 セッション内で、移行対象のライブラリメンバを含むソースライブラリにライブラリ参照名を割り当てます。 SAS/CONNECT に対して RSUBMIT および ENDRSUBMIT コマンドを使用します。

```
rsubmit; libname Srclib <engine> 'source-library-pathname'; endrsubmit;
```

現在のリリースのローカル(クライアント)セッションで、ステップ 2 と同じソースライブラリに割り当てます。 ただし、物理的な場所にライブラリ参照名を割り当てないでください。その代わり、ステップ 1 で SIGNON コマンドに割り当てたサーバー ID(この例では V8SRV)を SERVER=オプションに指定します。

```
libname Srclib <engine> server=v8srv;
```

ライブラリ参照名を 2 つ割り当てます。1 つはソースライブラリ、もう 1 つはターゲットライブラリが割り当て先です。 ソースライブラリはステップ 2 で割り当てたものと同じですが、異なるライブラリ参照名を使用する必要があります。

```
libname Source <engine> 'source-library-pathname'; libname Target <engine> 'target-libra
```

PROC MIGRATE で SLIBREF=オプションを使用します。 IN=オプションと OUT=オプションでは、ステップ 4 で割り当てたライブラリ参照名(この例では、それぞれ Source と Target)を指定します。SERVER=オプションを使用するライブラリ参照名(この例では、Srclib)に SLIBREF=を設定します。

```
proc migrate in=Source out=Target slibref=Srclib <options>; run;
```

あるいは、ライブラリ内のメンバの種類が CATALOG だけのときに SLIBREF=オプションを使用する場合は、IN=引数を省略します。

```
proc migrate out=Target slibref=Srclib <options>; run;
```

例 5: カタログの互換性がない場合の SAS®9 リリースからの移行

要素: PROC MIGRATE ステートメントのオプション
IN=
OUT=

注: 検証ツールを使用して PROC MIGRATE を実行することをお勧めします。“検証ツールを使用したライブラリの移行” (1186 ページ)を参照してください。

これが移行に対して正しい PROC MIGRATE 例かどうかを調べるには、次の SAS ウェブサイトの PROC MIGRATE カリキュレータを使用してください。support.sas.com/migration/planning/files/migratecalc

詳細

SAS®9 以降から移行する場合(SAS 9.1.3 から SAS 9.4 への移行など)、SLIBREF=は必要ありません。カタログに互換性がない場合は、IN=引数で SAS/CONNECT または SAS/SHARE サーバーのライブラリ参照名を指定して、SLIBREF=引数は省略します。

この例では、次のような場合について説明します。

- ソースライブラリとターゲットライブラリは、同一コンピュータ上に存在する場合も、異なるコンピュータ上に存在する場合もある。
- SLIBREF=引数が使用されていない。(SLIBREF=が必須かどうかを確かめるには、“SLIBREF=オプションの使用” (1186 ページ)を参照してください)。
- IN=引数は、SAS/CONNECT または SAS/SHARE ソフトウェアを使用して、互換性のないカタログも含めて、ソースライブラリでサポートされているファイルの種類すべてにアクセスします。この例では、SAS/CONNECT ソフトウェアを使用します。IN=引数に割り当てる SAS/CONNECT サーバーまたは SAS/SHARE サーバーは、SAS 9.1.3 以降にする必要があります。

プログラム

```
signon serv-ID sascmd='my-sas-invocation-command';

rsubmit; libname Source <engine> 'source-library-pathname'; endrsubmit;
```

```
libname Source <engine> server=serv-ID;

libname Target <engine> 'target-library-pathname';

proc migrate in=Source out=Target <options>; run;
```

プログラムの説明

現在の SAS のリリースのセッションから SIGNON コマンドをサブミットして SAS/CONNECT サーバーセッションを起動します。 複数のコンピュータにわたって作業をする場合は、サーバー ID にマシン名を指定してください。

```
signon serv-ID sascmd='my-sas-invocation-command';
```

このリモートセッション内で、移行対象のライブラリメンバを含むソースライブラリにライブラリ参照名を割り当てます。 SAS/CONNECT に対して RSUBMIT および ENDRSUBMIT コマンドを使用します。

```
rsubmit; libname Source <engine> 'source-library-pathname'; endrsubmit;
```

現在のリリースのローカル(クライアント)セッションで、ステップ 2 と同じソースライブラリ参照名を割り当てます。ただし、物理的な場所にライブラリ参照名を割り当てないでください。 その代わり、ステップ 1 で SIGNON コマンドに割り当てたサーバー ID(この例では serv-ID)を SERVER=オプションに指定します。

```
libname Source <engine> server=serv-ID;
```

ターゲットライブラリを割り当てます。

```
libname Target <engine> 'target-library-pathname';
```

PROC MIGRATE を使用します。

```
proc migrate in=Source out=Target <options>; run;
```

39 章

OPTIONS プロシジャ

概要: OPTIONS プロシジャ

OPTIONS プロシジャは、SAS システムオプションの現在の設定を SAS ログにリストします。

SAS システムオプションでは、SAS 出力形式による出力の制御、ファイルの処理、データセットの処理、動作環境との交互作用、単一の SAS プログラムや SAS データセットに固有ではないその他のタスクが実行されます。OPTIONS プロシジャを使用すると、オプションやオプショングループについての情報を得られます。OPTIONS プロシジャが提供する情報の一部を次に示します。

- オプションの現在の値とその設定方法
- オプションの説明
- オプションの有効な構文、有効なオプション値および値の範囲
- システムオプションの設定場所
- サイト管理者によるオプションの制限の可否
- オプションが制限されているかどうか

- システムオプショングループに属しているシステムオプション
- 動作環境に固有のシステムオプション
- オプション値が INSERT または APPEND システムオプションによって変更されているかどうか
- OPTSAVE プロシジャまたは DMOPTSAVE コマンドで保存可能なシステムオプション

SAS のシステムオプションの詳細については、*SAS システムオプション: リファレンス*を参照してください。

構文: OPTIONS プロシジャ

参照項目: “OPTIONS Procedure: UNIX” (*SAS Companion for UNIX Environments*)
“OPTIONS Procedure: Windows” (*SAS Companion for Windows*)
“OPTIONS Procedure: z/OS” (*SAS Companion for z/OS*)

PROC OPTIONS <*option(s)*>;

ステートメント	タスク	例
“PROC OPTIONS ステートメント”	現在のシステムオプション設定を SAS ログにリストします。	Ex. 1, Ex. 2, Ex. 3, Ex. 4

PROC OPTIONS ステートメント

SAS システムオプションの現在の設定を SAS ログにリストします。

例: “例 1: 簡易形式のオプションリストを作成する” (1208 ページ)
“例 2: 単一オプションの設定を表示する” (1209 ページ)
“例 3: 拡張パス環境変数の表示” (1210 ページ)
“例 4: INSERT オプションと APPEND オプションで指定可能なオプションのリスト” (1211 ページ)

構文

PROC OPTIONS <*option(s)*>;

オプション引数の要約

LISTGROUPS
システムオプショングループならびに各グループの説明を表示します。

Choose the format of the listing

DEFINE

オプション、オプショングループおよびオプションの種類の概要説明を表示します。

EXPAND
文字オプションの表示時に、オプション値の環境変数を環境変数の値に置き換えます。オプションがブール値オプション(CENTER や NOCENTER など)の場合、またはオプションの値が数値の場合、EXPAND は無視されます。

HEXVALUE
システムオプション文字値を 16 進値で表示します。

LOGNUMBERFORMAT
ロケール固有の句読点を使用して数値システムオプション値を表示します。

LONG
各システムオプションを説明付きで別の行にリストします。

NOEXPAND
パスの表示時に、環境変数の値ではなく環境変数を使用してパスを表示します。これがデフォルトです。

NOLOGNUMBERFORMAT
カンマやピリオドなどの句読点を使用せずに数値システムオプション値を表示します。これがデフォルトです。

SHORT
オプションの圧縮されたリストを説明を省いて表示するよう指定します。

VALUE
オプションの値とスコープ、さらにその値が設定された方法を表示します。

Restrict the number of options displayed

GROUP=*group-name*

GROUP=*(group-name–1 ...group-name-n*)
group-name で指定した 1 つ以上のグループのオプションを表示します。

HOST
ホストオプションのみを表示します。

LISTINSERTAPPEND
INSERT および APPEND システムオプションで値の変更が可能なシステムオプションをリストにします。

LISTOPTSAVE
PROC OPTSAVE コマンドまたは DMOPTSAVE コマンドとともに保存できるシステムオプションを表示します。

LISTRESTRICT
サイト管理者によって制限可能なシステムオプションをリストします。

NOHOST
ポータブルオプションのみを表示します。

OPTION=*option-name*

OPTION=(*option-name-1 … option-name-n*)
1 つ以上のシステムオプションについての情報を表示します。

RESTRICT
サイト管理者によって更新が制限されているシステムオプションを表示します。

オプション引数

DEFINE
オプション、オプショングループおよびオプションの種類の概要説明を表示します。SAS では、オプションをいつ設定できるか、オプションを制限できるかどうか、オプションの有効値、OPTSAVE プロシジャがオプションを保存するかどうかについての情報が表示されます。

操作 SHORT を指定すると、このオプションは無視されます。

例 “例 2: 単一オプションの設定を表示する” (1209 ページ)

EXPAND
文字オプションの表示時に、オプション値の環境変数を環境変数の値に置き換えます。オプションがブール値オプション(CENTER や NOCENTER など)の場合、またはオプションの値が数値の場合、EXPAND は無視されます。

制限事項 変数展開は、Windows および UNIX 動作環境でのみ有効です。

ヒント デフォルトでは、展開された変数と一緒に表示されるオプション値もあります。また、PROC OPTIONS ステートメントの EXPAND オプションが必須のオプションもあります。PROC OPTIONS ステートメントで DEFINE オプションを使用すると、デフォルトでオプション値によって変数が展開されるかどうか、または EXPAND オプションが必須かどうかを判断できます。PROC OPTIONS DEFINE からの出力で次の情報が示される場合は、EXPAND オプションを使用して変数値を展開する必要があります。

```
Expansion: Environment variables, within the option value,
           are not expanded
```

参照項目 “NOEXPAND” (1201 ページ) オプション。環境変数を表示するパスを示します。

例 “例 3: 拡張パス環境変数の表示” (1210 ページ)

GROUP=*group-name*
GROUP=*(group-name–1 ...group-name-n)*
group-name で指定した 1 つ以上のグループのオプションを表示します。

要件 2 つ以上のグループを指定する場合は、グループ名をかっこで囲み、スペースでグループ名を区切ります。

参照項目 “システムオプショングループの情報を表示する” (1205 ページ)

HEXVALUE
システムオプション文字値を 16 進値で表示します。

HOST
ホストオプションのみを表示します。

参照項目 “NOHOST” (1201 ページ) オプション。ポータブルオプションのみを表示します。

LISTINSERTAPPEND
INSERT および APPEND システムオプションで値の変更が可能なシステムオプションをリストにします。INSERT オプションは、システムオプション値リストの最初の値として挿入される値を指定します。APPEND オプションは、システムオプション値

リストの最終値として追加される値を指定します。どのシステムオプションで、値を値リストの最初に挿入したり最後に追加したりできるのかを表示するには、LISTINERTAPPEND を使用します。

参照項目 “INSERT= System Option” (*SAS System Options: Reference*) および “APPEND= System Option” (*SAS System Options: Reference*)

例 “例 4: INSERT オプションと APPEND オプションで指定可能なオプションのリスト” (1211 ページ)

LISTGROUPS
システムオプショングループならびに各グループの説明を表示します。

参照項目 “システムオプショングループの情報を表示する” (1205 ページ)

LISTOPTSAVE
PROC OPTSAVE コマンドまたは DMOPTSAVE コマンドとともに保存できるシステムオプションを表示します。

LISTRESTRICT
サイト管理者によって制限可能なシステムオプションをリストします。

参照項目 “RESTRICT” (1202 ページ) オプション。サイト管理者によって制限されているオプションをリストします。

LONG
各システムオプションを説明付きで別の行にリストします。これがデフォルトです。あるいは、説明なしの簡潔なリストを作成することもできます。

参照項目 “SHORT” (1202 ページ) オプション。説明なしの簡潔なリストを作成します。

例 “例 1: 簡易形式のオプションリストを作成する” (1208 ページ)

LOGNUMBERFORMAT
ロケール固有の句読点を使用して数値システムオプション値を表示します。

参照項目 “NOLOGNUMBERFORMAT” (1202 ページ) オプション。カンマを使用せずに数値オプション値を表示します。

例 “例 2: 単一オプションの設定を表示する” (1209 ページ)

NOEXPAND
パスの表示時に、環境変数の値ではなく環境変数を使用してパスを表示します。これがデフォルトです。

参照項目 “EXPAND” (1200 ページ) オプション。環境変数の値を展開してパスを表示します。

NOHOST
ポータブルオプションのみを表示します。

別名 PORTABLE または PORT

参照項目 “HOST” (1200 ページ)オプション。ホストオプションのみを表示します。

NOLOGNUMBERFORMAT
カンマやピリオドなどの句読点を使用せずに数値システムオプション値を表示します。これがデフォルトです。

参照項目 “LOGNUMBERFORMAT” (1201 ページ) オプション。カンマを使用して数値システムオプション値を表示します。

OPTION=*option-name*
OPTION=(*option-name-1 … option-name-n*)
概要説明、および(存在する場合は)*option-name* で指定したオプションの値を表示します。DEFINE オプションと VALUE オプションは、オプションについての詳細情報を提供します。

option-name
プロシジャへの入力として使用するオプションを指定します。

要件 SAS システムオプションで等号(PAGESIZE=など)が使用されている場合、OPTION=に対するオプションの指定時にはその等号を含めないでください。

例 “例 2: 単一オプションの設定を表示する” (1209 ページ)

RESTRICT
サイト管理者によって制限オプション構成ファイルに設定されたシステムオプションを表示します。ユーザーはこれらのオプションを変更できません。RESTRICT オプションは、制限されているオプションごとに、オプションの値、スコープおよび設定内容を表示します。

サイト管理者が一切のオプションを制限していない場合は、SAS ログに次のメッセージが表示されます。

```
Your Site Administrator has not restricted any SAS options.
```

参照項目 “LISTRESTRICT” (1201 ページ) オプション。サイト管理者が制限できるオプションをリストします。

SHORT
オプションの圧縮されたリストを説明を省いて表示するよう指定します。

参照項目 “LONG” (1201 ページ) オプション。オプションの説明があるリストを作成します。

VALUE
オプションの値とスコープ、さらにその値が設定された方法を表示します。構成ファイルを使用してこの値が設定された場合、SAS ログにはその構成ファイルの名前が表示されます。INSERT システムオプションまたは APPEND システムオプションを使用してこのオプションが設定された場合、SAS ログには挿入または追加された値が表示されます。

操作 SHORT を指定すると、このオプションは無効になります。

このオプションが Threaded Kernel(TK)システムオプションのグループ内にあるときは、**How option value set** の値が次のように表示されます

```
Internal
```

注 EMAILPW や METAPASS といったパスワードである SAS オプションにより、この値は実際のパスワード ではなく xxxxxxxx に戻ります。

例 “例 2: 単一オプションの設定を表示する” (1209 ページ)

システムオプションリストの表示

PROC OPTIONS の実行によって生じるログは、すべての動作環境で利用可能 なオプションに対する、および単一の動作環境に特化したオプションに対するシステムオプションを表示できます。すべての動作環境で利用可能なオプションはポータブルオプションといいます。単一の動作環境に特化したオプションはホストオプションといいます。

次の例では、ポータブルオプションの設定を表示しているログの一部を示します。

```
proc options;
run;
```

ログ 39.1 *SAS システムオプションのリストの一部を示す SAS ログ*

```
Portable Options:ANIMATION=STOP    Specifies whether to start or stop animation.ANIMDURATION=MIN
Specifies the number of seconds that each animation frame displays.ANIMLOOP=YES      Specifies the
number of iterations that animated images repeat.ANIMOVERLAY       Specifies that animation frames are
overlaid in order to view all frames.APPEND=           Specifies an option=value pair to insert the
value at the end of the existing option value.APPLETLOC=site-specific-path Specifies the location of
Java applets, which is typically a URL.ARMAGENT=         Specifies an ARM agent (which is an
executable module or keyword, such as LOG4SAS) that contains a specific implementation of the ARM
API.ARMLOC=ARMLOG.LOG Specifies the location of the ARM log.ARMSUBSYS=(ARM_NONE) Specifies the SAS ARM
subsystems to enable or disable.AUTOCORRECT       Automatically corrects misspelled procedure names
and keywords, and global statement names.
```

proc options;をサブミットすると、このログにはポータブルオプションとホストオプションの両方が表示されます。

ホストオプションのみを閲覧するには、このバージョンの OPTIONS プロシジャを使用してください。

```
proc options host;
run;
```

ログ 39.2 *SAS ログに表示されるホストオプションリストの一部*

```
Host Options:ACCESSIBILITY=STANDARD Specifies whether accessibility features are enabled in the
Customize Tool dialog box and in some Properties dialog boxes.ALIGNSASIOFILES   Aligns SAS files on a
page boundary for improved performance.ALTLOG=           Specifies the location for a copy of the SAS
log when SAS is running in batch mode.ALTPRINT=         Specifies the location for a copy of the SAS
procedure output when SAS is running in batch mode.AUTHPROVIDERDOMAIN= Specifies the authentication
provider that is associated with a domain.AUTHSERVER=       Specifies the domain server that finds and
authenticates secure server logins.AWSCONTROL=(SYSTEMMENU MINMAX TITLE) Specifies whether the main SAS
window includes a title bar, a system control menu, and minimize and maximize buttons.AWSDEF=(0 0 79
80) Specifies the location and dimensions of the main SAS window when SAS
initializes.AWSMENU           Displays the menu bar in the main SAS window.
```

オプションの情報の表示

1 つ以上の特定オプションの設定を表示するには、PROC OPTIONS ステートメントで OPTION=オプションと DEFINE オプションを使用します。次の例は、PROC OPTIONS が単一 SAS システムオプションに対して作成するログを示しています。

```
proc options option=errorcheck define;
run;
```

ログ 39.3 *単一 SAS システムオプションの設定*

```
5    proc options option=errorcheck define; 6    run; SAS (r) Proprietary Software Release 9.4  TS1M2
ERRORCHECK=NORMAL Option Definition Information for SAS Option ERRORCHECK Group= ERRORHANDLING Group
Description:Error messages and error conditions settings Description:Specifies whether SAS enters
syntax-check mode when errors are found in the LIBNAME, FILENAME,  %INCLUDE, and LOCK
statements.Type:The option value is of type CHARACTER Maximum Number of Characters:10 Casing:The
option value is retained uppercased Quotes:If present during "set", start and end quotes are removed
Parentheses:The option value does not require enclosure within parentheses.If present, the parentheses
are retained.Expansion:Environment variables, within the option value, are not expanded Number of
valid values:2 Valid value:NORMAL Valid value:STRICT When Can Set:Startup or anytime during the SAS
Session Restricted:Your Site Administrator can restrict modification of this option Optsave:PROC
Optsave or command Dmoptsave will save this option
```

2 つ以上のオプションの設定を表示するには、オプションをかっこで囲み、スペースでオプションを区切ります。

```
proc options option=(pdfsecurity pdfpassword) define;
run;
```

ログ 39.4 *2 つの SAS システムオプションの設定*

```
7    proc options option=(pdfsecurity pdfpassword) define; 8    run; SAS (r) Proprietary Software
Release 9.4  TS1M2 PDFSECURITY=NONE Option Definition Information for SAS Option PDFSECURITY Group=
PDF Group Description:PDF settings Group= SECURITY Group Description:Security settings
Description:Specifies the level of encryption to use for PDF documents.Type:The option value is of
type CHARACTER Maximum Number of Characters:4 Casing:The option value is retained uppercased Quotes:If
present during "set", start and end quotes are removed Parentheses:The option value does not require
enclosure within parentheses.If present, the parentheses are retained.Expansion:Environment variables,
within the option value, are not expanded Number of valid values:3 Valid value:HIGH Valid value:LOW
Valid value:NONE When Can Set:Startup or anytime during the SAS Session Restricted:Your Site
Administrator can restrict modification of this option Optsave:PROC Optsave or command Dmoptsave will
save this option PDFPASSWORD=xxxxxxxx
```

```
Option Definition Information for SAS Option PDFPASSWORD Group= PDF Group Description:PDF settings
Group= SECURITY Group Description:Security settings Description:Specifies the password to use to open
a PDF document and the password used by a PDF document owner.Type:The option value is of type
CHARACTER Maximum Number of Characters:2048 Casing:The option value is retained with original casing
Quotes:If present during "set", start and end quotes are removed Parentheses:The option value must be
enclosed within parentheses.The parentheses are retained.Password Option Value:Can not Print or
Display Expansion:Environment variables, within the option value, are not expanded When Can
Set:Startup or anytime during the SAS Session Restricted:Your Site Administrator cannot restrict
modification of this option Optsave:PROC Optsave or command Dmoptsave will not save this option
```

システムオプショングループの情報を表示する

各 SAS システムオプションは 1 つ以上のグループに属しています。このグループ分けは、エラー処理や並べ替えなどの機能に基づいています。システムオプショングループ、およびそのグループの 1 つ以上に属するシステムオプションのリストが表示されます。

システムオプショングループのリストを表示するには、LISTGROUPS オプションを使用します。

```
proc options listgroups;
run;
```

ログ 39.5 *SAS システムオプショングループのリスト*

```
26   proc options listgroups; 27   run; SAS (r) Proprietary Software Release
9.4  TS1M2 Option Groups GROUP=ADABAS                ADABAS
GROUP=ANIMATION              Animation GROUP=CODEGEN                Code
generation GROUP=COMMUNICATIONS         Networking and encryption
GROUP=DATACOM                Datacom GROUP=DATAQUALITY            Data Quality
GROUP=DB2                    DB2 GROUP=EMAIL                  E-mail
GROUP=ENVDISPLAY             Display
```

```
       GROUP=ENVFILES               Files GROUP=ERRORHANDLING          Error
handling GROUP=EXECMODES              Initialization and operation
GROUP=EXTFILES               External files GROUP=GRAPHICS               Driver
settings GROUP=HELP                   Help GROUP=IDMS                   IDMS
GROUP=IMS                    IMS GROUP=INPUTCONTROL           Data Processing
GROUP=INSTALL                Installation GROUP=ISPF                    ISPF
GROUP=LANGUAGECONTROL        Language control GROUP=LISTCONTROL
Procedure output GROUP=LOGCONTROL             SAS log
GROUP=LOG_LISTCONTROL        SAS log and procedure output
GROUP=MACRO                  SAS macro GROUP=MEMORY                 Memory
GROUP=META                   Metadata GROUP=ODSPRINT               ODS Printing
GROUP=PDF                    PDF GROUP=PERFORMANCE            Performance
GROUP=REXX                   REXX GROUP=SASFILES               SAS Files
GROUP=SECURITY               Security GROUP=SMF                    SMF
GROUP=SORT                   Procedure options GROUP=SQL                    SQL
GROUP=SVG                    SVG GROUP=TK                     TK
```

特定のグループに属するシステムオプションを表示するには、GROUP=オプションを使用します。1 つ以上のグループを指定できます。

```
proc options group=(svg graphics);
run;
```

ログ 39.6 *GROUP=オプションを使用したサンプル出力*

```
5    proc options group=(svg graphics); 6    run; SAS (r) Proprietary Software Release 9.4  TS1M2
Group=SVG ANIMATION=STOP    Specifies whether to start or stop animation.ANIMDURATION=MIN  Specifies
the number of seconds that each animation frame displays.ANIMLOOP=YES      Specifies the number of
iterations that animated images repeat.ANIMOVERLAY       Specifies that animation frames are overlaid
in order to view all frames.SVGAUTOPLAY       Starts animation when the page is loaded in the
browser.NOSVGCONTROLBUTTONS Does not display the paging control buttons and an index in a multipage
SVG document.SVGFADEIN=0       Specifies the number of seconds for the fade-in effect for a
graph.SVGFADEMODE=OVERLAP Specifies whether to use sequential frames or to overlap frames for the fade-
in effect of a graph.SVGFADEOUT=0      Specifies the number of seconds for a graph to fade out of
view.SVGHEIGHT=        Specifies the height of the viewport.Specifies the value of the height
attribute of the outermost SVG element.NOSVGMAGNIFYBUTTON Disables the SVG magnifier
tool.SVGPRESERVEASPECTRATIO= Specifies whether to force uniform scaling of SVG output.Specifies the
preserveAspectRatio attribute on the outermost SVG element.SVGTITLE=         Specifies the text in the
title bar of the SVG output.Specifies the value of the TITLE element in the SVG file.SVGVIEWBOX=
Specifies the coordinates, width, and height that are used to set the viewBox attribute on the
outermost SVG element.SVGWIDTH=         Specifies the width of the viewport.Specifies the value of the
width attribute of the outermost SVG element.SVGX=             Specifies the x-axis coordinate of one
corner of the rectangular region for an embedded SVG element.Specifies the x attribute in the
outermost SVG element.SVGY=             Specifies the y-axis coordinate of one corner of the
rectangular region for an embedded SVG element.Specifies the y attribute in the outermost SVG
element.Group=GRAPHICS DEVICE=           Specifies the device driver to which SAS/GRAPH sends
procedure output.GSTYLE            Uses ODS styles to generate graphs that are stored as GRSEG catalog
entries.GWINDOW           Displays SAS/GRAPH output in the GRAPH window.MAPS=("!sasroot\path-to-maps")
Specifies the location of SAS/GRAPH map data sets.MAPSGFK=(          "!sasroot\path-to-maps"  )
Specifies the location of GfK maps.MAPSSAS=(          "!sasroot\path-to-maps"  ) Specifies the
location of SAS map data sets.FONTALIAS=        Assigns a Windows font to one of the SAS fonts.
```

次のグループ名を GROUP=オプションの値として使用すると、グループ内のシステムオプションを表示できます。

ANIMATION	GRAPHICS	ODSPRINT
CODEGEN	HELP	PDF
COMMUNICATIONS	INPUTCONTROL	PERFORMANCE
DATAQUALITY	INSTALL	SASFILES
EMAIL	LANGUAGECONTROL	SECURITY
ENVDISPLAY	LOGCONTROL	SORK
ENVFILES	LOG_LISTCONTROL	SQL
ERRORHANDLING	MACRO	SVG
EXECMODES	MEMORY	TK
EXTFILES	META	

次のグループを使用して動作環境固有の値一覧を表示できます。この値は、GROUP=オプションを PROC OPTIONS と共に使用すると取得できる場合があります。

ADABAS	IDMS	REXX
CODEGEN	IMS	SMF
DATACOM	ISPF	
DB2	ORACLE	

動作環境の情報

これらのホスト固有のオプションの詳細については、動作環境に関する SAS ドキュメントを参照してください。

制限オプションの表示

サイト管理者は一部のシステムオプションを制限して、サイトに対して設定されたオプションに SAS セッションを適合させることができます。制限オプションを変更できるのはサイト管理者だけです。OPTIONS プロシジャでは、制限オプションについての情報を表示するオプションが 2 つ提供されます。RESTRICT オプションは、サイト管理者が制限したシステムオプションをリストします。LISTRESTRICT オプションは、サイト管理者による制限が可能なオプションをリストします。制限できないオプションのリストについては、“System Options That Cannot Be Restricted” (*SAS System Options: Reference*)を参照してください。

次の SAS ログには、RESTRICT オプション指定時の出力、および LISTRESTRICT オプション指定時の出力の一部が表示されています。

ログ 39.7 *サイト管理者によって制限されているオプションのリスト*

```
1 proc options restrict; 2    run; SAS (r) Proprietary Software Release 9.4
TS1M2 Option Value Information For SAS Option CMPOPT Option Value:(NOEXTRAMATH
NOMISSCHECK NOPRECISE NOGUARDCHECK NOGENSYMNAMES NOFUNCDIFFERENCING) Option
Scope:SAS Session How option value set:Site Administrator Restricted
```

ログ 39.8 *制限可能なオプションをリストするログの一部*

```
13   proc options listrestrict; 14   run; SAS (r) Proprietary Software Release 9.4  TS1M2 Your Site
Administrator can restrict the ability to modify the following Portable Options:ANIMATION
Specifies whether to start or stop animation.ANIMDURATION       Specifies the number of seconds that
each animation frame displays.ANIMLOOP           Specifies the number of iterations that animated
images repeat.ANIMOVERLAY        Specifies that animation frames are overlaid in order to view all
frames.APPLETLOC          Specifies the location of Java applets, which is typically a
URL.ARMAGENT           Specifies an ARM agent (which is an executable module or keyword, such as
LOG4SAS) that contains a specific implementation of the ARM API.ARMLOC             Specifies the
location of the ARM log.ARMSUBSYS          Specifies the SAS ARM subsystems to enable or
disable.AUTOCORRECT        Automatically corrects misspelled procedure names and keywords, and global
statement names.AUTOSAVELOC        Specifies the location of the Program Editor auto-saved file.
```

保存可能オプションの表示

PROC OPTSAVE コマンドまたは DMOPTSAVE コマンドを使用すれば、多くのシステムオプションを保存できます。これらのオプションは、後で、PROC OPTSAVE コマンドまたは DMOPTSAVE コマンドを使用して復元できます。PROC OPTIONS の LISTOPTSAVE オプションを使用すれば、保存して後で復元できるシステムオプションを表示できます。

次の SAS log は、PROC OPTSAVE コマンドまたは DMOPTSAVE コマンドを使用して保存できるオプションのリストの一部を示しています。

ログ 39.9 *保存可能なシステムオプションのリストの一部*

```
11   proc options listoptsave; run; SAS (r) Proprietary Software Release 9.4  TS1M2 Core options that
can be saved with OPTSAVE ANIMATION          Specifies whether to start or stop
animation.ANIMDURATION       Specifies the number of seconds that each animation frame
displays.ANIMLOOP           Specifies the number of iterations that animated images
repeat.ANIMOVERLAY        Specifies that animation frames are overlaid in order to view all
frames.APPLETLOC          Specifies the location of Java applets, which is typically a
URL.AUTOCORRECT        Automatically corrects misspelled procedure names and keywords, and global
statement names.AUTOSAVELOC        Specifies the location of the Program Editor auto-saved
file.AUTOSIGNON         Enables a SAS/CONNECT client to automatically submit the SIGNON command
remotely with the RSUBMIT command.BINDING            Specifies the binding edge type of duplexed
printed output.BOMFILE            Writes the byte order mark (BOM) prefix when a Unicode-encoded file
is written to an external file.BOTTOMMARGIN       Specifies the size of the margin at the bottom of a
printed page.BUFNO              Specifies the number of buffers for processing SAS data
sets.BUFSIZE            Specifies the size of a buffer page for output SAS data sets.
```

結果:OPTIONS プロシジャ

SAS では、SAS ログにオプションリストが書き込まれます。出力形式 ***option*** | **NO*option*** の SAS システムオプションは、現在の設定に応じて、***option*** か **NO*option*** のどちらかとしてリストされます。これらは常に肯定形式で並べ替えられます。たとえば、NOCAPS は C にリストされます。

OPTIONS プロシジャは、SAS ログでパスワードを表示する際、実際の長さにかかわらず 8 個の X で表示します。

動作環境の情報

PROC OPTIONS は、SAS システムの実行環境に固有の詳細情報を作成します。その詳細と、ホスト固有のオプションの説明については、動作環境に関する SAS ドキュメントを参照してください。

関連項目:

- *UNIX 版 SAS*
- *Windows 版 SAS*
- *z/OS 版 SAS*

例: OPTIONS プロシジャ

例 1: 簡易形式のオプションリストを作成する

要素: PROC OPTIONS ステートメントオプション
SHORT

詳細

この例では、SAS システムオプション設定の簡易形式のリストの生成方法を示します。この簡易形式を、“システムオプションリストの表示” (1203 ページ)に示された長い形式と比較してください。

プログラム

```
proc options short;
run;
```

プログラムの説明

すべてのオプションとその設定をリストします。 SHORT は、SAS システムオプションとその設定を説明なしでリストします。

```
proc options short;
run;
```

ログ

ログ 39.10 *SHORT オプションのリストの一部*

```
6    proc options short; 7    run; SAS (r) Proprietary Software Release 9.4  TS1M2 Portable
Options:ANIMATION=STOP ANIMDURATION=MIN ANIMLOOP=YES ANIMOVERLAY APPEND= APPLETLOC=your-directory
ARMAGENT= ARMLOC=ARMLOG.LOG ARMSUBSYS=(ARM_NONE) AUTOCORRECT AUTOEXEC= AUTOSAVELOC= NOAUTOSIGNON
BINDING=DEFAULT BOMFILE BOTTOMMARGIN=0.000 IN BUFNO=1 BUFSIZE=0 BYERR BYLINE BYSORTED NOCAPS
NOCARDIMAGE CATCACHE=0 CBUFNO=0 CENTER CGOPTIMIZE=3 NOCHARCODE NOCHKPTCLEAN CLEANUP NOCMDMAC CMPLIB=
CMPMODEL=BOTH CMPOPT=(NOEXTRAMATH NOMISSCHECK NOPRECISE NOGUARDCHECK NOGENSYMNAMES NOFUNCDIFFERENCING)
NOCOLLATE COLOPHON= COLORPRINTING COMAMID=TCP COMPRESS=NO NOCONNECTEVENTS CONNECTMETACONNECTION
CONNECTOUTPUT=BUFFERED CONNECTPERSIST CONNECTREMOTE= CONNECTSTATUS CONNECTWAIT COPIES=1 CPUCOUNT=2
CPUID CSTGLOBALLIB= CSTSAMPLELIB= DATAPAGESIZE=CURRENT DATASTMTCHK=COREKEYWORDS DATE DATESTYLE=MDY
NODBFMTIGNORE NODBIDIRECTEXEC DBSLICEPARM=(THREADED_APPS, 2) DBSRVTP=NONE DCSHOST=LOCALHOST
DCSPORT=7111 DECIMALCONV=COMPATIBLE DEFLATION=6 NODETAILS DEVICE= DFLANG=ENGLISH DKRICOND=ERROR
DKROCOND=WARN NODLCREATEDIR DLDMGACTION=REPAIR NODMR DMS DMSEXP DMSLOGSIZE=99999 DMSOUTSIZE=2147483647
DMSPGMLINESIZE=136 NODMSSYNCHK DQLOCALE= DQOPTIONS=
```

例 2: 単一オプションの設定を表示する

要素: PROC OPTIONS ステートメントオプション
- OPTION=
- DEFINE
- LOGNUMBERFORMAT
- VALUE

詳細

この例では、単一 SAS システムオプションの設定の表示方法を示します。ログには、SAS システムオプション MEMBLKSZ の現在の設定が表示されています。DEFINE オプションと VALUE オプションは、詳細情報を表示します。LOGNUMBERFORMAT は、カンマを使用して値を表示します。

プログラム

```
proc options option=memblksz define value lognumberformat;
run;
```

プログラムの説明

MEMBLKSZ SAS システムオプションを指定します。 OPTION=MEMBLKSZ は、オプション値情報を表示します。DEFINE と VALUE は、詳細情報を表示します。LOGNUMBERFORMAT は、カンマを使用した出力形式を値に適用するように指定します。

```
proc options option=memblksz define value lognumberformat;
run;
```

ログ

ログ 39.11 *MEMBLKSZ オプションの指定によるログ出力*

```
13   proc options option=memblksz define value lognumberformat; 14   run; SAS (r) Proprietary Software
Release 9.4 TS1M2 Option Value Information For SAS Option MEMBLKSZ Value:16,777,216 Scope:Default How
option value set:Shipped Default Option Definition Information for SAS Option MEMBLKSZ Group= MEMORY
Group Description:Memory settings Description:Specifies the memory block size for Windows memory-based
libraries.Type:The option value is of type INTMAX Range of Values:The minimum is 0 and the maximum is
9223372036854775807 Valid Syntax(any casing):MIN|MAX|n|nK|nM|nG|nT|hexadecimal Numeric Format:Usage of
LOGNUMBERFORMAT impacts the value format When Can Set:Session startup (command line or config) only
Restricted:Your Site Administrator can restrict modification of this option Optsave:PROC Optsave or
command Dmoptsave will not save this option
```

例 3: 拡張パス環境変数の表示

要素: PROC OPTIONS ステートメントオプション
- OPTION=
- EXPAND
- NOEXPAND
- HOST

詳細

この例では、パスを表示する際の環境変数の値を示します。

プログラム

```
proc options option=msg expand;
run;
proc options option=msg noexpand;
run;
```

プログラムの説明

環境変数の値を表示します。 EXPAND オプションでは、環境変数の代わりに環境変数の値が表示されます。NOEXPAND オプションでは、環境変数が表示されます。この例では、環境変数は!sasroot です。

```
proc options option=msg expand;
run;
proc options option=msg noexpand;
run;
```

ログ

ログ 39.12 OPTIONS プロシジャによる展開パス名と非展開パス名の表示

```
6    proc options option=msg expand; 7    run; SAS (r) Proprietary Software
Release 9.4  TS1M2 MSG=( 'C:\Program Files\SASHome\SASFoundation\9.4\core
\sasmsg' ) The path to the sasmsg directory NOTE:PROCEDURE OPTIONS used (Total
process time): real time           0.01 seconds cpu time            0.00 seconds
8    proc options option=msg noexpand; 9    run; SAS (r) Proprietary Software
Release 9.4  TSM2 0 MSG=( '!sasroot\core\sasmsg') The path to the sasmsg
directory
```

例 4: INSERT オプションと APPEND オプションで指定可能なオプションのリスト

要素: PROC OPTIONS ステートメントオプション
LISTINSERTAPPEND

詳細

この例では、INSERT および APPEND システムオプションで指定可能なオプションの表示方法を示します。

プログラム

```
proc options listinsertappend;
run;
```

プログラムの説明

INSERT オプションと APPEND オプションで指定可能なオプションをすべてリストします。 LISTINSERTAPPEND オプションは、これらのオプションのリストと説明を提供します。

```
proc options listinsertappend;
run;
```

ログ

ログ 39.13 *INSERT オプションと APPEND オプションで指定可能なオプションの表示*

```
9    proc options listinsertappend; 10   run; SAS (r) Proprietary Software Release xxx   TS1M2 Core
options that can utilize INSERT and APPEND AUTOEXEC           Specifies the location of the SAS
AUTOEXEC files.CMPLIB             Specifies one or more SAS data sets that contain compiler
subroutines to include during compilation.FMTSEARCH          Specifies the order in which format
catalogs are searched.MAPS               Specifies the location of SAS/GRAPH map data
sets.SASAUTOS           Specifies the location of one or more autocall libraries.SASHELP
Specifies the location of the Sashelp library.SASSCRIPT          Specifies one or more locations of
SAS/CONNECT server sign-on script files.Host options that can utilize INSERT and APPEND
HELPLOC            Specifies the location of the text and index files for the facility that is used to
view the online SAS Help and Documentation.MSG                Specifies the path to the library that
contains SAS error messages.SET               Defines a SAS environment variable.
```

40 章

OPTLOAD プロシジャ

概要: OPTLOAD プロシジャ

OPTLOAD プロシジャは、SAS レジストリや SAS データセットに保存されている SAS システムオプション設定を読み込み、有効化します。

SAS データセットまたはレジストリキーから SAS システムオプション設定をロードするには、次のどちらかの方法を使用します。

- SAS ウィンドウ環境のコマンド行の DMOPTLOAD コマンド。たとえば、このコマンドは、レジストリキーからシステムオプションをロードします。DMOPTLOAD key= “core\options”
- PROC OPTLOAD ステートメント。

オプションがサイト管理者によって制限されており、なおかつ PROC OPTLOAD によって設定されるオプション値がサイト管理者の設定したオプション値と異なる場合、ログに警告メッセージが出力されます。

構文: OPTLOAD プロシジャ

PROC OPTLOAD <*options*>;

ステートメント	タスク	例
“PROC OPTLOAD ステートメント”	SAS レジストリや SAS データセットに保存されている SAS システムオプション設定を使用します。	Ex. 1

PROC OPTLOAD ステートメント

SAS レジストリや SAS データセットに保存されている SAS システムオプションの保存設定をロードします。

構文

PROC OPTLOAD <*options*>;

オプション引数の要約

DATA=*libref.dataset*
既存のデータセットから SAS システムオプション設定をロードします。

KEY="*SAS registry key*"
既存のレジストリキーから SAS システムオプション設定をロードします。

オプション引数

DATA=*libref.dataset*
SAS システムオプション設定のロード元のライブラリとデータセットの名前を指定します。SAS 変数 OPTNAME には SAS システムオプション名の文字値が含まれ、SAS 変数 OPTVALUE には SAS システムオプション設定の文字値が含まれます。

デフォルト DATA=オプションと KEY=オプションを省略すると、プロシジャはデフォルトの SAS ライブラリとデータセットを使用します。デフォルトライブラリは、現在のユーザープロファイルが存在する場所です。ライブラリを指定しなければ、デフォルトライブラリは SASUSER です。SASUSER が別のアクティブな SAS セッションで使用されている場合は、一時 WORK ライブラリが、データセットのロード元のデフォルトの場所になります。デフォルトのデータセット名は MYOPTS です。

要件 SAS ライブラリとデータセットが存在する必要があります。

KEY="*SAS registry key*"
保存された SAS システムオプション設定の SAS レジストリの場所を指定します。レジストリは SASUSER で保持されます。SASUSER が使用できない場合は、一時 WORK ライブラリが使用されます。たとえば、KEY="OPTIONS"は、OPTIONS レジストリキーからシステムオプションをロードします。

要件 "*SAS registry key*"は、既存の SAS レジストリキーにする必要があります。

"*SAS registry key*"名は引用符で囲んでください。複数のキー名を並べる場合は、バックスラッシュ(\)で名前を区切ります。たとえば、KEY="CORE\OPTIONS"は、CORE\OPTIONS レジストリキーからシステムオプションをロードします。

例: 保存済みシステムオプションのデータセットのロード

要素: PROC OPTLOAD ステートメントのオプション
DATA=

詳細

この例では、OPTSAVE プロシジャを使用して現在のシステムオプション設定を保存し、YEARCUTOFF システムオプションを変更して、元のシステムオプションセットをロードします。

プログラム

```
libname mysas "c:\mysas";

proc options option=yearcutoff;
run;

proc optsave out=mysas.options;
run;

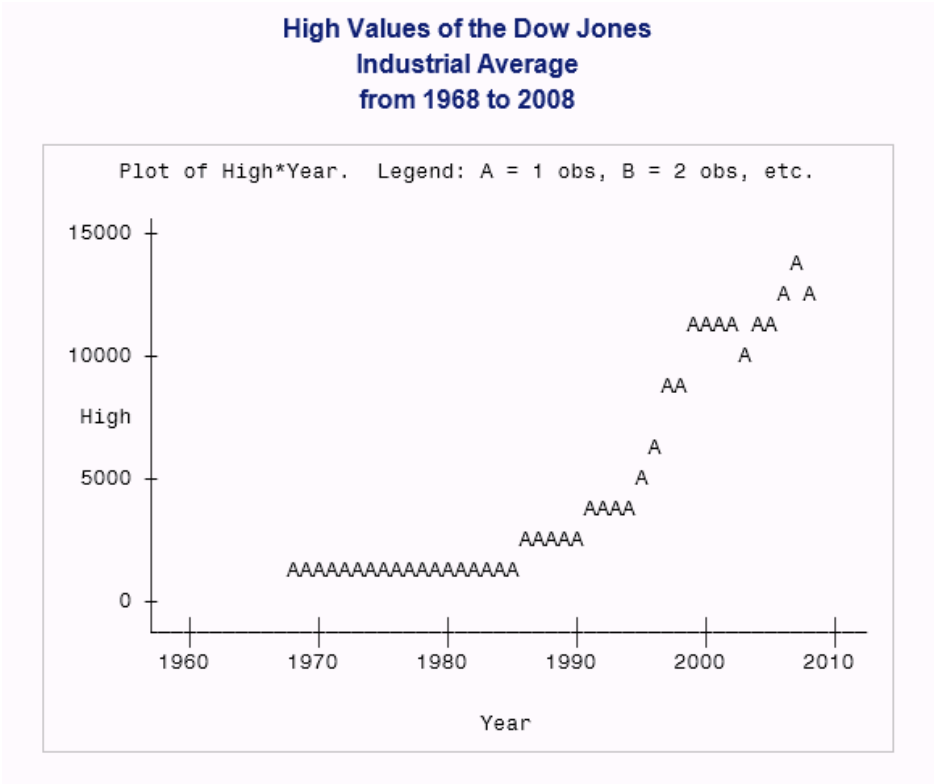
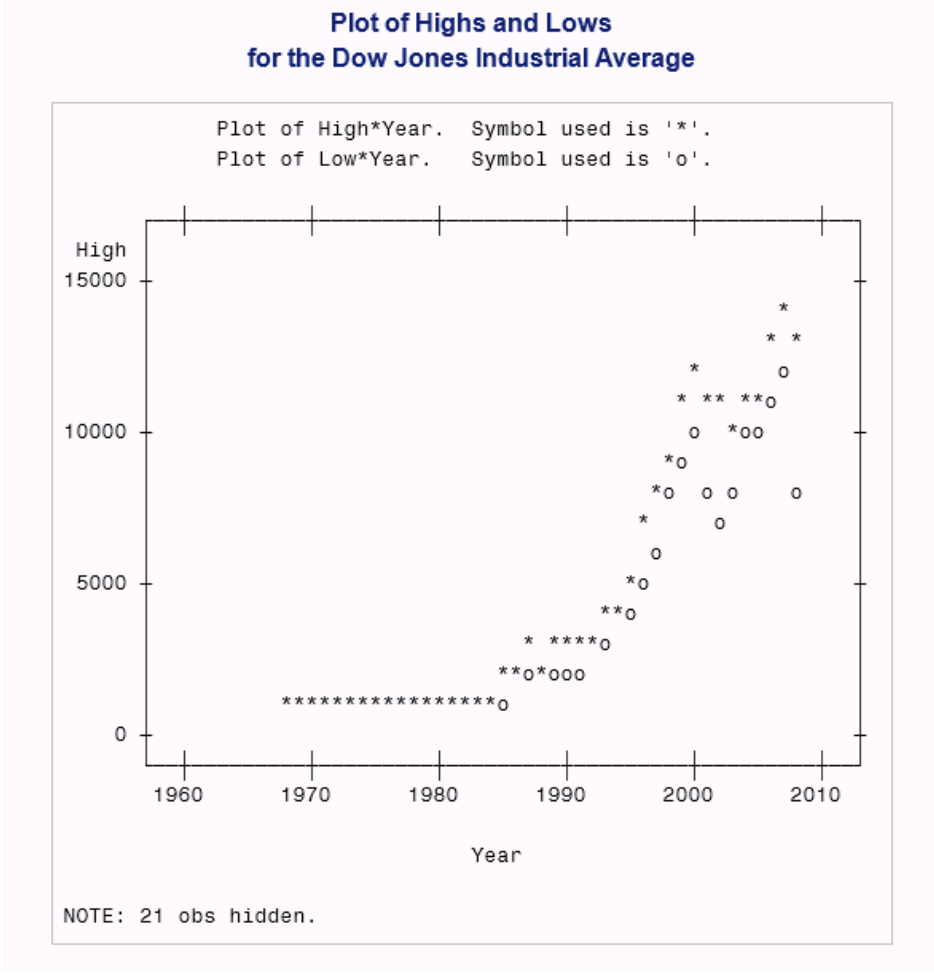
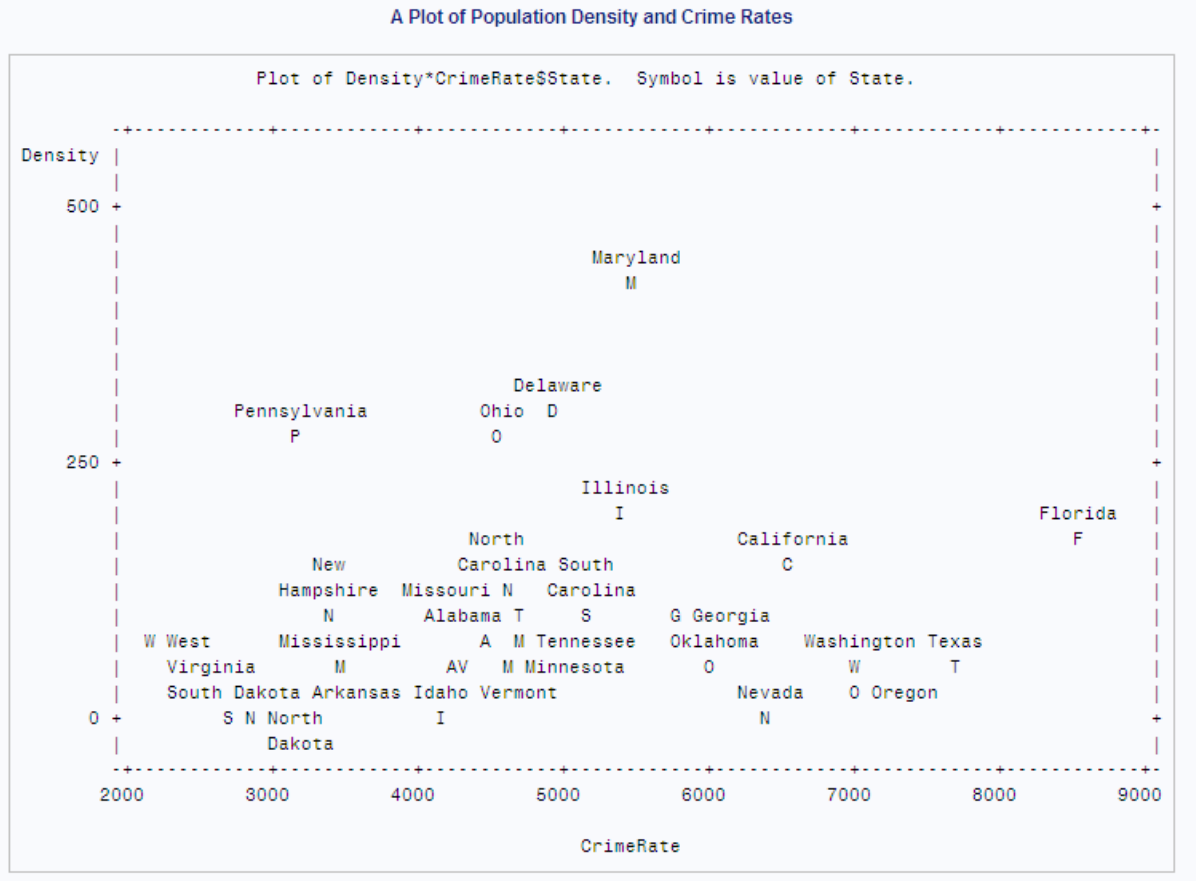
options yearcutoff=2000;

proc options option=yearcutoff;
run;

proc optload data=mysas.options;
run;


proc options option=yearcutoff;
run;
```

プログラムの説明

YEARCUTOFF オプションの表示を可能にするには、これらのステートメントとプロシジャを SAS プログラムとして実行するのではなく 1 つずつサブミットします。

ライブラリ参照名を割り当てます。

```
libname mysas "c:\mysas";
```

YEARCUTOFF=システムオプションの値を表示します。

```
proc options option=yearcutoff;
run;
```

現在のシステムオプション設定を mysas.options に保存します。

```
proc optsave out=mysas.options;
run;
```

OPTIONS ステートメントを使用して、YEARCUTOFF=システムオプションを値 2000 に設定します。

```
options yearcutoff=2000;
```

YEARCUTOFF=システムオプションの値を表示します。

```
proc options option=yearcutoff;
run;
```

保存したシステムオプション設定をロードします。

```
proc optload data=mysas.options;
run;
```

YEARCUTOFF=システムオプションの値を表示します。保存したシステムオプション設定をロードすると、YEARCUTOFF=オプションの値が元の値に戻ります。

```
proc options option=yearcutoff;
run;
```

ログ

ログ 40.1 *PROC OPTLOAD を使用してオプションをロードした後、SAS ログに YEARCUTOFF= 値を表示します。*

```
1    libname mysas "c:\mysas"; NOTE:Libref MYSAS was successfully assigned as
follows:Engine:        V9 Physical Name: c:\mysas 2    proc options
option=yearcutoff; 3    run; SAS (r) Proprietary Software Release 9.4  TS1M2
YEARCUTOFF=1926   Specifies the first year of a 100-year span that is used by
date informats and functions to read a two-digit year.NOTE:PROCEDURE OPTIONS
used (Total process time): real time           0.00 seconds cpu time
0.00 seconds 4    proc optsave out=mysas.options; 5    run; NOTE:The data set
MYSAS.OPTIONS has 259 observations and 2 variables.NOTE:PROCEDURE OPTSAVE used
(Total process time): real time           0.03 seconds cpu time            0.03
seconds 6    options yearcutoff=2000; 7    proc options option=yearcutoff; 8
run; SAS (r) Proprietary Software Release 9.4  TS1M2 YEARCUTOFF=2000   Specifies
the first year of a 100-year span that is used by date informats and functions
to read a two-digit year.NOTE:PROCEDURE OPTIONS used (Total process time): real
time           0.00 seconds cpu time            0.00 seconds 9    proc optload
data=mysas.options; 10   run; NOTE:PROCEDURE OPTLOAD used (Total process time):
real time           0.06 seconds cpu time            0.01 seconds
```

```
11   proc options option=yearcutoff; 12   run; SAS (r) Proprietary Software
Release 9.4  TS1M2 YEARCUTOFF=1926   Specifies the first year of a 100-year span
that is used by date informats and functions to read a two-digit
year.NOTE:PROCEDURE OPTIONS used (Total process time): real time           0.00
seconds cpu time            0.00 seconds
```

41 章

OPTSAVE プロシジャ

概要: OPTSAVE プロシジャ

PROC OPTSAVE は、SAS レジストリや SAS データセットに現在の SAS システムオプション設定を保存します。

SAS システムオプションは、SAS セッションをまたいで保存することはできません。SAS データセットやレジストリキーに SAS システムオプションの設定を保存するには、次のどちらかの方法を使用します。

- SAS ウィンドウ環境のコマンド行の DMOPTSAVE コマンド。次のようなコマンドを使用します。DMOPTSAVE <*save-location*>.
- PROC OPTSAVE ステートメント。

構文: OPTSAVE プロシジャ

PROC OPTSAVE <*options* >;

ステートメント	タスク	例
“PROC OPTSAVE ステートメント”	現在の SAS システムオプション設定を SAS レジストリや SAS データセットに保存します。	Ex. 1

PROC OPTSAVE ステートメント

現在の SAS システムオプション設定を SAS レジストリや SAS データセットに保存します。

構文

PROC OPTSAVE <*options* >;

オプション引数の要約

KEY="*SAS registry key*"
SAS システムオプション設定をレジストリキーに保存します。

OUT=*libref.dataset*
SAS システムオプション設定を SAS データセットに保存します。

オプション引数

KEY="*SAS registry key*"
保存された SAS システムオプション設定の SAS レジストリの場所を指定します。レジストリは SASUSER で保持されます。SASUSER が使用できない場合は、一時 WORK ライブラリが使用されます。たとえば、KEY="OPTIONS"により、システムオプションが OPTIONS レジストリキーに保存されます。

制限事項 “複数行にわたる“*SAS registry key*”名は使用できません。

要件 複数のキー名を並べる場合は、バックスラッシュ(\)で名前を区切ります。個々のキー名には、バックスラッシュ以外の任意の文字を含められます。

キー名の長さは 255 文字(バックスラッシュを含む)を超えないようにしてください。

“*SAS registry key*”名は引用符で囲んでください。

ヒント サブキーを指定するには、ルートキーで始まる複数のキー名を入力します。

注意 キーがすでに存在する場合は、上書きされます。指定したキーが現在の SAS レジストリに存在していない場合は、オプション設定が SAS レジストリに保存されるときに、自動的にキーが作成されます。

OUT=*libref.dataset*
SAS システムオプション設定の保存先のライブラリとデータセットの名前を指定します。SAS 変数 OPTNAME には、SAS システムオプション名の文字値が含まれます。SAS 変数 OPTVALUE には、SAS システムオプション設定の文字値が含まれます。

デフォルト OUT=オプションと KEY=オプションを省略すると、プロシジャはデフォルトの SAS ライブラリとデータセットを使用します。デフォルト SAS ライブラリは、現在のユーザープロファイルが存在する場所です。SAS ライブラリを指定しなければ、デフォルトライブラリは SASUSER です。SASUSER が別のアクティブな SAS セッションで使用されている場合は、一時 WORK ライブラリが、デ

ータセットを保存するデフォルトの場所になります。デフォルトのデータセット名は MYOPTS です。

注意 データセットがすでに存在する場合は、上書きされます。

単一オプションが保存可能かを指定する

オプションが保存可能かどうかを指定するには、OPTIONS プロシジャで DEFINE を指定します。ログの出力では、**Optsave:**で始まる行は、オプションを保存できることを示しています。

```
proc options option=pageno define;
run;
```

ログ 41.1 *オプションプロシジャ DEFINE オプションの出力を表示している SAS ログ*

```
8    proc options option=pageno define; 9    run; SAS (r) Proprietary Software
Release 9.4  TS1M0 PAGENO=1 Option Definition Information for SAS Option PAGENO
Group= LISTCONTROL Group Description:Procedure output and display settings
Description:Resets the SAS output page number.Type:The option value is of type
LONG Range of Values:The minimum is 1 and the maximum is 2147483647 Valid
Syntax(any casing):MIN|MAX|n|nK|nM|nG|nT|hexadecimal Numeric Format:Usage of
LOGNUMBERFORMAT impacts the value format When Can Set:Startup or anytime during
the SAS Session Restricted:Your Site Administrator can restrict modification of
this option Optsave:PROC Optsave or command Dmoptsave will save this option
```

保存可能なオプションのリストの作成

システムオプションには、保存できないものもあります。保存可能なオプションのリストを作成するには、SAS コードをサブミットします。

```
proc options listoptsave;
run;
```

保存可能なオプションのリストの一部をここに示します。

ログ 41.2 保存可能なオプションのリストの一部

```
51   proc options listoptsave; 52   run; SAS (r) Proprietary Software Release
9.4  TS1M2 Core options that can be saved with OPTSAVE ANIMATION
Specifies whether to start or stop animation.ANIMDURATION       Specifies the
number of seconds that each animation frame displays.ANIMLOOP
Specifies the number of iterations that animated images
repeat.ANIMOVERLAY        Specifies that animation frames are overlaid in order
to view all frames.APPLETLOC          Specifies the location of Java applets,
which is typically a URL.AUTOCORRECT        Automatically corrects misspelled
procedure names and keywords, and global statement names.AUTOSAVELOC
Specifies the location of the Program Editor auto-saved file.AUTOSIGNON
Enables a SAS/CONNECT client to automatically submit the SIGNON command remotely
with the RSUBMIT command.BINDING            Specifies the binding edge type of
duplexed printed output.BOMFILE            Writes the byte order mark (BOM)
prefix when a Unicode-encoded file is written to an external
file.BOTTOMMARGIN       Specifies the size of the margin at the bottom of a
printed page.BUFNO              Specifies the number of buffers for processing
SAS data sets.BUFSIZE            Specifies the size of a buffer page for output
SAS data sets.BYERR              SAS issues an error message and stops
processing if the SORT procedure attempts to sort a _NULL_ data
set.BYLINE             Prints the BY line above each BY group.BYSORTED
Requires observations in one or more data sets to be sorted in alphabetic or
numeric order.CAPS               Converts certain types of input, and all data
lines, into uppercase characters.CARDIMAGE          Processes SAS source code
and data lines as 80-byte records.CBUFNO            Specifies the number of
extra page buffers to allocate for each open SAS catalog.CENTER
Center SAS procedure output.
```

例: データセットのシステムオプションの保存

要素: PROC OPTSAVE ステートメントのオプション
OUT=

詳細

この例では、OPTSAVE プロシジャを使用して現在のシステムオプション設定を保存します。

プログラム

```
libname mysas "c:\mysas";


proc optsave out=mysas.options;
run;
```

プログラムの説明

ライブラリ参照名を作成します。

```
libname mysas "c:\mysas";
```

現在のシステムオプション設定を保存します。

```
proc optsave out=mysas.options;
run;
```

ログ

ログ 41.3 *SAS ログは PROC OPTSAVE の処理を示します。*

```
1    libname mysas "c:\mysas"; NOTE:Libref MYSAS was successfully assigned as
follows:Engine:        V9 Physical Name: c:\mysas 2    proc optsave
out=mysas.options; 3    run; NOTE:The data set MYSAS.OPTIONS has 289
observations and 2 variables.NOTE:PROCEDURE OPTSAVE used (Total process time):
real time           0.03 seconds cpu time            0.03 seconds
```

42 章

PLOT プロシジャ

概要: PLOT プロシジャ

PLOT プロシジャは、入力 SAS データセットの各オブザベーションの 2 つの変数の値をプロットします。プロットの各ポイントの座標は、入力データセットの 1 つ以上のオブザベーションの 2 つの変数値に対応します。

次の出力は、1968 年から 2008 年までのダウジョーンズ工業平均の高値を示す単純プロットです。PROC PLOT は、プロット記号と軸のスケールを決定します。出力を生成するステートメントは、次のとおりです。

```
options nodate pageno=1 linesize=64
   pagesize=25;

proc plot data=djia;
   plot high*year;
   title 'High Values of the Dow Jones';
   title2 'Industrial Average';
   title3 'from 1968 to 2008';
run;
```

アウトプット 42.1 *単純プロット*



また、次の出力に示すように、2 つのプロットを重ね合わせることもできます。一方のプロットが Djia の高値を示し、他方のプロットが安値を示します。また、このプロットは、プロット記号を指定し、プロットを枠で囲めることも示しています。次の出力を生成するステートメントは、“例 3: 2 つのプロットを重ね合わせる” (1255 ページ)に示されています。

アウトプット 42.2 2 つの値セットを一度にプロットする



また、PROC PLOT では、次に示すように、プロット上のポイントに変数の値を示すラベルを付けることもできます。プロットされたデータは、選択した米国州の人口密度と犯罪率を表します。次の出力を生成する SAS コードは、“例 11: PLACEMENT=オプションを使用し、プロット上のラベルを調整する” (1278 ページ)に示されています。

アウトプット 42.3 *プロット上のポイントにラベルを付ける*



概念: PLOT プロシジャ

RUN グループ

PROC PLOT は対話型プロシジャです。これは RUN ステートメントの実行後もアクティブなままです。通常、SAS では、RUN ステートメントの実行後にプロシジャは終了されます。PLOT プロシジャを開始すると、PROC PLOT ステートメントを再サブミットしなくても、任意の有効なステートメントを続けてサブミットできます。したがって、ラベルや目盛の値などの変更を簡単に試せます。PROC PLOT ステートメントは、別の PROC PLOT ステートメントをサブミットするまでアクティブです。

RUN ステートメントをサブミットすると、PROC PLOT は、前回の PROC PLOT または RUN ステートメント以後にサブミットされたステートメントをすべて実行します。各ステートメントグループは *RUN グループ*と呼ばれます。PROC PLOT は、RUN グループごとに新しいページを始め、存在する場合は VPERCENT=および HPERCENT=リストの最初の項目で始めます。

プロシジャを終了するには、QUIT ステートメント、DATA ステートメントまたは PROC ステートメントをサブミットします。RUN ステートメント同様に、それぞれのステートメントが RUN グループを完了します。RUN グループのステートメントを実行しない場合は、RUN CANCEL ステートメントを使用すると、プロシジャをすぐに終了できます。

BY ステートメントは対話形式で使用できます。BY ステートメントは、別の BY ステートメントを実行するか、プロシジャを終了するまで有効です。

PROC PLOT での RUN グループ処理の使用例については、“例 11: PLACEMENT= オプションを使用し、プロット上のラベルを調整する” (1278 ページ)を参照してください。

ラベルとプロットポイント

ポインタ記号

ラベル変数の使用時に、プロット記号を指定していないか、またはプロット記号として使用する変数の値が NULL('00'x)の場合、PROC PLOT はポインタ記号をプロット記号として使用します。ポインタ記号でラベル位置の大まかな方向を指し示すことによって、ポイントとラベルを関連付けます。PROC PLOT は、PLACEMENT=の S=および V=サブオプションの値に基づいて、4 つの異なるポインタ記号を使い分けます。次の表に、ポインタ記号を示します。

表 42.1 *ポインタ記号*

S=	V=	記号
LEFT	任意	<
RIGHT	任意	>
CENTER	>0	◦
CENTER	<=0	v

ポインタ記号の使用時に複数のポイントが同じ場所にある場合、PROC PLOT は、ポイントの数が 2 から 9 までであれば、そのポイントの数をプロット記号として使用します。ポイントの数が 9 を超える場合、プロシジャはアスタリスク(*)を使用します。

注: 動作環境ごとに文字セットが異なるため、S=CENTER かつ V>0 の場合のポインタ記号は、前述と異なる場合があります。

ペナルティについて

PROC PLOT は、ペナルティを使用して配置の質を評価します。すべてのラベルがゼロペナルティでプロットされる場合は、ラベルがまったく衝突せず、すべてのラベルが記号の近くに表示されます。すべてのラベルをゼロペナルティで配置することが不可能な場合、PROC PLOT は、合計ペナルティを最小限に抑えようとします。

次の表に、ペナルティの説明、ペナルティのデフォルト値、ペナルティを参照するためのインデックス、ペナルティを変更する場合に指定可能な値範囲を示します。各ペナルティの詳細については、表 42.3 (1228 ページ)で説明されています。

表 42.2 *ペナルティ表*

ペナルティ	デフォルトペナルティ	インデックス	範囲
ブランクなし	1	1	0-500

ペナルティ	デフォルトペナルティ	インデックス	範囲
不適切な分割(区切り文字の指定なし)	1	2	0-500
不適切な分割(区切り文字あり)	50	3	0-500
フリー水平シフト(*fhs*)	2	4	0-500
フリー垂直シフト(*fvs*)	1	5	0-500
垂直シフトの重み(*vsw*)	2	6	0-500
垂直/水平シフトの分母(*vhsd*)	5	7	1-500
衝突状態	500	8	0-10,000
(将来の使用のために予約)		9-14	
最初の文字なし	11	15	0-500
2 番目の文字なし	10	16	0-500
3 番目の文字なし	8	17	0-500
4 番目の文字なし	5	18	0-500
5 番目から 200 番目の文字なし	2	19–214	0-500

次の表に、前述の表のインデックス値、および対応するペナルティの説明を示します。

表 42.3 *ペナルティのインデックス値*

1	プロットのブランク以外の文字とラベルの埋め込みブランクが衝突しています。または、各ラベルフラグメントの前後にブランクまたはプロット境界がありません。
2	区切り文字を指定していないのに、ブランク以外の文字や句読点でない文字で分割が発生しています。
3	区切り文字を指定しているときに、L=サブオプションの指定とは異なる行数でラベルが配置されています。
4-7	対応するポイントから離れた位置にラベルが配置されています。PROC PLOT は、次の(整数演算)式に従ってペナルティを計算します。$[\mathrm{MAX}(\lvert H\rvert - fhs, 0) + vsw \times \mathrm{MAX}(\lvert V\rvert - (L + fvs + (V > 0))/2, 0)]/vhsd$ ペナルティ 4 から 7 がそのまま、ペナルティを決定するための公式の構成要素となっていることに注意してください。フリー水平シフトやフリー垂直シフトのペナルティを 500 などの大きな値に変更すると、大きな水平シフトや垂直シフトに対するペナルティが削除されるという効果があります。“例 6: 軸に日付値をプロットする” (1262 ページ)で、水平シフトペナルティの削除が有益な場合について説明しています。
8	ラベルとプロット記号が衝突する場合があります。プロット記号がブランクの場合は、衝突状態にはなりません。詳細については、“衝突状態” (1229 ページ)を参照してください。
15–214	ラベル文字がプロットに表示されません。デフォルトでは、最初の文字が印刷されないことに対するペナルティの方が、2 番目の文字が印刷されないことに対するペナルティよりも大きくなります。以降の文字についても同様です。ただし、デフォルトでは、5 番目以降の文字が印刷されないことに対するペナルティはすべて同じになります。

注: ラベルでは、ペナルティなしで文字を共有できます。

ペナルティの変更

PLOT ステートメントで PENALTIES=オプションを使用してデフォルトペナルティを変更できます。PROC PLOT はラベルの配置時にペナルティを考慮するので、デフォルトペナルティを変更すると、ラベルの配置も変更される可能性があります。

たとえば、ラベルがすべて 2 字の同一接頭辞から始まっている場合は、3、4、5 番目の文字が印刷されないようにするにはデフォルトペナルティを増加させ、1、2 番目の文字が印刷されないようにするにはデフォルトペナルティを減少させることをお勧めします。PENALTIES=オプションの使用方法の例は、"PENALTIES=オプションの使用" (1249 ページ) を参照してください。

衝突状態

衝突状態とは、ラベルとそのプロット記号が衝突する可能性のある配置状態です。衝突状態には 500 という大きなデフォルトペナルティが関連付けられているため、通常、PROC PLOT は衝突状態の使用を避けます。PROC PLOT は、配置状態が衝突状態かどうかを判断する際、特定ラベルの実際の長さや分割は考慮しません。PROC PLOT が衝突状態を判断するために使用するルールは次のとおりです。

- S=CENTER の配置状態では、ラベルの上下のシフトが不十分なせいで、すべてのラベルが記号とまったく異なる行にシフトされると、衝突状態になります。
- S=RIGHT の配置状態では、ラベルを、先に記号とまったく異なる上下の行にシフトせずに、ゼロ以上左側の位置にシフトすると、衝突状態になります。
- S=LEFT の配置状態では、ラベルを、先に記号とまったく異なる上下の行にシフトせずに、ゼロ以上右側の位置にシフトすると、衝突状態になります。

注: プロット記号を使用しない場合は、衝突状態にはなりません。

参照線

PROC PLOT は、プロットに参照線を引く前に、ラベルを配置し、ペナルティを計算します。プロシジャは、参照線を含む行や列を回避しようとはしません。

非表示のラベル文字

非表示のオブザベーションと非表示のプロット記号の数に加えて、PROC PLOT は、非表示のラベル文字数を印刷します。ラベル文字は、プロット記号や他のラベル文字によって非表示になる場合があります。

重ね合わせたラベルプロット

ラベルプロットと非ラベルプロットを重ね合わせると、PROC PLOT は、ラベルと非ラベルプロットの文字との衝突を回避しようとします。ラベル文字と非ラベルプロットの文字が衝突した場合、PROC PLOT は、ペナルティ合計に通常のペナルティを加算します。

2 つ以上のラベルプロットを重ね合わせた場合、衝突の回避や非表示文字カウントの計算では、すべてのラベルプロットが単一プロットとして扱われます。OVP システムオプションが有効でも、異なるプロットのラベルが重ね打ちされることは決してありません。

ラベルプロットに使用されるコンピュータリソース

このセクションでは、次の変数を使用して、PROC PLOT がラベルプロットの作成に時間とメモリをどれくらい使用するのかを説明します。

n
: ラベルの付いているポイントの数

len
ラベルの長さ

s
ラベルの一部(フラグメント)の数

p
PLACE=オプションで指定した配置状態の数

時間

指定したプロットサイズでプロットを作成するために要する時間は、$n \times len$におおよそ比例します。ラベルの分割に要する時間は、ns^2におおよそ比例します。一般的に、配置状態の指定が増えるほど、PROC PLOT がラベルの配置に必要とする時間も長くなります。ただし、水平シフトと垂直シフトの数が増えると、PROC PLOT はより柔軟に衝突を回避できるようになり、多くの場合、結果として、ラベルの配置に使用される時間も短縮されます。

メモリ

PROC PLOT は、内部配置状態リストのために $24p$ バイトのメモリを使用します。PROC PLOT は、内部ラベルリストに$n(84 + 5len + 4s(1 + 1.5(s + 1)))$バイトを使用します。PROC PLOT は、すべてのプロットをメモリ上で作成し、印刷位置ごとに 1 バイトずつメモリを使用します。

メモリ不足の場合は、各 PLOT ステートメントで要求するプロットの数を減らして、各 PLOT ステートメントの後に RUN ステートメントを挿入します。

構文: PLOT プロシジャ

要件 最低 1 つの PLOT ステートメントが必要です。

ヒント: PROC PLOT は RUN グループ処理をサポートしています。

ATTRIB、FORMAT、LABEL および WHERE ステートメントが使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。グローバルステートメントを使用することもできます。リストは、“グローバルステートメント” (24 ページ)および“Global Statements” (*SAS Statements: Reference*)を参照してください。

PROC PLOT <*option(s)*>;
BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2*…> <NOTSORTED>;
PLOT *plot-request(s)* </ *option(s)*>;

ステートメント	タスク	例
“PROC PLOT ステートメント”	プロットの作成を要求します	Ex. 10
“BY ステートメント”	BY グループごとに別々のプロットを作成します	Ex. 8
“PLOT ステートメント”	求めるプロットについて記述します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8,

ステートメント	タスク	例
		Ex. 9, Ex. 11, Ex. 12, Ex. 13

PROC PLOT ステートメント

プロットの作成を要求します。

ヒント: データセットオプションを DATA=オプションと使用できます。リストについては、“データセットオプション” (23 ページ)を参照してください。

例: “例 10: 欠損値を含むオブザベーションを除外する” (1275 ページ)

構文

PROC PLOT <*option(s)*>;

オプション引数の要約

DATA=*SAS-data-set*
: 入力データセットを指定します。

ENCRYPTKEY=*key-value*
: AES で暗号化されたデータセットをプロットする場合に必要となるキー値を指定します。

Control the appearance of the plot

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
: プロットの境界を構成する文字を指定します。

NOLEGEND
: プロット上部の凡例を非表示にします。

VTOH=*aspect-ratio*
: 出力デバイスでの文字の縦横比を印刷します。

Control the axes

MISSING
: 欠損文字変数値が含まれます。

NOMISS
: 欠損値を含むオブザベーションを除外します。

UNIFORM
: 複数の BY グループにわたって軸のスケールを均等に揃えます。

Control the size of the plot

HPERCENT=*percent(s)*
: 各プロットで使用可能な横のスペースの割合を指定します。

VPERCENT=*percent(s)*
: 各プロットで使用可能な縦のスペースの割合を指定します。

オプション引数

DATA=*SAS-data-set*
入力 SAS データセットを指定します。

参照項目 2 章, “Base SAS プロシジャの使用に必要な基本概念” (21 ページ)

ENCRYPTKEY=*key-value*
AES で暗号化されたデータセットをプロットする場合に必要となるキー値を指定します。入力データセットが ENCRYPT=AES で作成された場合は、ENCRYPTKEY=値を指定して対象データをプロットする必要があります。たとえば、DATA ステートメントを使用して secretPlot という名前のデータセットが作成される場合は、

```
data secretPlot(encrypt=AES encryptkey=Ib007)
```

次の PROC ステートメントを指定して secretPlot 内の対象データをプロットしてください。

```
proc plot data=secretPlot(encryptkey=Ib007);
```

参照項目 “ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*) では、ENCRYPTKEY=データセットオプションの詳細について説明しています。

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
は、プロットの境界を構成するために使用する文字を定義します。

position(s)
SAS フォーマット文字列における 1 つ以上の文字の位置を識別します。スペースまたはカンマで位置を区切ります。

デフォルト *position(s)*)の省略は、20 のすべての可能な SAS フォーマット文字を順番に指定することと同じです。

範囲 PROC PLOT は、フォーマット文字 1、2、3、5、7、9、11 を使用します。次の表に、PROC PLOT が使用するフォーマット文字を示します。

表 42.4 *文字位置*

位置	デフォルト	表示に使用
1	|	垂直区切り線
2	-	水平区切り線
3 5 9 1 1	-	角
7	+	垂直区切り線と水平区切り線の交点

formatting-character(s)
指定位置に使用する文字をリストします。PROC PLOT は、*formatting-character(s)*の文字を表示されている順序で *position(s)*に割り当てます。たとえば、次のオプションではアスタリスク(*)を 3 番目のフォーマット文字に、数字記号(#)を 7 番目の文字に割り当てます。その他の文字は変更されません。

```
formchar(3,7)='*#'
```

操作 SAS システムオプション FORMCHAR=では、デフォルトのフォーマット文字を指定します。システムオプションは、フォーマット文字の全体の文字列を定義します。プロシジャの FORMCHAR=オプションでは、選択した文字を再定義できます。

ヒント 16 進数文字を含む *formatting-characters* の文字を使用できます。16 進数文字を使用する場合、**x** を終了引用符の後に付ける必要があります。たとえば、次のオプションでは、16 進文字 2-D が 3 番目のフォーマット文字に、16 進文字 7C が 7 番目の文字にそれぞれ割り当てられます。その他の文字は変わりません。**formchar(3,7)='2D7C'x**

*formatting-character(s)*に対しすべてブランクを指定すると、境界のないプロットが作成されます。たとえば、次のコードではすべてのブランクが指定されます。**formchar (1,2,7)=''**.

HPERCENT=*percent(s)*
各プロットで使用するための使用可能な横のスペースの割合を 1 つ以上指定します。HPERCENT=では、複数のプロットを 1 ページにまとめられます。PROC PLOT は、可能な限り多くのプロットを 1 ページに収めようとします。各 *percent(s)*の使用後に、PROC PLOT はリストの最初に戻ります。リストにゼロがあると、次のプロットを同じページに収めることが可能であっても、PROC PLOT は新しいページに移動します。

HPERCENT=33
プロットをページごとに 3 つずつ横に並べて印刷します。各プロットの幅は 1 ページの 3 分の 1 です。

HPERCENT=50 25 25
プロットをページごとに 3 つずつ印刷します。最初のプロットは他の 2 つのプロットの 2 倍の幅になります。

HPERCENT=33 0
1 ページの 3 分の 1 の幅のプロットを作成します。各プロットは別々のページに印刷されます。

HPERCENT=300
3 ページ分の幅のプロットを作成します。

各 BY グループの初めと各 RUN ステートメントの後で、PROC PLOT は *percent(s)* の初めに戻って新しいページの印刷を開始します。

別名 HPCT=

デフォルト 100

例 “例 4: ページごとに複数のプロットを作成する” (1258 ページ)

MISSING
軸の構成に欠損文字変数値が含まれます。これは数値変数では無効です。

操作 文字変数に対する NOMISS オプションより優先されます。

NOLEGEND
各プロット上部の凡例を非表示にします。凡例には、プロットされている変数名と、プロットで使用されるプロット記号がリストされます。

NOMISS
どちらかの変数が欠損しているオブザベーションは軸の計算から除外します。通常、PROC PLOT は、他の変数が欠損しているポイントも含めて、プロットされているすべての変数値に基づいて軸を描きます。

操作 HAXIS=オプションは、横軸で NOMISS を無効化します。VAXIS=オプションは縦軸で無効化します。

文字変数に対する MISSING が NOMISS より優先されます。

例 “例 10: 欠損値を含むオブザベーションを除外する” (1275 ページ)

UNIFORM
複数の BY グループにわたって軸のスケールを均等に揃えます。スケールを均等に揃えると、異なる BY 変数値のプロットを直接比較できます。

制限事項 別のユーザーが同時にデータセットを更新している場合、同時アクセスをサポートするエンジンでは PROC PLOT で UNIFORM オプションを使用できません。

VPERCENT=*percent(s)*
各プロットで使用するための使用可能な縦のスペースの割合を 1 つ以上指定します。100 よりも大きい割合を使用すると、PROC PLOT は連続するページにプロットのセクションを印刷します。

別名 VPCT=

デフォルト 100

参照項目 “HPERCENT=*percent(s)* ” (1233 ページ)

例 “例 4: ページごとに複数のプロットを作成する” (1258 ページ)

VTOH=*aspect-ratio*
出力デバイスでの文字の縦横比を指定します。*aspect-ratio* は正の実数です。VTOH=オプションを使用すると、PROC PLOT が目盛間にスペースを入れて、縦と横の目盛の間隔をほぼ等しくします。たとえば、文字の高さを幅の 2 倍にする場合は、VTOH=2 を指定します。

操作 PLOT ステートメントで HSPACE=および VSPACE=オプションを使用した場合、VTOH=は無効になります。

注 使用できる最小値は 0 です。

参照項目 “HAXIS=*axis-specification* ” (1238 ページ) では所定の距離が両軸の同一データ範囲を示すように軸を等しくする方法方法について、説明しています。

BY ステートメント

BY グループごとに、別々のプロットを作成し、新しいページを開始します。

参照項目: “BY” (68 ページ)

例: “例 8: BY グループのプロット” (1268 ページ)

構文

BY <DESCENDING> *variable-1* <DESCENDING> *variable-2* …<NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは指定するすべての変数によって並べ替えるか、適切にインデックスを付ける必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
オブザベーションが BY ステートメントの文字 DESCENDING の直後に続く変数で降順に並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。データは、時系列などの別の方法でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

PLOT ステートメント

PROC PLOT によるプロットの作成を要求します。

ヒント: 複数の PLOT ステートメントを使用できます。

例:
“例 1: プロット記号の指定” (1251 ページ)
“例 2: X 軸の制御と参照線の追加” (1253 ページ)
“例 3: 2 つのプロットを重ね合わせる” (1255 ページ)
“例 4: ページごとに複数のプロットを作成する” (1258 ページ)
“例 5: 対数尺度にデータをプロットする” (1261 ページ)
“例 6: 軸に日付値をプロットする” (1262 ページ)
“例 7: 等高線グラフの作成” (1265 ページ)
“例 8: BY グループのプロット” (1268 ページ)
“例 9: プロットにラベルを追加する” (1272 ページ)
“例 11: PLACEMENT=オプションを使用し、プロット上のラベルを調整する” (1278 ページ)
“例 12: マクロを使用し、プロット上のラベルを調整する” (1283 ページ)
“例 13: デフォルトのペナルティを変更する” (1286 ページ)

構文

PLOT *plot-request(s)* </ *option(s)*>;

オプション引数の要約

BOX
プロットの周りを枠で囲みます。

OVERLAY
プロットを重ね合わせます。

Control the axes

HAXIS=*axis-specification*
横軸の目盛値を指定します。

HEXPAND
横軸を拡張します。

HPOS=*axis-length*
横軸の印刷位置の数を指定します。

HREVERSE
横軸上の値の順序を逆にします。

HSPACE=*n*
横軸上の目盛間隔を指定します。

HZERO
横軸の最初の目盛にゼロの値を割り当てます。

VAXIS=*axis-specification*
縦軸の目盛値を指定します。

VEXPAND
縦軸を拡張します。

VPOS=*axis-length*
縦軸の印刷位置の数を指定します。

VREVERSE
縦軸上の値の順序を逆にします。

VSPACE=*n*
縦軸上の目盛間隔を指定します。

VZERO
縦軸の最初の目盛にゼロの値を割り当てます。

Label points on a plot

LIST<=*penalty-value*>
ペナルティとポイントの配置状態をリストします。

OUTWARD='*character*'
強制的にラベルを原点から離します。

PENALTIES<(*index-list*)>=*penalty-list*
デフォルトペナルティを変更します。

PLACEMENT=(*expression(s)*)
ラベル配置のための場所を指定します。

SPLIT='*split-character*'
ラベルの区切り文字を指定します。

STATES
有効な配置状態をすべてリストします。

Produce a contour plot

CONTOUR<=*number-of-levels*>
等高線グラフを書きます。

Scontour-level='*character-list*'

1 つの等高線レベルに対してプロット記号を指定します。

Produce a countour plot

SLIST='*character-list-1*' <'*character-list-2* …'>
複数の等高線レベルに対してプロット記号を指定します。

Specify reference lines

HREF=*value-specification*
横軸上の指定値に対して直角を成す直線を引きます。

HREFCHAR='*character*'
水平方向の参照線を引くために使用する文字を指定します。

VREF=*value-specification*
縦軸上の指定値に対して直角を成す直線を引きます。

VREFCHAR='*character*'
垂直参照線を引くために使用する文字を指定します。

必須引数

plot-request(s)
プロットする変数(縦と横)、およびプロットのポイントをマークするためのプロット記号を指定します。

*plot-request(s)*の各形式がラベル変数をサポートしています。ラベル変数の前にドル記号($)が置かれ、プロットのポイントにラベルを付ける値を含む変数が指定されます。

```
plot y*x $ label-variable
```

```
plot y*x='*' $ label-variable
```

詳細については、“ラベルとプロットポイント” (1227 ページ)を参照してください。さらに、“例 9: プロットにラベルを追加する” (1272 ページ)、およびこれ以降のすべての例を参照してください。

*plot-request(s)*には、次のうち 1 つ以上が可能です。

vertical*horizontal <$ label-variable>
縦軸でプロットする変数と横軸でプロットする変数を指定します。

たとえば次のステートメントでは、Y×X のプロットが要求されます。

```
plot y*x;
```

Y は縦軸上、Xは横軸上に表示されます。

この形式のプロット要求では、プロットポイントをマークするプロット記号を選択するためのデフォルトメソッドが使用されます。プロットの 1 つのポイントがデータセット内の 1 つのオブザベーションを表す場合、PLOC PLOT はそのポイントに文字 A を置きます。1 つのポイントが 2 つのオブザベーションの値を表す場合、文字 B が表示されます。1 つのポイントが 3 つのオブザベーションを表す場合は、文字 C が表示され、それ以降もアルファベットの最後まで同様に処理されます。文字 Z は、同じ印刷位置にオブザベーションが 26 以上発生した場合に使用されます。

vertical*horizontal='character' <$ label-variable>
縦軸と横軸でプロットする変数、およびプロットの各ポイントをマークするプロット記号を指定します。1 文字で、1 つ以上のオブザベーションの値を表します。

たとえば、次のステートメントは二変量のプロットを要求しており、プロットの各ポイントはプラス記号(+)で表されます。

```
plot y*x='+';
```

*vertical*horizontal=variable <$ label-variable>*
縦軸と横軸でプロットする変数、およびその値がプロットの各ポイントをマークする変数を指定します。これら変数は、数値または文字になります。(2 つ以上の値の開始文字が同じ場合でも)フォーマットされた変数値の最初(左端)のブランク以外の文字がプロット記号として使用されます。同じプロット位置にオブザベーションが 2 つ以上マップされた場合は、最初のオブザベーションの値によってポイントがマークされます。たとえば、次のステートメントでは、GENDER は **FEMALE** と **MALE** の値を含む文字変数です。**F** 値と **M** 値でプロットの各オブザベーションがマークされます。

```
plot height*weight=gender;
```

参照項目 “プロット要求の変数リストの指定” (1248 ページ) および “変数の組み合わせの指定” (1248 ページ)

オプション引数

BOX
左側と下部だけではなく、プロット全体の周りに境界線を引きます。

例 “例 3: 2 つのプロットを重ね合わせる” (1255 ページ)

CONTOUR<=*number-of-levels*>
プロット記号を使って等高線グラフを書きます。*number-of-levels* が *variable* の範囲を分割するためのレベル数である部分の網かけの程度はさまざまです。プロット要求は *vertical*horizontal=variable* の形式で指定し、*variable* はデータセット内の数値変数にする必要があります。網かけ濃度はこの変数の値によって決定されます。

CONTOUR を使用する場合、PROC PLOT は *variable* の欠損値を含むオブザベーションはプロットしません。

OVP システムオプションで重ね打ちを有効化している場合は、重ね打ちを使用して網かけが生成されます。有効化していない場合は、さまざまな濃さの単一文字が使用されます。CONTOUR オプションは、プロットが密集している場合が最も効果的です。

デフォルト 10

範囲 1-10

例 “例 7: 等高線グラフの作成” (1265 ページ)

HAXIS=*axis-specification*
横軸の目盛値を指定します。

- 数値の場合、*axis-specification* は、明示的な値リストか BY 増分のどちらか、または両方の組み合わせです。

  n < … *n*>

  BY *increment*

  n TO *n* BY *increment*

  値は、昇順または降順である必要があります。降順を指定するには、*increment* に負の値を使用します。指定される値は、値が均等に分布しない場合も、横軸に沿って均等にスペースが調整されます。数値は、次の方法で指定できます。

表 42.5 HAXIS=数値の指定

HAXIS=値	コメント
`10 to 100 by 5`	値は、10 で開始して 100 で終了する 5 の増分として示されます。
`by 5`	値が 5 ずつ増加します。PROC PLOT が目盛の最小値と最大値を決定します。
`10 100 1000 10000`	値は均等に分布しません。この指定によって対数プロットが作成されます。PROC PLOT は、軸の指定によって暗示される関数を決定できない場合、ポイント間に単純な線形補間を使用します。PROC PLOT が関数を正確に補間するかどうかを決定するには、DATA ステップで関数を決定するデータを生成し、プロット時に線形に表示されるかどうかを確認します。例については、“例 5: 対数尺度にデータをプロットする” (1261 ページ)を参照してください。
`1 2 10 to 100 by 5`	前述の指定の組み合わせ。

- *文字変数の場合*、, *axis-specification* は引用符で囲まれた一意の値のリストです。

  '*value-1*' <…'*value-n*'>

  たとえば、次のステートメントは横軸の目盛値を示すために 3 都市を割り当てます。

  ```
  haxis='Paris' 'London' 'Tokyo'
  ```

  文字列では、大文字と小文字が区別されます。文字変数に出力形式が関連付けられている場合、*axis-specification* にはフォーマットされた値を指定する必要があります。値はどのような順序でも指定できます。

- *日時値を含む軸の変数の場合*、*axis-specification* は、明示的な値リストか、増分を指定した開始値と終了値のどちらかです。

  '*date-time-value*'*i* <…'*date-time-value*'*i*>

  '*date-time-value*'*i* TO <…'*date-time-value*'*i*> <BY *increment*>

'date-time-value'i
SAS 関数 INTCK および INTNX に対して記述された任意の SAS 日付値、SAS 時間値または SAS 日時値。接尾辞 *i* は次のいずれかになります。

D 日付

T 時間

DT 日時

increment
INTCK または INTNX 関数の有効な引数のいずれかです。*increment* は次のいずれかになります。

表 42.6 INTCK 値および INTNX 値

日付	日時	時間
DAY	DTDAY	HOUR
WEEK	DTWEEK	MINUTE
MONTH	DTMONTH	SECOND
QTR	DTQTR	
YEAR	DTYEAR	

次の例にはデータ増分が含まれています。

```
haxis='01JAN95'd to '01JAN96'd
by month

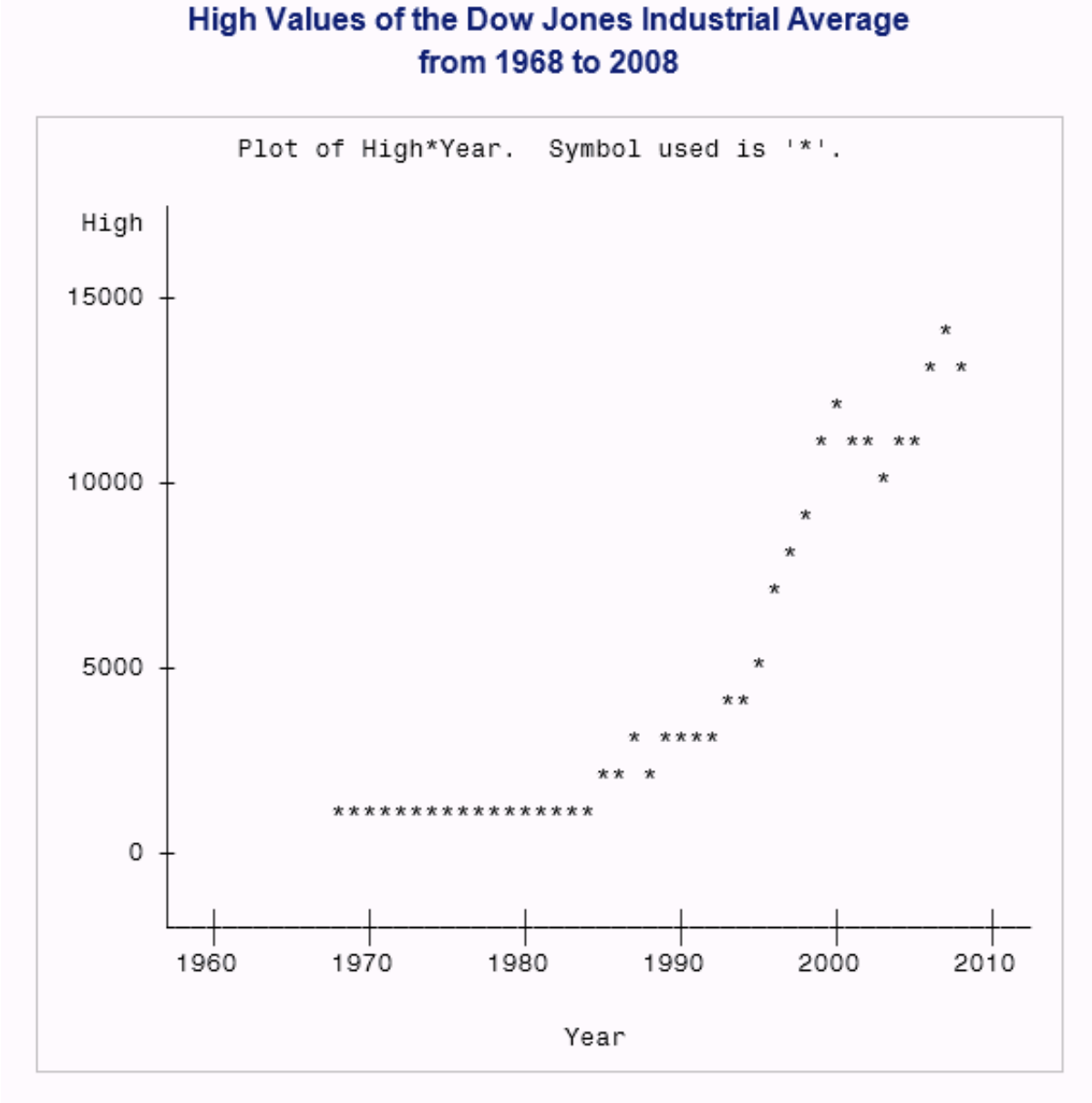
haxis='01JAN95'd to '01JAN96'd
by qtr
```

注: わかりやすい出力形式で目盛値を出力するには、FORMAT ステートメントを使用する必要があります。

操作 軸を等しくするには、HAXIS=および VAXIS=オプションを VTOH=オプションとあわせて使用します。データが適切な場合は、HAXIS=BY*n* と VAXIS=BY*n* で *n* に同じ値を指定し、VTOH=オプションの値を指定します。横の目盛を区切る列の数は、縦の目盛を区切る行の数に VTOH=オプションの値を乗じた数にほぼ等しくなります。場合によっては、PROC PLOT が 3 つの値すべてを同時に使用することができず、値を 1 つ以上変更することがあります。

例 “例 2: X 軸の制御と参照線の追加” (1253 ページ)

“例 5: 対数尺度にデータをプロットする” (1261 ページ)

“例 6: 軸に日付値をプロットする” (1262 ページ)

HEXPAND
可能であれば、横軸を拡張して、プロットの両側の余白を最小限に抑え、目盛の間隔を最大限にします。

HEXPAND を使用すると、PROC PLOT はデータのスペースについての情報を無視します。このオプションを指定して作成したプロットは、スペースの無駄は少ないですが、本質的な変数の関係が不明瞭になる場合があります。

HPOS=*axis-length*
横軸の印刷位置の数を指定します。プロットを 1 ページに収めることが可能な *axis-length* の最大値は、LINESIZE=システムオプションの値から 3 を引いた位置です。これは、プロシジャが縦軸の隣に情報を印刷するスペースを必要とするためです。正確な最大値は縦軸の変数値の文字数によって異なります。*axis-length* が大きすぎて 1 行に収まらない場合、PROC PLOT はオプションを無視します。

HREF=*value-specification*
プロットで横軸上の指定値に対して直角を成す直線を引きます。PROC PLOT は、HREF=オプションで指定した横軸の値を含みます。ただし、HAXIS=オプションで他の値を指定した場合は別です。

value-specification の構文については、“HAXIS=*axis-specification* ” (1238 ページ)を参照してください。

例 “例 8: BY グループのプロット” (1268 ページ)

HREFCHAR='*character*'
水平方向の参照線を引くために使用する文字を指定します。

デフォルト 縦棒(|)

参照項目 “FORMCHAR <(*position(s)*)>='*formatting-character(s)*'” (1232 ページ) および “HREF=*value-specification* ” (1240 ページ)

HREVERSE
横軸上の値の順序を逆にします。

HSPACE=*n*
横軸上で印刷位置 *n* ごとに目盛が発生するように指定します。このとき、*n* は HSPACE=の値です。

HZERO
横軸の最初の目盛にゼロの値を割り当てます。

操作 PROC PLOT は、横軸変数に負の値がある場合や、HAXIS=オプションでゼロ以外の値から始まる範囲を指定している場合、HZERO を無視します。

LIST<=*penalty-value*>
縦軸と横軸の値、ペナルティ、および *penalty-value* 以上のペナルティでプロットされるすべてのポイントの配置状態をリストします。*penalty-value* 以上のペナルティを有するプロットされたポイントがない場合、リストは印刷されません。

ヒント LIST は LIST=0 に相当します。

参照項目 “ペナルティについて” (1227 ページ)

例 “例 11: PLACEMENT=オプションを使用し、プロット上のラベルを調整する” (1278 ページ)

OUTWARD='*character*'
原点(0,0)の方向にある *character* と一致する記号の隣の位置を保護することにより、ポイントラベルを強制的に外側に移動してプロットの原点から離そうとします。アルゴリズムが保護された位置へのラベル付けを回避しようとするので、通常は外側に移動します。

ヒント このオプションは、変数値を示すラベルをポイントに付ける場合のみ有益です。

OVERLAY
1 つの軸セット上に PLOT ステートメントで指定したすべてのプロットを重ね合わせます。軸のラベル付けには、最初のプロットの変数名(存在する場合は変数ラベル)が使用されます。HAXIS=または VAXIS=オプションを指定しなければ、PROC PLOT は、変数がすべてうまく収まるように軸のスケールを自動的に調整します。

SAS システムオプション OVP が有効で、重ね打ちが可能な場合、プロットは重なり合います。逆に NOOVP が有効な場合、PROC PLOT は、最初のプロットのプロット記号を使用して、2 つ以上のプロットに表示されるポイントを表します。その場合、非表示のオブザベーションがいくつあるかを示すメッセージが出力に含まれます。

例 “例 3: 2 つのプロットを重ね合わせる” (1255 ページ)

PENALTIES<(*index-list*)>=*penalty-list*
デフォルトペナルティを変更します。*index-list* では、ペナルティリストにおけるペナルティの位置が提供されます。*penalty-list* には、*index-list* で示されるペナルティに対して指定する値が含まれます。*index-list* と *penalty-list* には、1 つ以上の整数を含められます。さらに、*index-list* および *penalty-list* の両方で、**`value TO value`** という形式が受け入れられます。

参照項目 “ペナルティについて” (1227 ページ)

例 “例 13: デフォルトのペナルティを変更する” (1286 ページ)

PLACEMENT=(*expression(s)*)
座標に関連するラベルの配置可能な場所を指定することによって、ラベルの配置を制御します。各 *expression* は、アスタリスク(*)またはコロン(:)で結ばれる 1 つ以上のサブオプション(H=、L=、S=、V=)のリストで構成されます。PROC PLOT は、アスタリスクとコロンを使用し、各式を展開して、4 つの使用可能なサブオプションの値の組み合わせにします。アスタリスクでは、式リストで可能な値の組み合わせがすべて作成されます。コロンでは、対の組み合わせのみが作成されます。コロンはアスタリスクより優先されます。コロンでは、一方のリストが他方のリストより短い場合、短いリストの値は必要に応じて再利用されます。

配置を制御するには、次のサブオプションを使用します。

H=*integer(s)*
横のスペース(列)の数を指定し、ラベルの開始位置をシフトします。正の整数と負の整数の両方が有効です。正の整数ではラベルが右にシフトされ、負の整数では左にシフトされます。たとえば、H=サブオプションは次のように使用できます。

```
place=(h=0 1 -1 2 -2)
```

このリストではキーワード BY ALT を使用できます。BY ALT で作成される一連の数字では、正と負の符号が交互に並び、1 対ごとに 1 つずつ絶対値が変わります。たとえば、次の PLACE=の指定は同等です。

```
place=(h=0 -1 to -3 by alt)
place=(h=0 -1 1 -2 2 -3 3)
```

この一連の数字にゼロが含まれる場合は、ゼロが 2 回出現します。たとえば、次の PLACE=オプションは同等です。

```
place=(h= 0 to 2 by alt)
place=(h=0 0 1 -1 2 -2)
```

デフォルト H=0

範囲 −500 から 500

L=*integer(s)*
ラベルの分割が可能な行数を指定します。

デフォルト L=1

範囲 1-200

S=*start-position(s)*
ラベル印刷の開始位置を指定します。*start-position* の値には、次のうちから 1 つ以上を指定できます。

CENTER
プロシジャは、プロット記号を中央にしてラベルを配置します。

RIGHT
ラベルは、プロット記号の場所から開始され、右へと続きます。

LEFT
ラベルは、プロット記号の左から開始され、プロット記号の場所で終了します。

デフォルト CENTER

V=*integer(s)*
縦のスペース(行)の数を指定し、ラベルの開始位置をシフトします。V=の動作は前述の H=サブオプションと同じです。

サブオプションの前に演算子が置かれていない場合、新しい式が開始されます。各式を囲むかっこは任意指定です。指定すると、リスト内の個々の式がわかりやすくなります。ただし、式リスト全体は、次の例に示すように、かっこで囲む必要があります。後述の表では、次の式の展開を示し、各配置状態について説明しています。

```
place=((v=1)
    (s=right left : h=2 -2)
    (v=-1)
    (h=0 1 to 2 by alt * v=1 -1)
    (l=1 to 3 * v=1 to 2 by alt *
     h=0 1 to 2 by alt))
```

値の各組み合わせが*配置状態*です。プロシジャは、配置状態リストの出現順序どおりに配置状態を使用するので、最優先する配置状態を最初に指定します。プロシジャは、ラベルごとにすべての状態を試し、最小限のペナルティでラベルを配置する最初の状態を使用します。すべてのラベルの初期配置時に、プロシジャはプロットを複数回繰り返し表示して、配置を体系的に改善します。改善ステップでは、ペナルティを最小化することと、状態リストの最初に近い方の配置を使用することの両方が試みられます。ただし、PROC PLOT はヒューリスティック方法を使用して配置を行うので、最善の配置セットが見つかるとは限りません。

表 42.7 *式リストの配置状態への展開*

式	配置状態	意味
(V=1)	S=CENTER L=1 H=0 V=1	ポイントの上の行で、ポイントを中心としてラベルを配置します。ラベルのために 1 行を使用します。
(S=RIGHT LEFT :H=2 −2)	S=RIGHT L=1 H=2 V=0	ポイントの 2 列右からラベルを開始します。ラベルのために 1 行を使用します。
	S=LEFT L=1 H=−2 V=0	ポイントの 2 列左でラベルを終了します。ラベルのために 1 行を使用します。
(V=-1)	S=CENTER L=1 H=0 V=− 1	ポイントの下の行で、ポイントを中心としてラベルを配置します。ラベルのために 1 行を使用します。

式	配置状態	意味
(H=0 1 to 2 BY ALT * V=1 −1)	S=CENTER L=1 H=0 V=1	ポイントの上の行で、ポイントを中心としてラベルを配置します。
	S=CENTER L=1 H=0 V=-1	ポイントの下の行で、ポイントを中心としてラベルを配置します。
	S=CENTER L=1 H=1 V=1	ポイントの上の行で、ラベルを中心から右に 1 列シフトします。
	S=CENTER L=1 H=1 V=-1	ポイントの下の行で、ラベルを中心から右に 1 列シフトします。
	S=CENTER L=1 H=-1 V=1	ポイントの上の行で、ラベルを中心から左に 1 列シフトします。
	S=CENTER L=1 H=− 1 V=−1	ポイントの下の行で、ラベルを中心から左に 1 列シフトします。
	S=CENTER L=1 H=2 V=1 S=CENTER L=1 H=2 V=−1	最初にポイントの上の行、次に下の行で、ラベルを中心から右に 2 列シフトします。
	S=CENTER L=1 H=-2 V=1 S=CENTER L=1 H=-2 V=-1	最初にポイントの上の行、次に下の行で、ラベルを中心から左に 2 列シフトします。
(L=1 to 3 * V=1 to 2 BY ALT * H=0 1 to 2 BY ALT)	S=CENTER L=1 H=0 V=1	ポイントの上の行で、ポイントを中心としてラベルを配置します。ラベルのために 1 行を使用します。
	S=CENTER L=1 H=1 V=1 S=CENTER L=1 H=−1 V=1 S=CENTER L=1 H=2 V=1 S=CENTER L=1 H=−2 V=1	ポイントの上の行で、ラベルを右または左に 1 列または 2 列シフトします。ラベルのために 1 行を使用します。
	S=CENTER L=1 H=0 V=-1	ポイントの下の行で、ポイントを中心としてラベルを配置します。ラベルのために 1 行を使用します。
	S=CENTER L=1 H=1 V=−1 S=CENTER L=1 H=−1 V=−1 S=CENTER L=1 H=2 V=−1 S=CENTER L=1 H=−2 V=−1	ポイントの下の行で、ラベルを右および左に 1 列または 2 列シフトします。
	.	ポイントの 2 行上とポイントの 2 行下で、同じ水平シフトを使用します。
	S=CENTER L=1 H=− 2 V=-2	

式	配置状態	意味
	S=CENTER L=2 H=0 V=1	ラベルを 2 行に分割する処理全体を繰り返します。次に、ラベルを 3 行に分割する処理を繰り返します。
	S=CENTER L=3 H=− 2 V=-2	

別名 PLACE=

デフォルト PLACE=オプションにはデフォルトが 2 つあります。プロット記号としてブランクを使用すると、デフォルトの配置状態は PLACE=(S=CENTER :V=0 :H=0 :L=1)となり、ラベルが中央揃えになります。ブランク以外を使用すると、デフォルトは PLACE=((S=RIGHT LEFT :H=2 −2) (V=1 −1 * H=0 1 -1 2 -2))となります。記号とあわせて配置されるラベルのデフォルトには、プロット記号のまわりの位置が複数含まれるので、プロシジャは密集したプロットで柔軟にラベルを配置できます。

ヒント 配置状態リストを印刷するには STATES オプションを使用します。

参照項目 “ラベルとプロットポイント” (1227 ページ)

例 “例 11: PLACEMENT=オプションを使用し、プロット上のラベルを調整する” (1278 ページ)

“例 12: マクロを使用し、プロット上のラベルを調整する” (1283 ページ)

Scontour-level='character-list'

単一等高線レベルに対して使用するプロット記号を指定します。PROC PLOT は等高線グラフを作成する際、濃度レベルごとに使用する記号を自動的に選択します。これらの記号を無効にして固有の指定を行うには、S=オプションを使用します。*character-list* には 3 文字まで含められます。重ね打ちができない場合、PROC PLOT は最初の文字のみを使用します。

たとえば、Z 変数に対して 3 レベルの網かけを指定するには、次のステートメントを使用します。

```
plot y*x=z /
    contour=3 s1='A' s2='+' s3='X0A';
```

また、プロット記号を 16 進定数で指定することもできます。

```
plot y*x=z /
    contour=3 s1='7A'x  s2='7F'x s3='A6'x;
```

これは 16 進定数でグレースケール埋め字を示せるプリンタ向けの機能です。

範囲 1 から最高までの等高線レベル(CONTOUR オプションで設定)。

参照項目 “SLIST='*character-list-1*' <'*character-list-2 …*'>” (1246 ページ)および “CONTOUR<=*number-of-levels*>” (1238 ページ)

SLIST='*character-list-1*' <'*character-list-2* …'>
複数の等高線レベルに対してプロット記号を指定します。*character-list* ごとに 1 つの等高線レベルに対するプロット記号が指定されます。最初の *character-list* では最初のレベル、2 番目の *character-list* では 2 番目のレベル、というように処理されます。次の例は、5 つの等高線レベルに対して異なる 5 つのプロット記号を付与する方法を示しています。

```
plot y*x=z /
     contour=5  slist='.' ':' '!' '=' '+O';
```

各等高線レベルのプロット記号を省略すると、PROC PLOT はデフォルト記号を使用します。

```
slist='.' ',' '-' '=' '+' 'O' 'X'        'W' '*' '#'
```

制限事項 SLIST=オプションを使用する場合は、PLOT ステートメントの最後にリストする必要があります。

参照項目 “Scontour-level='character-list'” (1245 ページ) および
“CONTOUR<=*number-of-levels*>” (1238 ページ)

SPLIT='*split-character*'
プロットポイントにラベルを付ける際、複数行にまたがるラベルを分割する場所を指定します。ラベルは PLACEMENT=オプションに対する L=サブオプションで指定した行数に分割されます。区切り文字を指定すると、プロシジャは、適切な配置が見つからなくても、その文字が出現するたびに常にラベルを分割します。L=2 以上を指定しているのに区切り文字を指定していない場合、プロシジャは、ブランクや句読点でラベルを分割しようとしますが、必要であれば単語も分割します。

PROC PLOT は、分割ラベルを個々のフラグメント単位ではなくブロック単位でシフトします(*フラグメント*は 1 行に含まれる分割ラベルの一部です)。たとえば、強制的に `This is a label` を `a` の後ろで分割するには、`This is a*label` に変更して `SPLIT='*'`を指定します。

参照項目 “ラベルとプロットポイント” (1227 ページ)

STATES
有効な配置状態をすべてリストします。STATES は、PLACE=オプションで指定した順序で配置状態を印刷します。

VAXIS=*axis-specification*
縦軸の目盛値を指定します。VAXIS=は、HAXIS=オプションと同じルールに従います。

例 “例 7: 等高線グラフの作成” (1265 ページ)

“例 12: マクロを使用し、プロット上のラベルを調整する” (1283 ページ)

VEXPAND
可能であれば、縦軸を拡張して、プロットの上下の余白を最小限に抑え、縦目盛間のスペースを最大限にします。

参照項目 “HEXPAND ” (1240 ページ)

VPOS=*axis-length*
縦軸の印刷位置の数を指定します。プロットを 1 ページに収めることが可能な *axis-length* の最大値は、SAS システムオプション PAGESIZE=の値から 8 行を引いた値です。これは、プロシジャが横軸の下に情報を印刷する余地を残しておく必要があるためです。正確な最大値は、使用するタイトル、プロットの重ね合わせの有無、および CONTOUR 指定の有無によって異なります。*axis-length* 値の指定に

よってプロットが 1 ページに収まらなくなった場合、プロットは複数ページにわたって出力されます。

参照項目 “HPOS=*axis-length* ” (1240 ページ)

VREF=*value-specification*
プロットで縦軸上の指定値に対して直角を成す直線を引きます。PROC PLOT は、VREF=オプションで指定した縦軸の値を含みます。ただし、VAXIS=オプションで他の値を指定した場合は別です。*value-specification* の構文については、“HAXIS=*axis-specification* ” (1238 ページ)を参照してください。

例 “例 2: X 軸の制御と参照線の追加” (1253 ページ)

VREFCHAR='*character*'
垂直参照線を引くために使用する文字を指定します。

デフォルト 横棒(-)

参照項目 “FORMCHAR <(*position(s)*)>='*formatting-character(s)*'” (1232 ページ), “HREFCHAR='*character*'” (1241 ページ), および “VREF=*value-specification* ” (1247 ページ)

VREVERSE
縦軸上の値の順序を逆にします。

VSPACE=*n*
縦軸上で印刷位置 *n* ごとに目盛が発生するように指定します。このとき、*n* は VSPACE=の値です。

VZERO
縦軸の最初の目盛にゼロの値を割り当てます。

操作 PROC PLOT は、縦軸変数に負の値がある場合や、VAXIS=オプションでゼロ以外の値から始まる範囲を指定している場合、VZERO オプションを無視します。

PLOT プロシジャの使用

プログラムステートメントを使用してデータを作成する

プロットするデータを作成する際の適切なルールは、作成するオブザベーション数を、横軸の位置数よりも少なくすることです。その場合、PROC PLOT は、横軸変数の増分を目盛の間隔として使用します。

PROC PLOT はオブザベーションごとに 1 文字ずつ印刷するので、SAS プログラムステートメントを使用して PROC PLOT のデータセットを作成すると、連続するプロットの有効性を高められます。たとえば、次の方程式をプロットするためにデータを作成するとします。この方程式では、x の範囲は 0 から 100 までです。

$$y = 2.54 + 3.83x$$

次のステートメントをサブミットします。

```
options linesize=80;
```

```
data generate;
   do x=0 to 100 by 2;
      y=2.54+3.83*x;
      output;
   end;
run;
proc plot data=generate;
   plot y*x;
run;
```

LINESIZE=値 80 でプロットを印刷する場合、X 値を示す横軸上ではおよそ 75 の位置が使用可能になります。したがって、2 が適切な増分です。横軸で使用可能な位置数 75 よりも少ない 51 のオブザベーションが生成されます。

ただし、LINESIZE=値 132 でプロットを印刷する場合は、増分を 2 にすると、プロット記号の間にスペースが空いたプロットが作成されます。滑らかな線にするには、1 の方が適切な増分です。その場合、101 のオブザベーションが生成されるからです。

プロット要求の変数リストの指定

プロット要求で SAS 変数リストを使用できます。有効なプロット要求の例を次に示します。

表 42.8 プロット要求

プロット要求	プロット対象
(a - - d)	a*b a*c a*d b*c b*d c*d
(x1 - x4)	x1*x2 x1*x3 x1*x4 x2*x3 x2*x4 x3*x4
(_numeric_)	数値変数のすべての組み合わせ
y*(x1 - x4)	y*x1 y*x2 y*x4 y*x4

垂直と水平の両方の指定で 1 つ以上の変数が要求されて、変数が両方のリストに存在する場合、その変数が自身に対してプロットされることはありません。たとえば、次のステートメントでは、B*B および C*C はプロットされません。

```
plot (a b c)*(b c d);
```

変数の組み合わせの指定

要求での演算子はアスタリスク(*)かコロン(:)のどちらかです。アスタリスクは、リストの変数を組み合わせて、*x* 変数と *y* 変数で可能な組み合わせをすべて生成します。たとえば、次のプロット要求は同等です。

```
plot (y1-y2) * (x1-x2);

plot y1*x1 y1*x2 y2*x1 y2*x2;
```

コロンは、変数を対にして組み合わせます。したがって、各リストの最初の変数同士を組み合わせてプロットを要求し、2 番目、3 番目以降も同様に処理します。たとえば、次のプロット要求は同等です。

```
plot (y1-y2) : (x1-x2);

plot y1*x1 y2*x2;
```

PENALTIES=オプションの使用

次の例では、PENALTIES=オプションにより、3、4、5 番目の文字を印刷しない場合としてデフォルトペナルティを 11、10、8 に上げ、1、2 番目の文字を印刷しない場合としてデフォルトペナルティを 2 まで下げています。

```
penalties(15 to 20)=2 2 11 10 8 2
```

この例では、ペナルティリストが拡張されます。20 番目のペナルティの 2 は、6 番目から 200 番目までの文字が印刷されないことに対するペナルティです。最後のインデックス *i* が 18 より大きい場合、最後のペナルティは(*i* − 14)番目以降の文字に対して使用されます。

また、開始インデックスを指定するだけでもペナルティリストを拡張できます。たとえば、次の PENALTIES=オプションは、前述の指定に相当します。

```
penalties(15)=2 2 11 10 8 2
```

結果: PLOT プロシジャ

軸の尺度

通常、PROC PLOT は、各変数の最も水準の低い順位 5 つの各ペア間の最小差異(*デルタ*)を調べ、可能な場合は、最終スケール調整軸でその間隔が印刷位置ごとに 1 つずつしかないことを確認します。この間隔配置の余地がない場合、および PROC PLOT が、データが人為的に生成されていると推測した場合は、各印刷位置にデルタの固定数が置かれます。その他の場合は、PROC PLOT は値を無視します。

表示された出力

PLOT ステートメントの VPOS=および HPOS=オプション、PROC PLOT ステートメントの VPERCENT=もしくは HPERCENT=オプション、または PAGESIZE=および LINESIZE=システムオプションでプロットのサイズを変更しない限り、プロットごとに 1 ページ全体が使用されます。タイトル、凡例および変数ラベルが各ページの上部に印刷されます。それぞれの軸に、変数の名前かまたは(存在する場合は)変数のラベルがラベル付けされます。

通常、PROC PLOT は、新しいページで新しいプロットを開始します。ただし、VPERCENT=および HPERCENT=オプションを使用すると、1 ページに 2 つ以上のプロットを印刷できます。VPERCENT=および HPERCENT=については、“PROC PLOT ステートメント” (1231 ページ) で前述しています。

PROC PLOT は、RUN ステートメントの後と BY グループの最初で常に新しいページを開始します。

ODS テーブル名

PLOT プロシジャは、作成する各テーブルに名前を割り当てます。これらの名前を使用して、Output Delivery System (ODS)を使用してテーブルを選択し、出力データセットを作成する際にそのテーブルを参照できます。詳細については、“ODS Table Names and the Base SAS Procedures That Produce Them” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

表 42.9 *PLOT プロシジャによって生成される ODS テーブル*

テーブル名	説明	テーブル生成時の条件
Plot	単一プロット	OVERLAY オプションを指定しない場合。
Overlaid	単一軸セット上の 2 つ以上のプロット	OVERLAY オプションを指定する場合。

PROC PLOT による ODS 出力のポータビリティ

特定の状況で PROC PLOT を Output Delivery System と使用すると、ポータブルでないファイルが生成されます。SAS セッションの SAS システムオプション FORMCHAR= で非標準の線描文字が使用されると、SAS Monospace フォントがインストールされていない動作環境では線の代わりに不正な文字が出力に含まれていることがあります。この問題を回避するには、PROC PLOT を実行する前に次の OPTIONS ステートメントを指定します。

```
options formchar="|----|+|---+=|-/\<>*";
```

欠損値

プロットする変数のどちらかの値が欠損している場合、PROC PLOT はプロットにそのオブザベーションを含めません。ただし、Y*X のプロットでは、対応する Y が欠損値の X 値は、PROC PLOT ステートメントで NOMISS オプションを指定しない限り、X 軸のスケーリングに含まれます。

非表示のオブザベーション

デフォルトでは、PROC PLOT は、値がプロット上の同じ場所にあるオブザベーションを表すには、異なるプロット記号(A、B、C など)を使用します。ただし、独自のプロット記号を指定する場合や、OVERLAY オプションを使用する場合は、同じ場所にある値を認識できない場合があります。

プロット記号を指定すると、PROC PLOT は、値が同じ場所にあるオブザベーション数に関係なく、同じ記号を使用します。OVERLAY オプションを使用し、なおかつ重ね打ちが無効な場合、PROC PLOT は最初のプロット要求の記号を使用します。どちらの場合も、非表示のオブザベーションがいくつあるかを示すメッセージが出力に含まれます。

例: PLOT プロシジャ

例 1: プロット記号の指定

要素: PROC PLOT ステートメントオプション
FORMCHAR
PLOT ステートメント

データセット: Dija

詳細

この例では、異なるプロット記号を指定することによって、アウトプット 42.1 (1224 ページ)について詳しく説明します。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
...more data lines...
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;

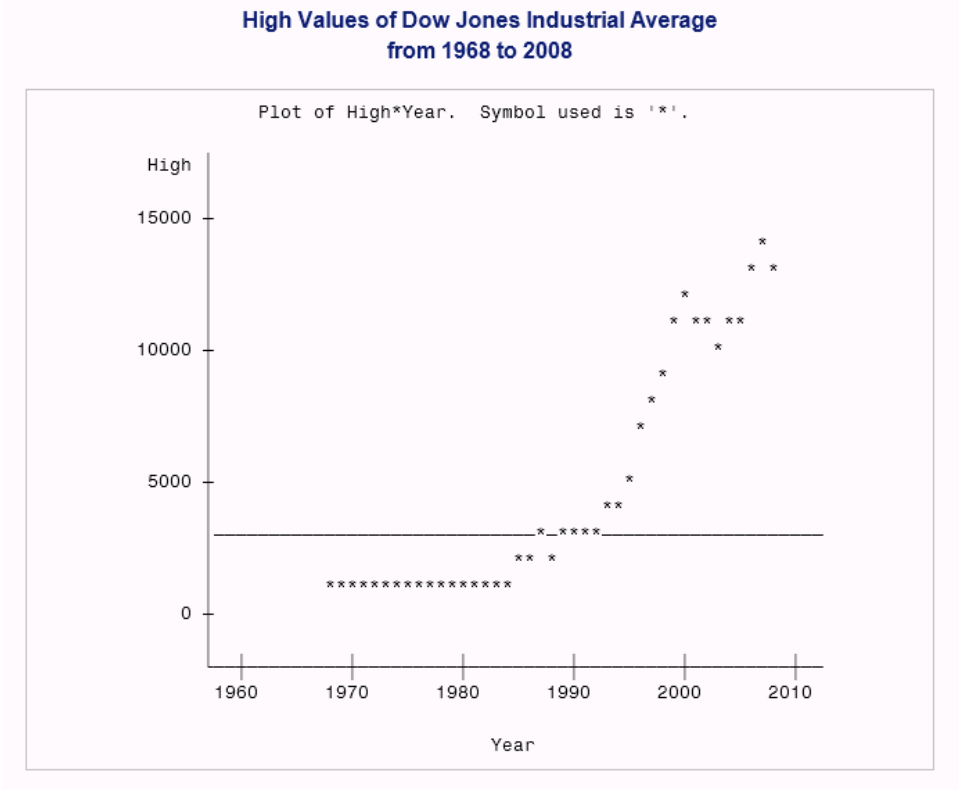
proc plot data=djia;
      plot high*year='*'
      / vspace=5 vaxis=by 1000;

   title 'High Values of the Dow Jones Industrial Average';
   title2 'from 1968 to 2008';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

Dija データセットを作成します。 Dija には、1968 年から 2008 年までのダウジョーンズ工業平均の最高終値と最安終値が含まれます。DATA ステップでこのデータセットが作成されます。

```
data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
...more data lines...
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;
```

プロットを作成します。 プロット要求によって、縦軸に High の値、横軸に Year の値がプロットされます。また、アスタリスクがプロット記号として指定されます。VAXIS=オプションと VSPACE=オプション。VAXIS=by 1000 オプションでは、縦軸の目盛値の増分として 1,000 が指定されます。VSPACE=のオプションでは、縦軸の目盛間にある、印刷スペースの値が指定されます。

```
proc plot data=djia;
      plot high*year='*'
      / vspace=5 vaxis=by 1000;
```

タイトルを指定します。

```
   title 'High Values of the Dow Jones Industrial Average';
   title2 'from 1968 to 2008';
run;
```

出力

PROC PLOT は、両軸の目盛と尺度を決定します。

アウトプット 42.4 *プロット記号がアスタリスクのプロット*



例 2: X 軸の制御と参照線の追加

要素: PROC PLOT ステートメントオプション
FORMCHAR
PLOT ステートメント
PLOT ステートメントオプション
HAXIS=
VREF=

データセット: Dija

詳細

この例では、横軸の値を指定し、縦軸から参照線を引きます。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
...more data lines...
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;

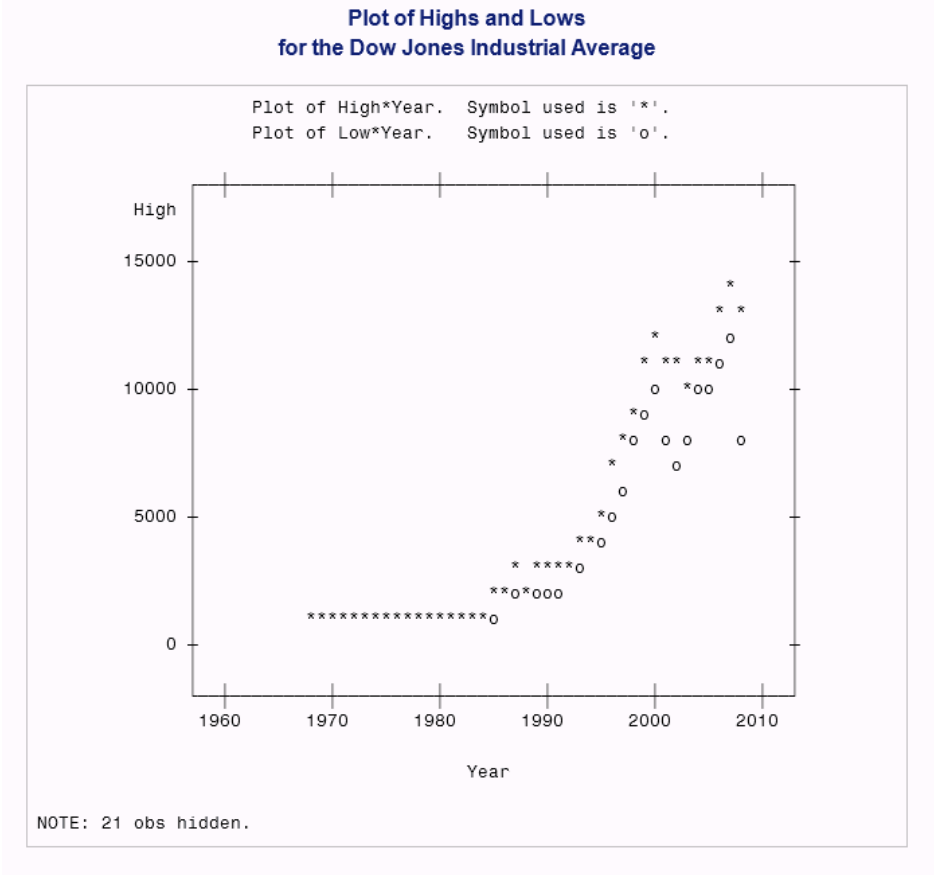
proc plot data=djia;
   plot high*year='*'

     / haxis=1965 to 2020 by 10 vref=3000;

title 'High Values of Dow Jones Industrial Average';
title2 'from 1968 to 2008';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

Dija データセットを作成します。 Dija には、1968 年から 2008 年までのダウジョーンズ工業平均の最高終値と最安終値が含まれます。DATA ステップでこのデータセットが作成されます。

```
data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
...more data lines...
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;
```

プロットを作成します。 プロット要求によって、縦軸に High の値、横軸に Year の値がプロットされます。また、アスタリスクがプロット記号として指定されます。

```
proc plot data=djia;
   plot high*year='*'
```

横軸をカスタマイズし、参照線を引きます。 HAXIS=では、横軸に値 1968 から 2008 までを 10 年の増分で表示するように指定します。VREF=では、縦軸の値 3000 を起点とする参照線が引かれます。

```
       / haxis=1965 to 2020 by 10 vref=3000;
```

タイトルを指定します。

```
title 'High Values of Dow Jones Industrial Average';
title2 'from 1968 to 2008';
run;
```

出力

アウトプット 42.5 *参照線のあるプロット*



例 3: 2 つのプロットを重ね合わせる

要素: PROC PLOT ステートメントオプション
FORMCHAR

PLOT ステートメント

PLOT ステートメントオプション
- BOX
- HAXIS
- OVERLAY
- VAXIS

データセット: Dija

詳細

この例では、2 つのプロットを重ね合わせ、プロットのまわりを枠で囲みます。

プログラム

```
options formchar="|";

data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
...more data lines...
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;

proc plot data=djia formchar="|----|+|---+=|-/\<>*";
   plot high*year='*'
        low*year='o' / overlay box
        haxis=by 10
        vaxis=by 5000;

   title 'Plot of Highs and Lows';
   title2 'for the Dow Jones Industrial Average';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|";
```

Dija データセットを作成します。 Dija には、1968 年から 2008 年までのダウジョーンズ工業平均の最高終値と最安終値が含まれます。DATA ステップでこのデータセットが作成されます。

```
data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
```

```
...more data lines...
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;
```

プロットを作成します。 PROC ステートメントを使用してプロットを作成します。FORMCHAR オプションを設定します。FORMCHAR オプションをこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。最初のプロット要求では、縦軸に High、横軸に Year がプロットされ、プロット記号としてアスタリスクが指定されます。2 番目のプロット要求では、縦軸に Low、横軸に Year がプロットされ、プロット記号として'o'が指定されます。OVERLAY は、最初のプロットに 2 番目のプロットを重ね合わせます。BOX は、プロットのまわりを枠で囲みます。OVERLAY と BOX は、両方のプロット要求に適用されます。HAXIS=では、横軸に値 1968 から 2008 までを 10 年の増分で表示するように指定します。VAXIS=では、縦軸の値が増分として 5,000 を示すよう指定されます。

```
proc plot data=djia formchar="|----|+|---+=|-/\<>*";
   plot high*year='*'
        low*year='o' / overlay box
        haxis=by 10
        vaxis=by 5000;
```

タイトルを指定します。

```
   title 'Plot of Highs and Lows';
   title2 'for the Dow Jones Industrial Average';
run;
```

出力

アウトプット 42.6 *異なるプロット記号を使用して重ね合わせた 2 つのプロット*



例 4: ページごとに複数のプロットを作成する

要素: PROC PLOT ステートメントオプション
FORMCHAR
HPERCENT=
VPERCENT=
PLOT ステートメント

データセット: Dija

詳細

この例では、出力の 1 ページに 3 つのプロットを配置します。

プログラム

```
options formchar="|----|+|---+=|-/\<>*" pagesize=40 linesize=120;

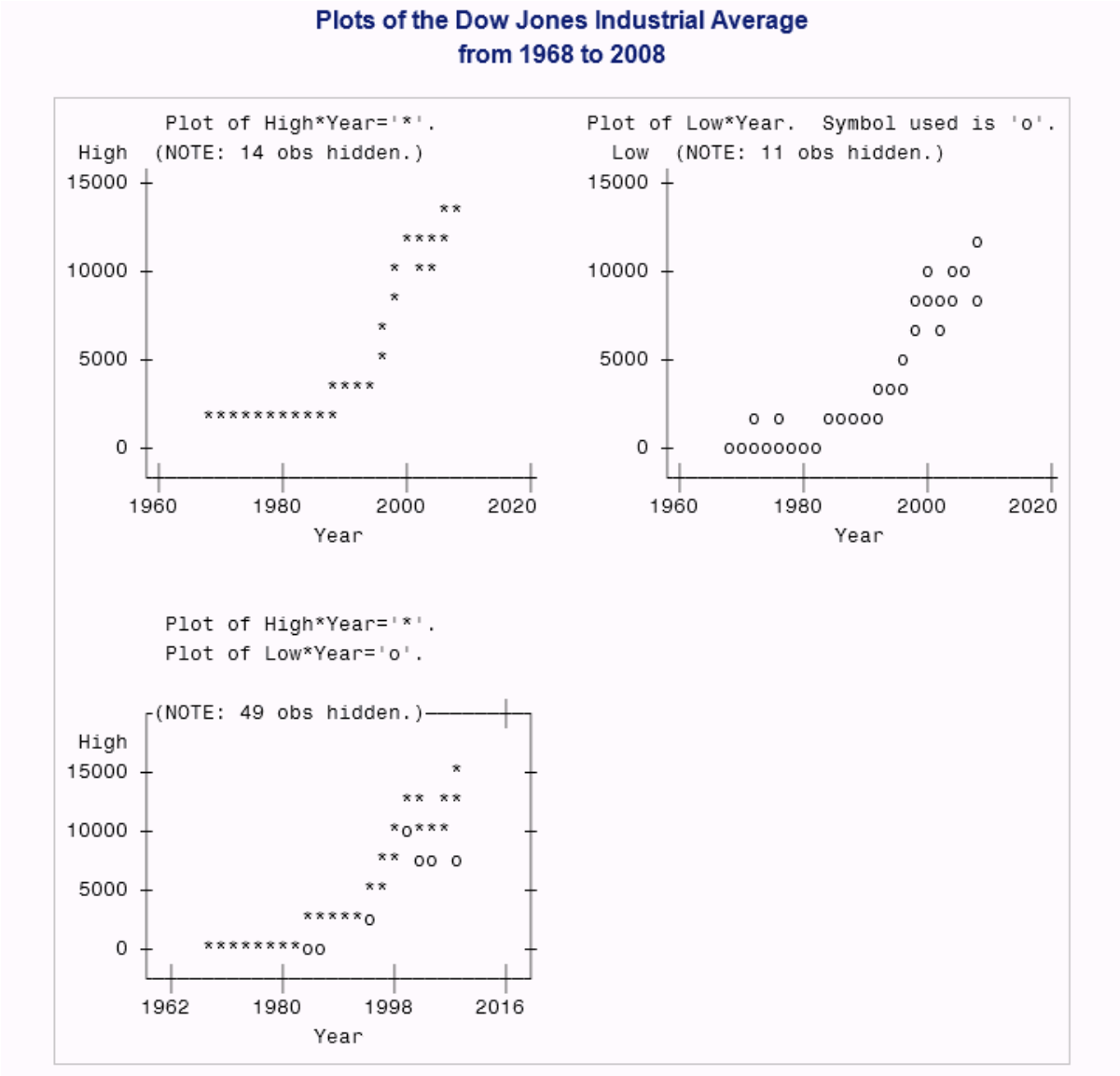
data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
```

```
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
...more data lines...
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;

proc plot data=djia vpercent=50 hpercent=50;

   plot high*year='*';

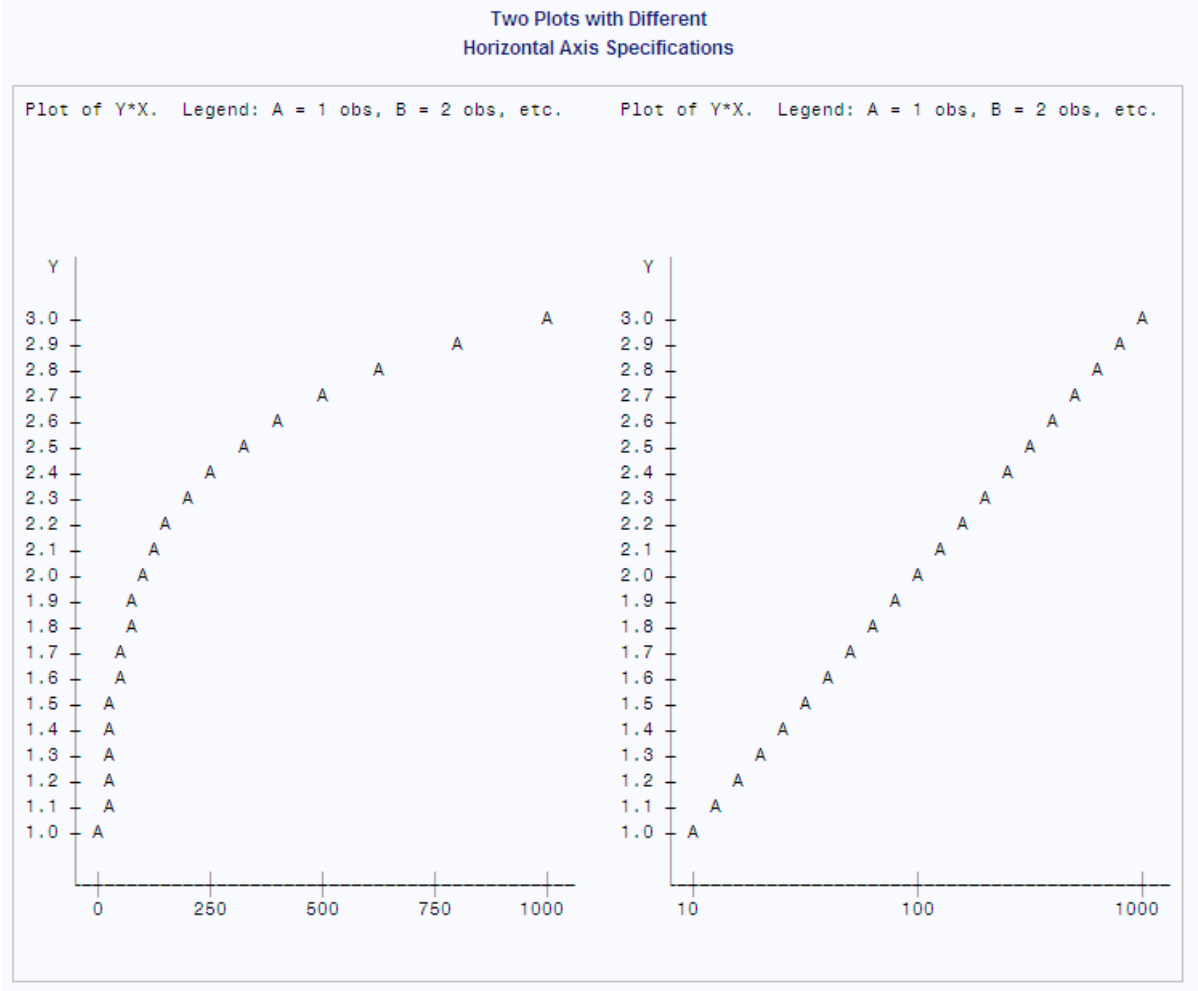
   plot low*year='o';

   plot high*year='*' low*year='o' / overlay box;

   title 'Plots of the Dow Jones Industrial Average';
   title2 'from 1968 to 2008';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。PAGESIZE=オプションで出力行数が 40 に設定され、LINESIZE=オプションで出力ウィンドウの文字数が 120 文字に設定されます。

```
options formchar="|----|+|---+=|-/\<>*" pagesize=40 linesize=120;
```

Dija データセットを作成します。 Dija には、1968 年から 2008 年までのダウジョーンズ工業平均の最高終値と最安終値が含まれます。DATA ステップでこのデータセットが作成されます。

```
data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
...more data lines...
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;
```

プロットサイズを指定します。 VPERCENT=では、各プロットで出力ページの縦のスペースの 50%を使用することが指定されます。HPERCENT=では、各プロットで横のスペースの 50%を使用することが指定されます。

```
proc plot data=djia vpercent=50 hpercent=50;
```

最初のプロットを作成します。 このプロット要求によって、縦軸に High の値、横軸に Year の値がプロットされます。また、アスタリスクがプロット記号として指定されます。

```
   plot high*year='*';
```

---

2 番目のプロットを作成します。 このプロット要求によって、縦軸に Low の値、横軸に Year の値がプロットされます。また、アスタリスクがプロット記号として指定されます。

```
   plot low*year='o';
```

3 番目のプロットを作成します。 最初のプロット要求では、縦軸に High、横軸に Year がプロットされ、プロット記号としてアスタリスクが指定されます。2 番目のプロット要求では、縦軸に Low、横軸に Year がプロットされ、プロット記号として'o 'が指定されます。OVERLAY は、最初のプロットに 2 番目のプロットを重ね合わせます。BOX は、プロットのまわりを枠で囲みます。OVERLAY と BOX は、両方のプロット要求に適用されます。

```
      plot high*year='*' low*year='o' / overlay box;
```

タイトルを指定します。

```
      title 'Plots of the Dow Jones Industrial Average';
      title2 'from 1968 to 2008';
   run;
```

出力

アウトプット 42.7 *1 ページに 3 つのプロット*



例 5: 対数尺度にデータをプロットする

要素: PROC PLOT ステートメントオプション
PLOT ステートメント

PLOT ステートメントオプション
HAXIS=
VSPACE=

他の要素: DATA ステップ

データセット: EQUA

詳細

この例では、DATA ステップを使用して EQUA データセットを生成します。DATA ステップでは、反復 DO ステートメントを使用してデータが作成されます。PROC PLOT ステップでは、同じデータのプロットが 2 つ表示されます。一方のプロットは横軸の指定がなしで、もう一方のプロットは横軸に対数スケールが指定されます。

プログラム

```
data equa;
   do Y=1 to 3 by .1;
      X=10**y;
      output;
   end;
run;

proc plot data=equa hpercent=50;

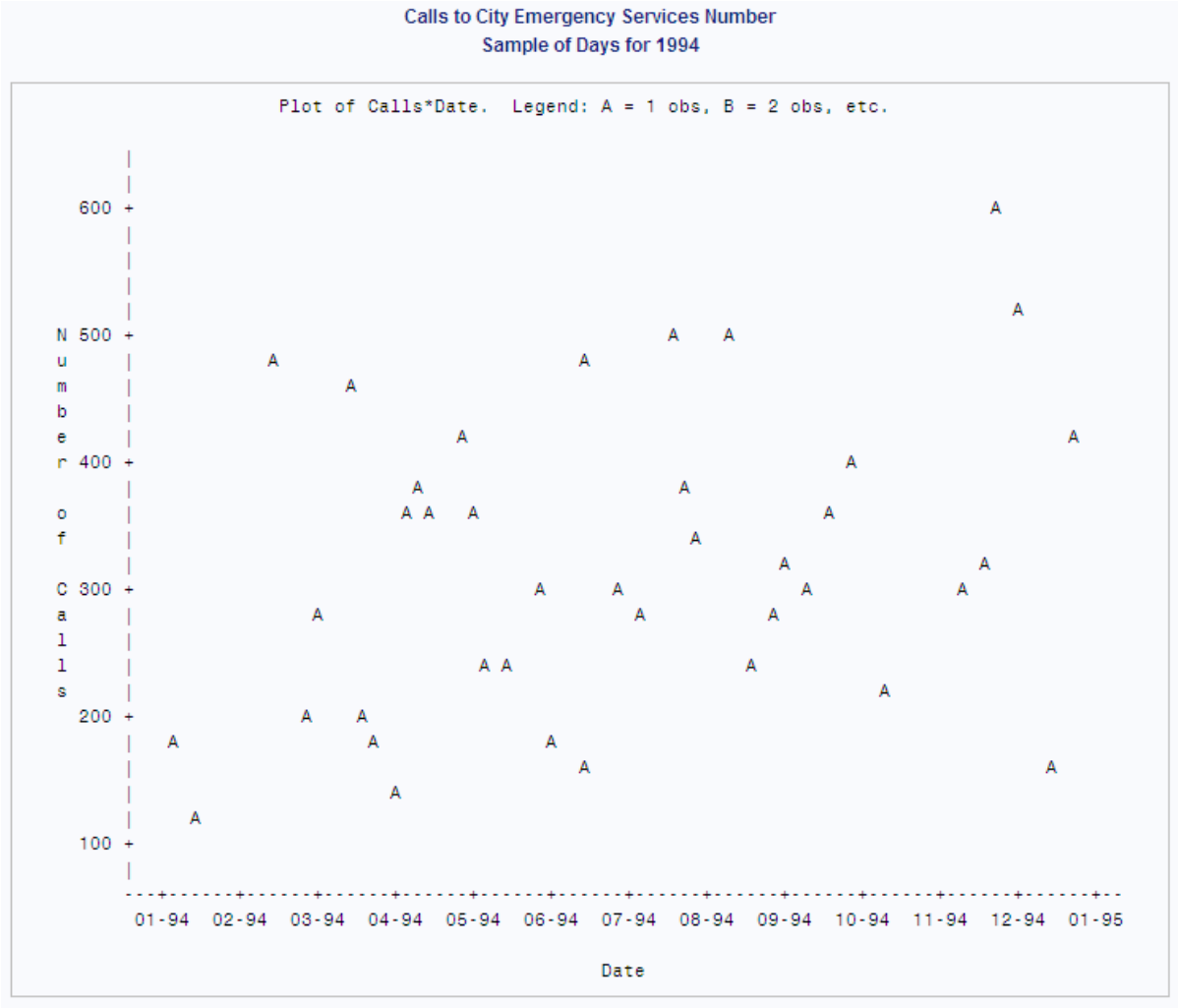
    plot y*x / vspace=1;
    plot y*x / haxis=10 100 1000 vspace=1;

    title 'Two Plots with Different';
    title2 'Horizontal Axis Specifications';
run;
```

プログラムの説明

EQUA データセットを作成します。 EQUA では、変数 Y の値を増分を 0.1 として、1 から最大 3 まで増加させることによって、X と Y の値を作成します。各 X の値は、10^{Y} で算出されます。

```
data equa;
   do Y=1 to 3 by .1;
      X=10**y;
      output;
   end;
run;
```

プロットサイズを指定します。 HPERCENT=では、2 つのプロットを並べる場所を空けるために、各プロットで横のスペースを 50%ずつ使用することが指定されます。

```
proc plot data=equa hpercent=50;
```

プロットを作成します。 PLOT ステートメント要求によって、縦軸に Y、横軸に X がプロットされます。HAXIS=では、2 番目のプロットの横軸に対して対数スケールが指定されます。VSPACE=オプションでは、目盛間の印刷スペースの値が指定されます。

```
plot y*x / vspace=1;
plot y*x / haxis=10 100 1000 vspace=1;
```

タイトルを指定します。

```
title 'Two Plots with Different';
title2 'Horizontal Axis Specifications';
run;
```

出力

アウトプット 42.8 *横軸の指定が異なる 2 つのプロット*



例 6: 軸に日付値をプロットする

要素: PROC PLOT ステートメントオプション
FORMCHAR

PLOT ステートメント

PLOT ステートメントオプション
HAXIS=

他の要素: DATA ステップ

データセット: Emergency_calls

詳細

この例では、DATA ステップを使用して Emergency_calls データセットを作成し、軸に日付値を指定する方法を示します。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

data emergency_calls;
   input Date : date7. Calls @@;
   label calls='Number of Calls';
   datalines;
1APR94 134   11APR94 384  13FEB94 488
2MAR94 289   21MAR94 201  14MAR94 460
3JUN94 184   13JUN94 152  30APR94 356
4JAN94 179   14JAN94 128  16JUN94 480
5APR94 360   15APR94 350  24JUL94 388
6MAY94 245   15DEC94 150  17NOV94 328
7JUL94 280   16MAY94 240  25AUG94 280
8AUG94 494   17JUL94 499  26SEP94 394
9SEP94 309   18AUG94 248  23NOV94 590
19SEP94 356  24FEB94 201  29JUL94 330
10OCT94 222  25MAR94 183  30AUG94 321
11NOV94 294  26APR94 412   2DEC94 511
27MAY94 294  22DEC94 413  28JUN94 309
;

proc plot data=emergency_calls;
   plot calls*date / haxis='1JAN94'd to '1JAN95'd by month vaxis=by 100 vspace=5;

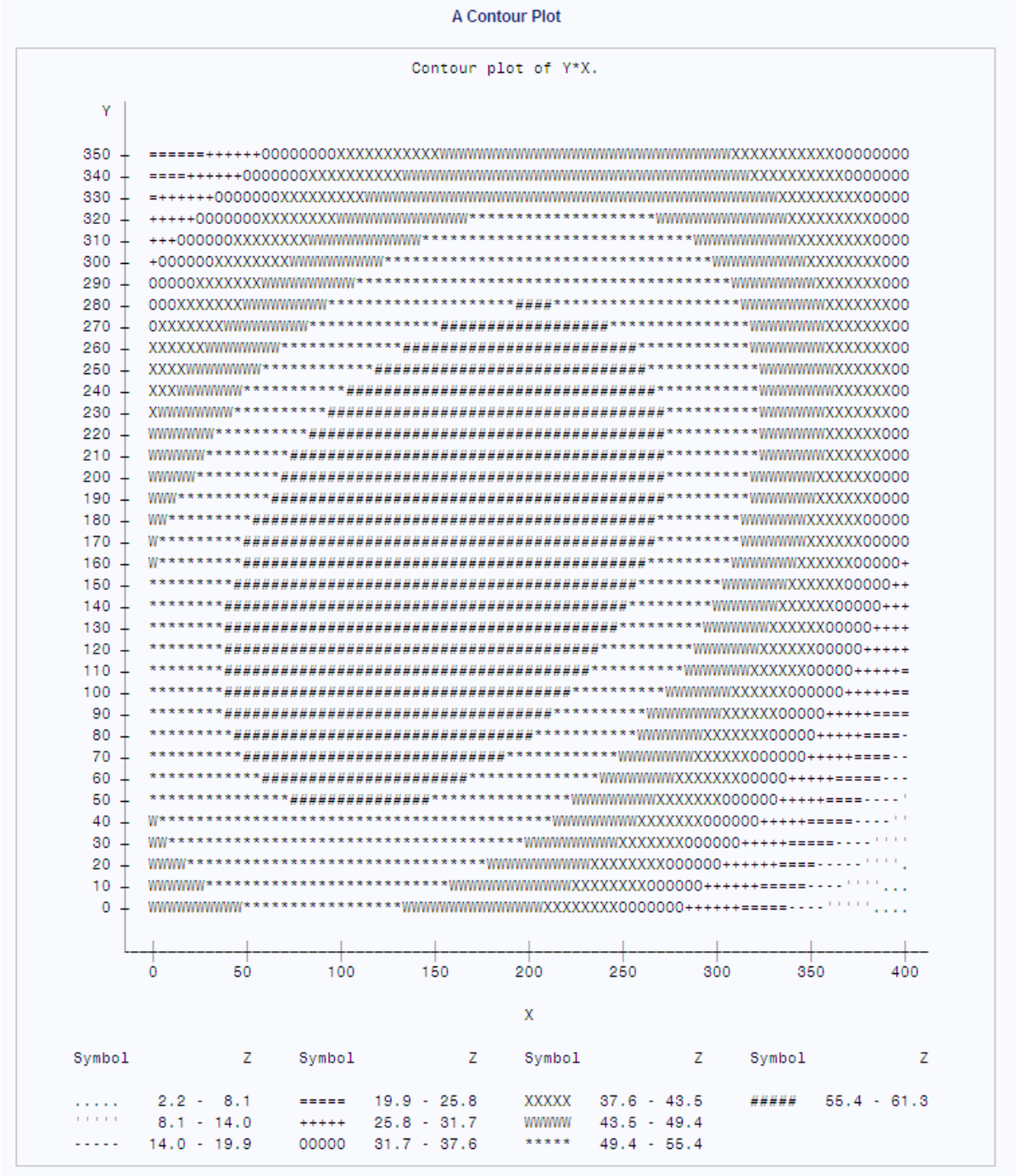
   format date mmyyd5.;

   title 'Calls to City Emergency Services Number';
   title2 'Sample of Days for 1994';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

Emergency_calls データセットを作成します。 Emergency_calls には、各日付の緊急ヘルプラインへの電話回数が含まれます。

```
data emergency_calls;
   input Date : date7. Calls @@;
   label calls='Number of Calls';
```

```
   datalines;
1APR94 134   11APR94 384  13FEB94 488
2MAR94 289   21MAR94 201  14MAR94 460
3JUN94 184   13JUN94 152  30APR94 356
4JAN94 179   14JAN94 128  16JUN94 480
5APR94 360   15APR94 350  24JUL94 388
6MAY94 245   15DEC94 150  17NOV94 328
7JUL94 280   16MAY94 240  25AUG94 280
8AUG94 494   17JUL94 499  26SEP94 394
9SEP94 309   18AUG94 248  23NOV94 590
19SEP94 356  24FEB94 201  29JUL94 330
10OCT94 222  25MAR94 183  30AUG94 321
11NOV94 294  26APR94 412   2DEC94 511
27MAY94 294  22DEC94 413  28JUN94 309
;
```

プロットを作成します。 プロット要求によって、縦軸に Calls、横軸に Date がプロットされます。HAXIS=では、横軸に月単位の時間を使用します。**'1JAN94'd** の表記は日付定数です。値**'1JAN95'd** を指定すると、確実に軸上で 12 月のオブザベーションの場所が十分に取れます。

```
proc plot data=emergency_calls;
   plot calls*date / haxis='1JAN94'd to '1JAN95'd by month vaxis=by 100 vspace=5;
```

DATE 値をフォーマットします。 FORMAT ステートメントでは、MMYYD5.出力形式が Date に割り当てられます。この出力形式では、ハイフンで区切られた 2 桁の月と 2 桁の年の値が使用されます。

```
   format date mmyyd5.;
```

タイトルを指定します。

```
   title 'Calls to City Emergency Services Number';
   title2 'Sample of Days for 1994';
run;
```

出力

アウトプット 42.9 *横軸に日付値が示されるプロット*



例 7: 等高線グラフの作成

要素: PROC PLOT ステートメントオプション
FORMCHAR
PLOT ステートメント
PLOT ステートメントオプション
CONTOUR=

他の要素: DATA ステップ
PROC PRINT
OBS=データセットオプション
NOOBS システムオプション

データセット: CONTOURS

詳細

この例では、DATA ステップを使用して CONTOURS データセットを作成します。ここでは、2 次元のプロットで 3 つの変数の値を表す方法を示します。そのためには、変数の 1 つを CONTOUR 変数として設定します。変数 X と Y が軸上に表示され、Z が等高線変数です。プログラムステートメントによってプロットのオブザベーションが生成され、次の式では等高線の表面について記述されます。

$$z = 46.2 + .09x - .0005x^2 + .1y - .0005y^2 + .0004xy$$

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

data contours;
   format Z 5.1;
   do X=0 to 400 by 5;
      do Y=0 to 350 by 10;
         z=46.2+.09*x-.0005*x**2+.1*y-.0005*y**2+.0004*x*y;
         output;
      end;
   end;
run;

proc print data=contours(obs=5) noobs;
   title 'CONTOURS Data Set';
   title2 'First 5 Observations Only';
run;

proc plot data=contours;
plot y*x=z / contour=10;

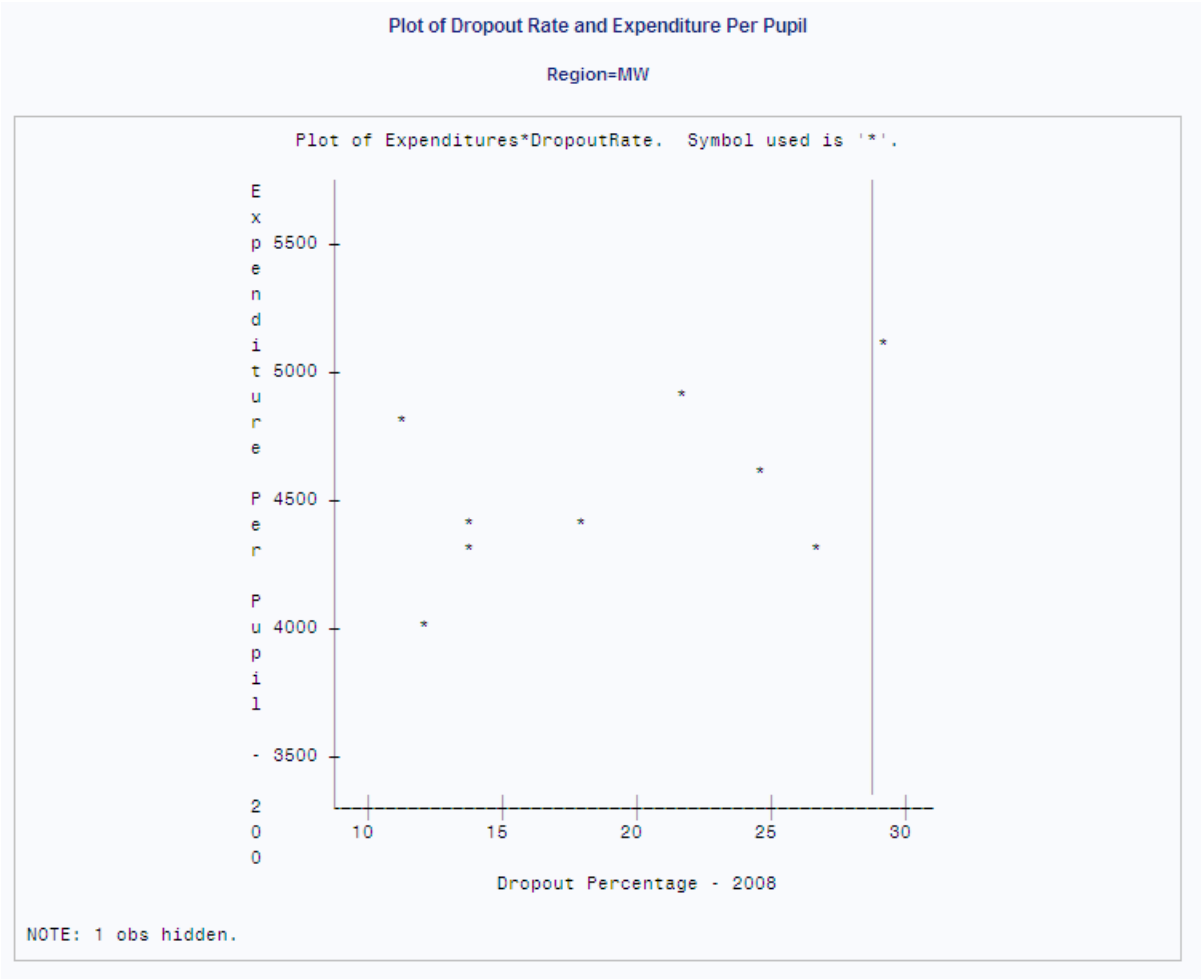
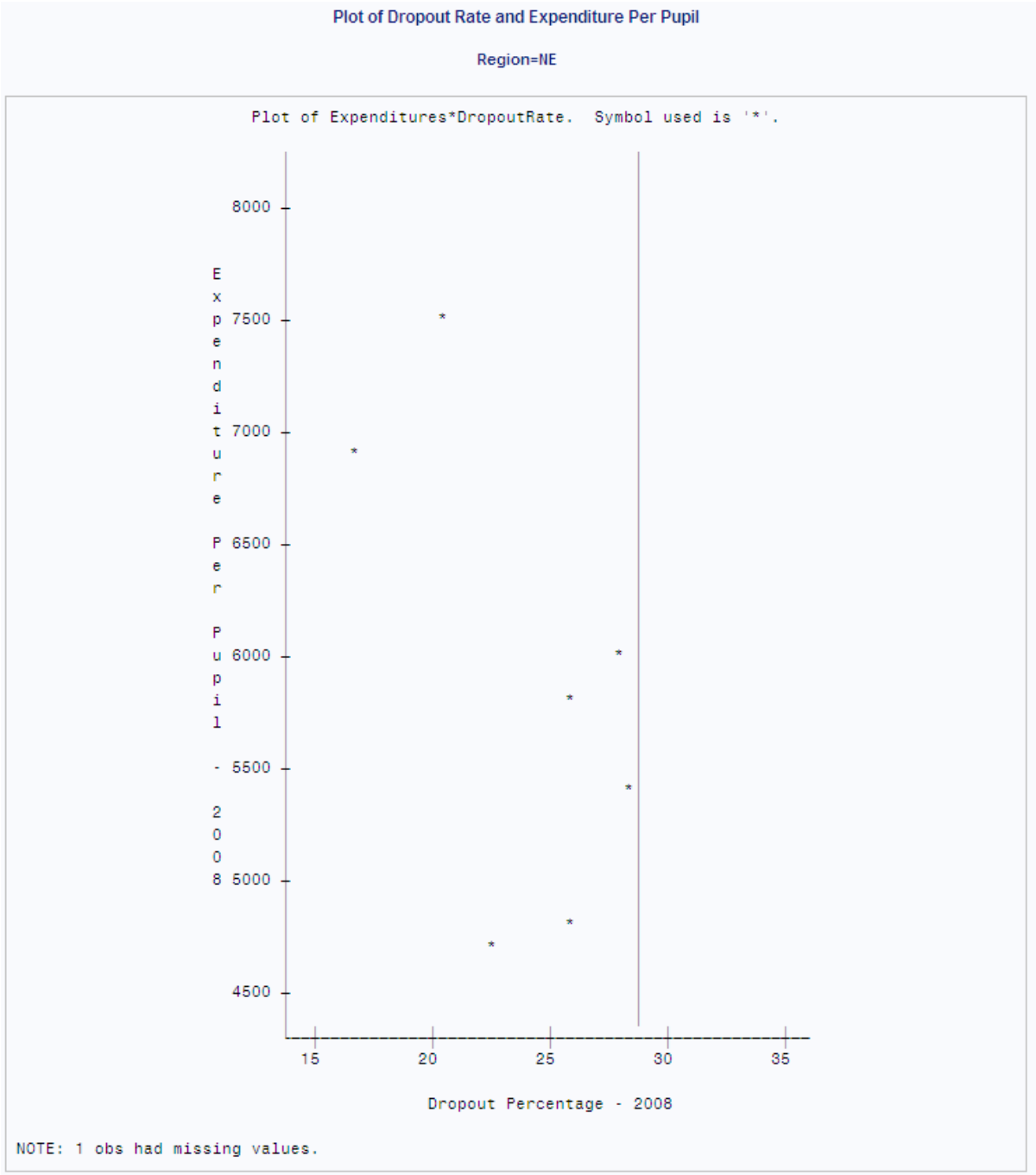
    title 'A Contour Plot';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

CONTOURS データセットを作成します。 CONTOURS データセットに含まれるオブザベーションの X 値は 5 刻みで 0 から 400 までで、Y 値は 10 刻みで 0 から 350 までです。

```
data contours;
   format Z 5.1;
   do X=0 to 400 by 5;
      do Y=0 to 350 by 10;
         z=46.2+.09*x-.0005*x**2+.1*y-.0005*y**2+.0004*x*y;
         output;
      end;
   end;
run;
```

CONTOURS データセットを印刷します。 OBS=データセットオプションは、印刷を最初の 5 つのオブザベーションのみに制限します。NOOBS は、オブザベーション番号の印刷を抑制します。

```
proc print data=contours(obs=5) noobs;
   title 'CONTOURS Data Set';
   title2 'First 5 Observations Only';
run;
```

プロットを作成します。 PLOT ステートメントでは、縦軸に Y、横軸に X がプロットされ、Z が等高線変数として指定されます。CONTOUR=10 を指定すると、プロットでは、Z 値が 10 の増分に分割され、各増分に異なるプロット記号が割り当てられます。

```
proc plot data=contours;
plot y*x=z / contour=10;
```

タイトルを指定します。

```
    title 'A Contour Plot';
run;
```

出力

Z 値に関連付けられた網かけがプロットの下部に表示されます。プロット記号#は、高い Z 値が発生する場所を示しています。

アウトプット 42.10 *CONTOURS データセットの最初の 5 つのオブザベーション*

CONTOURS Data Set
First 5 Observations Only

Z	X	Y
46.2	0	0
47.2	0	10
48.0	0	20
48.8	0	30
49.4	0	40

アウトプット 42.11 *Y*X の等高線グラフ*



例 8: BY グループのプロット

要素: PROC PLOT ステートメントオプション
FORMCHAR
BY ステートメント
PLOT ステートメント
PLOT ステートメントオプション

HAXIS=
HREF=
HSPACE=
VAXIS=
VSPACE=

他の要素: PROC SORT
DATA ステップ

データセット: Education

詳細

この例では、“EDUCATION” (2150 ページ) データセットを使用して、PROC PLOT での BY グループ処理を示します。

プログラム

```
options formchar=|----|+|---+=|-/\<>*";

data education;
   input State $14. +1 Code $ DropoutRate Expenditures MathScore
         Region $;
   label dropout='Dropout Percentage - 1989'
          expend='Expenditure Per Pupil - 1989'
            math='8th Grade Math Exam - 1990';
   datalines;
Alabama        AL 22.3 3197 252 SE
Alaska         AK 35.8 7716 .   W
...more data lines...
New York       NY 35.0 .    261 NE
North Carolina NC 31.2 3874 250 SE
North Dakota   ND 12.1 3952 281 MW
Ohio           OH 24.4 4649 264 MW
;

proc sort data=education;
   by region;
run;

proc plot data=education;
   by region;

    plot expenditures*dropoutrate='*' / href=28.6
         vaxis=by 500 vspace=5
         haxis=by 5 hspace=12;

    title 'Plot of Dropout Rate and Expenditure Per Pupil';
 run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar=|----|+|---+=|-/\<>*";
```

Education データセットを作成します。 “EDUCATION” (2150 ページ) には、米国の一部の州に関する教育データ(出典:米国教育省)が含まれています。DropoutRate は、高校の中退率です。Expenditures は、州が各生徒に使ったドル金額です。MathScore は、標準数学テストでの 8 年生のスコアです。ただし、すべての州が数学テストに参加したわけではありません。

```
data education;
   input State $14. +1 Code $ DropoutRate Expenditures MathScore
         Region $;
   label dropout='Dropout Percentage - 1989'
          expend='Expenditure Per Pupil - 1989'
            math='8th Grade Math Exam - 1990';
   datalines;
Alabama        AL 22.3 3197 252 SE
Alaska         AK 35.8 7716 .   W
...more data lines...
New York       NY 35.0 .    261 NE
North Carolina NC 31.2 3874 250 SE
North Dakota   ND 12.1 3952 281 MW
Ohio           OH 24.4 4649 264 MW
;
```

Education データセットを並べ替えます。 PROC SORT は、EDUCATION を Region 順に並べ替えて、PROC PLOT で Region を BY 変数として使用できるようにします。

```
proc sort data=education;
   by region;
run;
```

BY グループごとに別々のプロットを作成します。 BY ステートメントは、Region の各値に対し別のプロットを作成します。

```
proc plot data=education;
   by region;
```

参照線のあるプロットを作成します。 PLOT ステートメントでは、縦軸に Expenditures、横軸に DropoutRate がプロットされ、プロット記号としてアスタリスクが指定されます。HREF=では、横軸の 28.6 を起点とする参照線が引かれます。参照線は全国平均を表します。縦軸と横軸に目盛を設定するには、VAXIS と HAXIS が使用されます。VSPACE=オプションでは、縦軸の目盛間の印刷スペースの値が指定されます。

```
   plot expenditures*dropoutrate='*' / href=28.6
        vaxis=by 500 vspace=5
        haxis=by 5 hspace=12;
```

タイトルを指定します。

```
   title 'Plot of Dropout Rate and Expenditure Per Pupil';
run;
```

出力

PROC PLOT は、BY グループごとにプロットを作成します。中西部と北東部のプロットのみを示します。

アウトプット 42.12 BY グループ別プロット(中西部地域)



アウトプット 42.13 *BY グループ別プロット(北東部地域)*



例 9: プロットにラベルを追加する

要素: PROC PLOT ステートメントオプション
FORMCHAR
BY ステートメント
PLOT ステートメント

他の要素: PROC SORT

データセット: Education

詳細

この例では、データセットで変数を使用してプロット上のポイントにラベルを設定する方法を示します。この例では、“例 8: BY グループのプロット” (1268 ページ) の例からの出力にラベルを追加します。PROC SORT はまず Region ごとのデータセットの並べ替えに使用されます。これで、最初の PLOT ステートメントで Region を BY 変数として使用できます。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

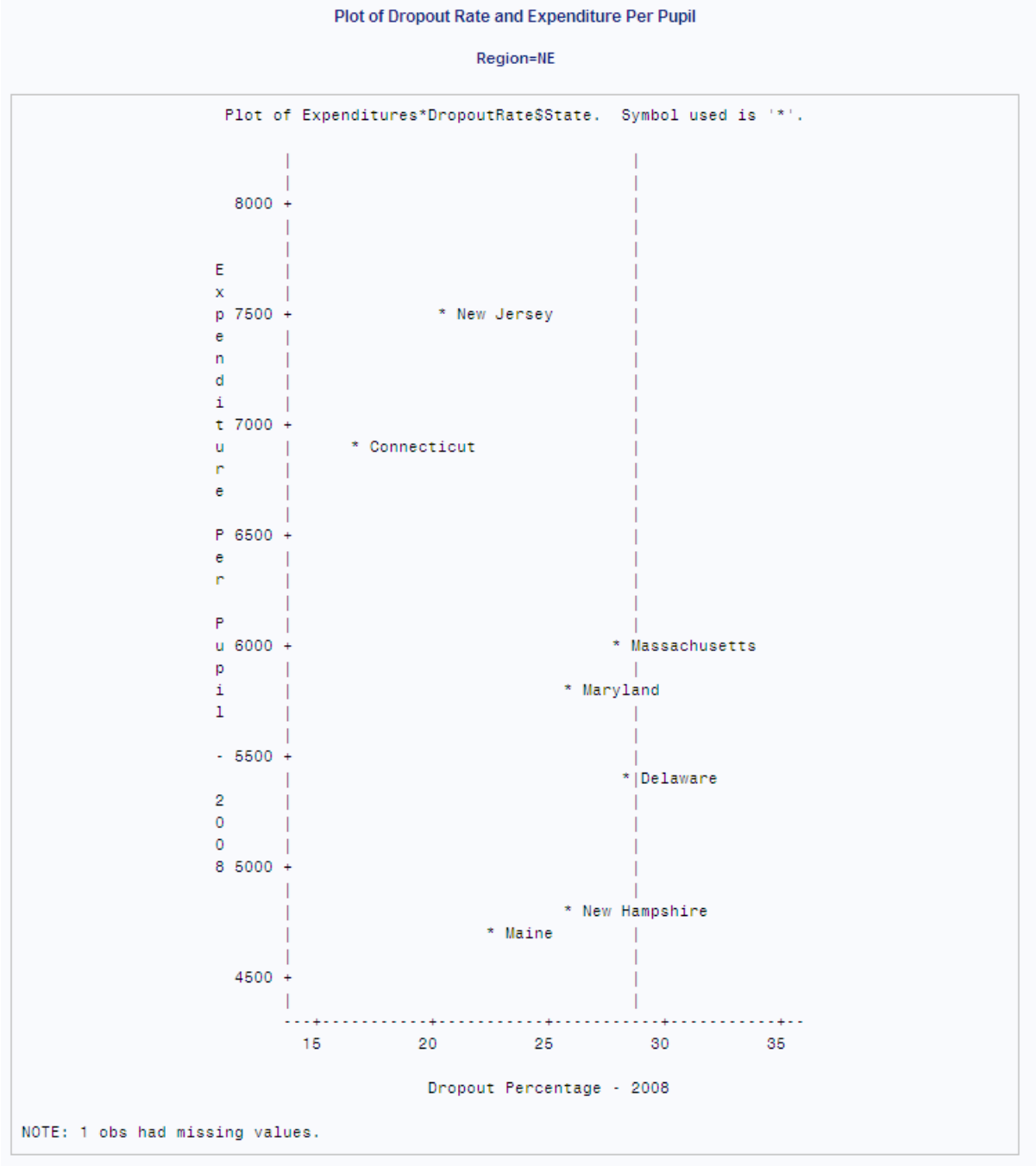
proc sort data=education;
   by region;
run;

proc plot data=education;
   by region;

   plot expenditures*dropoutrate='*' $ state / href=28.6
        vaxis=by 500 vspace=5
        haxis=by 5 hspace=12;

   title 'Plot of Dropout Rate and Expenditure Per Pupil';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

Education データセットを並べ替えます。 PROC SORT は、EDUCATION を Region 順に並べ替えて、PROC PLOT で Region を BY 変数として使用できるようにします。

```
proc sort data=education;
   by region;
run;
```

BY グループごとに別々のプロットを作成します。 BY ステートメントは、Region の各値に対し別のプロットを作成します。

```
proc plot data=education;
   by region;
```

参照線のあるプロットと各データポイントのラベルを作成します。 プロット要求では、縦軸に Expenditures、横軸に DropoutRate がプロットされ、プロット記号としてアスタリスクが指定されます。PLOT ステートメントでラベル変数を指定すると(`$ state`)、プロット上の各ポイントに対応する州名のラベルが付けられます。HREF=では、横軸の 28.6 を起点とする参照線が引かれます。参照線は全国平均を表します。縦軸と横軸に目盛を設定するには、VAXIS と HAXIS が使用されます。HSPACE=12 のオプションは、横軸の目盛間に 12 の印刷スペースを指定します。

```
   plot expenditures*dropoutrate='*' $ state / href=28.6
```

```
          vaxis=by 500 vspace=5
          haxis=by 5 hspace=12;
```

タイトルを指定します。

```
     title 'Plot of Dropout Rate and Expenditure Per Pupil';
run;
```

出力

PROC PLOT は、BY グループごとにプロットを作成します。**Midwest** と **Northeast** のプロットのみを示します。

アウトプット 42.14 *ラベル付きプロット(中西部地域)*

Plot of Dropout Rate and Expenditure Per Pupil

Region=MW

```
 Plot of Expenditures*DropoutRate$State.  Symbol used is '*'.

E       |                                                  |
x       |                                                  |
p 5500 +                                                  |
e       |                                                  |
n       |                                                  |
d       |                                                  |
i       |                                        Michigan|*
t 5000 +                                                  |
u       |                                * Illinois        |
r       |     * Minnesota                                  |
e       |                                                  |
        |                                       * Ohio     |
P 4500 +                                                  |
e       |  Nebraska *          * Kansas                    |
r       |      Iowa * Indiana                      * Missouri
        |                                                  |
P       |                                                  |
u 4000 +       * North Dakota                             |
p       |                                                  |
i       |                                                  |
l       |                                                  |
        |                                                  |
- 3500 +                                                  |
        |                                                  |
2       ---+-----------+-----------+-----------+-----------+--
0         10          15          20          25          30
0
                        Dropout Percentage - 2008

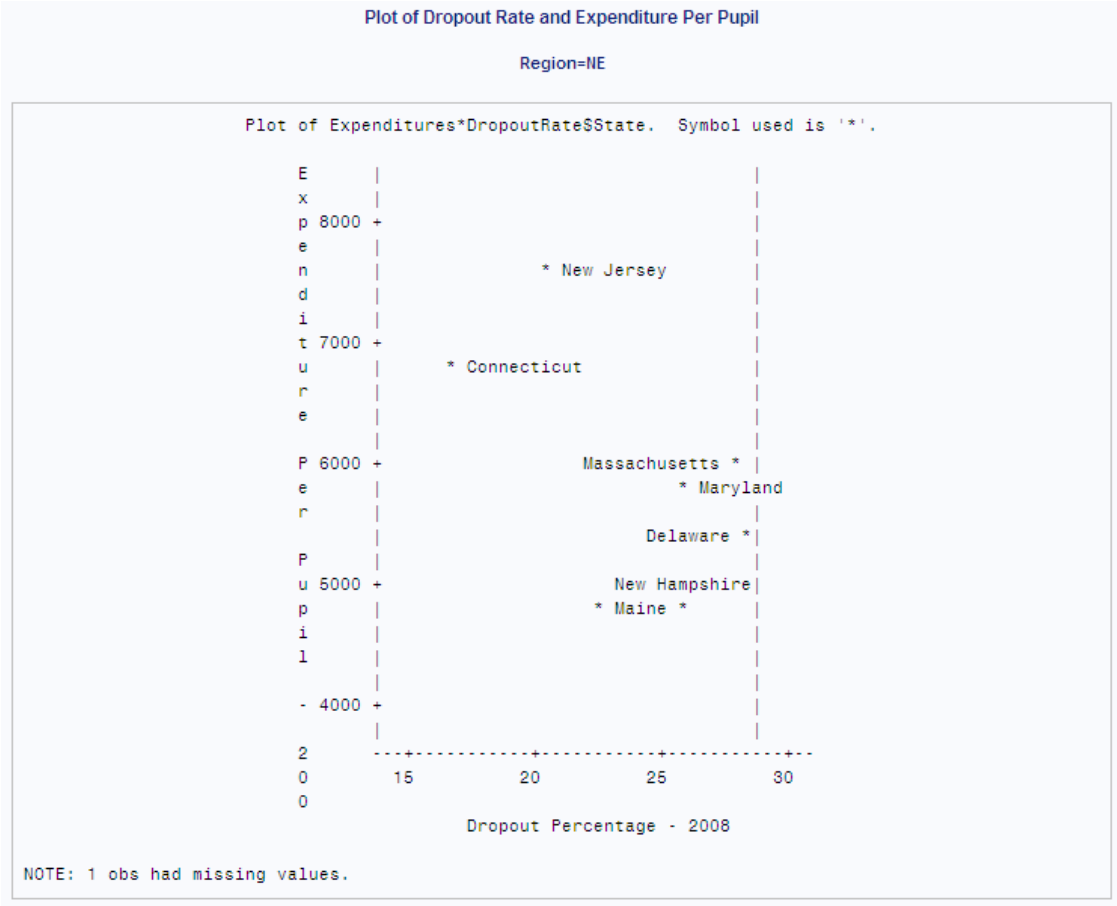
NOTE: 1 obs hidden.
```

アウトプット 42.15 ラベル付きプロット(北東部地域)



例 10: 欠損値を含むオブザベーションを除外する

要素: PROC PLOT ステートメントオプション
FORMCHAR
NOMISS

BY ステートメント

PLOT ステートメント

PLOT ステートメントオプション

HAXIS=
HREF=
HSPACE=
VAXIS=
VSPACE=

他の要素: PROC SORT
WHERE statement

データセット: Education

詳細

この例では、欠損値が軸の計算に及ぼす影響について示します。この例では、“EDUCATION” (2150 ページ) データセットを使用しています。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

proc sort data=education;
   by region;
run;

proc plot data=education nomiss;

    by region;

    plot expenditures*dropoutrate='*' $ state / href=28.6
         vaxis=by 500 vspace=5
         haxis=by 5 hspace=12;

    title 'Plot of Dropout Rate and Expenditure Per Pupil';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

Education データセットを並べ替えます。 PROC SORT は、EDUCATION を Region 順に並べ替えて、PROC PLOT で Region を BY 変数として使用できるようにします。

```
proc sort data=education;
   by region;
run;
```

欠損値のあるデータポイントを除外します。 NOMISS は、軸変数のどちらかに欠損値があるオブザベーションを除外します。

```
proc plot data=education nomiss;
```

BY グループごとに別々のプロットを作成します。 BY ステートメントは、Region の各値に対し別のプロットを作成します。

```
    by region;
```

参照線のあるプロットと各データポイントのラベルを作成します。 プロット要求では、縦軸に Expenditures、横軸に DropoutRate がプロットされ、プロット記号としてアスタリスクが指定されます。PLOT ステートメントでラベル変数を指定すると(**$ state**)、プロット上の各ポイントに対応する州名のラベルが付けられます。HREF=では、横軸の 28.6 を起点とする参照線が引かれます。参照線は全国平均を表します。縦軸と横軸に目盛を設定するには、VAXIS と HAXIS が使用されます。VSPACE=5 オプションは、縦軸の目盛間に 5 のスペースを指定し、HSPACE=12 は横軸の目盛間に 12 のスペースを指定します。

```
      plot expenditures*dropoutrate='*' $ state / href=28.6
           vaxis=by 500 vspace=5
           haxis=by 5 hspace=12;
```

タイトルを指定します。

```
      title 'Plot of Dropout Rate and Expenditure Per Pupil';
run;
```

出力

PROC PLOT は、BY グループごとにプロットを作成します。**Northeast** のプロットのみを示します。**New York** の Expenditures に欠損値があるため、オブザベーションが除外され、PROC PLOT は横軸の計算に DropoutRate の値 35 を使用しません。この

出力の横軸と、“例 9: プロットにラベルを追加する” (1272 ページ)の **Northeast** のプロットの横軸を比較してください。

アウトプット 42.16 *欠損値を除外したプロット*



例 11: PLACEMENT=オプションを使用し、プロット上のラベルを調整する

要素: PROC PLOT ステートメントオプション
- FORMCHAR

PLOT ステートメント

PLOT ステートメントオプション
- BOX=
- LIST=
- PLACEMENT=
- HAXIS=
- HSPACE=
- VAXIS=
- VSPACE=

他の要素: DATA ステップ

RUN グループ処理

データセット: Census

詳細

この例では、ラベルのデフォルト配置と、密集したプロット上でのラベル配置の調整方法について説明します。ラベルはデータセット“CENSUS” (2120 ページ)の変数の値です。([1]

この例には、PROC PLOT の RUN グループ処理も含まれています。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

data census;
   input Density CrimeRate State $ 14-27 PostalCode $ 29-30;
   datalines;
263.3 4575.3 Ohio           OH
62.1 7017.1  Washington     WA

...more data lines...

111.6 4665.6 Tennessee      TN
120.4 4649.9 North Carolina NC
;


proc plot data=census;
   plot density*crimerate=state $ state /

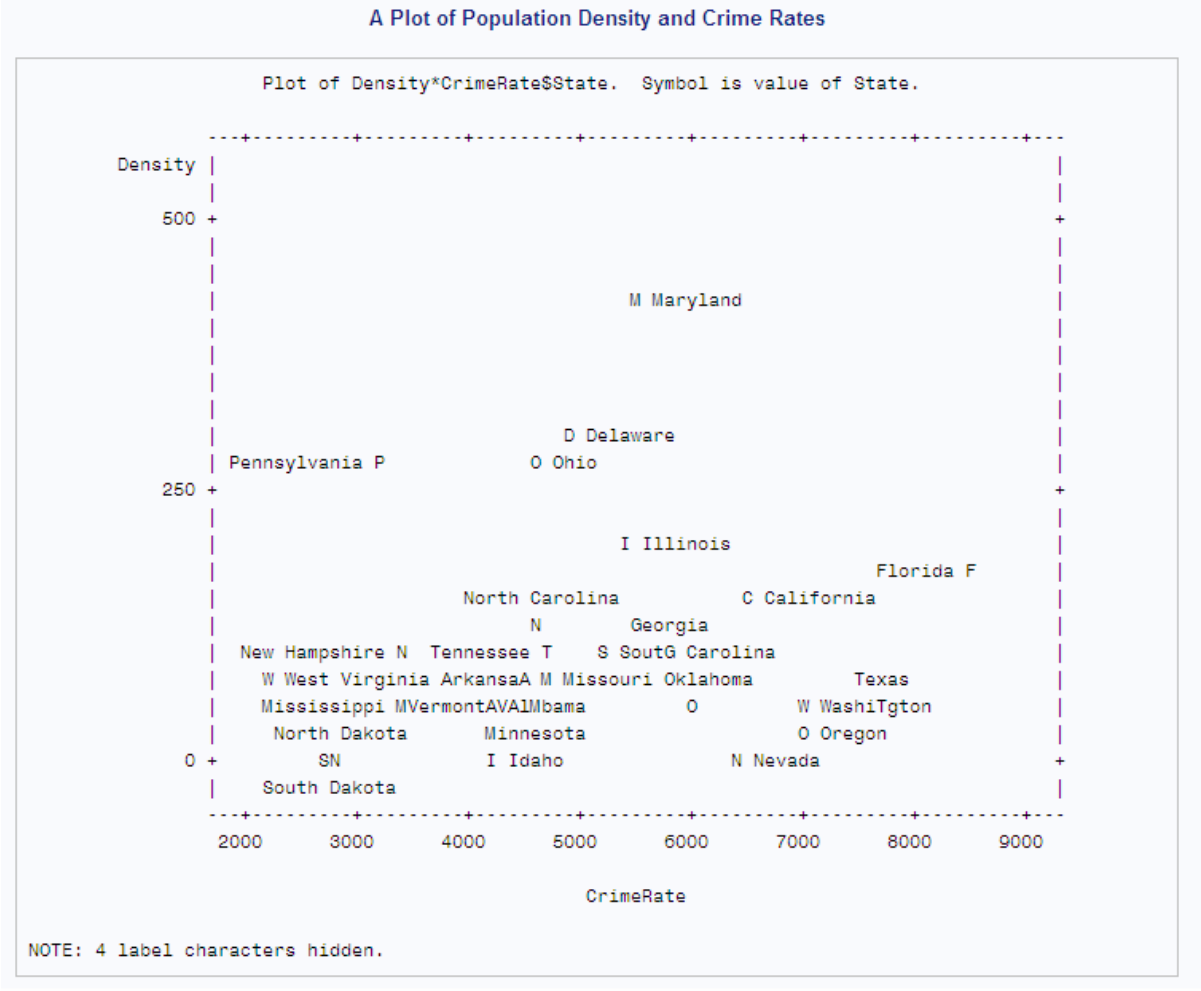
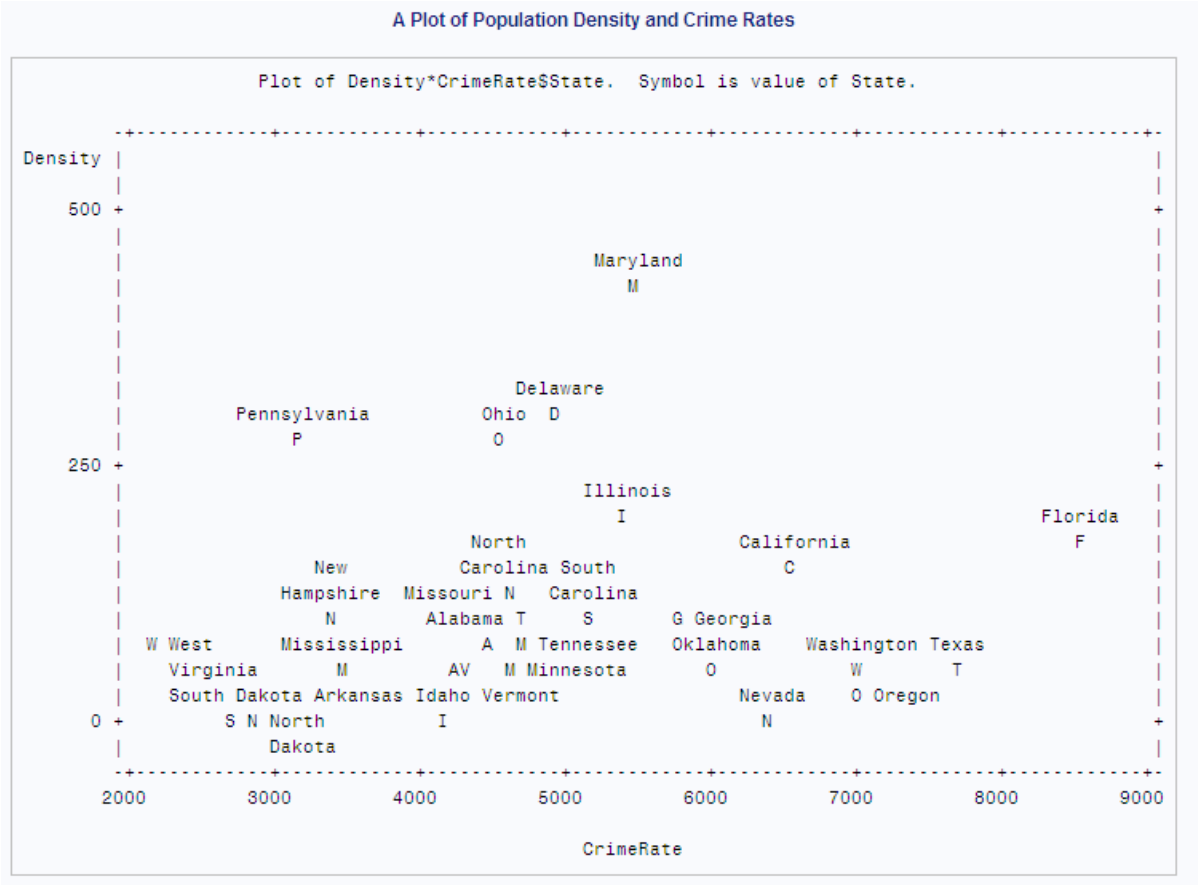
   box
   list=1
   haxis=by 1000
   vaxis=by 250
   vspace=10
   hspace=10;

    plot density*crimerate=state $ state /
              box
              list=1
              haxis=by 1000
              vaxis=by 250
              vspace=10

          placement=((v=2 1 : l=2 1)
                ((l=2 2 1 : v=0 1 0) * (s=right left : h=2 -2))
                (s=center right left * l=2 1 * v=0 1 -1 2 *
                 h=0 1 to 5 by alt));

   title 'A Plot of Population Density and Crime Rates';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

1 出典:米国国税調査局および 1987 年統一経済白書(FBI))

Census データセットを作成します。 Census には、選択した州の変数 CrimeRate および Density が含まれます。CrimeRate は、100,000 人当たりの犯罪数です。Density は、1980 年の国税調査における 1 平方マイル当たりの人口密度です。DATA ステップ “CENSUS” (2120 ページ) は、次のデータセットを作成します。

```
data census;
   input Density CrimeRate State $ 14-27 PostalCode $ 29-30;
   datalines;
263.3 4575.3 Ohio           OH
62.1 7017.1  Washington     WA

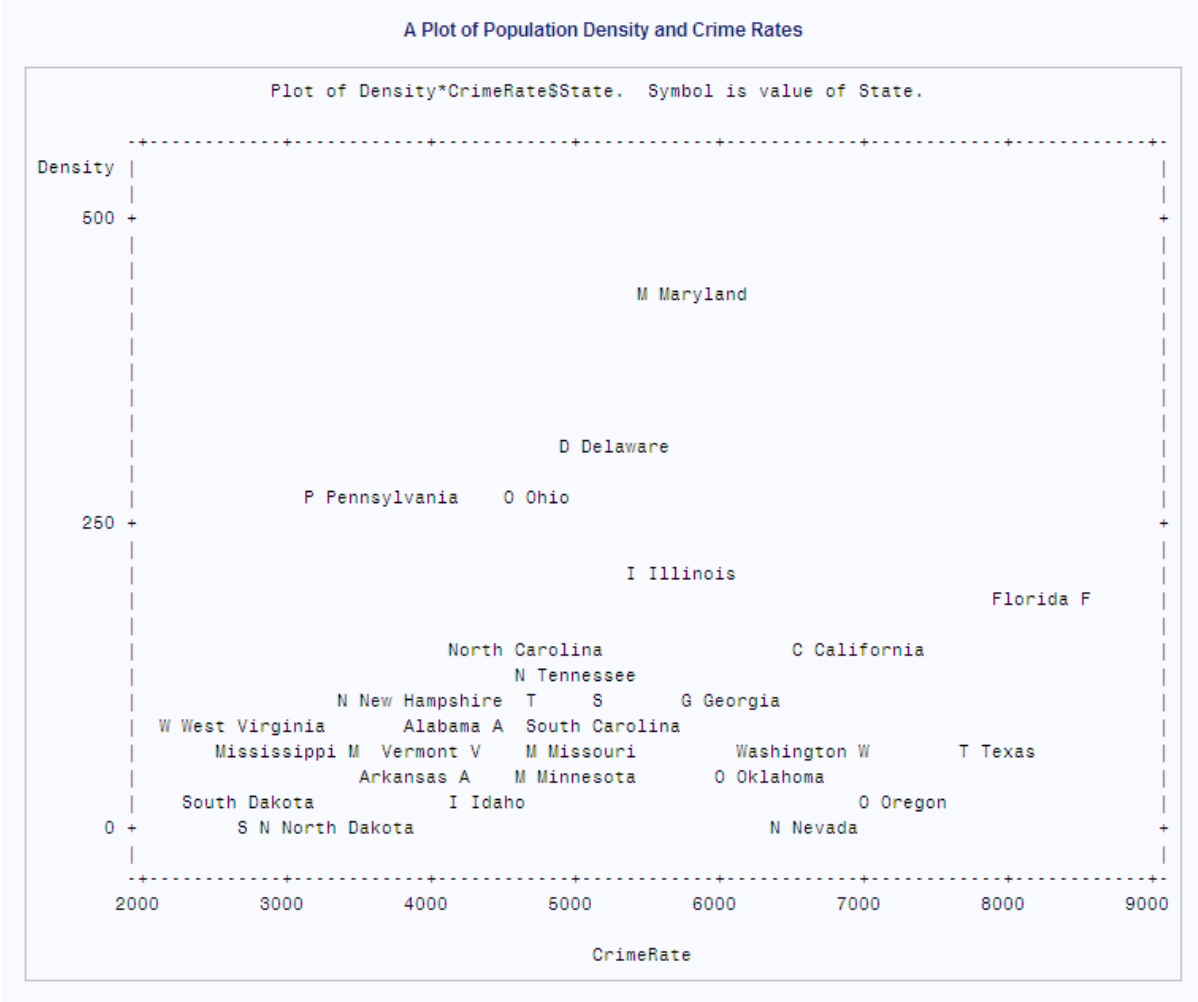
...more data lines...

111.6 4665.6 Tennessee      TN
120.4 4649.9 North Carolina NC
;
```

各データポイントにラベルが付いたプロットを作成します。 プロット要求では、縦軸に Density、横軸に CrimeRate をプロットし、State の 1 字目をプロット記号として使用します。これにより、記号とラベルを簡単に結び付けられるようになります。PLOT ステートメントでラベル変数を指定すると(**$ state**)、各ポイントに対応する州名のラベルが付けられます。

```
proc plot data=census;
   plot density*crimerate=state $ state /
```

プロットオプションを指定します。 BOX は、プロットのまわりを枠で囲みます。LIST=は、ペナルティが 1 以上のラベルをリストします。HAXIS=と VAXIS=は、増分のみを指定します。PROC PLOT は、データを使用して軸の範囲を決定します。VSPACE=10 オプションは、縦軸の目盛間に 10 のスペースを指定し、HSPACE=10 は横軸の目盛間に 10 のスペースを指定します。

```
box
list=1
haxis=by 1000
vaxis=by 250
vspace=10
hspace=10;
```

2 番目のプロットを要求します。 PROC PLOT は対話型なので、プログラムのこの時点でもなおプロシジャは実行中です。プロット要求をもう 1 つサブミットするためにプロシジャを再開始する必要はありません。LIST=1 では、1 以上のペナルティがないため出力は作成されません。

```
 plot density*crimerate=state $ state /
           box
           list=1
           haxis=by 1000
           vaxis=by 250
           vspace=10
```

配置オプションを指定します。 PLACEMENT=によって、PROC PLOT でラベルの配置により多くの配置状態を使用できるようになります。PLACEMENT=には 3 つの式が含まれています。最初の式では、ラベルの優先位置が指定されます。最初の式では、1 行か 2 行のラベルを付け、プロット記号の上で中央揃えの配置状態にすることが決定さ

れます。2 番目と 3 番目の式では、PROC PLOT でプロット記号のまわりの複数の位置にラベルを配置できる配置状態が決定されます。

```
placement=((v=2 1 : l=2 1)
      ((l=2 2 1 : v=0 1 0) * (s=right left : h=2 -2))
      (s=center right left * l=2 1 * v=0 1 -1 2 *
       h=0 1 to 5 by alt));
```

タイトルを指定します。

```
   title 'A Plot of Population Density and Crime Rates';
run;
```

出力

ラベル **Tennessee**、**South Carolina**、**Arkansas**、**Minnesota**、**South Dakota** にはペナルティがあります。デフォルト配置状態では、PROC PLOT がポイントの近接に対して与えられるペナルティを回避できる可能性は不十分です。4 つのラベル文字が非表示です。

アウトプット 42.17 ペナルティのあるプロット



List of Point Locations, Penalties, and Placement States

Label	Vertical Axis	Horizontal Axis	Penalty	Starting Position	Lines	Vertical Shift	Horizontal Shift
South Carolina	103.40	5161.9	2	Right	1	0	2
Alabama	76.60	4451.4	9	Center	1	-1	2
Arkansas	43.90	4245.2	2	Center	1	1	-1
Vermont	55.20	4271.2	2	Left	1	0	-2
Washington	62.10	7017.1	2	Right	1	0	2

プロットで衝突は発生していません。

アウトプット 42.18 *ラベルの衝突を避けて配置したプロット*



例 12: マクロを使用し、プロット上のラベルを調整する

要素: PROC PLOT ステートメントオプション
FORMCHAR
PLOT ステートメント
PLOT ステートメントオプション
BOX=
LIST=
HAXIS=
VAXIS=
VSPACE=

他の要素: %IF statement
%MACRO statement

データセット: Census

詳細

この例では、ラベルのデフォルト配置と、ラベル配置を調整するマクロの使用法について説明します。ラベルは“CENSUS” (2120 ページ)データセットの変数の値です。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

%macro place(n);
   %if &n > 13 %then %let n = 13;
       placement=(
   %if &n <= 0 %then (s=center); %else (h=2 -2 : s=right left);
   %if &n = 1 %then (v=1 * h=0 -1 to -2 by alt);
   %else %if &n = 2 %then (v=1 -1 * h=0 -1 to -5 by alt);
   %else %if &n > 2 %then (v=1 to 2 by alt * h=0 -1 to -10 by alt);
   %if &n > 3 %then
       (s=center right left * v=0 1 to %eval(&n - 2) by alt *
       h=0 -1 to %eval(-3 * (&n - 2)) by alt *
       l=1 to %eval(2 + (10 * &n - 35) / 30)); )
   %if &n > 4 %then penalty(7)=%eval((3 * &n) / 2);
%mend;

proc plot data=census;
   plot density*crimerate=state $ state /

           box
           list=1
           haxis=by 1000
           vaxis=by 250
           vspace=12
           %place(4);

   title 'A Plot of Population Density and Crime Rates';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

条件付きロジックを使用して配置を決定します。 %PLACE マクロは PLACEMENT=オプションの代わりに使用できます。n の値が高くなるほど、PROC PLOT はラベルを自由に配置できるようになります。

```
%macro place(n);
   %if &n > 13 %then %let n = 13;
       placement=(
   %if &n <= 0 %then (s=center); %else (h=2 -2 : s=right left);
   %if &n = 1 %then (v=1 * h=0 -1 to -2 by alt);
   %else %if &n = 2 %then (v=1 -1 * h=0 -1 to -5 by alt);
   %else %if &n > 2 %then (v=1 to 2 by alt * h=0 -1 to -10 by alt);
   %if &n > 3 %then
       (s=center right left * v=0 1 to %eval(&n - 2) by alt *
       h=0 -1 to %eval(-3 * (&n - 2)) by alt *
       l=1 to %eval(2 + (10 * &n - 35) / 30)); )
   %if &n > 4 %then penalty(7)=%eval((3 * &n) / 2);
%mend;
```

プロットを作成します。 プロット要求では、縦軸に Density、横軸に CrimeRate をプロットし、State の 1 字目をプロット記号として使用します。PLOT ステートメントでラベル変数を指定すると($ state)、各ポイントに対応する州名のラベルが付けられます。

```
proc plot data=census;
   plot density*crimerate=state $ state /
```

プロットオプションを指定します。 BOX は、プロットのまわりを枠で囲みます。LIST=は、ペナルティが 1 以上のラベルをリストします。HAXIS=と VAXIS=は、増分のみを指定します。PROC PLOT は、データを使用して軸の範囲を決定します。VSPACE=12 オプションでは、縦軸の目盛間に 12 のスペースが指定されます。PLACE マクロは、ラベルの配置を決定します。

```
            box
            list=1
            haxis=by 1000
            vaxis=by 250
            vspace=12
            %place(4);
```

タイトルを指定します。

```
   title 'A Plot of Population Density and Crime Rates';
run;
```

出力

アウトプット 42.19 *マクロを使用してラベルを配置したプロット*



例 13: デフォルトのペナルティを変更する

要素: PROC PLOT ステートメントオプション
- FORMCHAR

PLOT ステートメント

PLOT ステートメントオプション
- HAXIS=
- LIST=
- PENALTIES=
- PLACEMENT=
- VAXIS=
- VSPACE=

データセット: Census

詳細

この例では、デフォルトペナルティの変更がラベル配置に及ぼす影響について説明します。目標は、作成したプロットのラベルがポイントの散布を損なわないようにすることです。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

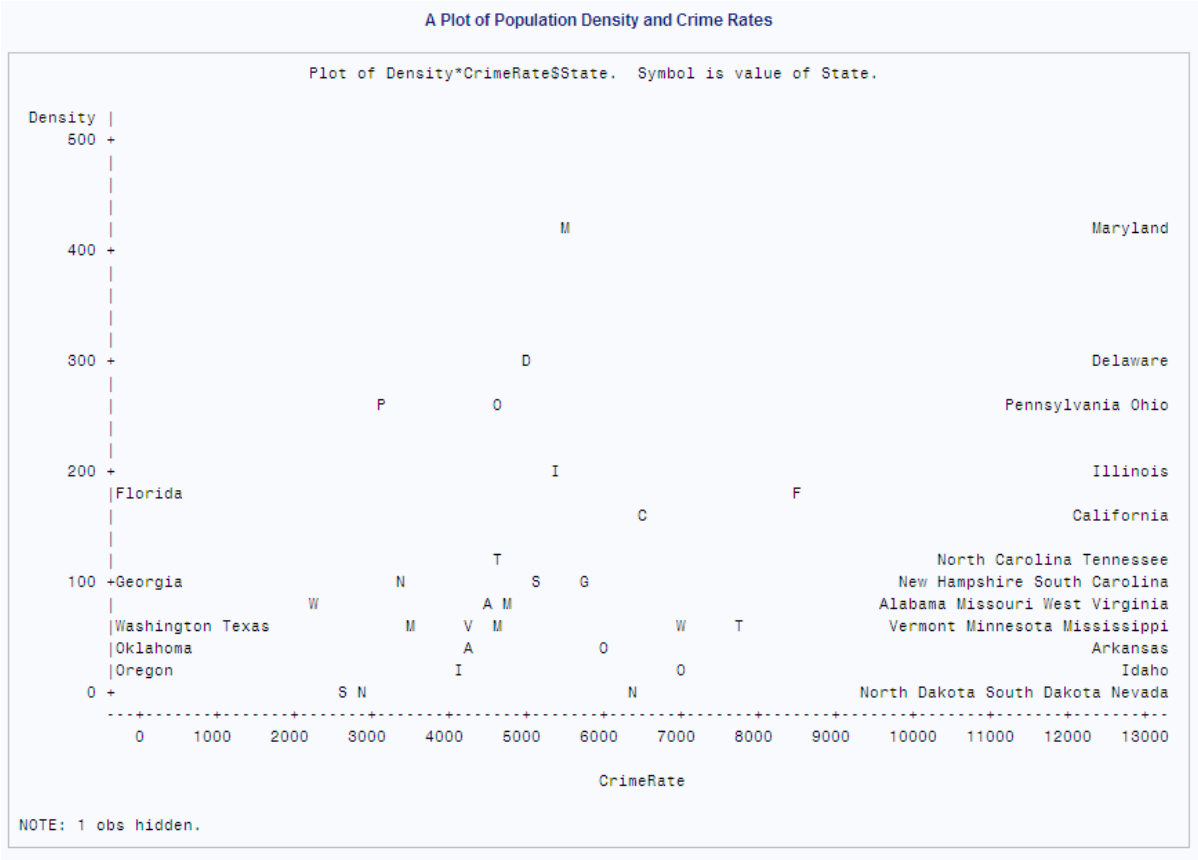
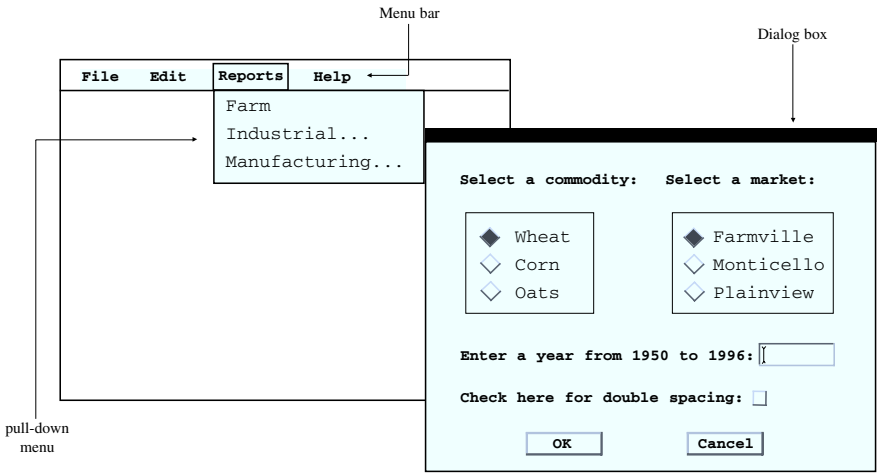
   proc plot data=census;
      plot density*crimerate=state $ state /

           placement=(h=100 to 10 by alt * s=left right)

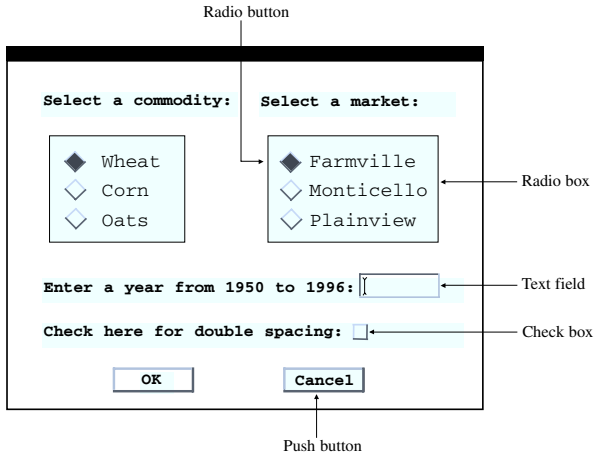
           penalties(4)=500  list=0

           haxis=0 to 13000 by 1000
           vaxis=by 100
           vspace=5;

   title 'A Plot of Population Density and Crime Rates';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";

   proc plot data=census;
      plot density*crimerate=state $ state /
```

配置を指定します。 PLACEMENT=では、優先配置状態として、ポイントと同じ行のポイントの左右 100 列が指定されます。

```
           placement=(h=100 to 10 by alt * s=left right)
```

デフォルトペナルティを変更します。 PENALTIES(4)=では、フリー水平シフトのデフォルトペナルティが 500 に変更され、水平シフトのペナルティがすべて削除されます。LIST=では、PROC PLOT がラベルをそれぞれのポイントからどの程度遠くにシフトしたかが示されます。

```
           penalties(4)=500  list=0
```

軸をカスタマイズします。 HAXIS=では、プロットの両側にラベル用のスペースが十分空いた横軸が作成されます。VAXIS=では、縦軸の値の増分として 100 が指定され、VSPACE=5 オプションでは、縦軸の目盛間に 5 のスペースが指定されます。

```
           haxis=0 to 13000 by 1000
           vaxis=by 100
           vspace=5;
```

タイトルを指定します。

```
   title 'A Plot of Population Density and Crime Rates';
run;
```

出力

アウトプット 42.20 *デフォルトペナルティを調整したプロット*



アウトプット 42.21 ポイント位置、ペナルティおよび配置状態のリスト

List of Point Locations, Penalties, and Placement States

Label	Vertical Axis	Horizontal Axis	Penalty	Starting Position	Lines	Vertical Shift	Horizontal Shift
Maryland	428.70	5477.6	0	Right	1	0	55
Delaware	307.60	4938.8	0	Right	1	0	59
Pennsylvania	264.30	3163.2	0	Right	1	0	65
Ohio	263.30	4575.3	0	Right	1	0	66
Illinois	205.30	5416.5	0	Right	1	0	56
Florida	180.00	8503.2	0	Left	1	0	-64
California	151.40	6506.4	0	Right	1	0	45
Tennessee	111.60	4665.6	0	Right	1	0	61
North Carolina	120.40	4649.9	0	Right	1	0	46
New Hampshire	102.40	3371.7	0	Right	1	0	52
South Carolina	103.40	5161.9	0	Right	1	0	52
Georgia	94.10	5792.0	0	Left	1	0	-42
West Virginia	80.80	2190.7	0	Right	1	0	76
Alabama	76.60	4451.4	0	Right	1	0	41
Missouri	71.20	4707.5	0	Right	1	0	47
Mississippi	53.40	3438.6	0	Right	1	0	68
Vermont	55.20	4271.2	0	Right	1	0	44
Minnesota	51.20	4615.8	0	Right	1	0	49
Washington	62.10	7017.1	0	Left	1	0	-49
Texas	54.30	7722.4	0	Left	1	0	-49
Arkansas	43.90	4245.2	0	Right	1	0	65
Oklahoma	44.10	6025.6	0	Left	1	0	-43
Idaho	11.50	4156.3	0	Right	1	0	69
Oregon	27.40	6969.9	0	Left	1	0	-53
South Dakota	9.10	2678.0	0	Right	1	0	67
North Dakota	9.40	2833.0	0	Right	1	0	52
Nevada	7.30	6371.4	0	Right	1	0	50

43 章

PMENU プロシジャ

概要: PMENU プロシジャ

PMENU プロシジャでは、DATA ステップウィンドウ、マクロウィンドウ、SAS/AF および SAS/FSP の両ウィンドウ、またはカスタマイズメニューの指定が可能などの SAS アプリケーションでも使用できるメニューが定義されます。

メニューは、コマンドを実行する方法としてコマンド行のかわりに使用できます。メニューをアクティブにするには、任意のコマンド行から PMENU コマンドを発行します。メニューを表示するためには、アクティブにする必要があります。

メニューをアクティブにすると、各アクティブウィンドウにメニューバーが表示され、そこに選択可能な項目がリストされます。選択した項目に応じて、SAS では、コマンドが処理されるか、メニューもしくはサブメニューが表示されるか、またはダイアログボックスの情報入力を完了するよう要求されます。ダイアログボックスは、アクションを実行す

る前に回答が必要な質問や選択肢を示す単なるボックスです。次の図は、PROC PMENU で作成できる機能を示しています。

図 43.1 *メニューバー、メニューおよびダイアログボックス*



注: 一部の動作環境では、メニューバーがポップアップメニューとして表示されたり、ウィンドウの下部に表示されたりすることがあります。

PMENU プロシジャでは、目に見える出力がすぐに作成されるわけではありません。単に、後からアプリケーションで使用できる種類 PMENU のカタログエントリが作成されるだけです。

概念:PMENU プロシジャ

プロシジャの実行

プロシジャの初期化

複数のメニューを定義するには、RUN ステートメントで定義を区切ります。RUN ステートメントで終わるステートメントグループは、RUN グループと呼ばれます。RUN ステートメントをサブミットする前に、PMENU カタログエントリの定義を完了しておく必要があります。RUN ステートメントの後でプロシジャを再開する必要はありません。

メニューバーを定義する初期 MENU ステートメントを含め、さらに、すべての ITEM ステートメントおよび任意の SELECTION、MENU、SUBMENU、DIALOG ステートメント、ならびに同じ RUN グループ内の DIALOG ステートメントに関連付けられたステートメントを含める必要があります。たとえば、次のステートメントでは 2 つの別々の PMENU カタログエントリが定義されます。両方とも同じカタログに格納されますが、各 PMENU カタログエントリは互いに独立しています。例では、どちらの PMENU カタログエントリでも、ユーザーによる選択、実行が可能なウィンドウ環境コマンドをリストするだけのメニューバーが作成されます。

```
libname proclib 'SAS-data-library';

proc pmenu catalog=proclib.mycat;
   menu menu1;
```

```
   item end;
   item bye;
run;

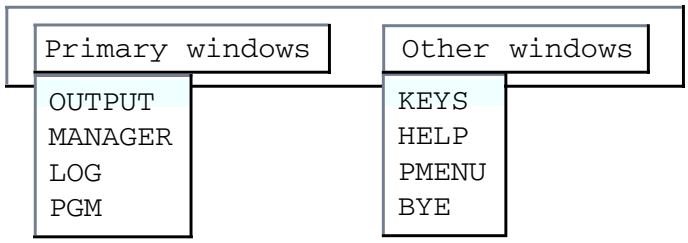
   menu menu2;
   item end;
   item pgm;
   item log;
   item output;
run;
```

これらのステートメントをサブミットすると、PMENU エントリが作成されたことを知らせるメッセージが表示されます。これらのメニューバーのいずれかを表示するには、“PMENU カタログエントリの作成と使用のステップ” (1293 ページ)で説明しているように、PMENU カタログエントリをウィンドウに関連付けて、有効化されたメニューを備えるウィンドウをアクティブにします。

プロシジャの終了

未実行のステートメントを実行するには、QUIT、DATA または新しい PROC ステートメントをサブミットし、PMENU プロシジャを終了します。未実行のステートメントをキャンセルするには、RUN CANCEL ステートメントをサブミットし、PMENU プロシジャを終了します。

PMENU カタログエントリの作成と使用のステップ

ほとんどの場合、PNENU エントリを作成、使用するには、次の操作が必要です。

1. PROC PMENU を使用して、メニューバー、メニュー、およびその他の必要な機能を定義します。SAS カタログに PROC PMENU の出力を格納します。詳細については、“メニューとウィンドウの関連付け” (1327 ページ)を参照してください。
2. SAS/AF および SAS/FSP ソフトウェア、または Base SAS ソフトウェアの WINDOW もしくは%WINDOW ステートメントを使用してウィンドウを定義します。
3. 次のいずれかを使用して、ステップ 1 で作成した PMENU カタログエントリをウィンドウに関連付けます。
   - Base SAS ソフトウェアの WINDOW ステートメントの MENU=オプション。詳細については、“メニューとウィンドウの関連付け” (1327 ページ)を参照してください。
   - マクロ機能の%WINDOW ステートメントの MENU=オプション。
   - SAS/AF ソフトウェアの PROGRAM エントリの **GATTR** ウィンドウの**コマンドメニュー**フィールド。
   - SAS/AF ソフトウェアの FRAME エントリの Keys、Pmenu および**コマンド**ウィンドウ。“例 5: メニューと FRAME アプリケーションの関連付け” (1329 ページ)を参照してください。
   - SAS/AF および SAS/FSP ソフトウェアの PMENU 関数。
   - SAS/FSP ソフトウェアの SETPMENU コマンド。“例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)を参照してください。
4. 作成したウィンドウをアクティブにします。メニューが有効になっていることを確認します。

PROC PMENU ステップのコーディングのテンプレート

次のコーディングテンプレートには、PMENU プロシジャでのステートメントの使用方法が要約されています。詳細については、ステートメントの説明を参照してください。

- 単純なメニューバーを作成します。メニューバーの項目はすべてウィンドウ環境コマンドです。

```
proc pmenu;
   menu menu-bar;
   item command;
   ...more-ITEM-statements...
run;
```

- メニューを作成する項目を含むメニューバーを作成します。

```
proc pmenu;
    menu menu-bar;
    item 'menu-item' menu=pull-down-menu;
    ...more-ITEM-statements...
    menu pull-down-menu;
    ...ITEM-statements-for-pull-down-menu...
run;
```

- メニューバーに表示されている以外のコマンドをサブミットする項目を含むメニューバーを作成します。

```
proc pmenu;
   menu menu-bar;
   item 'menu-item' selection=selection;
   ...more-ITEM-statements...
   selection selection 'command-string';
run;
```

- ダイアログボックスを開く項目を含むメニューバーを作成します。ダイアログボックスでは、情報が表示され、テキスト入力が要求されます。

```
proc pmenu;
   menu menu-bar;
   item 'menu-item' menu=pull-down-menu;
   ...more-ITEM-statements...
   menu pull-down-menu;
      item 'menu-item' dialog=dialog-box;
      dialog dialog-box 'command @1';
         text #line @column 'text';
         text #line @column LEN=field-length;
run;
```

- ダイアログボックスを開く項目を含むメニューバーを作成します。ダイアログボックスでは、使用可能値リストからいずれかを選択できます。

```
proc pmenu;
   menu menu-bar;
   item 'menu-item' menu=pull-down-menu;
   ...more-ITEM-statements...
   menu pull-down-menu;
      item 'menu-item' dialog=dialog-box;
      dialog dialog-box 'command %1';
         text #line @column 'text';
```

```
        radiobox default=button-number;
        rbutton #line @column
                'text-for-selection';
        ...more-RBUTTON-statements...
run;
```

- ダイアログボックスを開く項目を含むメニューバーを作成します。ダイアログボックスでは、複数の独立した選択が可能です。

```
proc pmenu;
   menu menu-bar;
   item 'menu-item' menu=pull-down-menu;
   ...more-ITEM-statements...
   menu pull-down-menu;
      item 'menu-item' dialog=dialog-box;
      dialog dialog-box 'command &1';
         text #line @column 'text';
         checkbox #line @column 'text';
         ...more-CHECKBOX-statements...
run;
```

構文: PMENU プロシジャ

制限事項: MENU ステートメントを少なくとも 1 つ使用し、その後に ITEM ステートメントを少なくとも 1 つ指定する必要があります。

ヒント: RUN グループ処理をサポートします。

グローバルステートメントを使用することもできます。リストは、“グローバルステートメント” (24 ページ)および“Global Statements” (*SAS Statements: Reference*)を参照してください。

参照項目: “PMENU Procedure: Windows” (*SAS Companion for Windows*)

“PMENU Procedure: UNIX” (*SAS Companion for UNIX Environments*)

“PMENU Procedure: z/OS” (*SAS Companion for z/OS*)

```
PROC PMENU <CATALOG=<libref.>catalog>
<DESC 'entry-description'>;
    MENU menu-bar;
        ITEM command <option(s)> <action-option(s)>;
        ITEM 'menu-item' <option(s)> <action-option(s)>;
            DIALOG dialog-box 'command-string field-number-specification';
                CHECKBOX <ON> #line @column 'text-for-selection'
                    <COLOR=color> <SUBSTITUTE='text-for-substitution'>;
                RADIOBOX DEFAULT=button-number;
                    RBUTTON <NONE> #line @column'text-for-selection'
                      <COLOR=color> <SUBSTITUTE='text-for-substitution'>;
                TEXT #line @column field-description
                  <ATTR=attribute> <COLOR=color>;
            MENU pull-down-menu;
            SELECTION selection 'command-string';
            SEPARATOR;
            SUBMENU submenu-name SAS-file;
```

ステートメント	タスク	例
“PROC PMENU ステートメント”	カスタマイズメニューの定義	Ex. 1
“CHECKBOX ステートメント”	ダイアログボックスでのユーザーの選択肢の定義	
“DIALOG ステートメント”	メニューの項目に関連づけられたダイアログボックスの説明	Ex. 2, Ex. 3, Ex. 4
“ITEM ステートメント”	メニューバーまたはメニューにリストされる項目の指定	Ex. 1, Ex. 3, Ex. 5
“MENU ステートメント”	カタログエントリ名の指定かまたはメニューの定義	Ex. 1, Ex. 5
“RADIOBOX ステートメント”	ダイアログボックス内の相互排他的な選択肢のリストと定義	Ex. 3
“RBUTTON ステートメント”	ダイアログボックス内の相互排他的な選択肢のリストと定義	Ex. 3
“SELECTION ステートメント”	項目の選択時にサブミットされるコマンドの定義	Ex. 1, Ex. 4
“SEPARATOR ステートメント”	メニューの項目間の線引き	
“SUBMENU ステートメント”	項目に関連付ける共通サブメニューの定義	Ex. 1
“TEXT ステートメント”	ダイアログボックスのテキストと入力フィールドの指定	Ex. 2

PROC PMENU ステートメント

PMENU プロシジャを呼び出し、PROC PMENU ステップで作成されるすべての PMENU カタログエントリの格納場所を指定します。

例: “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)

構文

PROC PMENU <CATALOG=<*libref.*>*catalog*>
<DESC '*entry-description*'>;

オプション引数

CATALOG=<*libref.*>*catalog*
PMENU エントリを格納するカタログを指定します。

デフォルト *libref* を省略すると、PMENU エントリは SASUSER ライブラリのカタログに格納されます。CATALOG=を省略すると、エントリは SASUSER.PROFILE カタログに格納されます。

例 “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)

DESC *'entry-description'*
ステップで作成された PMENU カタログエントリの説明を提供します。

デフォルト メニューの説明

注 ウィンドウ環境での**カタログ**ウィンドウかまたは CATALOG プロシジャでの CONTENTS ステートメントの使用時に、これらの説明が表示されます。

CHECKBOX ステートメント

ユーザーがダイアログボックス内で指定できる選択肢を定義します。

制限事項: DIALOG ステートメントの後で使用する必要があります。

構文

CHECKBOX <ON> #*line* @*column* '*text-for-selection*'
<COLOR=*color*> <SUBSTITUTE='*text-for-substitution*'>;

必須引数

column
ダイアログボックスでチェックボックスとテキストを配置する列を指定します。

line
ダイアログボックスでチェックボックスとテキストを配置する行を指定します。

text-for-selection
このチェックボックスについて説明するテキストを定義します。このテキストは、ウィンドウに表示され、さらに、SUBSTITUTE=オプションを使用しない場合は、ユーザーがチェックボックスを選択すると、前述の DIALOG ステートメントのコマンドにも挿入されます。

オプション引数

COLOR=*color*
チェックボックスの色と、説明テキストを定義します。

ON
デフォルトでは、このチェックボックスがアクティブになることを示します。このオプションを使用する場合は、CHECKBOX キーワードの直後に指定する必要があります。

SUBSTITUTE=*'text-for-substitution'*
DIALOG ステートメントのコマンドに挿入するテキストを指定します。

詳細

ダイアログボックスのチェックボックス

CHECKBOX ステートメントごとに、ユーザーが他の選択とは無関係に選択可能な項目が 1 つずつ定義されます。したがって、5 つの CHECKBOX ステートメントで 5 つの選択肢を定義すると、ユーザーはそれらの選択肢を任意に組み合わせて選択できます。ユーザーが選択肢を選ぶと、選択肢に関連付けられた text-for-selection 値が、前述の DIALOG ステートメントで先頭にアンパサンド(&)が付くフィールド位置のコマンド文字列に挿入されます。

DIALOG ステートメント

メニューの項目に関連付けられたダイアログボックスについて記述します。

制限事項: この後に少なくとも 1 つ TEXT ステートメントを指定する必要があります。

例: “例 2: ダイアログボックスでのユーザー入力の収集” (1312 ページ)
“例 3: 複数の変数を検索するダイアログボックスの作成” (1316 ページ)
“例 4: DATA ステップウィンドウアプリケーションのメニューの作成” (1323 ページ)

構文

DIALOG *dialog-box* '*command-string field-number-specification*';

必須引数

command-string
項目の選択時に実行されるコマンドかまたはコマンドの一部です。代入の実行後に、その結果として生じる *command-string* の制限は、各自の動作環境に対するコマンド行制限です。通常、コマンド行制限はおよそ 80 文字です。

'*command-string field-number-specification*'の制限は 200 文字です。

注: PROC PMENU を使用して、**PROGRAM EDITOR** ウィンドウでのみ有効な任意のコマンド(INCLUDE コマンドなど)をサブミットする場合は、ウィンドウ環

境を実行し、**PROGRAM EDITOR** ウィンドウにコントロールを戻す必要があります。

dialog-box
前述の ITEM ステートメントの DIALOG=オプションで指定したものと同じ名前です。

field-number-specification
次のうちから 1 つ以上が指定されます。

@1…@*n*　%1…%*n*　&1…&*n*

@1、%1 または&1 などのフィールド番号をコマンド文字列に埋め込み、コマンド文字列内で異なる種類のフィールド番号を併用できます。フィールド番号の数値部分は、TEXT、RADIOBOX および CHECKBOX ステートメントの相対位置に対応しています。これらのステートメントの実数に対応しているわけではありません。

@1…@*n*
サブミット前のコマンドに情報を追加できる任意の TEXT ステートメント番号です。アットマーク(@)の後に指定される数字は、LEN=オプションを使用して入力フィールドを定義する TEXT ステートメントに対応しています。

%1…%*n*
サブミット前のコマンドに情報を追加できる任意の RADIOBOX ステートメント番号です。パーセント記号(%)の後に指定される数字は、DIALOG ステートメントの後に続く RADIOBOX ステートメントに対応してしまいます。

注 番号が RBUTTON ステートメントではなく RADIOBOX ステートメントに対応していることに注意してください。

&1…&*n*
サブミット前のコマンドに情報を追加できる任意の CHECKBOX ステートメント番号です。アンパサンド(&)の後に指定される数字は、DIALOG ステートメントの後に続く CHECKBOX ステートメントに対応しています。

注 *command-string* で@ (アットマーク)、% (パーセント記号)または&(アンパサンド)の文字を指定するには、次のように 2 文字を使用します。@@ (アットマーク)、%% (パーセント記号)または&&(アンパサンド)

詳細

- ダイアログボックスの配置は制御できません。ダイアログボックスはスクロールできません。ダイアログボックスのサイズと配置は、各自のウィンドウ環境によって決定されます。
- DIALOG ステートメントを使用するには、ITEM ステートメントで DIALOG=オプションを指定します。
- ITEM ステートメントでは、メニューバーまたはメニューのエントリが作成され、DIALOG=オプションでは、どの DIALOG ステートメントでダイアログボックスについて記述するかが指定されます。
- ダイアログボックスのコンテンツを定義するには、CHECKBOX、RADIOBOX および RBUTTON ステートメントを使用します。
- 次の図は、一般的なダイアログボックスを示しています。ダイアログボックスでは、3 つの方法で情報が要求されます。
  - フィールドに入力します。ユーザーからのテキストを受け入れるフィールドは、テキストフィールドと呼ばれます。

- 相互排他的な選択肢のリストから選択します。この種の選択肢のグループはラジオボタンと呼ばれ、個々の選択肢はラジオボタンと呼ばれます。
- 他の独立した選択肢を選択するかどうかを示します。たとえば、さまざまなオプションの使用を選択するには、リストされた選択肢の一部またはすべてを選択します。この種の選択肢はチェックボックスと呼ばれます。

図 43.2 *一般的なダイアログボックス*



ダイアログボックスには 2 つ以上のボタン(OK とキャンセルなど)があり、自動的にボックスに組み込まれます。ボタンによってアクションが引き起こされます。

注: ボタンの実際の名前は、ウィンドウ環境によって異なります。

ITEM ステートメント

メニューバーまたはメニューにリストする項目を識別します。

例: “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)
“例 3: 複数の変数を検索するダイアログボックスの作成” (1316 ページ)
“例 5: メニューと FRAME アプリケーションの関連付け” (1329 ページ)

構文

ITEM *command* <*option(s)*> <*action-option(s)*>;
ITEM '*menu-item*' <*option(s)*> <*action-option(s)*>;

オプション引数の要約

ACCELERATE=*name-of-key*
項目を選択するかわりに使用可能なキーシーケンスを定義します。

action-option
項目のアクションを指定します。

GRAY
項目がこのウィンドウではアクティブな選択肢ではないことを示します。

HELP='*help-text*'
ユーザーがメニュー項目を表示したときに表示されるテキストを指定します。

ID=*integer*
メニューの項目の ID として使用する値を指定します。

MNEMONIC=*character*
項目を選択できる文字を 1 つ定義します。

STATE=CHECK|RADIO
チェックボックスまたはラジオボタンを項目の隣に配置します。

必須引数

command
メニューが表示されるウィンドウの有効な SAS コマンドとなる単語。2 語以上のコマンド(WHERE CLEAR など)は、一重引用符で囲む必要があります。*command* は、メニューバーに大文字で表示されます。

メニュー上で SAS コマンドを大文字にするか小文字にするかを制御する場合は、コマンドを一重引用符で囲みます。その場合、使用した文字がそのままメニューに表示されます。

'*menu-item*'
引用符で囲まれた単語またはテキスト文字列で、ユーザーがこの項目を選択した際に発生するアクションについて記述します。メニュー項目の開始文字にパーセント記号(%)は使用できません。

オプション引数

ACCELERATE=*name-of-key*
項目を選択するかわりに使用可能なキーシーケンスを定義します。ユーザーがキーシーケンスを押すと、メニューバーやメニューから項目を選択する場合と同じ結果になります。

制限事項 このオプションの機能性は数文字に限られます。詳細については、動作環境に関する SAS のドキュメントを参照してください。

このオプションはすべての動作環境で使用できるわけではありません。このオプションを含めても、動作環境で使用できない場合は無視されます。

action-option
次のいずれかになります。

DIALOG=*dialog-box*
関連付けられた DIALOG ステートメントの名前を指定します。これにより、ユーザーがこの項目を選択すると、ダイアログボックスが表示されます。

例 “例 3: 複数の変数を検索するダイアログボックスの作成” (1316 ページ)

MENU=*pull-down-menu*
関連付けられた MENU ステートメントの名前を指定します。これにより、ユーザーがこの項目を選択すると、メニューが表示されます。

参照項目 “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)

SELECTION=*selection*
関連付けられた SELECTION ステートメントの名前を指定します。これにより、ユーザーがこの項目を選択すると、コマンドがサブミットされます。

参照項目 “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)

SUBMENU=*submenu*
項目と共通サブメニューを関連付けます。

関連付けられた SUBMENU ステートメントの名前を指定します。これにより、ユーザーがこの項目を選択すると、ポップメニューエントリが表示されます。

参照項目 “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)

DIALOG=、MENU=、SELECTION=、SUBMENU=オプションのいずれも指定しなかった場合、*command* または *menu-item* テキスト文字列は、ユーザーが項目を選択すると、コマンド行コマンドとしてサブミットされます。

GRAY
項目がこのウィンドウではアクティブな選択肢ではないことを示します。このオプションは、多数のウィンドウに対する標準項目リストを定義する場合に有益ですが、すべての項目がすべてのウィンドウで有効とは限りません。このオプションを設定すると、ユーザーが項目を選択しても、アクションは発生しません。

HELP='*help-text*'
ユーザーがメニュー項目を表示したときに表示されるテキストを指定します。たとえば、マウスを使用してメニューをプルダウンする場合は、項目にマウスポインタを合わせると、テキストが表示されます。

制限事項 このオプションはすべての動作環境で使用できるわけではありません。このオプションを含めても、動作環境で使用できない場合は無視されます。

ヒント テキストが表示される場所は、動作環境によって異なります。

ID=*integer*
メニューの項目の ID として使用する値を指定します。SAS/AF アプリケーション内でこの ID を使用すると、メニューの項目を選択的にアクティブや非アクティブにしたり、チェックボックスやラジオボタンで項目の状態を設定したりできます。

制限事項 0 から 3000 までの整数は、動作環境および SAS 用に予約されています。

このオプションはすべての動作環境で使用できるわけではありません。このオプションを含めても、動作環境で使用できない場合は無視されます。

注 使用できる最小値は 3001 です。

ヒント ID=は SAS コンポーネント言語の WINFO 関数と併用すると有益です。

同じ ID を 2 つ以上の項目に使用できます。

参照項目 “STATE=CHECK|RADIO” (1303 ページ)

MNEMONIC=*character*
メニューに表示されるテキスト文字列で最初に出現した *character* に下線を引きます。*character* はテキスト文字列にする必要があります。

character は通常、Alt などの別のキーと組み合わせて使用します。キーシーケンスを使用すると、項目にカーソルを置いたときと同じ結果になります。ただし、その項目が制御するアクションは*呼び出されません*。

制限事項 このオプションはすべての動作環境で使用できるわけではありません。このオプションを含めても、動作環境で使用できない場合は無視されます。

STATE=CHECK|RADIO
選択した項目の隣にチェックボックスまたはラジオボタンを配置できます。

制限事項 このオプションはすべての動作環境で使用できるわけではありません。このオプションを含めても、動作環境で使用できない場合は無視されます。

ヒント STATE=は、ID=オプションおよび SAS コンポーネント言語の WINFO 関数とあわせて使用されます。

詳細

メニューバーの項目の定義

ITEM ステートメントを使用して、メニューバーに表示されるすべての項目の名前を指定する必要があります。また、ITEM ステートメントを使用して、任意のメニューに表示される項目の名前も指定します。ITEM ステートメントで指定できる項目は、ユーザーの項目選択時に発行されるコマンドや、関連付けされた DIALOG、MENU、SELECTION、SUBMENU ステートメントによって実行される他のアクションの説明などです。

メニューの All ITEM ステートメントは、MENU ステートメントの直後、および任意の DIALOG、SELECTION、SUBMENU やその他の MENU ステートメントの前に配置する必要があります。一部の動作環境では、ITEM ステートメントの間に SEPARATOR ステートメントを挿入すると、メニュー上の項目のグループを区切る行を作成できます。詳細については、“SEPARATOR ステートメント” (1307 ページ)を参照してください。

注: ウィンドウに対して長すぎるメニューバーを指定すると、切り捨てられるか、折り返されて複数行になる場合があります。

MENU ステートメント

メニューを格納するかまたはメニューを定義するカタログエントリ名を指定します。

例: “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)
“例 5: メニューと FRAME アプリケーションの関連付け” (1329 ページ)

構文

MENU *menu-bar*;
MENU *pull-down-menu*;

必須引数

次のいずれかの引数が必須です。

menu-bar
メニューを格納するカタログエントリ名を指定します。

pull-down-menu
ユーザーがメニューバーの項目を選択したときに表示されるメニューの名前を指定します。*pull-down-menu* の値は、前述の ITEM ステートメントの MENU=オプションで指定する *pull-down-menu* 名と一致させる必要があります。

詳細

メニューの定義

MENU ステートメントは、メニューの定義に使用する場合、MENU=オプションを指定する ITEM ステートメントの後に指定する必要があります。メニューの ITEM ステートメントと MENU ステートメントは両方とも、PMENU カタログエントリのメニューバーを定義する MENU ステートメントとして、同じ RUN グループ内に存在する必要があります。

メニューバーとメニューのどちらでも、MENU ステートメントの後に続けて、メニューに表示される各項目を定義する ITEM ステートメントを指定します。メニューの ITEM ステートメントをすべてひとまとめにします。たとえば、次の PROC PMENU ステップでは、カタログエントリ WINDOWS が 1 つ作成され、そこで 2 つの項目 **Primary windows** と **Other windows** を含むそメニューバーが作成されます。これらの項目のいずれかを選択すると、メニューが表示されます。

```
libname proclib 'SAS-data-library';

proc pmenu cat=proclib.mycat;

      /* create catalog entry */
   menu windows;
   item 'Primary windows' menu=prime;
   item 'Other windows' menu=other;

      /* create first menu */
   menu prime;
   item output;
   item manager;
   item log;
   item pgm;

      /* create second menu */
   menu other;
   item keys;
   item help;
   item pmenu;
   item bye;

   /* end of run group */
run;
```

次の図は、結果として表示されるメニュー選択を示しています。

図 43.3 メニュー



RADIOBOX ステートメント

ダイアログボックス内に、相互排他的な選択肢を含むボックスを定義します。

制限事項: DIALOG ステートメントの後で使用する必要があります。
その後に RBUTTON ステートメントを 1 つ以上指定する必要があります。

例: “例 3: 複数の変数を検索するダイアログボックスの作成” (1316 ページ)

構文

RADIOBOX DEFAULT=*button-number*;

必須引数

DEFAULT=*button-number*
どのラジオボタンがデフォルトかを示します。

デフォルト 1

詳細

RADIOBOX ステートメントでは、選択リストの開始が示されます。RADIOBOX ステートメントの直後に、ユーザーが行える各選択に対する RBUTTON ステートメントをリストする必要があります。ユーザーが選択を行うと、その選択肢に関連付けられたテキスト値が、前述の DIALOG ステートメントのコマンド文字列で先頭にパーセント記号(%)が付くフィールド位置に挿入されます。

RBUTTON ステートメント

ダイアログ内に、相互排他的な選択肢をリストします。

制限事項: RADIOBOX ステートメントの後で使用する必要があります。

例: “例 3: 複数の変数を検索するダイアログボックスの作成” (1316 ページ)

構文

RBUTTON <NONE> #*line* @*column*'*text-for-selection*'
<COLOR=*color*> <SUBSTITUTE='*text-for-substitution*'>;

必須引数

column
ラジオボタンとテキストを配置するダイアログボックスの列を指定します。

line
ラジオボタンとテキストを配置するダイアログボックスの行を指定します。

text-for-selection
ダイアログボックスに表示されるテキストを定義し、SUBSTITUTE=オプションが使用されていない場合は、その前の DIALOG ステートメントのコマンドに挿入されるテキストを定義します。

注: テキストとラジオボタンを配置する際は、列と行が重複しないように注意してください。テキストとボタンが重複すると、エラーメッセージが表示されます。また、他のテキストとラジオボタンの間のスペースも指定します。

オプション引数

COLOR=*color*
ラジオボタンの色と、ボタンの説明テキストを定義します。

制限事項 このオプションはすべての動作環境で使用できるわけではありません。このオプションを含めても、動作環境で使用できない場合は無視されます。

NONE
その他のどの選択肢も示さないボタンを定義します。このボタンを定義すると、その他の選択肢をすべて無視できます。DIALOG ステートメントに文字(ブランクを含む)はまったく挿入されません。

制限事項 このオプションを使用する場合は、RBUTTON キーワードの直後に指定する必要があります。

SUBSTITUTE=*'text-for-substitution'*
DIALOG ステートメントのコマンドに挿入するテキストを指定します。

参照項目 “例 3: 複数の変数を検索するダイアログボックスの作成” (1316 ページ)

SELECTION ステートメント

項目の選択時にサブミットされるコマンドを定義します。

制限事項: ITEM ステートメントの後で使用する必要があります。

例: “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)
“例 4: DATA ステップウィンドウアプリケーションのメニューの作成” (1323 ページ)

構文

SELECTION *selection* '*command-string*';

必須引数

selection
前述の ITEM ステートメントの SELECTION=オプションで指定したものと同じ名前です。

command-string
ユーザーの項目選択時にコマンド行コマンドとしてサブミットされる、引用符で囲まれたテキスト文字列。*command-string* には 200 文字の制限があります。ただし、およそ 80 文字のコマンド行制限を超えることはできません。コマンド行制限は、さまざまな動作環境によって多少異なります。

注: SAS では、SELECTION ステートメントで指定した項目の最初の 8 文字のみ使用されます。ユーザーがメニューリストから項目を選択する際、リスト内の各項目名の最初の 8 文字は一意である必要があります。これにより、SAS ではリスト内の正しい項目を選択できます。最初の 8 文字が一意ではない場合、SAS では、リスト内の最後の項目が選択されます。

詳細

ITEM ステートメントで項目名を定義し、SELECTION=オプションを指定して、その項目を後続の SELECTION ステートメントと関連付けます。次に、SELECTION ステートメントで、ユーザーがメニューバーまたはメニューの項目を選択した際にサブミットされる実際のコマンドが定義されます。

SELECTION ステートメントを使用してコマンド文字列を定義する場合もあります。ITEM ステートメントを使用して、単純な別名を作成します。これにより、SELECTION ステートメントに定義された長いコマンド文字列が呼び出されます。たとえば、メニューバーに WINDOW ステートメントを呼び出す項目を含めて、データエントリを可能にできます。ユーザーがこの項目を選択した際に処理される実際のコマンドは、アプリケーションを含めてサブミットするコマンドです。

注: PROC PMENU を使用して、**PROGRAM EDITOR** ウィンドウでのみ有効な任意のコマンド(INCLUDE コマンドなど)を発行する場合は、ウィンドウ環境を実行する必要があります。また、**PROGRAM EDITOR** ウィンドウにコントロールを戻す必要もあります。

SEPARATOR ステートメント

メニューの項目間に線を引きます。

制限事項: ITEM ステートメントの後で使用する必要があります。
すべての動作環境で使用できるわけではありません。

構文

SEPARATOR;

SUBMENU ステートメント

項目に関連付けられた共通のサブメニューを含む SAS ファイルを指定します。

例: “例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)

構文

SUBMENU *submenu-name SAS-file*;

必須引数

submenu-name
サブメニューステートメントの名前を指定します。サブメニューとメニュー項目を関連付けるには、*submenu-name* が ITEM ステートメントの SUBMENU=アクションオプションと一致している必要があります。

SAS-file
共通のサブメニューを含む SAS ファイルの名前を指定します。

TEXT ステートメント

ダイアログボックスのテキストと入力フィールドを指定します。

制限事項: DIALOG ステートメントの後でのみ使用可能です。

例: “例 2: ダイアログボックスでのユーザー入力の収集” (1312 ページ)

構文

TEXT #*line* @*column field-description*
<ATTR=*attribute*> <COLOR=*color*>;

必須引数

column
テキストまたは入力フィールドの開始列を指定します。

field-description
TEXT ステートメントの使用方法を定義します。*field-description* は、次のうちのいずれかになります。

LEN=*field-length*
ユーザーが情報を入力できる入力フィールドの長さです。LEN=引数を使用すると、フィールドに入力した情報が、前述の DIALOG ステートメントのコマンド文字列で先頭に記号(@)が付くフィールド位置に挿入されます。

参照項目 “例 2: ダイアログボックスでのユーザー入力の収集” (1312 ページ)

'*text*'
ダイアログ内の *line* と *column* で定義した位置に表示されるテキスト文字列です。

line
テキストまたは入力フィールドの行数を指定します。

オプション引数

ATTR=*attribute*
テキストまたは入力フィールドの属性を定義します。有効な属性値は次のとおりです。

- BLINK

- HIGHLIGH
- REV_VIDE
- UNDERLIN

制限事項 このオプションはすべての動作環境で使用できるわけではありません。このオプションを含めても、動作環境で使用できない場合は無視されます。

使用しているハードウェアで、これらの属性の一部がサポートされていない場合もあります。

COLOR=*color*
テキストまたは入力フィールド文字の色を定義します。使用可能な色の値は次のとおりです。

BLACK	BROWN
GRAY	MAGENTA
PINK	WHITE
BLUE	CYAN
GREEN	ORANGE
RED	YELLOW

制限事項 このオプションはすべての動作環境で使用できるわけではありません。このオプションを含めても、動作環境で使用できない場合は無視されます。

使用しているハードウェアで、これらの色の一部がサポートされていない場合もあります。

例: PMENU プロシジャ

例 1: FSEDIT アプリケーションのメニューバーの作成

要素: PROC PMENU ステートメントオプション
CATALOG=
ITEM ステートメントオプション
MENU=
SELECTION=
SUBMENU=
MENU ステートメント
SELECTION ステートメント
SUBMENU ステートメント

詳細

この例では、FSEDIT アプリケーションでデフォルトメニューバーのかわりに使用できるメニューバーを作成します。これらのメニューで使用可能な選択肢では、エンドユーザーによるオブザベーションの削除、複製はできません。

注:

- PROC PMENU のウィンドウ例は UNIX 環境で作成したものであり、その他の動作環境における同ウィンドウとは多少表示が異なる場合があります。
- メニューの表示や既存 SAS メニューとの結合に影響を及ぼす可能性がある動作環境固有のシステムオプションについて知っておく必要があります。詳細については、動作環境に関する SAS のドキュメントを参照してください。

プログラム

```
libname proclib
'SAS-data-library';

proc pmenu catalog=proclib.menucat;

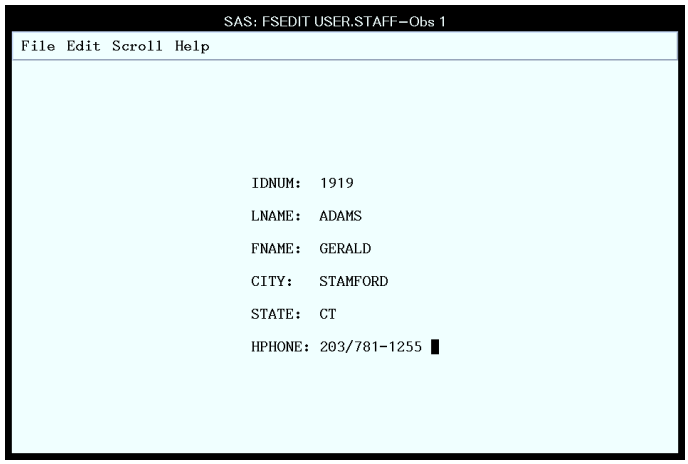
   menu project;

      item 'File' menu=f;
      item 'Edit' submenu=editmnu;
      item 'Scroll' menu=s;
      item 'Help' menu=h;

      menu f;
         item 'Goback' selection=g;
         item 'Save';
         selection g 'end';

      submenu editmnu sashelp.core.edit;

      menu s;
         item 'Next Obs' selection=n;
         item 'Prev Obs' selection=p;
         item 'Top';
         item 'Bottom';
         selection n 'forward';
         selection p 'backward';

      menu h;
         item 'Keys';
         item 'About this application' selection=hlp;
         selection hlp 'sethelp user.menucat.staffhlp.help;help';
quit;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリは、メニュー定義の格納に使用されます。

```
libname proclib
'SAS-data-library';
```

メニュー定義を格納するカタログを指定します。 メニュー定義は PROCLIB.MENUCAT カタログに格納されます。

```
proc pmenu catalog=proclib.menucat;
```

カタログエントリ名を指定します。 MENU ステートメントで、カタログエントリ名として PROJECT を指定します。メニューはカタログエントリ PROCLIB.MENUCAT.PROJECT.PMENU に格納されます。

```
menu project;
```

メニューバーを設計します。 ITEM ステートメントで、メニューバーの項目を指定します。MENU=オプションの値が後続の MENU ステートメントで使用されます。Edit 項目では共通の事前定義サブメニューが使用され、その他の項目のメニューはこの PROC ステップで定義されます。

```
item 'File' menu=f;
item 'Edit' submenu=editmnu;
item 'Scroll' menu=s;
item 'Help' menu=h;
```

File メニューを設計します。 このステートメントグループでは、メニューバーの File で使用可能な選択肢が定義されます。最初の ITEM ステートメントでは、File の最初の選択肢として **Goback** が指定されます。SELECTION=オプションの値は後続の SELECTION ステートメントと一致しています。このステートメントでは、その選択肢に対して発行されるコマンドとして END が指定されます。2 番目の ITEM ステートメントでは、その選択肢に対して SAVE コマンドが発行されることを指定します。

```
menu f;
   item 'Goback' selection=g;
   item 'Save';
   selection g 'end';
```

EDITMNU サブメニューを追加します。 SUBMENU ステートメントでは、SAS ファイル SASHELP.CORE.EDIT に存在する事前定義されたサブメニューが、メニューバーの Edit 項目に関連付けられます。SUBMENU ステートメントの名前は EDITMNU です。これは Edit 項目に対する ITEM の SUBMENU=アクションオプションの名前と一致しています。

```
submenu editmnu sashelp.core.edit;
```

Scroll メニューを設計します。 このステートメントグループでは、メニューバーの Scroll で使用可能な選択肢が定義されます。

```
menu s;
   item 'Next Obs' selection=n;
   item 'Prev Obs' selection=p;
   item 'Top';
   item 'Bottom';
   selection n 'forward';
   selection p 'backward';
```

Help メニューを設計します。 このステートメントグループでは、メニューバーの Help で使用可能な選択肢が定義されます。SETHELP コマンドでは、FSEDIT アプリケーションのユーザー作成情報を含む HELP エントリが指定されます。HELP エントリ名の後にセミコロンを記述すると、HELP コマンドを文字列に含められます。その HELP コマンドによって、HELP エントリが呼び出されます。

```
        menu h;
           item 'Keys';
           item 'About this application' selection=hlp;
           selection hlp 'sethelp user.menucat.staffhlp.help;help';
quit;
```

メニューバーと FSEDIT セッションの関連付け

次の SETPMENU コマンドでは、カスタマイズされたメニューバーが **FSEDIT** ウィンドウに関連付けられます。

```
setpmenu proclib.menucat.project.pmenu;pmenu on
```

また、FSEDIT セッションのコマンド行を使用したり、SAS コンポーネント言語(SCL)の CALL EXECCMD コマンドを発行したりして、メニューバーを指定することもできます。

カスタマイズされたメニューバーを **FSEDIT** ウィンドウに関連付けるその他の方法については、“メニューバーと FSEDIT セッションの関連付け” (1320 ページ)を参照してください。

次の **FSEDIT** ウィンドウにメニューバーが表示されます。

図 43.4 *FSEDIT ウィンドウのメニューバーの例*



例 2: ダイアログボックスでのユーザー入力の収集

要素: DIALOG ステートメント
TEXT ステートメントオプション
LEN=

詳細

この例では、“例 1: FSEDIT アプリケーションのメニューバーの作成” (1309 ページ)で作成したメニューにダイアログボックスを追加します。ダイアログボックスを使用すると、WHERE 句を使用して SAS データセットをサブセット化できます。

タスクには次が含まれます。

- ダイアログボックスでのユーザー入力の収集

- FSEDIT アプリケーション用カスタマイズメニューの作成

プログラム

```
libname proclib
'SAS-data-library';

proc pmenu catalog=proclib.menucat;

   menu project;

      item 'File' menu=f;
      item 'Edit' menu=e;
      item 'Scroll' menu=s;
      item 'Subset' menu=sub;
      item 'Help' menu=h;

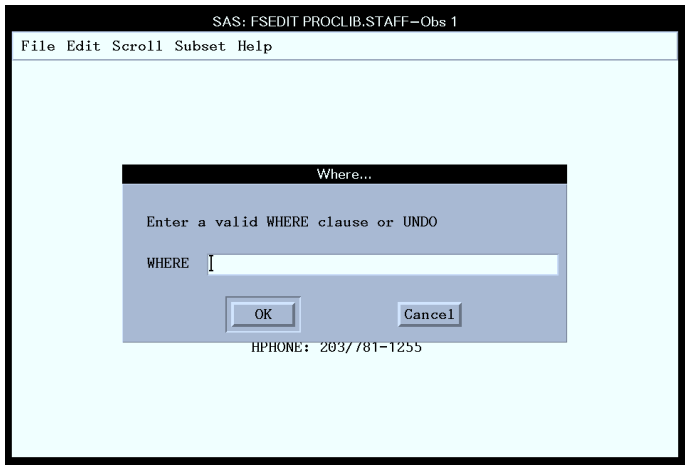
      menu f;
         item 'Goback' selection=g;
         item 'Save';
         selection g 'end';

      menu e;
         item 'Cancel';
         item 'Add';

      menu s;
         item 'Next Obs' selection=n;
         item 'Prev Obs' selection=p;
         item 'Top';
         item 'Bottom';
         selection n 'forward';
         selection p 'backward';

      menu sub;
         item 'Where' dialog=d1;
         item 'Where Clear';

      menu h;
         item 'Keys';
         item 'About this application' selection=hlp;
         selection hlp 'sethelp proclib.menucat.staffhlp.help;help';

      dialog d1 'where @1';
         text #2 @3 'Enter a valid WHERE clause or UNDO';
         text #4 @3 'WHERE ';
         text #4 @10 len=40;
quit;
```

プログラムの説明

PROCLIB ライブラリを宣言します。PROCLIB ライブラリは、メニュー定義の格納に使用されます。

```
libname proclib
'SAS-data-library';
```

メニュー定義を格納するカタログを指定します。メニュー定義は PROCLIB.MENUCAT カタログに格納されます。

```
proc pmenu catalog=proclib.menucat;
```

カタログエントリ名を指定します。 MENU ステートメントで、カタログエントリ名として PROJECT を指定します。メニューはカタログエントリ PROCLIB.MENUCAT.PROJECT.PMENU に格納されます。

```
menu project;
```

メニューバーを設計します。 ITEM ステートメントで、メニューバーの項目を指定します。MENU=オプションの値が後続の MENU ステートメントで使用されます。

```
item 'File' menu=f;
item 'Edit' menu=e;
item 'Scroll' menu=s;
item 'Subset' menu=sub;
item 'Help' menu=h;
```

File メニューを設計します。 このステートメントグループでは、メニューバーの File の選択肢が定義されます。最初の ITEM ステートメントでは、File の最初の選択肢として **Goback** が指定されます。SELECTION=オプションの値は後続の SELECTION ステートメントと一致しています。このステートメントでは、その選択肢に対して発行されるコマンドとして END が指定されます。2 番目の ITEM ステートメントでは、その選択肢に対して SAVE コマンドが発行されることを指定します。

```
menu f;
   item 'Goback' selection=g;
   item 'Save';
   selection g 'end';
```

Edit メニューを設計します。 このステートメントグループでは、メニューバーの Edit で使用可能な選択肢が定義されます。

```
menu e;
   item 'Cancel';
   item 'Add';
```

Scroll メニューを設計します。 このステートメントグループでは、メニューバーの Scroll で使用可能な選択肢が定義されます。

```
menu s;
   item 'Next Obs' selection=n;
   item 'Prev Obs' selection=p;
   item 'Top';
   item 'Bottom';
   selection n 'forward';
   selection p 'backward';
```

Subset メニューを設計します。 このステートメントグループでは、メニューバーの Subset で使用可能な選択肢が定義されます。DIALOG=オプションの値 d1 が後続の DIALOG ステートメントで使用されます。

```
menu sub;
   item 'Where' dialog=d1;
   item 'Where Clear';
```

Help メニューを設計します。 このステートメントグループでは、メニューバーの Help で使用可能な選択肢が定義されます。SETHELP コマンドでは、FSEDIT アプリケーションのユーザー作成情報を含む HELP エントリが指定されます。セミコロンを使用すると、

HELP コマンドを文字列に含められます。その HELP コマンドによって、HELP エントリが呼び出されます。

```
menu h;
   item 'Keys';
   item 'About this application' selection=hlp;
   selection hlp 'sethelp proclib.menucat.staffhlp.help;help';
```

ダイアログボックスを設計します。 DIALOG ステートメントによって、WHERE コマンドが作成されます。WHERE コマンドの引数は、ユーザー入力によって、3 つの TEXT ステートメントにより記述されるテキストエントリフィールドに提供されます。@1 表記は、テキストフィールドのユーザー入力に対するプレースホルダです。TEXT ステートメントでは、ダイアログボックスのテキストと入力フィールド長が指定されます。

```
dialog d1 'where @1';
   text #2 @3 'Enter a valid WHERE clause or UNDO';
   text #4 @3 'WHERE ';
   text #4 @10 len=40;
quit;
```

メニューバーと FSEDIT ウィンドウの関連付け

次の SETPMENU コマンドでは、カスタマイズされたメニューバーが **FSEDIT** ウィンドウに関連付けられます。

```
setpmenu proclib.menucat.project.pmenu;pmenu on
```

また、FSEDIT セッションのコマンド行を使用したり、SAS コンポーネント言語(SCL)の CALL EXECCMD コマンドを発行したりして、メニューバーを指定することもできます。SCL に関する完全なドキュメントについては、*SAS(R) Component Language 9.3:Reference* を参照してください。

カスタマイズされたメニューバーを **FSEDIT** ウィンドウに関連付けるその他の方法については、“メニューバーと FSEDIT セッションの関連付け” (1320 ページ)を参照してください。

ユーザーが **Subset**、**Where** の順に選択すると、次のダイアログボックスが表示されます。

図 43.5 *Where ダイアログボックスの例*



例 3: 複数の変数を検索するダイアログボックスの作成

要素: DIALOG ステートメント
SAS マクロ呼び出し
ITEM ステートメント
DIALOG=オプション
RADIOBOX ステートメントオプション
DEFAULT=
RBUTTON ステートメントオプション
SUBSTITUTE=

他の要素: SAS マクロ呼び出し

詳細

この例では、FSEDIT セッションのメニューバーを変更して、複数の変数から値を 1 つ検索できるようにする方法を示します。例では、FSEDIT セッションで使用するカスタマイズメニューを作成します。メニュー構造は、**WHERE** ダイアログボックスを除いて、前述の例と同じです。

メニュー項目を選択すると、マクロが呼び出されます。ユーザー入力がマクロパラメータの値になります。マクロによって生成される WHERE コマンドが拡張され、検索に必要な変数がすべて含められます。

タスクには次が含まれます。

- カスタマイズメニューと FSEDIT セッションの関連付け
- WHERE 句での複数の変数の検索
- SAS マクロでの PROC PMENU 機能の拡張

プログラム

```
libname proclib
'SAS-data-library';

proc pmenu catalog=proclib.menucat;

   menu project;

      item 'File' menu=f;
      item 'Edit' menu=e;
      item 'Scroll' menu=s;
      item 'Subset' menu=sub;
      item 'Help' menu=h;

      menu f;
         item 'Goback' selection=g;
         item 'Save';
         selection g 'end';

      menu e;
         item 'Cancel';
         item 'Add';

      menu s;
         item 'Next Obs' selection=n;
```

```
      item 'Prev Obs' selection=p;
      item 'Top';
      item 'Bottom';
      selection n 'forward';
      selection p 'backward';

   menu sub;
      item 'Where' dialog=d1;
      item 'Where Clear';

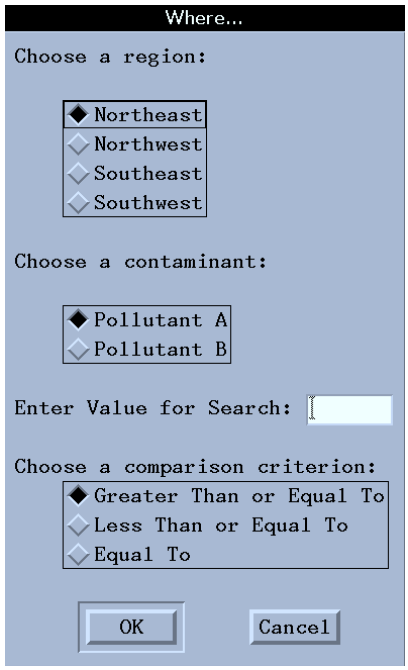
   menu h;
      item 'Keys';
      item 'About this application' selection=hlp;
      selection hlp 'sethelp proclib.menucat.staffhlp.help;help';

    dialog d1 '%%wbuild(%1,%2,@1,%3)';

       text #1 @1 'Choose a region:';
       radiobox default=1;
          rbutton #3 @5 'Northeast' substitute='NE';
          rbutton #4 @5 'Northwest' substitute='NW';
          rbutton #5 @5 'Southeast' substitute='SE';
          rbutton #6 @5 'Southwest' substitute='SW';

       text #8 @1 'Choose a contaminant:';
       radiobox default=1;
          rbutton #10 @5  'Pollutant A' substitute='pol_a,2';
          rbutton #11 @5  'Pollutant B' substitute='pol_b,4';

       text #13 @1 'Enter Value for Search:';
       text #13 @25 len=6;

       text #15 @1 'Choose a comparison criterion:';
       radiobox default=1;
          rbutton #16 @5 'Greater Than or Equal To'
                         substitute='GE';
          rbutton #17 @5 'Less Than or Equal To'
                         substitute='LE';
          rbutton #18 @5 'Equal To' substitute='EQ';
quit;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリは、メニュー定義の格納に使用されます。

```
libname proclib
'SAS-data-library';
```

メニュー定義を格納するカタログを指定します。 メニュー定義は PROCLIB.MENUCAT カタログに格納されます。

```
proc pmenu catalog=proclib.menucat;
```

カタログエントリ名を指定します。 MENU ステートメントで、カタログエントリ名として STAFF を指定します。メニューはカタログエントリ PROCLIB.MENUCAT.PROJECT.PMENU に格納されます。

```
    menu project;
```

メニューバーを設計します。 ITEM ステートメントで、メニューバーの項目を指定します。MENU=オプションの値が後続の MENU ステートメントで使用されます。

```
item 'File' menu=f;
item 'Edit' menu=e;
item 'Scroll' menu=s;
item 'Subset' menu=sub;
item 'Help' menu=h;
```

File メニューを設計します。 このステートメントグループでは、メニューバーの File の選択肢が定義されます。最初の ITEM ステートメントでは、File の最初の選択肢として **Goback** が指定されます。SELECTION=オプションの値は後続の SELECTION ステートメントと一致しています。このステートメントでは、その選択肢に対して発行されるコマンドとして END が指定されます。2 番目の ITEM ステートメントでは、その選択肢に対して SAVE コマンドが発行されることを指定します。

```
menu f;
   item 'Goback' selection=g;
   item 'Save';
   selection g 'end';
```

Edit メニューを設計します。 ITEM ステートメントでは、メニューバーの Edit の選択肢が定義されます。

```
menu e;
   item 'Cancel';
   item 'Add';
```

Scroll メニューを設計します。 このステートメントグループでは、メニューバーの Scroll の選択肢が定義されます。ITEM ステートメントで、引用符で囲まれた文字列が有効なコマンドではない場合、SELECTION=オプションは後続の SELECTION ステートメントと一致します。このステートメントでは、有効なコマンドが指定されます。

```
menu s;
   item 'Next Obs' selection=n;
   item 'Prev Obs' selection=p;
   item 'Top';
   item 'Bottom';
   selection n 'forward';
   selection p 'backward';
```

Subset メニューを設計します。 このステートメントグループでは、メニューバーの Subset の選択肢が定義されます。DIALOG=オプションでは、後続の DIALOG ステートメントで定義されるダイアログボックスの名前が指定されます。

```
menu sub;
   item 'Where' dialog=d1;
   item 'Where Clear';
```

Help メニューを設計します。 このステートメントグループでは、メニューバーの Help の選択肢が定義されます。SETHELP コマンドでは、FSEDIT アプリケーションのユーザー作成情報を含む HELP エントリが指定されます。HELP エントリ名の後にセミコロンを記述すると、HELP コマンドを文字列に含められます。その HELP コマンドによって、HELP エントリが呼び出されます。

```
menu h;
   item 'Keys';
   item 'About this application' selection=hlp;
```

```
selection hlp 'sethelp proclib.menucat.staffhlp.help;help';
```

ダイアログボックスを設計します。 WBUILD は SAS マクロです。PROC PMENU がフィールド番号が後に続くと予期しないようにするには、WBUILD の前に置かれた二重のパーセント記号が必要です。フィルード番号%1、%2 および%3 は、ユーザーがラジオボタンで指定した値と同じです。フィールド番号@1 は、ユーザーが入力する検索値と同じです。

```
dialog d1 '%%wbuild(%1,%2,@1,%3)';
```

地域を選択するためのラジオボタンを追加します。 TEXT ステートメントでは、1 行目の 1 列目から表示される、ダイアログボックスのテキストが指定されます。RADIOBOX ステートメントでは、ダイアログボックスにラジオボタンを表示することが指定されます。DEFAULT=では、最初のラジオボタン(Northeast)がデフォルトで選択されることが指定されます。RBUTTON ステートメントでは、ラジオボタンの相互排他的な選択肢(Pollutant A または Pollutant B)が指定されます。次の例では、北東部(NE)、北西部(NW)、南東部(SE)、南西部(SW)を指定しています。SUBSTITUTE=では、ラジオボタンが選択された場合、前述の DIALOG ステートメントの%1 のかわりに使用される値が指定されます。

```
text #1 @1 'Choose a region:';
radiobox default=1;
   rbutton #3 @5 'Northeast' substitute='NE';
   rbutton #4 @5 'Northwest' substitute='NW';
   rbutton #5 @5 'Southeast' substitute='SE';
   rbutton #6 @5 'Southwest' substitute='SW';
```

汚染物質を選択するためのラジオボタンを追加します。 TEXT ステートメントでは、8 行目(#8)の 1 列目(@1)から表示される、ダイアログボックスのテキストが指定されます。RADIOBOX ステートメントでは、ダイアログボックスにラジオボタンを表示することが指定されます。DEFAULT=では、最初のラジオボタン(Pollutant A)がデフォルトで選択されることが指定されます。RBUTTON ステートメントでは、ラジオボタンの相互排他的な選択肢(Pollutant A または Pollutant B)が指定されます。SUBSTITUTE=では、ラジオボタンが選択された場合、前述の DIALOG ステートメントの%2 のかわりに使用される値が指定されます。

```
text #8 @1 'Choose a contaminant:';
radiobox default=1;
   rbutton #10 @5  'Pollutant A' substitute='pol_a,2';
   rbutton #11 @5  'Pollutant B' substitute='pol_b,4';
```

入力フィールドを追加します。 最初の TEXT ステートメントでは、13 行目の 1 列目から表示される、ダイアログボックスのテキストが指定されます。2 番目の TEXT ステートメントでは、13 行目の 25 列目から始まる長さ 6 バイトの入力フィールドが指定されます。ユーザーがフィールドに入力する値が、前述の DIALOG ステートメントの@1 のかわりに使用されます。

```
text #13 @1 'Enter Value for Search:';
text #13 @25 len=6;
```

比較演算子を選択するためのラジオボタンを追加します。 TEXT ステートメントでは、15 行目の 1 列目から表示される、ダイアログボックスのテキストが指定されます。RADIOBOX ステートメントでは、ダイアログボックスにラジオボタンを表示することが指定されます。DEFAULT=では、最初のラジオボタン(Greater Than or Equal To)がデフォルトで選択されることが指定されます。RBUTTON ステートメントでは、ラジオボタンの相互排他的な選択肢が指定されます。SUBSTITUTE=では、ラジオボタンが選択された場合、前述の DIALOG ステートメントの%3 のかわりに使用される値が指定されます。

```
            text #15 @1 'Choose a comparison criterion:';
            radiobox default=1;
               rbutton #16 @5 'Greater Than or Equal To'
                              substitute='GE';
               rbutton #17 @5 'Less Than or Equal To'
                              substitute='LE';
               rbutton #18 @5 'Equal To' substitute='EQ';
quit;
```

ユーザーが Subset、Where の順に選択すると、次のダイアログボックスが表示されます。

アウトプット 43.1 *Where ダイアログボックスの例*



詳細

メニューバーと FSEDIT セッションの関連付け

SAS データセット PROCLIB.LAKES には、複数の湖についてのデータが存在します。2 つの汚染物質(Pollutant A および Pollutant B)について各湖でテストが行われました。Pollutant A のテストは各湖で 2 回ずつ行われ、その結果が変数 POL_A1 と POL_A2 に記録されます。Pollutant B のテストは各湖で 4 回ずつ行われ、その結果が POL_B1 から POL_B4 までの変数に記録されます。各湖は、4 つの地域のいずれかに位置します。次の例に、PROCLIB.LAKES のコンテンツリストを示します。

例のコード 43.1 *Contents of the Proclib.Lakes Data Set*

```
PROCLIB.LAKES                                  1 region    lake          pol_a1    pol_a2    pol_b1
pol_b2    pol_b3    pol_b4 NE      Carr           0.24      0.99      0.95      0.36      0.44
0.67 NE      Duraleigh     0.34      0.01      0.48      0.58      0.12      0.56 NE
Charlie       0.40      0.48      0.29      0.56      0.52      0.95 NE      Farmer         0.60
0.65      0.25      0.20      0.30      0.64 NW      Canyon         0.63      0.44      0.20
0.98      0.19      0.01 NW      Morris         0.85      0.95      0.80      0.67      0.32      0.81
NW      Golf          0.69      0.37      0.08      0.72      0.71      0.32 NW      Falls
0.01      0.02      0.59      0.58      0.67      0.02 SE      Pleasant      0.16      0.96
0.71      0.35      0.35      0.48 SE      Juliette      0.82      0.35      0.09      0.03
0.59      0.90 SE      Massey         1.01      0.77      0.45      0.32      0.55      0.66 SE
Delta         0.84      1.05      0.90      0.09      0.64      0.03 SW      Alumni         0.45
0.32      0.45      0.44      0.55      0.12 SW      New Dam       0.80      0.70      0.31
0.98      1.00      0.22 SW      Border         0.51      0.04      0.55      0.35      0.45      0.78
SW      Red           0.22      0.09      0.02      0.10      0.32 01
```

“PROCLIB.LAKES” (2161 ページ)DATA ステップでは、Proclib.Lakes が作成されます。

次のステートメントでは、PROCLIB.LAKES に対する PROC FSEDIT セッションが開始されます。

```
proc fsedit data=proclib.lakes screen=proclib.lakes;
run;
```

カスタマイズされたメニューバーのメニューを FSEDIT セッションに関連付けるには、次のいずれかを実行します。

- コマンド行で SETPMENU コマンドを入力します。この例のコマンドは次のとおりです。

  ```
  setpmenu proclib.menucat.project.pmenu
  ```

  コマンド行で PMENU ON を入力して、メニューをオンにします。
- **Command** ウィンドウに SETPMENU コマンドを入力します。
- カスタマイズされたメニューを使用したりオンにしたりする FSEDIT セッションに、SCL プログラムを含めます。たとえば、

  ```
  fseinit:
    call execcmd('setpmenu proclib.menucat.project.pmenu;
                  pmenu on;');
  return;
  init:
  return;
  main:
  return;
  term:
  return;
  ```

WBUILD マクロの機能

Southwest 地域の湖で、Pollutant A の値が.50 以上というテスト結果が出たものがあるか確かめる方法を考えてみましょう。カスタマイズされたメニュー項目がない場合は、**FSEDIT** ウィンドウで次の WHERE コマンドを発行することになります。

```
where region="SW" and (pol_a1 ge .50 or pol_a2 ge .50);
```

カスタムメニュー項目を使用する場合は、**Southwest**、**Pollutant A** を選択し、値として.50 を入力し、比較演算子として **Greater Than or Equal To** を入力します。2 つの湖(**New Dam** および **Border**)が基準を満たしています。

WBUILD マクロでは、ダイアログボックスからの 4 つの情報を使用して、WHERE コマンドが生成されます。

- 地域の値のうちの 1 つ(**NE**、**NW**、**SE**、**SW** のいずれか)が、マクロパラメータ REGION の値になります。
- **pol_a,2** か **pol_b,4** のどちらかが、PREFIX および NUMVAR マクロパラメータの値になります。カンマは WBUILD マクロに渡される値の一部で、2 つのパラメータ(PREFIX および NUMVAR)を区切るために使用されます。
- ユーザーが検索のために入力する値は、マクロパラメータ VALUE の値になります。
- ユーザーが選択した演算子は、マクロパラメータ OPERATOR の値になります。

マクロの機能を確認するために、もう一度次の例について考えてみましょう。ここでは、Southwest の湖で、Pollutant A の値が.50 以上というテスト結果が出たものがあるか確認するとします。次の表には、マクロパラメータの値が含まれます。

表 43.1 *マクロパラメータの値*

REGION	***SW***
PREFIX	**pol_a**
NUMVAR	2
VALUE	.50
OPERATOR	GE

最初の%IF ステートメントでは、ユーザーが値を入力したことを確認するためのチェックが行われます。値が入力されていれば、マクロによる WHERE コマンドの生成が開始されます。まず、マクロでは WHERE コマンドの開始部分が作成されます。

```
where region="SW" and (
```

次に、%DO ループが実行されます。NUMVAR=2 なので、Pollutant A に対して 2 回実行されます。マクロ定義では、**&prefix.&i** の期間によって、**pol_a** が **1** および **2** と連結されます。ループの反復ごとに、マクロによって PREFIX、OPERATOR および VALUE が解決され、WHERE コマンドの一部が生成されます。最初の反復では、**pol_a1 GE .50** が生成されます。

ループ内の%IF ステートメントでは、そのループが最後の反復で機能するかどうか決定するためにチェックが行われます。機能しない場合、マクロでは、個々の句の間に **OR** を挿入することによって、複合 WHERE コマンドが作成されます。WHERE コマンドの次の部分は **OR pol_a2 GE .50** になります。

Pollutant A を 2 回実行後にループが終了し、マクロで WHERE コマンドの終了部分が生成されます。

```
)
```

マクロの結果はコマンド行に置かれます。次のコードは WBUILD マクロの定義です。強調表示されているコードは、WHERE コマンドのうち、マクロでは解決されないテキスト文字列の部分を示しています。

```
%macro wbuild(region,prefix,numvar,value,operator);
       /* check to see if value is present */
    %if &value ne %then %do;
```

```
      where region="&region" AND (
              /* If the values are character,     */
              /* enclose &value in double quotation marks. */
        %do i=1 %to &numvar;
            &prefix.&i &operator &value
               /* if not on last variable,  */
               /* generate 'OR'             */
           %if &i ne &numvar %then %do;
                OR
           %end;
        %end;
         )
    %end;

%mend wbuild;
```

例 4: DATA ステップウィンドウアプリケーションのメニューの作成

要素: DIALOG ステートメント
SELECTION ステートメント

他の要素: FILENAME ステートメント

詳細

この例では、アプリケーションを定義して、ユーザーがさまざまな部門の人事データを入力し、そのデータエントリによって作成されたデータセットからのレポートを要求できるようにします。

例の最初の部分では、メニューを作成する PROC PMENU ステップが記述されます。後続セクションでは、DATA ステップウィンドウアプリケーションでのメニューの使用方法が記述されます。

タスクには次が含まれます。

- カスタマイズメニューと DATA ステップウィンドウの関連付け
- DATA ステップウィンドウのメニューの作成
- メニュー選択による SAS コードのサブミット
- ダイアログボックスを呼び出すメニュー選択肢の作成

プログラム

```
libname proclib
'SAS-data-library';

filename de     'external-file';
filename prt    'external-file';

 proc pmenu catalog=proclib.menus;

    menu select;

   item 'File' menu=f;
   item 'Data_Entry' menu=deptsde;
   item 'Print_Report' menu=deptsprt;
```

```
   menu f;
      item 'End this window' selection=endwdw;
      item 'End this SAS session' selection=endsas;
      selection endwdw 'end';
      selection endsas 'bye';

    menu deptsde;
       item 'For Dept01' selection=de1;
       item 'For Dept02' selection=de2;
       item 'Other Departments' dialog=deother;

       selection de1 'end;pgm;include de;change xx 01;submit';
       selection de2 'end;pgm;include de;change xx 02;submit';

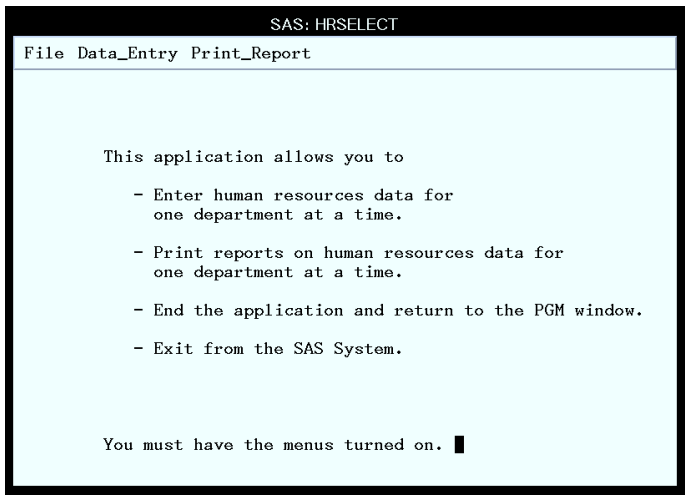
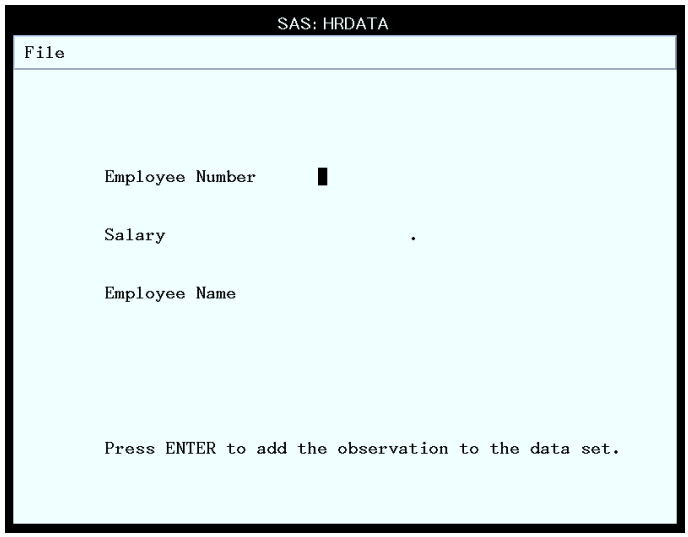
       dialog deother 'end;pgm;include de;c deptxx @1;submit';
          text #1 @1 'Enter department name';
          text #2 @3 'in the form DEPT99:';
          text #2 @25 len=7;

      menu deptsprt;
         item 'For Dept01' selection=prt1;
         item 'For Dept02' selection=prt2;
         item 'Other Departments' dialog=prother;

         selection prt1
               'end;pgm;include prt;change xx 01 all;submit';
         selection prt2
               'end;pgm;include prt;change xx 02 all;submit';

      dialog prother 'end;pgm;include prt;c deptxx @1 all;submit';
         text #1 @1 'Enter department name';
         text #2 @3 'in the form DEPT99:';
         text #2 @25 len=7;

   run;

     menu entrdata;
        item 'File' menu=f;
        menu f;
           item 'End this window' selection=endwdw;
           item 'End this SAS session' selection=endsas;
           selection endwdw 'end';
           selection endsas 'bye';
   run;
quit;
```

プログラムの説明

PROCLIB ライブラリを宣言します。PROCLIB ライブラリは、メニュー定義の格納に使用されます。

```
libname proclib
'SAS-data-library';
```

DE および PRT ファイル名を宣言します。FILENAME ステートメントでは、ウィンドウ作成プログラムを格納する外部ファイルが定義されます。

```
filename de     'external-file';
filename prt    'external-file';
```

メニュー定義を格納するカタログを指定します。 メニュー定義は PROCLIB.MENUCAT カタログに格納されます。

```
proc pmenu catalog=proclib.menus;
```

カタログエントリ名を指定します。 MENU ステートメントで、カタログエントリ名として SELECT を指定します。メニューはカタログエントリ PROCLIB.MENUS.SELECT.PMENU に格納されます。

```
menu select;
```

メニューバーを設計します。 ITEM ステートメントでは、メニューバー上の 3 項目が指定されます。MENU=オプションの値が後続の MENU ステートメントで使用されます。

```
item 'File' menu=f;
item 'Data_Entry' menu=deptsde;
item 'Print_Report' menu=deptsprt;
```

File メニューを設計します。 このステートメントグループでは、File の選択肢が定義されます。SELECTION=オプションの値が後続の SELECTION ステートメントで使用されます。

```
menu f;
   item 'End this window' selection=endwdw;
   item 'End this SAS session' selection=endsas;
   selection endwdw 'end';
   selection endsas 'bye';
```

Data_Entry メニューを設計します。 このステートメントグループでは、メニューバーの Data_Entry の選択肢が定義されます。ITEM ステートメントでは、Data_Entry の下に For Dept01 および For Dept02 を表示することが指定されます。SELECTION=オプションの値は後続の SELECTION ステートメントと同じです。これには、実際にサブミットされるコマンドの文字列が含まれます。DIALOG=オプションの値は後続の DIALOG ステートメントと同じです。ここには、この項目の選択時に表示されるダイアログボックスが記述されます。

```
menu deptsde;
   item 'For Dept01' selection=de1;
   item 'For Dept02' selection=de2;
   item 'Other Departments' dialog=deother;
```

Data_Entry メニューのコマンドを指定します。 ユーザーが For Dept01 または ForDept02 を選択すると、一重引用符で囲まれたコマンドがサブミットされます。END コマンドでは、現在のウィンドウが終了し、PROGRAM EDITOR ウィンドウに戻って、追加コマンドをサブミットできるようになります。INCLUDE コマンドでは、データエントリウィンドウを作成する SAS ステートメントがインクルードされます。CHANGE コマンドでは、インクルードされたプログラムの DATA ステートメントが変更され、それにより正しいデータセットが作成されます。SUBMIT コマンドでは、DATA ステッププログラムがサブミットされます。

```
   selection de1 'end;pgm;include de;change xx 01;submit';
   selection de2 'end;pgm;include de;change xx 02;submit';
```

DEOTHER ダイアログボックスを設計します。 DIALOG ステートメントでは、ユーザーが Other Departments を選択したときに表示するダイアログボックスが定義されます。DIALOG ステートメントでは、コマンド文字列が変更されます。これにより、ユーザーの入力する部門名を使用して、インクルードされる SAS プログラムの **deptxx** が変更されます。最初の 2 つの TEXT ステートメントでは、ダイアログボックスに表示される

テキストが指定されます。3 番目の TEXT ステートメントでは、入力フィールドが指定されます。このフィールドに入力される名前が、DIALOG ステートメントの@1 のかわりに使用されます。

```
dialog deother 'end;pgm;include de;c deptxx @1;submit';
   text #1 @1 'Enter department name';
   text #2 @3 'in the form DEPT99:';
   text #2 @25 len=7;
```

Print_Report メニューを設計します。このステートメントグループでは、Print_Report 項目の選択肢が定義されます。これらの ITEM ステートメントでは、メニューに For Dept01 および For Dept02 を表示することが指定されます。SELECTION=オプションの値は後続の SELECTION ステートメントと同じです。これには、実際にサブミットされるコマンドの文字列が含まれます。

```
menu deptsprt;
   item 'For Dept01' selection=prt1;
   item 'For Dept02' selection=prt2;
   item 'Other Departments' dialog=prother;
```

Print_Report メニューのコマンドを指定します。ユーザーが For Dept01 または ForDept02 を選択すると、一重引用符で囲まれたコマンドがサブミットされます。END コマンドでは、現在のウィンドウが終了し、PROGRAM EDITOR ウィンドウに戻って、追加コマンドをサブミットできるようになります。INCLUDE コマンドでは、レポートを印刷する SAS ステートメントがインクルードされます。詳細については、“プログラムの印刷” (1329 ページ)を参照してください。)CHANGE コマンドでは、インクルードされたプログラムの PROC PRINT ステップが変更され、それにより、正しいデータセットが印刷されます。SUBMIT コマンドでは、PROC PRINT プログラムがサブミットされます。

```
   selection prt1
         'end;pgm;include prt;change xx 01 all;submit';
   selection prt2
         'end;pgm;include prt;change xx 02 all;submit';
```

PROTHER ダイアログボックスを設計します。DIALOG ステートメントでは、ユーザーが Other Departments を選択したときに表示するダイアログボックスが定義されます。DIALOG ステートメントでは、コマンド文字列が変更されます。これにより、ユーザーの入力する部門名を使用して、インクルードされる SAS プログラムの **deptxx** が変更されます。最初の 2 つの TEXT ステートメントでは、ダイアログボックスに表示されるテキストが指定されます。3 番目の TEXT ステートメントでは、入力フィールドが指定されます。このフィールドに入力される名前が、DIALOG ステートメントの@1 のかわりに使用されます。

```
dialog prother 'end;pgm;include prt;c deptxx @1 all;submit';
   text #1 @1 'Enter department name';
   text #2 @3 'in the form DEPT99:';
   text #2 @25 len=7;
```

この RUN グループを終了します。

```
run;
```

---

2 つ目のカタログエントリとメニューバーを指定します。MENU ステートメントでは、この RUN グループが作成するカタログエントリの名前として ENTRDATA が指定されます。メニューバー上の項目は File のみです。使用可能な選択肢は、End this window と End this SAS session です。

```
menu entrdata;
   item 'File' menu=f;
```

```
      menu f;
         item 'End this window' selection=endwdw;
         item 'End this SAS session' selection=endsas;
         selection endwdw 'end';
         selection endsas 'bye';
   run;
quit;
```

その他の例

メニューとウィンドウの関連付け

最初のステートメントグループによって、アプリケーションのプライマリウィンドウが定義されます。これらのステートメントは、HRWDW ファイル参照名によって参照されるファイルに格納されます。

WINDOW ステートメントでは、**HRSELECT** ウィンドウが作成されます。MENU=では、PROCLIB.MENUS.SELECT.PMENU エントリとこのウィンドウが関連付けられます。

```
data _null_;
   window hrselect menu=proclib.menus.select
   #4  @10 'This application allows you to'
   #6  @13 '- Enter human resources data for'
   #7  @15 'one department at a time.'
   #9  @13 '- Print reports on human resources data for'
   #10 @15 'one department at a time.'
   #12 @13 '- End the application and return to the PGM window.'
   #14 @13 '- Exit from the SAS System.'
   #19 @10 'You must have the menus turned on.';
```

DISPLAY ステートメントでは、**HRSELECT** ウィンドウが表示されます。

```
   display hrselect;
run;
```

DISPLAY ステートメントによって表示される **HRSELECT** ウィンドウ

図 43.6 *HRSELECT ウィンドウ*



データエントリプログラムの使用

ユーザーが **HRSELECT** ウィンドウのメニューバーから **Data_Entry** を選択すると、メニューが表示されます。ユーザーが、リストされている部門のいずれかを選択するか、異なる部門の入力を選択すると、次のステートメントが呼び出されます。これらのステートメントは、DE ファイル参照名によって参照されるファイルに格納されます。

WINDOW ステートメントでは、**HRDATA** ウィンドウが作成されます。MENU=では、PROCLIB.MENUS.ENTRDATA.PMENU エントリとウィンドウが関連付けられます。

```
data proclib.deptxx;
    window hrdata menu=proclib.menus.entrdata
    #5  @10 'Employee Number'
    #8  @10 'Salary'
    #11 @10 'Employee Name'
    #5  @31 empno $4.
    #8  @31 salary 10.
    #11 @31 name $30.
    #19 @10 'Press ENTER to add the observation to the data set.';
```

DISPLAY ステートメントでは、**HRDATA** ウィンドウが表示されます。

```
    display hrdata;
run;
```

%INCLUDE ステートメントでは、ファイル HRWDW のステートメントがリコールされます。HRWDW のステートメントでは、プライマリウィンドウが再表示されます。次を参照してください。HRSELECT ウィンドウ (1327 ページ)

```
filename hrwdw 'external-file';
%include hrwdw;
run;
```

PROC PMENU ステップの SELECTION および DIALOG ステートメントによってこのプログラムの DATA ステートメントが変更され、データセットの作成時に正しい部門名が使用されるようになります。したがって、ユーザーが **Other Departments** を選択して **DEPT05** を入力すると、DATA ステートメントは、DIALOG ステートメントのコマンド文字列によって、次のように変更されます。

```
 data proclib.dept05;
```

次の図は、データエントリウィンドウ **HRDATA** を示しています。

図 43.7 *HRDATA ウィンドウ*



プログラムの印刷

ユーザーがメニューバーから **Print_Report** を選択すると、メニューが表示されます。ユーザーが、リストされている部門のいずれかを選択するか、異なる部門の入力を選択すると、次のステートメントが呼び出されます。これらのステートメントは、PRT ファイル参照名によって参照される外部ファイルに格納されます。

PROC PRINTTO で、出力が外部ファイルに送られます。

```
   proc printto
file='external-file' new;
   run;

   libname proclib
'SAS-data-library';

   proc print data=proclib.deptxx;
      title 'Information for deptxx';
   run;
```

PROC PRINTTO ステップでは、デフォルトの出力先が復元されます。46 章, “PRINTTO プロシジャ,” (1427 ページ)を参照してください。

```
   proc printto;
   run;
```

%INCLUDE ステートメントでは、ファイル HRWDW のステートメントがリコールされます。HRWDW のステートメントでは、プライマリウィンドウが再表示されます。

```
   filename hrwdw 'external-file';
   %include hrwdw;
   run;
```

例 5: メニューと FRAME アプリケーションの関連付け

要素: ITEM ステートメント

MENU ステートメント

他の要素: SAS/AF ソフトウェア

詳細

この例では、FRAME エントリに対するメニューを作成し、メニューを SAS/AF ソフトウェアからの FRAME エントリと関連付けるために必要な操作について説明します。

プログラム

```
libname proclib
'SAS-data-library';

proc pmenu catalog=proclib.menucat;

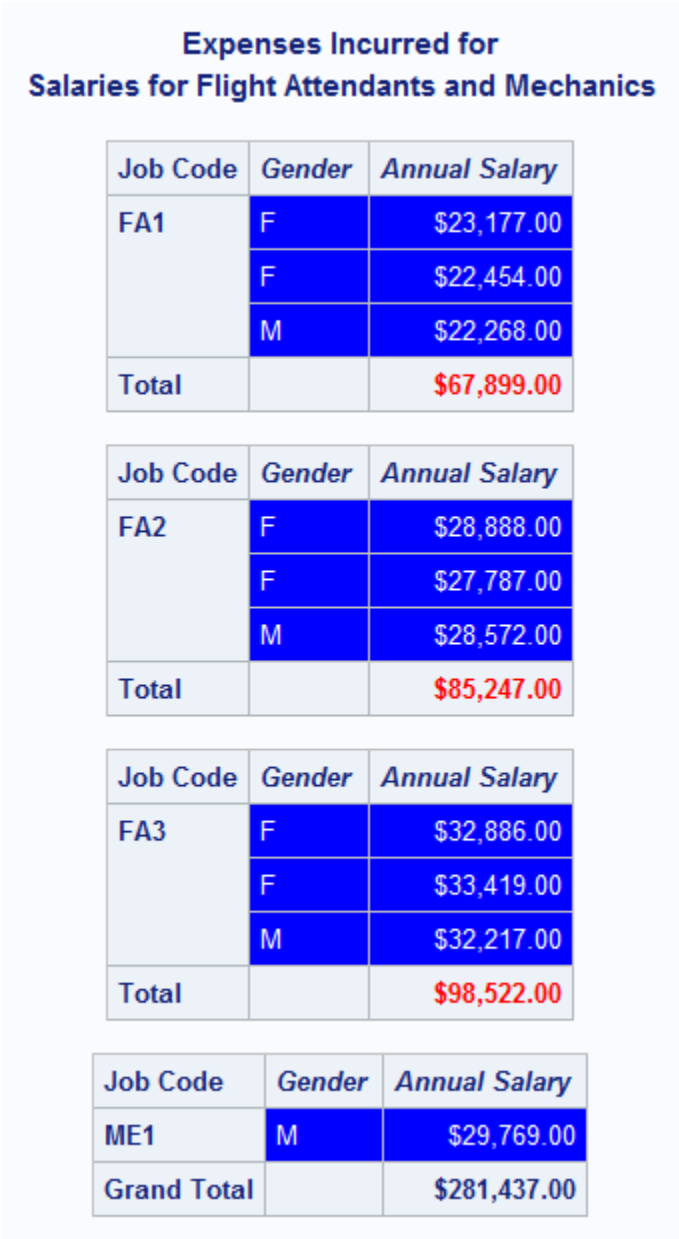
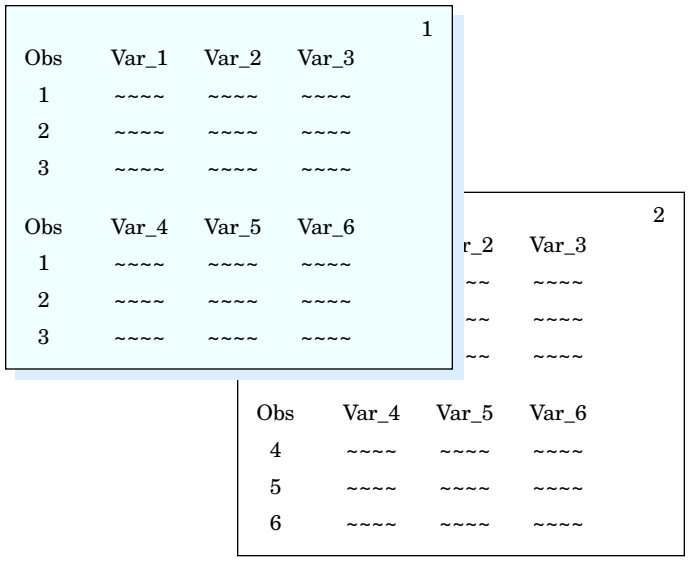
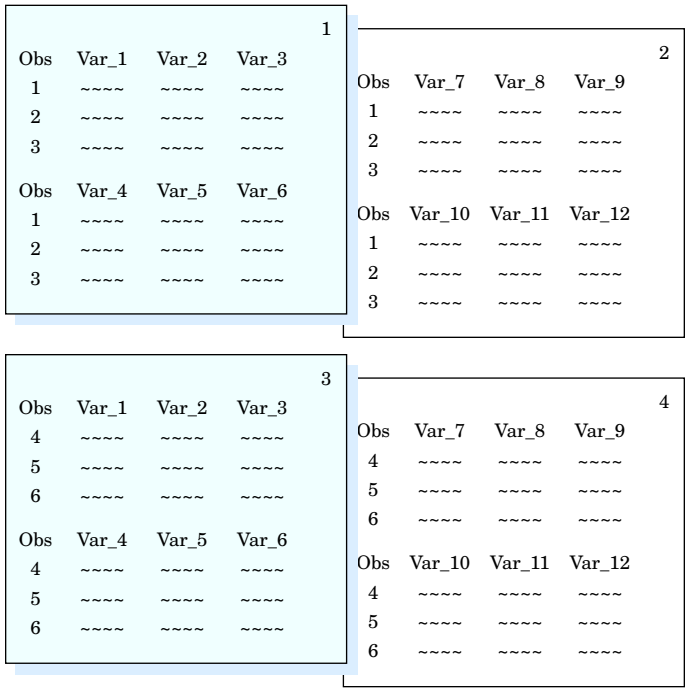
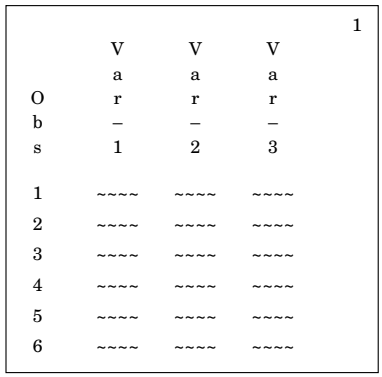
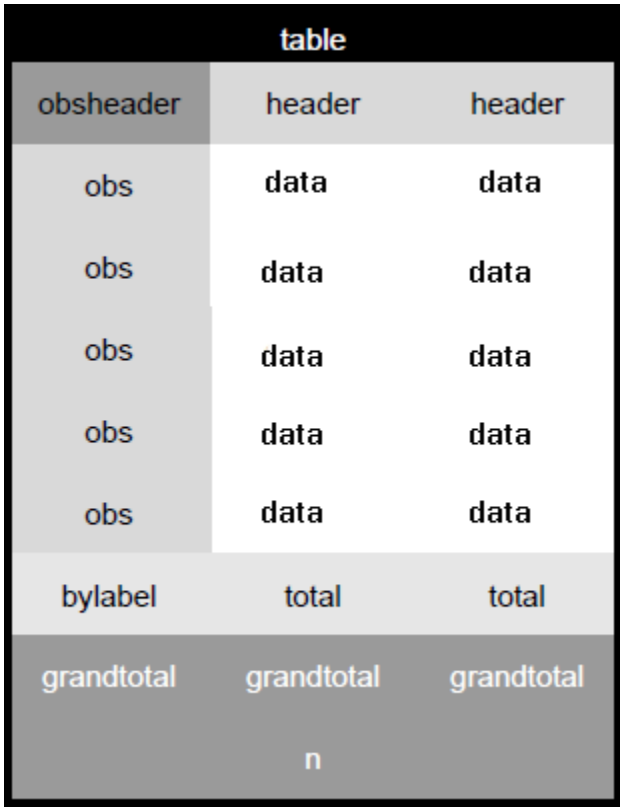
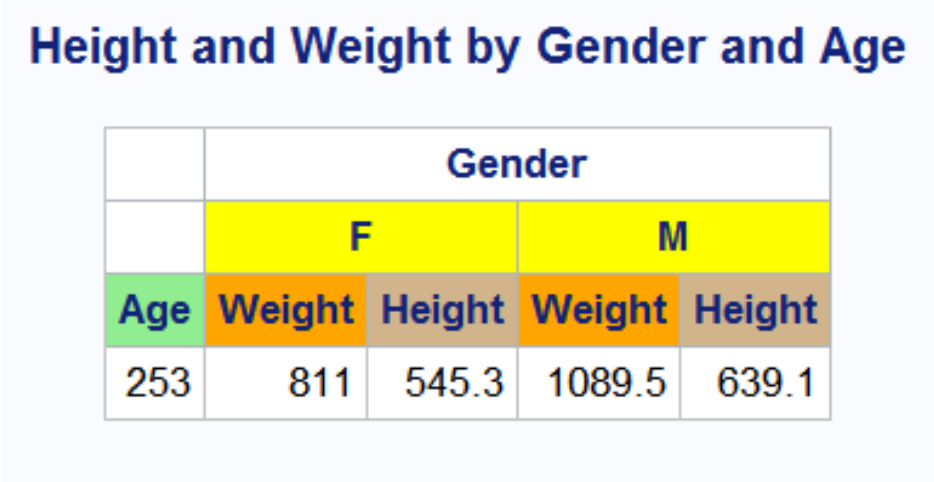
   menu frame;

      item 'File' menu=f;
      item 'Help' menu=h;

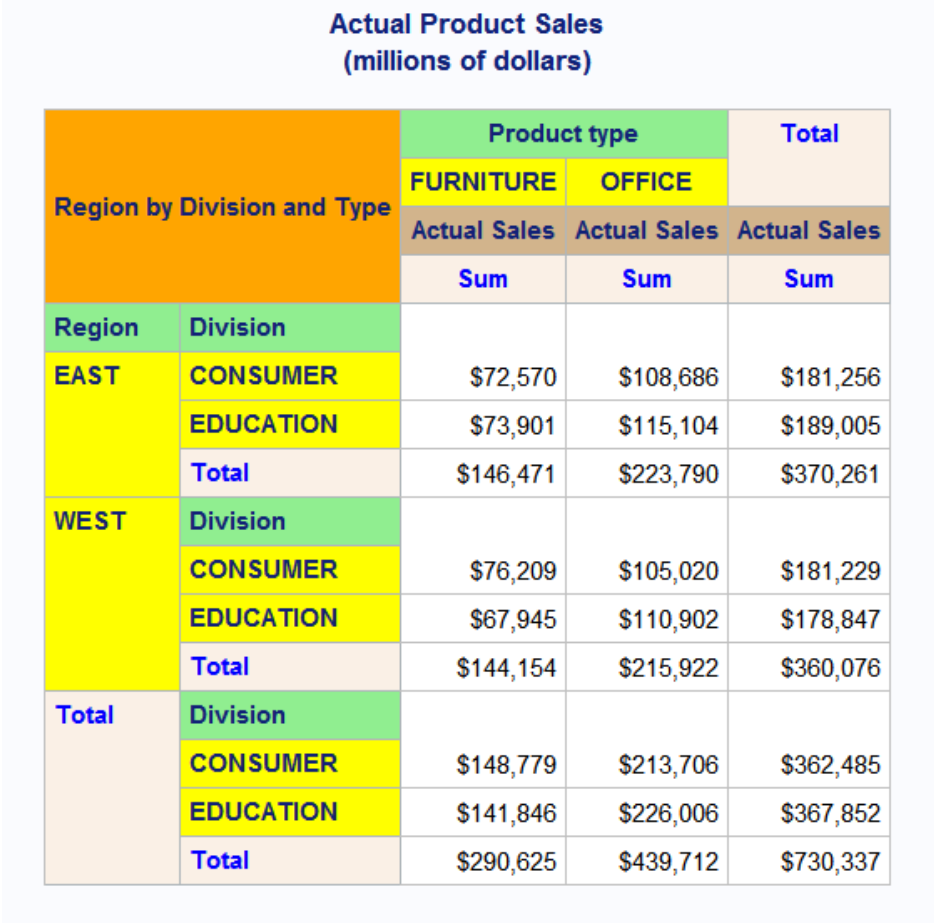
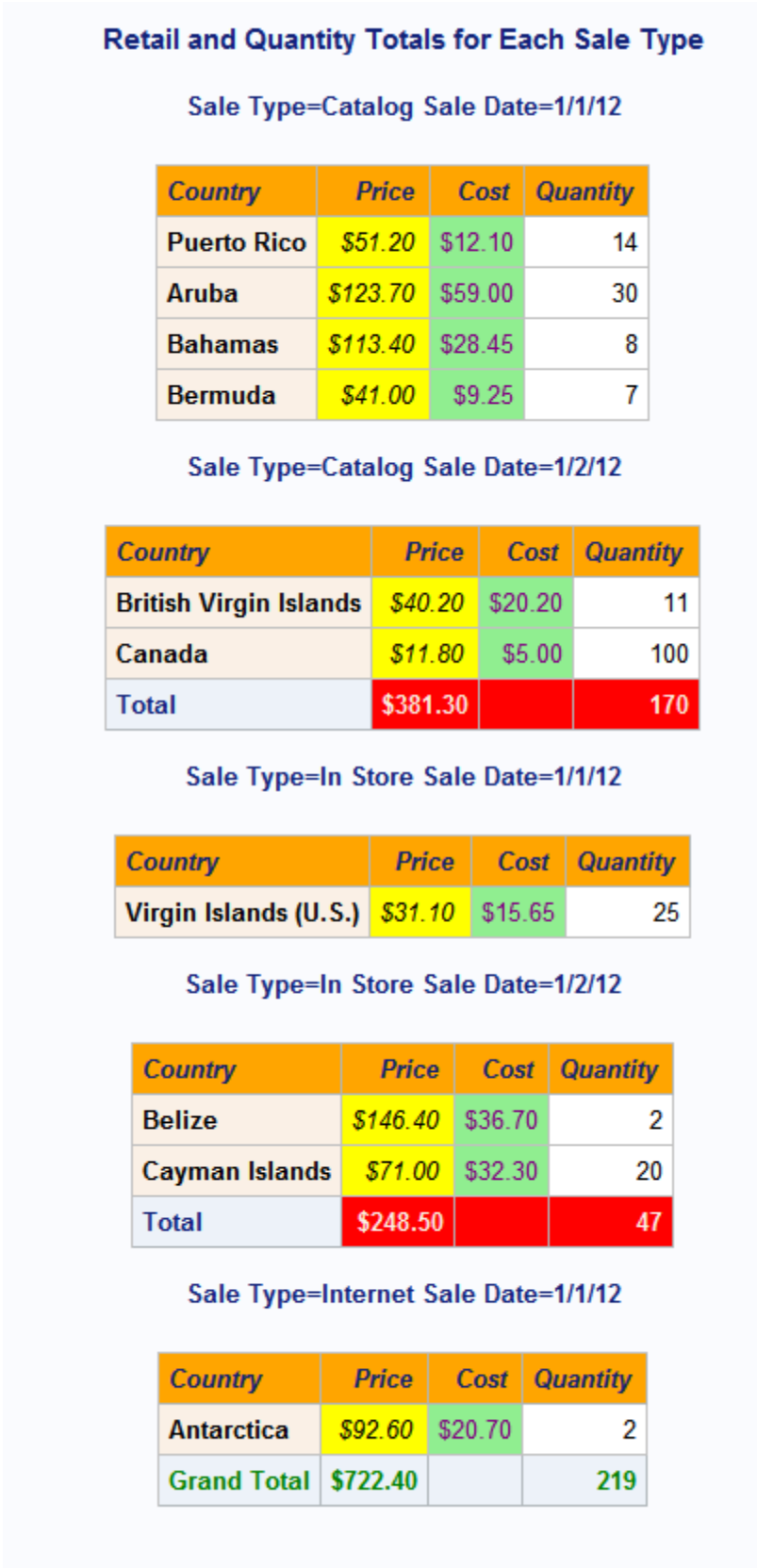
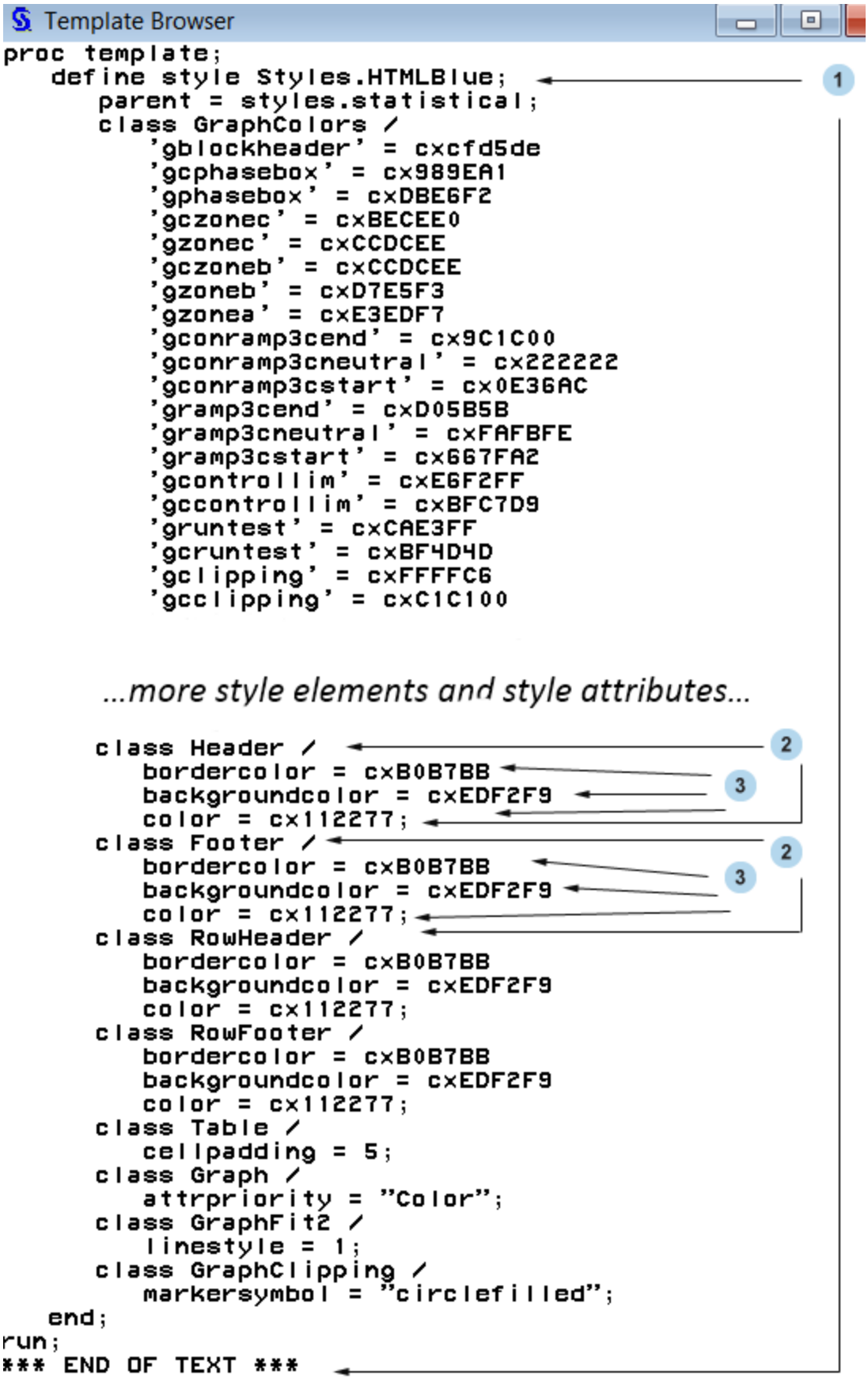
      menu f;
         item 'Cancel';
         item 'End';

      menu h;
         item 'About the application' selection=a;
         item 'About the keys'   selection=k;

            selection a 'sethelp proclib.menucat.app.help;help';
            selection k 'sethelp proclib.menucat.keys.help;help';
run;
quit;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリは、メニュー定義の格納に使用されます。

```
libname proclib
'SAS-data-library';
```

メニュー定義を格納するカタログを指定します。 メニュー定義は PROCLIB.MENUCAT カタログに格納されます。

```
proc pmenu catalog=proclib.menucat;
```

カタログエントリ名を指定します。 MENU ステートメントで、カタログエントリ名として FRAME を指定します。メニューはカタログエントリ PROCLIB.MENUS.FRAME.PMENU に格納されます。

```
   menu frame;
```

メニューバーを設計します。 ITEM ステートメントで、メニューバーの項目を指定します。MENU=の値は、後続の MENU ステートメントと一致します。

```
      item 'File' menu=f;
      item 'Help' menu=h;
```

File メニューを設計します。 MENU ステートメントは、前述の ITEM ステートメントの MENU=オプションと同じです。ITEM ステートメントでは、メニューバーの File で使用可能な選択肢が指定されます。

```
menu f;
   item 'Cancel';
   item 'End';
```

Help メニューを設計します。 MENU ステートメントは、前述の ITEM ステートメントの MENU=オプションと同じです。ITEM ステートメントでは、メニューバーの Help で使用可能な選択肢が指定されます。SELECTION=オプションの値は、後続の SELECTION ステートメントと同じです。

```
menu h;
   item 'About the application' selection=a;
   item 'About the keys'   selection=k;
```

Help メニューのコマンドを指定します。 SETHELP コマンドでは、このアプリケーションのユーザー作成情報を含む HELP エントリが指定されます。HELP エントリ名の後にセミコロンを記述すると、HELP コマンドを文字列に含められます。その HELP コマンドによって、HELP エントリが呼び出されます。

```
            selection a 'sethelp proclib.menucat.app.help;help';
            selection k 'sethelp proclib.menucat.keys.help;help';
run;
quit;
```

メニューと FRAME を関連づける操作

1. FRAME エントリの BUILD 環境で、メニューバーから**表示** ⇨ **プロパティウィンドウ**を選択します。
2. **プロパティ**ウィンドウで、pmenuEntry 属性名の**値**フィールドを選択します。**エントリの選択**ウィンドウが表示されます。
3. **エントリの選択**ウィンドウで、メニューを作成する PROC PMENU ステップで指定されるカタログエントリ名を入力します。
4. FRAME のメニューバーから**作成** ⇨ **テスト**を選択して、FRAME をテストします。次の表示で、メニューと FRAME が現在関連付けられていることが示されます。

図 43.8 FRAME ウィンドウ

FRAME エントリを使用したプログラミングの詳細については、*Getting Started with SAS/AF(R) 9.3 and Frames* を参照してください。

44 章

PRESENV プロシジャ

PRESENV プロシジャの動作について

PRESENV プロシジャは、ある SAS セッションから別の SAS セッションにかけて、SAS コードのすべてのグローバルステートメントとマクロ変数を保存します。このプロシジャが SAS セッションの最後に呼び出されると、グローバルステートメントとマクロ変数はすべて、1 つのファイルに書き込まれます。Work データセットとマクロカタログは、補助ディレクトリに書き込まれます。これで SAS セッションを 終了できます。後でこのセッションを再開し、保存したグローバルステートメントとマクロ変数設定を再度実行することができます。Work データセットを現行の Work ディレクトリにコピーバックすれば、このセッションを再開させることができます。

PRESENV プロシジャは PRESENV システムオプションとともに機能して、SAS プログラムとデータセットを保存します。このオプションはいつでも有効または無効にできます。PRESENV システムオプションが無効になっているときは、グローバルステートメントコレクションが停止されます。有効にすればコレクションは再開します。このコレクションが破棄されることはありません。ただし、このオプションは初回のオプション有効化まで開始されません。

この機能は、後で別のノードで Enterprise Guide (EG)セッションを終了し再開する必要があるグリッド環境においては、非常に便利です。

保存できるグローバルステートメント

OPTIONS ステートメントにおいて、または起動時に、PRESENV システムオプションを有効してジョブの最後に PROC PRESENV を実行すれば、プログラムで使用している次のグローバルステートメントがメモリ内で収集されます。

AXIS	LOCK
CATNAME	MISSING
FILENAME	OPTIONS
FOOTNOTE	PATTERN
GOPTIONS	SASFILE
LEGEND	SYMBOL
LIBNAME	TITLE

プログラムで使用しているマクロ変数もメモリ内で収集されますが、X コマンドなどの別のグローバルステートメントは収集されません。プログラム実行中にコンパイルされるマクロは、Work ディレクトリに保存され、このディレクトリは PROC PRESENV の実行の一部としてコピーされます。

構文: PRESENV プロシジャ

要件 PRESENV システムオプションを PRESENV(デフォルトは NOPRESENV)に設定してグローバルステートメントを保持する必要があります。これで PROC PRESENV により、これらグローバルステートメントのコピーが指定したファイルに保存されます。グローバルステートメントは PRESENV システムオプションが有効になった時点から保持されます。

PROC PRESENV PERMDIR=*libref* SASCODE=*fileref* <SHOW_COMMENTS>;
RUN;

ステートメント	タスク
"PROC PRESENV ステートメント"	1 つの SAS セッションから別の SAS セッションにかけて、すべてのグローバルステートメントとマクロ変数を保持する

PROC PRESENV ステートメント

1 つの SAS セッションから別の SAS セッションにかけて、すべてのグローバルステートメントとマクロ変数を保持します。

構文

PROC PRESENV PERMDIR=*libref* SASCODE=*fileref* <SHOW_COMMENTS>;

オプション引数の要約

SHOW_COMMENTS

すべてのグローバルステートメントを表示します。冗長なグローバルステートメントがコメントアウトされます。

必須引数

PERMDIR=*libref*
Work データセット、カタログ、マクロのすべてが書き込まれる libref を指定します。

SASCODE=*fileref*
SAS プログラムが書き込まれる fileref を指定します。SAS プログラムには、環境を再保存する場合に必要なコードの全てが含まれています。

オプション引数

SHOW_COMMENTS
すべてのグローバルステートメントを表示します。冗長なグローバルステートメントがコメントアウトされます。このオプションを使用しなければ、グローバルステートメントは抑制されます。

ヒント このオプションは、生成中のテキスト数を大幅に増加させる可能性があるため、プログラムをデバッグする場合にのみ使用してください。

PROC PRESENV の実行

環境を保持するには、ジョブの最後に PRESENV プロシジャを実行します。

```
proc presenv save permdir=permdir sascode=sascode;
run;
```

PERMDIR の値は、Work データセットとカタログのすべて(work.sasmacr を含む)が書き込まれる libref です。SASCODE の値は SAS プログラムが書き込まれる fileref です。SAS プログラムには、環境を再保存する場合に必要なコードの全てが含まれています。

後続ジョブでの環境の復元

後続ジョブで環境を復元するには、PRESENV システムオプションを含まない SAS を呼び出して次のコードをサブミットします。

```
%include 'restore-file';
run;
```

Restore-file は、元のジョブの SASCODE=引数の fileref に関連付けられているファイル名です。このプログラムを実行すると、すべてのマクロ、マクロ変数、オプション、グローバルステートメントがそれぞれ元の値に戻ります。

45 章

PRINT プロシジャ

概要: PRINT プロシジャ

PRINT プロシジャの動作について

PRINT プロシジャはすべての変数または一部の変数を使用して、SAS データセットのオブザベーションを印刷します。簡易 LISTING から、データの分類および数値変数に関する合計や小計の計算を行う高度にカスタマイズされたレポートに至るまで、さまざまなレポートの作成が可能です。

簡易 LISTING レポート

次の出力に、生成可能な最も単純な種類のレポートを示します。出力を生成するステートメントは次のとおりです。“例 1: 印刷する変数の選択” (1376 ページ)では、データセット EXPREV を作成します。

```
options obs=10;

proc print data=exprev;
run;
```

The SAS System

Obs	Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
1	Antarctica	99999999	1/1/12	1/7/12	Internet	2	92.6	20.70
2	Puerto Rico	99999999	1/1/12	1/5/12	Catalog	14	51.2	12.10
3	Virgin Islands (U.S.)	99999999	1/1/12	1/4/12	In Store	25	31.1	15.65
4	Aruba	99999999	1/1/12	1/4/12	Catalog	30	123.7	59.00
5	Bahamas	99999999	1/1/12	1/4/12	Catalog	8	113.4	28.45
6	Bermuda	99999999	1/1/12	1/4/12	Catalog	7	41.0	9.25
7	Belize	120458	1/2/12	1/2/12	In Store	2	146.4	36.70
8	British Virgin Islands	99999999	1/2/12	1/5/12	Catalog	11	40.2	20.20
9	Canada	99999999	1/2/12	1/5/12	Catalog	100	11.8	5.00
10	Cayman Islands	120454	1/2/12	1/2/12	In Store	20	71.0	32.30

レポートのカスタマイズ

次の HTML レポートは、ODS を使用して PROC PRINT によって生成されるカスタマイズされたレポートです。このレポートを作成するステートメントを使用して、次のことを行います。

- タイトルと列ヘッダーをカスタマイズする
- レポートの表示をカスタマイズする
- 数値出力でドル記号とカンマを使用する
- 変数の順序をレポートに選択的に含めて制御する
- JobCode 別にデータをグループ化する
- ジョブコード別およびジョブコード全体に対するサラリーの数値を合計し、要約行と総計行の各ラベルを追加する

このレポートを作成するプログラムの説明については、“プログラム:STYLE オプションを使用した HTML レポートの作成” (1419 ページ)を参照してください。

図 45.1 ODS を使用して PROC PRINT によって生成されたカスタマイズレポート



Expenses Incurred for
Salaries for Flight Attendants and Mechanics

Job Code	Gender	Annual Salary
FA1	F	$23,177.00
	F	$22,454.00
	M	$22,268.00
Total		$67,899.00

Job Code	Gender	Annual Salary
FA2	F	$28,888.00
	F	$27,787.00
	M	$28,572.00
Total		$85,247.00

Job Code	Gender	Annual Salary
FA3	F	$32,886.00
	F	$33,419.00
	M	$32,217.00
Total		$98,522.00

Job Code	Gender	Annual Salary
ME1	M	$29,769.00
Grand Total		$281,437.00

概念: PRINT プロシジャ

PROC PRINT 出力について

デフォルトでは、ウィンドウ環境で SAS を運用するときは PROC PRINT により HTML レポートが生成されます。他のオペレーティングモードではいずれも、デフォルトの出力先は LISTING です。PRINT プロシジャのステートメントおよびオプションを使用すれば当該レポートの体裁を変更できます。PRINT プロシジャステートメントである

PROC PRINT、BY、PAGEBY、SUMBY、ID、SUM、VAR は、レポートの内容を管理するものです。各ステートメントのオプションによりレポートの体裁が管理されます。

レポートの ODS 出力先を変更するには、PROC PRINT ステートメントの前に ODS ステートメントを使用してください。HTML 出力が不要な場合には、本プロシジャを運用する前に必ず ODS の HTML 出力先を閉じてください。ODS の使用の詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

本プロシジャが生成するレポートの種類のサンプリングについては、PRINT プロシジャ例 (1376 ページ)を参照してください。

デフォルトの ODS 出力先である HTML のページレイアウト

ODS の HTML 出力のページの縦や横の長さに制限はありません。従って、1 つのテーブル内の各オブザベーションは単一行に印刷され、レポートが印刷を指定するオブザベーションはすべて、HTML 出力の 1 ページに表示されます。

デフォルトでは、PROC PRINT が動作するたびに、SAS により HTML 出力の後に改ページが追加されます。横罫線で出力を分断すれば、改ページがレンダリングされます。詳細については、“ODS HTML Statement ” (*SAS Output Delivery System: User's Guide*)を参照してください。

ページのサイズが制限されている場合のページレイアウト

Observations

PROC PRINT は、ページの縦横の長さが制限されている出力を生成する ODS 出力先の 1 ページ上の全オブザベーションに対しては、同一レイアウトを使用します。ODS 出力先には RTF、PDF、LISTING などがあります。まず、次の図のように、単一行のオブザベーションの印刷を試行します。

図 45.2 *単一行のオブザベーションの印刷*

1

Obs	Var_1	Var_2	Var_3
1	~~~~	~~~~	~~~~
2	~~~~	~~~~	~~~~
3	~~~~	~~~~	~~~~
4	~~~~	~~~~	~~~~
5	~~~~	~~~~	~~~~
6	~~~~	~~~~	~~~~

PROC PRINT は、単一行のすべての変数に適合しない場合、オブザベーションを 2 つ以上のセクションに分割し、各行の始めにオブザベーション番号または ID 変数を印刷します。たとえば、次の図では、PROC PRINT は各ページの最初のセクションの最初の 3 つの変数の値と、各ページの 2 番目のセクションの次の 3 つの変数の値を印刷します。

図 45.3 *1 ページの複数のセクションへのオブザベーションの分割*



PROC PRINT が 1 ページのすべての変数に適合しない場合、プロシジャはすべての変数を印刷するまで、同じオブザベーションを含む後続ページを印刷します。たとえば、次の図では、PROC PRINT は最初の 2 ページを使用して最初の 3 つのオブザベーションの値を印刷し、その次の 2 ページを使用してその他のオブザベーションの値を印刷します。

図 45.4 *複数ページにまたがるオブザベーションの分割*



注: リスと出力先については、PROC PRINT ステートメントの ROWS=オプションでページレイアウトを変更できます。(ROWS=オプション (1350 ページ)に関する内容を参照してください)。

注: データセットが RADIX アドレス指定可能でない場合、PROC PRINT による出力はやや異なります。バージョン 6 の圧縮ファイルは RADIX アドレス指定可能では

ありませんが、バージョン 7 以降では圧縮ファイルが RADIX 指定可能です。(データの整合性は損なわれません。本プロシジャは単にオブザベーションに異なる番号を付与するものです。

列ヘッダー

PROC PRINT が列ヘッダーを横方向または縦方向に印刷するかどうかは、スペースの数で指定します。図 45.2 (1341 ページ), 図 45.3 (1342 ページ)および 図 45.4 (1342 ページ)は、すべて横方向のヘッダーを示します。次の図に、縦方向のヘッダーを示します。

図 45.5 *縦方向のヘッダーの使用*



				1
	V	V	V	
	a	a	a	
O	r	r	r	
b	–	–	–	
s	1	2	3	
1	~~~~	~~~~	~~~~	
2	~~~~	~~~~	~~~~	
3	~~~~	~~~~	~~~~	
4	~~~~	~~~~	~~~~	
5	~~~~	~~~~	~~~~	
6	~~~~	~~~~	~~~~	

注: LABEL を使用し、少なくとも 1 つの変数にラベルがある場合、HEADING=VERTICAL を指定しない限り、PROC PRINT はすべての列ヘッダーを横方向に印刷します。

列幅

デフォルトでは、PROC PRINT は変数のフォーマットされた幅を列幅として使用します。(WIDTH=オプションは、リスと出力先に対して本デフォルト動作を無効にします。) 変数にフィールド幅を明示的に指定する出力形式が含まれていない場合、PROC PRINT はそのページの変数の最大幅データ値を列幅として使用します。

文字変数のフォーマットされた値またはフォーマットされていない文字変数のデータ幅がすべての ID 変数の長さを引いたページサイズを超える場合、PROC PRINT は値を切り捨てることがあります。次の状況について考えます。

- ページサイズが 80 です。
- IdNumber が長さ 10 の文字変数です。ID 変数として使用されます。
- State は長さ 2 の文字変数です。ID 変数として使用されます。
- コメントは長さ 200 の文字変数です。

PROC PRINT は、1 行にこれら 3 つの変数を印刷する際、2 つの ID 変数に 14 の印刷位置を、それぞれの後にスペースを使用します。この調整により COMMENT の印刷位置は、80–14 または 66 のままです。それよりも長い COMMENT の値は切り捨てられます。

WIDTH=はリスと出力先の列幅を制御します。

注: 列幅は、可変幅だけでなく、列ヘッダーの長さの影響も受けます。列ヘッダーが長いと、WIDTH=の有用性が低下することがあります。

構文: PRINT プロシジャ

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化して、ログに適切に書き込まれるようにする必要があります。

Output Delivery System をサポートします。詳細については、“Output Delivery System: Basic Concepts” (*SAS Output Delivery System: User's Guide*)を参照してください。

ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。*SAS ステートメント: リファレンス*を参照してください。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

PROC PRINT <*option(s)*>;
  BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …> <NOTSORTED>;
    PAGEBY *BY-variable*;
    SUMBY *BY-variable*;
  ID *variable(s)*
    </ STYLE <(*location(s)*)>=<*style-override*>>;
  SUM *variable(s)*
    </ STYLE <(*location(s)*)>=<*style-override*>>;
  VAR *variable(s)*
    </ STYLE <(*location(s)*)>=<*style-override*> >;

ステートメント	タスク	例
“PROC PRINT ステートメント”	データセットのオブザベーションを印刷します	Ex. 1, Ex. 2, Ex. 3, Ex. 5, Ex. 8
“BY ステートメント”	BY グループごとに別のレポートセクションを生成します	Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 8
“ID ステートメント”	オブザベーション番号ではなく、リストする変数のフォーマットされた値によってオブザベーションを識別します	Ex. 7
“PAGEBY ステートメント”	ページが埋まる前に発生するページ排出を制御します	Ex. 3
“SUMBY ステートメント”	レポートに表示する合計数を制限します	Ex. 4, Ex. 5, Ex. 6, Ex. 8
“SUM ステートメント”	数値変数の合計値	Ex. 6
“VAR ステートメント”	レポートに表示する変数を選択し、その順序を決定します	Ex. 1, Ex. 2, Ex. 8

PROC PRINT ステートメント

一部の変数またはすべての変数を使用して SAS データセットのオブザベーションを印刷します。

構文

PROC PRINT <*option(s)*>;

オプション引数の要約

CONTENTS=*link-text*
: HTML コンテンツファイルのリンク用テキストを指定します。

DATA=*SAS-data-set*
: 印刷する SAS データセットを指定します。

Control column format

GRANDTOTAL_LABEL='*ラベル*'
: 総計行にラベルを表示します。

HEADING=*direction*
: 列ヘッダーの方向を制御します。

LABEL
: 変数のラベルを列ヘッダーとして使用するように指定します。

SPLIT='*split-character*'
: 列ヘッダーの改行を制御する区切り文字を指定します。

STYLE <(*locations(s)*)>=<*style-override(s)*>
: レポートの特定の領域のデフォルトのスタイル要素および属性を変更するために、ODS スタイルのオーバーライドを 1 つ以上指定します。

SUMLABEL

NOSUMLABEL

SUMLABEL='*ラベル*'
: BY グループの要約行へのラベル表示の有無を指定します。

Control general format

BLANKLINE=*n*

BLANKLINE=(COUNT=*n* <STYLE=[*style-attibute-specification(s)*]>)
: *n* オブザベーションの後にブランクを書き込みます。

DOUBLE
: オブザベーション間にブランク行を書き込みます。

N<="*string-1*" <"*string-2*">>
: データセット、BY グループのいずれか、またはその両方のオブザベーション数を印刷し、その数と印刷する説明テキストを指定します。

NOOBS
: 各オブザベーションを番号で識別する出力の列を非表示にします。

OBS="*column-header*"
: 各オブザベーションを番号で識別する列に対して列ヘッダーを指定します。

ROUND
: フォーマットされていない数値を小数点以下 2 桁に丸めます。

Control page format

ROWS=*page-format*
1 ページの行をフォーマットします。

UNIFORM
各変数のフォーマットされた幅をすべてのページで列幅として使用するように指定します。

WIDTH=*column-width*
変数ごとの列幅を決定します。

オプション引数

BLANKLINE=*n*
BLANKLINE=(COUNT=*n* <STYLE=[*style-attibute-specification(s)*]>)
各 *n* オブザベーションの後にブランク行を挿入するように指定します。オブザベーション数は、すべての ODS 出力先に対する各 BY グループの開始時に 0 にリセットされます。

n* | COUNT = *n
後ろにブランク行が挿入されるオブザベーション番号を指定します。

STYLE=[*style-attribute-specification(s)*]
ブランク行に使用するスタイル属性を指定します。

デフォルト	DATA
ヒント	BACKGROUNDCOLOR スタイル属性を使用して、オブザベーションを色によって視覚的に区別できます。
参照項目	STYLE=オプション (1351 ページ)(有効なスタイル属性)
例	“例 1: 印刷する変数の選択” (1376 ページ)

CONTENTS=*link-text*
PROC PRINT ステートメントによって生成される出力への HTML コンテンツファイルのリンク用テキストを指定します。

制限事項	CONTENTS=は、HTML Body ファイルに影響しません。HTML コンテンツファイルにのみ影響します。
	CONTENTS=は、ODS LISTING 出力先には無効です。
参照項目	HTML 出力の詳細については、HTML 出力先によって生成されるファイル と “ODS HTML Statement ” (*SAS Output Delivery System: User's Guide*)を参照してください。

DATA=*SAS-data-set*
印刷する SAS データセットを指定します。

参照項目	“入力データセット” (25 ページ)

DOUBLE
オブザベーション間にブランク行を書き込みます。

別名	D
制限事項	DOUBLE は、ODS LISTING 出力先にのみ有効です。
例	“例 1: 印刷する変数の選択” (1376 ページ)

GRANDTOTAL_LABEL='*ラベル*'
総計行にラベルを表示します。#BYVAR 変数および#BYVAL 変数を'*label*'に含めることができます。

別名 GRAND_LABEL
GRANDTOT_LABEL
GTOT_LABEL
GTOTAL_LABEL

制限事項 #BYVAR 変数と#BYVAL 変数は LISTING 出力先には非対応です。

例 “例 5: 複数の BY 変数を使用し、数値変数を合計する” (1399 ページ)

HEADING=*direction*
列ヘッダーの方向を制御します。*direction* は、次のうちいずれかになります。

HORIZONTAL
すべての列ヘッダーを横方向に印刷します。

別名 H

VERTICAL
すべての列ヘッダーを縦方向に印刷します。

別名 V

制限事項 LISTING 出力については、ページに対して列ヘッダーが長すぎる場合には、ラベルのかわりに変数名を使用します。

デフォルト ヘッダーはすべて横方向、すべて縦方向のいずれかになります。HEADING=を省略すると、PROC PRINT は列ヘッダーの方向を次のように決定します。

LABEL を使用しない場合、スペースによって列ヘッダーが縦方向または横方向であるかが決定されます。
LABEL を使用し、少なくとも 1 つの変数にラベルがある場合、すべてのヘッダーは横方向になります。

LABEL
変数のラベルを列ヘッダーとして使用するように指定します。

別名 L

デフォルト PROC PRINT は、次の 2 つの状況では変数名を列ヘッダーとして使用します。

1. PROC PRINT ステップに LABEL ステートメントが含まれている場合でも PROC PRINT ステートメントで LABEL オプションを省略する場合
2. 変数にラベルが含まれていない場合

操作 デフォルトでは、LABEL を指定し、少なくとも 1 つの変数にラベルが含まれている場合は PROC PRINT はすべての列ヘッダーを横方向に印刷します。そのため、LABEL を使用すると出力のページ数が増えることがあります。(PROC PRINT ステートメントで HEADING=VERTICAL を使用して、列ヘッダーを縦方向に印刷します。

PROC PRINT は、ラベルを複数行にわたって分割し、スペースを節約する場合があります。PROC PRINT ステートメントで SPLIT=を使用して、これらの分割が発生する箇所を制御します。SPLIT=を使用する場合、LABEL を使用する必要はありません。

注 SAS システムオプション LABEL は、プロシジャがラベルを使用するために有効である必要があります。詳細については、“LABEL System Option” (*SAS System Options: Reference*)を参照してください。

ヒント 変数に対しブランクの列ヘッダーを作成するには、PROC PRINT ステップでこの LABEL ステートメントを使用します。

```
label variable-name='00'x;
```

参照項目 LABEL ステートメントを使用してプロシジャで一時ラベルを作成する詳細については、3 章, “複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

LABEL ステートメントを DATA ステップで使用して永久ラベルを作成する詳細については、“LABEL Statement” (*SAS Statements: Reference*)を参照してください。

例 “例 3: オブザベーショングループごとのレポートの別のセクションの作成” (1386 ページ)

N<=“*string-1*” <“*string-2*”>>

データセット、BY グループのいずれか、またはその両方のオブザベーション数を印刷し、その数と印刷する説明テキストを指定します。

N オプション使用	PROC PRINT アクション
BY ステートメントおよび SUM ステートメントと使用しない	レポートの最後にデータセットのオブザベーション数を印刷し、その数を *string-1* とラベル付けします。
BY ステートメントと使用	各 BY グループの最後に BY グループのオブザベーション数を印刷し、その数を *string-1* の値とラベル付けします。
BY ステートメントおよび SUM ステートメントと使用	各 BY グループの最後に BY グループのオブザベーション数を印刷し、レポートの最後にデータセットのオブザベーション数を印刷します。BY グループのオブザベーション数は *string-1* とラベル付けされます。データセット全体のオブザベーション数は *string-2* とラベル付けされます。

例 “例 2: 列ヘッダーテキストのカスタマイズ” (1380 ページ)

“例 3: オブザベーショングループごとのレポートの別のセクションの作成” (1386 ページ)

“例 4: 1 つの BY グループを使用し、数値変数を合計する” (1394 ページ)

NOOBS
各オブザベーションを番号で識別する出力の列を非表示にします。

例 “例 3: オブザベーショングループごとのレポートの別のセクションの作成” (1386 ページ)

OBS=“*column-header*”
各オブザベーションを番号で識別する列に対して列ヘッダーを指定します。

ヒント OBS=は、区切り文字を有効化します(SPLIT=オプション (1350 ページ)の説明を参照)。

例 “例 2: 列ヘッダーテキストのカスタマイズ” (1380 ページ)

ROUND
フォーマットされていない数値を小数点以下 2 桁に丸めます。(フォーマットされている値は、出力形式により指定小数点以下桁数にすでに丸められています。) フォーマットされている変数とフォーマットされていない変数について、PROC PRINT はこれらの丸められた値を使用してレポートの合計を計算します。

ROUND を省略すると、*PROC PRINT はフォーマットされた(丸められた)値を表示しますが*、行の実際の値を追加して合計を取得します。合計も出力形式によって丸められますが、この合計には丸めエラーが 1 つだけ含まれます。これは、実際の値の合計の丸めエラーです。ROUND オプションは値を合計する前に丸めるため、複数の丸めエラーがある場合があります。ROUND を使用しない方が結果はより正確なものになりますが、ROUND は合計が印刷された(丸められた)値の合計であることが重要なパブリッシュされたレポートに有用です。

ROUND オプションを使用した PROC PRINT の結果が、PROC MEANS、DATA ステップなどのその他の方法を使用して同じデータを合計した結果と異なる場合があることに注意してください。次が True である単純な場合について考えます。

- データセットには X に対する 3 つの値、.003、.004 および.009 が含まれている。
- X に出力形式 5.2 がある。

合計の計算方法によって、3 つの異なる答え、0.02、0.01、0.016 を得ます。次の図に、PROC PRINT (ROUND オプションを使用しない場合と使用した場合)と PROC MEANS を使用した合計の計算結果を示します。

図 45.6 変数を合計する 3 つの方法

Actual Values	PROC PRINT without the ROUND option		PROC PRINT with the ROUND option		PROC MEANS
	OBS	X	OBS	X	Analysis Variable : X
.003	1	0.00	1	0.00	Sum
.004	2	0.00	2	0.00	0.0160000
.009	3	0.01	3	0.01	
=====		=====		=====	
.016		0.02		0.01	

ROUND オプションを使用せずに生成される合計(.02)が、ROUND を使用して生成される合計(0.01)よりも実際の結果(0.16)に近いことに注意してください。ただし、ROUND を使用して生成される合計は、レポートに表示される数を反映しています。

別名 R

注意 **ROUND を PICTURE 出力形式と使用しないでください。** ROUND は、数値と使用します。SAS プロシジャは、ピクチャ出力形式の変数を文字変数として扱います。ROUND をそのような変数と使用すると、予期せぬ結果が発生することがあります。

ROWS=*page-format*

1 ページの行をフォーマットします。現在、PAGE は *page-format* に使用できる唯一の値です。

PAGE

ページごとの各オブザベーションに対し 1 行の変数のみ印刷します。ROWS=PAGE を使用すると、PROC PRINT はページをセクションに分割せず、各ページにできるだけ多くのオブザベーションを印刷します。出力の最後のページを埋めるのに十分なオブザベーションがない場合、PROC PRINT は最後のページをセクションに分割し、最後の少数のオブザベーションに対しすべての変数を印刷します。

制限事項 ROWS=は、ODS LISTING 出力先にのみ有効です。そのため、ROWS=を使用する場合、PROC PRINT からの HTML 出力は同じです。

ヒント データセットに多くの変数とオブザベーションが含まれている場合、PAGE 値により出力のページ数を減らすことができます。ただし、データセットに含まれる変数の数が多く、オブザベーションの数が少ない場合、PAGE 値により出力のページ数を増やすことができます。

参照項目 “ページのサイズが制限されている場合のページレイアウト” (1341 ページ) (デフォルトレイアウトの説明)

例 “例 7: 多数の変数を使用し、レポートのレイアウトを制御する” (1411 ページ)

SPLIT='*split-character*'

列ヘッダーの改行を制御する区切り文字を指定します。また、ラベルを列ヘッダーとして使用します。PROC PRINT は区切り文字に達すると列ヘッダーを改行し、次の行ヘッダーを続けます。区切り文字は発生するたびラベルの最大 256 文字に考慮されますが、列ヘッダーの一部ではありません。

別名 S=

操作 SPLIT=はラベルの使用を意味するため、LABEL と SPLIT=の両方を使用する必要はありません。

OBS=オプションは区切り文字を有効化します。(“OBS=“*column-header*”” (1349 ページ)の説明を参照)。

注 PROC PRINT は、SPLIT=を指定しても各 BY グループまたは要約ラベルまたは総計レベルの前におかれるヘッダーの BY 変数のラベルを分割しません。代わりに、PROC PRINT は区切り文字をブランクと置き換えます。

例 “例 2: 列ヘッダーテキストのカスタマイズ” (1380 ページ)

STYLE <(*locations(s)*)>=<*style-override(s)*>
レポートの特定の領域のデフォルトのスタイル要素および属性を変更するために、ODS スタイルのオーバーライドを 1 つ以上指定します。

スタイルの無効化は次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2 …*>]

location
STYLE オプションが影響するレポートの部分を識別します。*location(s)*が指定されていない場合、スタイルのオーバーライドが適用される位置は、PROC PRINT によってステートメント、指定されたスタイル要素、およびスタイル属性に基づき決定されます。

次の図に、利用可能な場所と、それらを指定できるその他のステートメントを示します。

表 45.1 *STYLE オプションでの場所の指定*

場所	場所の別名	影響を受けるレポート部分	次のステートメントでも使用可能
BYLABEL	BYSUMLABEL BYLBL BYSUMLBL	SUM 合計を含む行の BY 変数のラベル	なし

場所	場所の別名	影響を受けるレポート部分	次のステートメントでも使用可能
DATA	COLUMN COL	OBS 列または ID 列のデータを除くすべてのデータ または ID ステートメントの STYLE=オプションで DATA 位置が指定されているときの ID 列のデータ	VAR ID SUM
GRANDTOTAL	GRANDTOT GRAND GTOTAL GTOT	レポート全体の総計を含む SUM 行	SUM
HEADER	HEAD HDR	OBS 列または ID 列を除くすべてのヘッダー * または ステートメントの STYLE=オプションで HEADER 位置が指定されているときの ID 列の列ヘッダーすべて	VAR ID SUM
N	なし	N=テーブルおよびコンテンツ	なし
OBS	OBSDATA OBSCOLUMN OBSCOL	ID ステートメントの STYLE=オプションで DATA 位置が指定されていないときの OBS 列または ID 列のデータ	なし
OBSHEADER	OBSHEAD OBSHDR	ID ステートメントの STYLE=オプションで DATA 位置が指定されていないときの OBS 列または ID 列のヘッダー *	なし
TABLE	REPORT	レポートの構造部分(境界の幅、セル間のスペースなどの設定に使用される基本テーブル)	なし

場所	場所の別名	影響を受けるレポート部分	次のステートメントでも使用可能
TOTAL	TOT BYSUMLINE BYLINE BYSUM	BY グループごとの合計を含む SUM 行	SUM

* SAS 9.4 より前のバージョンでは、PROC PRINT ステートメントの STYLE=オプションの HEADER 位置を指定すると、HEADER スタイル属性を用いてすべての列ヘッダーがレンダリングされていました。SAS 9.4 では、PROC PRINT ステートメントの STYLE=オプションの OBSHEADER 位置を使用して、OBS 列と ID 列のフォーマットします。既存プログラムにおける PROC PRINT ステートメントの STYLE=オプションには、HEADER 位置のみならず OBSHEADER 位置も含まれている必要があります。

図 45.7 *PROC PRINT 領域と対応するステートメント*



PROC PRINT ステートメント以外のステートメントでのスタイル指定は、PROC PRINT ステートメントの同じスタイル指定に優先します。ただし、別のステートメントでのスタイル指定によるスタイルを優先しない場合、PROC PRINT ステートメントで指定するスタイル属性が継承されます。たとえば、PROC PRINT ステートメントですべての列ヘッダーに青い背景と白い前景を指定し、ID 列ヘッダーに薄い灰色の背景を指定すると、ID 列ヘッダーの背景が薄い灰色になり、前景は白になります(PROC PRINT ステートメントで指定されているとおり)。この PRINT プロシジャは、ID 列ヘッダーにおける白色のインヘリタンスを示します。

```
proc print data=exprev style(header)={backgroundcolor=blue color=white};
   id country / style(obsheader)=[backgroundcolor=light gray];
run;
```

The SAS System

Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
Antarctica	99999999	1/1/12	1/7/12	Internet	2	92.6	20.70

PROC PRINT ステートメントの OBSHEADER 位置と ID ステートメントの HEADER 位置に対して同一のスタイル属性が表示されていれば、OBSHEADER 属性よりも HEADER 位置属性が優先されます。PROC PRINT ステートメントと ID ステートメントの両方の ID 列に対する他のすべてのスタイル属性は、統合されてその ID 列のスタイルを作成します。たとえば PROC PRINT ステートメントでは、OBSHEADER の位置属性は**{fontsize=5 fontweight=bold}**になります。ID ステートメントでは、HEADER の位置属性は**[fontsize=6 fontstyle=italic]**になります。その結果、ID 列のスタイルは**[fontsize=6 fontweight=bold fontstyle=italic]**となります。

```
proc print data=exprev style(obsheader)={fontsize=5 fontweight=bold};
   id country / style(header)=[fontsize=6 fontstyle=italic];
run;
```

The SAS System

Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
Antarctica	99999999	1/1/12	1/7/12	Internet	2	92.6	20.70

PROC PRINT ステートメントの OBS 位置と ID ステートメントの DATA 位置に対して同一のスタイル属性が表示されていれば、OBS 属性よりも DATA 位置属性が優先されます。PROC PRINT ステートメントと ID ステートメントの両方の ID 列に対する他のすべてのスタイル属性は、統合されてその ID 列のスタイルを作成します。たとえば PROC PRINT ステートメントでは、OBS の位置属性は**{backgroundcolor=light gray color=blue}**になります。ID ステートメントでは、DATA の位置属性は**[color=white fontstyle=italic]**になります。その結果、ID 列のスタイルは**[backgroundcolor=light gray color=white fontstyle=italic]**となります。

```
proc print data=exprev style(obs)={backgroundcolor=light gray color=blue};
   id country / style(data)=[color=white fontstyle=italic];
run;
```

The SAS System

Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
Antarctica	99999999	1/1/12	1/7/12	Internet	2	92.6	20.70
Puerto Rico	99999999	1/1/12	1/5/12	Catalog	14	51.2	12.10
Virgin Islands (U.S.)	99999999	1/1/12	1/4/12	In Store	25	31.1	15.65

style-element-name

Output Delivery System で登録されるスタイルテンプレートにおけるスタイル要素の名前です。SAS では、一部の スタイルテンプレートを提供しています。ユーザーは TEMPLATE プロシジャを使用して、独自のスタイルテンプレートを作成できます。*SAS 9.4 Output Delivery System: Procedures Guide* を参照してください。

スタイル要素の処理時は、より詳細な特定のスタイル要素がより詳細でないものに優先します。各 OPROC PRINT 位置のデフォルトのスタイル要素およびスタイル属性の表は、表 45.3 (1374 ページ)を参照してください。

ヒント スタイル要素名には、複合名やフォーマットを使用できます。複合スタイル要素名の使用例として、`style(obsheader)=data.italic.red;` があります。フォーマット要素名の使用例として、`style=$cities` があります。フォーマットの使用の詳細については、*SAS 9.4 Output Delivery System: Procedures Guide* を参照してください。

style-attribute-specification
変更するスタイル属性を記述します。*style-attribute-specification* にはそれぞれ、次の一般的な形式があります。

style-attribute-name=style-attribute-value

次のスタイル属性を TABLE 場所で設定できます。

BACKGROUNDCOLOR=	FONTWIDTH= *
BACKGROUNDIMAGE=	COLOR= *
BORDERCOLOR=	FRAME=
BORDERCOLORDARK=	HTMLCLASS=
BORDERCOLORLIGHT=	TEXTALIGN=
BORDERWIDTH=	OUTPUTWIDTH=
CELLPADDING=	POSTHTML=
CELLSPACING=	POSTIMAGE=
FONT= *	POSTTEXT=
FONTFAMILY= *	PREHTML=
FONTSIZE= *	PREIMAGE=
FONTSTYLE= *	PRETEXT=
FONTWEIGHT= *	RULES=

* これらの属性は使用時に属性 PRETEXT=、POSTTEXT=、PREHTML=、POSTHTML=で指定されるテキストにのみ影響します。表に表示されるテキストの前景色またはフォントを変更するには、表ではなくセルに影響する場所で対応する属性を設定する必要があります。

次のスタイル属性を TABLE 以外のすべての場所で設定できます。

ASIS=	FONTWIDTH=
BACKGROUNDCOLOR=	HREFTARGET=
BACKGROUNDIMAGE=	CLASS=
BORDERCOLOR=	TEXTALIGN=

BORDERCOLORDARK=	NOBREAKSPACE=
BORDERCOLORLIGHT=	POSTHTML=
BORDERWIDTH=	POSTIMAGE=
HEIGHT=	POSTTEXT=
CELLWIDTH=	PREHTML=
FLYOVER=	PREIMAGE=
FONT=	PRETEXT=
FONTFAMILY=	PROTECTSPECIALCHARACTERS=
FONTSIZE=	TAGATTR=
FONTSTYLE=	URL=
FONTWEIGHT=	VERTICALALIGN=

制限事項 STYLE=は、ODS LISTING または ODS OUTPUT 出力先には無効です。

参照項目 PROC TABULATE、PROC REPORT および PROC PRINT とともに使用できるスタイル属性の表は、表 45.2 (1371 ページ)を参照してください。

各 PROC PRINT 位置のデフォルトのスタイル要素およびスタイル属性の表は、表 45.3 (1374 ページ)を参照してください。

PROC PRINT でのスタイルの使用の詳細については、“BASE SAS レポート作成プロシジャでの ODS スタイルの使用” (1364 ページ)を参照してください。

スタイル属性および PROC TEMPLATE の詳細については、*SAS 9.4 Output Delivery System: Procedures Guide* の DEFINE スタイルステートメントを参照してください。

SUMLABEL
NOSUMLABEL
SUMLABEL='*ラベル*'
BY グループの要約行へのラベル表示の有無を指定します。

SUMLABEL
変数ラベルがあれば、変数名のかわりに要約ラインのラベルとして使用するよう指定します。

NOSUMLABEL
要約行ブランクにそのラベルを残すよう指定します。または、SUMLABEL="" (間にスペースがない一重または二重引用符)を使用して要約行のブランクを表示することもできます。

SUMLABEL='*label*'
BY グループの要約行のラベルとしてテキストを使用するよう指定します。#BYVAR 変数および#BYVAL 変数を'*label*'に含めることができます。

制限事項 #BYVAR 変数と#BYVAL 変数は LISTING 出力先には非対応です。

デフォルト SUMLABEL を省略すると、PROC PRINT は要約行の BY 変数名を使用します。

例 “例 4: 1 つの BY グループを使用し、数値変数を合計する” (1394 ページ)

“例 5: 複数の BY 変数を使用し、数値変数を合計する” (1399 ページ)

UNIFORM
WIDTH=UNIFORM (1357 ページ)を参照してください。

WIDTH=*column-width*
変数ごとの列幅を決定します。*column-width* の値は、次のうちいずれかである必要があります。

FULL
変数のフォーマットされた幅を列幅として使用します。変数にフィールド幅を明示的に指定する出力形式がない場合、PROC PRINT はデフォルトの幅を使用します。文字変数の場合、デフォルトの幅は、変数の長さになります。数値変数の場合、デフォルトの幅は 12 となります。WIDTH=FULL を使用すると、列幅がページによって異なることはありません。

ヒント WIDTH=FULL を使用すると、実行時間を削減できます。

MINIMUM
変数のすべての値に対応する最小列幅を各変数に使用します。

別名 MIN

UNIFORM
各変数のフォーマットされた幅をすべてのページで列幅として使用します。変数にフィールド幅を明示的に指定する出力形式が含まれていない場合、PROC PRINT は最大幅データ値を列幅として使用します。WIDTH=UNIFORM を指定すると、PROC PRINT は通常データセットを二度読み込む必要があります。ただし、データセットのすべての変数にフィールド幅を明示的に指定する出力形式(たとえば BEST.ではなく BEST12.)が含まれている場合、PROC PRINT はデータセットを一度だけ読み込みます。

別名 U

制限事項 一部の変数に幅を明示的に指定する出力形式が含まれていない場合、別のユーザーがデータセットを同時に更新している場合は WIDTH=UNIFORM を同時アクセスをサポートするエンジンと使用できません。

ヒント データセットが大きく、一定のレポートが必要な場合、PROC PRINT がデータを一度だけ読み込むように、フィールド幅を明示的に指定する出力形式を使用してコンピュータリソースを節約できます。

WIDTH=UNIFORM は、UNIFORM と同じです。

UNIFORMBY
変数のフォーマットされた幅を列幅として使用して、BY グループ内のすべての列に出力形式を一様に適用します。変数にフィールド幅を明示的に指定する出力形式が含まれていない場合、PROC PRINT は最大幅データ値を列幅として使用します。

別名 UBY

制限事項 UNIFORMBY を順次データセットと使用できません。

デフォルト WIDTH=を省略し、UNIFORM オプションを指定しない場合、PROC PRINT は個別に出力の各ページを作成します。プロシジャはページごとにデータを分析し、最適な表示方法を決定します。そのため、列幅がページごとに異なる場合があります。

制限事項 WIDTH=は LISTING 出力先に対してのみ有効です

ヒント 列幅は、可変幅だけでなく、列ヘッダーの長さの影響も受けます。列ヘッダーが長いと、WIDTH=の有用性が低下することがあります。

参照項目 デフォルトの列幅の説明については、“列幅” (1343 ページ)を参照してください。

BY ステートメント

BY グループごとにレポートの別のセクションを生成します。

参照項目: 3 章, “複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)

例: “例 3: オブザベーショングループごとのレポートの別のセクションの作成” (1386 ページ)
“例 4: 1 つの BY グループを使用し、数値変数を合計する” (1394 ページ)
“例 5: 複数の BY 変数を使用し、数値変数を合計する” (1399 ページ)
“例 6: レポート内の合計数を制限する” (1406 ページ)
“例 8: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1415 ページ)

構文

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …> <NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは指定するすべての変数別に並べ替えるか、適切にインデックス化する必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
データセットが BY ステートメントで単語 DESCENDING の直後に続く変数を基準にして降順で並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。データは、時系列などの別の方法でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定した場合は、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ別の BY グループとして扱います。

詳細

BY ステートメントと ID ステートメントとの使用

すべての BY 変数が ID ステートメントの開始時に同じ順序で表示される場合、PROC PRINT は特別なレイアウトを使用します(“例 8: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1415 ページ)を参照)。

BY ステートメントと NOBYLINE オプションとの使用

BY ステートメントを、BY グループ処理によって生成される出力に通常表示される BY 行を非表示にする SAS システムオプション NOBYLINE と使用すると、PROC PRINT は常に BY グループごとに新しいページを開始します。この動作により、BY グループ情報をタイトルに挿入し、デフォルトの BY 行を NOBYLINE で非表示にしてカスタマイズした BY 行を作成すると、タイトルの情報がページのレポートと一致するようになります。

並べ替えられていないデータの印刷時の BY 変数の使用

値が並べ替えられていない BY 変数を指定すると、最初の並べ替えられていないグループの処理時にデータセットの印刷が停止します。メッセージが SAS ログに書き込まれます。

ID ステートメント

オブザベーション番号ではなく、リストする変数のフォーマットされた値を使用して、オブザベーションを識別します。

例: “例 7: 多数の変数を使用し、レポートのレイアウトを制御する” (1411 ページ)
“例 8: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1415 ページ)

構文

ID *variable(s)*
</ STYLE <(*location(s)*)>=<*style-override(s)*> >;

必須引数

variable(s)
レポートの各行の開始時にオブザベーション番号の代わりに印刷する 1 つ以上の変数を指定します。

制限事項 ID 変数が非常に多くのスペースを占めているためにその行に少なくとも 1 つのその他の変数のためのスペースがない場合、PROC PRINT は警告を SAS ログに書き込み、すべての ID 変数を ID 変数として扱いません。

操作 ID ステートメントの変数が VAR ステートメントにも表示される場合、出力にはその変数に対する 2 つの列が含まれます。

オプション引数

STYLE <(*locations(s)*)>=<*style-override(s)*>
ID ステートメントで作成される ID 列に使用する 1 つ以上のスタイルのオーバーライドを指定します。

スタイルの無効化は次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2 …*>]

制限事項 OBSHEADER 位置のスタイル指定は ID ステートメントには無効です。

操作 ID ステートメントで STYLE(HEADER)=オプションを、PROC PRINT ステートメントで STYLE(OBSHEADER)=を指定すると、ID ステートメントのために指定されているスタイル属性が PROC PRINT ステートメントで指定したスタイル要素に優先します。その後、PROC PRINT ステートメントの STYLE(OBSHEADER)=オプションのスタイル属性は ID ステートメントの STYLE(HEADER)=オプションと結合し、ID 列ヘッダーの出力がレンダリングされます。

ヒント 異なる ID 列に対して異なるスタイルのオーバーライドを指定するには、各変数に別の ID ステートメントを使用して、異なる STYLE オプションを各 ID ステートメントに追加します。

参照項目 このオプションの引数およびその使用方法の詳細については、PROC PRINT ステートメントの STYLE= (1351 ページ)オプションを参照してください。

詳細

BY ステートメントと ID ステートメントとの使用

すべての BY 変数が ID ステートメントの開始時に同じ順序で表示される場合、PROC PRINT は特別なレイアウトを使用します (“例 8: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1415 ページ)を参照)。

PAGEBY ステートメント

ページが埋まる前に発生するページ排出を制御します。

要件 BY ステートメント

例: “例 3: オブザベーショングループごとのレポートの別のセクションの作成” (1386 ページ)

構文

PAGEBY *BY-variable*;

必須引数

BY-variable
PROC PRINT ステップの BY ステートメントに表示される変数を識別します。BY 変数の値が変わる、または BY ステートメントで前にくる BY 変数の値が変わると、PROC PRINT は新しいページの印刷を開始します。

操作 BY ステートメントを、BY グループ処理によって生成される出力に通常表示される BY 行を非表示にする SAS システムオプション NOBYLINE と使用すると、PROC PRINT は常に BY グループごとに新しいページを開始します。この動作により、BY グループ情報をタイトルに挿入し、デフォルトの BY 行を NOBYLINE で非表示にしてカスタマイズした BY 行を作成すると、タイトルの情報がページのレポートと一致するようになります(“BY グループの情報を含むタイトルの作成 ” (53 ページ)を参照)。

SUM ステートメント

数値変数の値を総計します。

例: “例 4: 1 つの BY グループを使用し、数値変数を合計する” (1394 ページ)
“例 5: 複数の BY 変数を使用し、数値変数を合計する” (1399 ページ)
“例 6: レポート内の合計数を制限する” (1406 ページ)
“例 8: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1415 ページ)

構文

SUM *variable(s)*
</ STYLE <(*location(s)*)>=<*style-override(s)*> >;

必須引数

variable(s)
レポートで総計する数値変数を識別します。

オプション引数

STYLE <(*locations(s)*)>=<*style-override(s)*>
SUM ステートメントで作成される合計を含むセルに使用する 1 つ以上のスタイルのオーバーライドを指定します。

スタイルの無効化は次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2 …*>]

ヒント 合計をレポートする異なるセルに異なるスタイルのオーバーライドを指定するには、変数ごとに別の SUM ステートメントを使用して、異なる STYLE オプションを各 SUM ステートメントに追加します。

STYLE オプションが同じ場所に影響する複数の SUM ステートメントで使用される場合、最後の SUM ステートメントの STYLE オプションが使用されます。

参照項目 このオプションの引数およびその使用方法の詳細については、PROC PRINT ステートメントのオプション STYLE= (1346 ページ) を参照してください。

詳細

SUM と BY ステートメントとの使用

SUM ステートメントと、1 つの BY 変数を含む BY ステートメントを使用すると、PROC PRINT は複数のオブザベーションを含む BY グループごとに SUM 変数を合計し、すべての BY グループについて総計します(“例 4: 1 つの BY グループを使用し、数値変数を合計する” (1394 ページ)を参照)。

SUM ステートメントと、複数の BY 変数を含む BY ステートメントを使用すると、PROC PRINT は BY 変数を 1 つだけ使用する場合と同じように複数のオブザベーションを含む BY グループごとに SUM 変数を合計します。ただし、BY グループが変わるときに値が変わるこれらの BY 変数に対する合計のみ提供されます(“例 5: 複数の BY 変数を使用し、数値変数を合計する” (1399 ページ)を参照)。

注: BY 変数の値が変わると、SAS システムは BY ステートメントでその後にリストされるすべての変数の値も変わるとみなします。

SUMBY ステートメント

レポートに表示される合計数を制限します。

要件 BY ステートメント

例: “例 6: レポート内の合計数を制限する” (1406 ページ)

構文

SUMBY *BY-variable*;

必須引数

BY-variable
PROC PRINT ステップの BY ステートメントに表示される変数を識別します。BY 変数の値が変わる、または BY ステートメントでその前にくる BY 変数の値が変わると、PROC PRINT は SUM ステートメントにリストされるすべての変数の合計を印刷します。

詳細

要約される変数

SUM ステートメントを使用すると、PROC PRINT は SUM 変数のみ小計します。その他の場合は、PROC PRINT は ID ステートメントと BY ステートメントでリストされる変数を除く、データセットのすべての数値変数を小計します。

VAR ステートメント

レポートに表示される変数を選択し、その順序を決定します。

ヒント: VAR ステートメントを省略すると、PROC PRINT はデータセットのすべての変数を印刷します。

例: “例 1: 印刷する変数の選択” (1376 ページ)
“例 8: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1415 ページ)

構文

VAR *variable(s)*
</ STYLE <(*location(s)*)>=<*style-override(s)*> >;

必須引数

variable(s)
印刷する変数を識別します。PROC PRINT は、変数をリストする順序で印刷します。

操作 PROC PRINT 出力では、ID ステートメントにリストされる変数が VAR ステートメントにリストされる変数よりも前に置かれます。ID ステートメントの変数が VAR ステートメントにも表示される場合、出力にはその変数に対する 2 つの列が含まれます。

オプション引数

STYLE <(*locations(s)*)>=<*style-override(s)*>
VAR ステートメントによって作成されるすべての列に使用する 1 つ以上のスタイルのオーバーライドを指定します。

スタイルの無効化は次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2 …*>]

ヒント 異なる列に異なるスタイルのオーバーライドを指定するには、別の VAR ステートメントを使用して変数ごとに列を作成し、異なる STYLE オプションを各 VAR ステートメントに追加します。

参照項目 このオプションの引数およびその使用方法の詳細については、PROC PRINT ステートメントのオプション STYLE= (1351 ページ)を参照してください。

BASE SAS レポート作成プロシジャでの ODS スタイルの使用

概要

ODS をサポートする Base SAS プロシジャの多くは、1 つ以上のテーブルテンプレートを使用して出力オブジェクトを生成します。これらのテーブルテンプレートには、テーブル要素(列、ヘッダー、フッター)に対するテンプレートが含まれています。各テーブル要素で、出力のさまざまな部分に 1 つ以上のスタイル要素を使用することを指定できます。これらのスタイル要素はプロシジャの構文内で指定することはできませんが、使用する ODS 出力先に合わせてカスタマイズされたスタイルを使用できます。テーブルとスタイルのカスタマイズの詳細については、“TEMPLATE Procedure: Creating a Style Template” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。

Base SAS レポートプロシジャの PROC PRINT、PROC REPORT および PROC TABULATE を使用してデータをすばやく分析し、読みやすいテーブルに整理することができます。これらのプロシジャステートメントで STYLE=オプションを使用してレポートの表示を変更できます。STYLE=オプションを使用すると、すべての出力のデフォルトスタイルを変更せずに、出力の各セションに変更を加えることができます。プロシジャ内の特定のステートメントで STYLE=オプションを指定して、プロシジャ出力の特定のセクションをカスタマイズすることができます。

次のプログラムでは、STYLE=オプションを使用して以下の PROC REPORT 出力の背景色を作成します。

```
title "Height and Weight by Gender and Age";
proc report nowd data=sashelp.class
   style(header)=[background=white];
   col age (('Gender' sex),(weight height));
   define age / style(header)=[background=lightgreen];
   define sex / across style(header)=[background=yellow] ' ';
   define weight / style(header)=[background=orange];
```

```
   define height / style(header)=[background=tan];
run;
```

図 45.8 PROC REPORT 出力の拡張



Height and Weight by Gender and Age

	Gender			
	F		M	
Age	Weight	Height	Weight	Height
253	811	545.3	1089.5	639.1

次のプログラムでは、STYLE=オプションを使用して以下の PROC TABULATE 出力の色を作成します。

```
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

proc tabulate data=prdsale;
   class region division prodtype / style=[background=lightgreen];
   classlev region division prodtype  / style=[background=yellow];
   var actual  / style=[background=tan];
   keyword all sum / style=[background=linen color=blue];
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         box=[label='Region by Division and Type' style=[backgroundcolor=orange]];

title 'Actual Product Sales';
title2 '(millions of dollars)';
run;
```

図 45.9 PROC TABULATE 出力の拡張



Actual Product Sales
(millions of dollars)

Region by Division and Type		Product type		Total
		FURNITURE	OFFICE	
		Actual Sales	Actual Sales	Actual Sales
		Sum	Sum	Sum
Region	Division			
EAST	CONSUMER	$72,570	$108,686	$181,256
	EDUCATION	$73,901	$115,104	$189,005
	Total	$146,471	$223,790	$370,261
WEST	Division			
	CONSUMER	$76,209	$105,020	$181,229
	EDUCATION	$67,945	$110,902	$178,847
	Total	$144,154	$215,922	$360,076
Total	Division			
	CONSUMER	$148,779	$213,706	$362,485
	EDUCATION	$141,846	$226,006	$367,852
	Total	$290,625	$439,712	$730,337

次のプログラムでは、STYLE=オプションを使用して以下の PROC PRINT 出力の色を作成します。

```
proc print data=exprev noobs sumlabel='Total' GRANDTOTAL_LABEL="Grand Total"
      style(table)=[frame=box rules=groups]
      style(bysumline)=[background=red foreground=linen]
      style(grandtotal)=[foreground=green]
      style(header)=[font_style=italic background=orange];
   by sale_type order_date;
   sum price quantity;
   sumby sale_type;
   label sale_type='Sale Type' order_date='Sale Date';
format price dollar10.2 cost dollar10.2;
   var Country / style(data)=[font_face=arial font_weight=bold background=linen];
   var Price / style(data)=[font_style=italic background=yellow];
   var Cost / style(data)=[foreground=hgt. background=lightgreen];
title 'Retail and Quantity Totals for Each Sale Type';
run;
```

完全な入力データセットについては、“EXPREV” (2154 ページ)を参照してください。

図 45.10 PROC PRINT 出力の拡張



Retail and Quantity Totals for Each Sale Type

Sale Type=Catalog Sale Date=1/1/12

Country	Price	Cost	Quantity
Puerto Rico	$51.20	$12.10	14
Aruba	$123.70	$59.00	30
Bahamas	$113.40	$28.45	8
Bermuda	$41.00	$9.25	7

Sale Type=Catalog Sale Date=1/2/12

Country	Price	Cost	Quantity
British Virgin Islands	$40.20	$20.20	11
Canada	$11.80	$5.00	100
Total	$381.30		170

Sale Type=In Store Sale Date=1/1/12

Country	Price	Cost	Quantity
Virgin Islands (U.S.)	$31.10	$15.65	25

Sale Type=In Store Sale Date=1/2/12

Country	Price	Cost	Quantity
Belize	$146.40	$36.70	2
Cayman Islands	$71.00	$32.30	20
Total	$248.50		47

Sale Type=Internet Sale Date=1/1/12

Country	Price	Cost	Quantity
Antarctica	$92.60	$20.70	2
Grand Total	$722.40		219

スタイル、スタイル要素およびスタイル属性

SAS 出力の表示はスタイルテンプレート(スタイル)で制御されます。スタイルは、SAS 出力の視覚側面(色、フォント、罫線、マーカーなど)を定義する ODS テンプレートの一種です。スタイルにより、そのスタイルを使用するドキュメントの全体的な表示が決まります。スタイルテンプレートは、スタイル要素とスタイル属性で構成されています。

- スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性の名前付きコレクションです。ODS 出力の各領域には、その領域に関連付けられているスタイル要素名があります。スタイル要素名で、スタイル属性が適用される場所を指定します。たとえば、スタイル要素に、列ヘッダーの表示またはセル内のデータの表示に関する指示を含めることができます。スタイル要素で、スタイルを使用する出力のデフォルトの色およびフォントも指定することもできます。
- スタイル属性は、色、フォントのプロパティ、罫線の特性などの視覚プロパティで、予約された名前と値を使用して ODS で定義されます。スタイル属性は、スタイルテンプレート内のスタイル要素によりまとめて参照されます。各*スタイル属性*は、表示の 1 つの側面の値を指定します。たとえば、BACKGROUNDCOLOR=属性では、HTML テーブルまたは印刷出力の色付きテーブルの背景色を指定します。FONTSTYLE=属性では、Roman フォントとイタリックフォントのどちらを使用するか指定します。

注: スタイルはデータの表示を制御するので、LISTING、DOCUMENT または OUTPUT の出力先に移動する出力オブジェクトには影響しません。

使用可能なスタイルは、SASHELP.TMPLMST アイテムストアに含まれています。SAS Enterprise Guide では、スタイルシートのリストがスタイルウィザードで表示されます。バッチモードまたは SAS Studio では、次のコードをサブミットして使用可能なスタイルテンプレートのリストを表示できます。

```
proc template;
list styles / store=sashelp.tmplmst;
run;
```

ODS スタイルの表示に関する詳細は、“Viewing ODS Styles Supplied by SAS” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

デフォルトでは、HTML 出力は HTMLBlue スタイルテンプレートを使用します。スタイル、スタイル要素およびスタイル属性を熟知するために、これらの関係を確認します。以下の図は、スタイル、スタイル要素およびスタイル属性の関係を示しています。次の図は、スタイルの構造を示す例です。

図 45.11 HtmlBlue スタイルの図



以下のリストは、上記の図の番号の付いた項目に対応しています。

1 Styles.HtmlBlue は*スタイル*です。スタイルは、SAS 出力の表示側面(色、フォント、フォントのサイズなど)の表示方法を記述します。スタイルにより、そのスタイルを使用する ODS ドキュメントの全体的な表示が決まります。HTML 出力のデフォルトのスタイルは HtmlBlue です。各スタイルはスタイル要素で構成されています。各出力先には、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルが 1 つあります。

- HTML 出力のデフォルトのスタイルは HTMLBlue です。
- PRINTER の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

を使用して新しいスタイルを作成できます。“DEFINE STYLE Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*).新しいスタイルは既存のスタイルとは別に作成することも、既存のスタイルに基づいて作成することもできます。既存のスタイルから新しいスタイルを作成するには、“PARENT= Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*) を使用できます。ODS スタイルに関する詳細は、“Style Templates” (*SAS Output Delivery System: User's Guide*)を参照してください。

2 *スタイル要素*の例として、ヘッダーとフッターがあります。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。たとえば、スタイル要素に、列ヘッダーの表示またはテーブルセル内のデータの表示に関する指示を含めることができます。スタイル要素で、スタイルを使用する出力のデフォルトの色およびフォントも指定することもできます。スタイル要素はスタイル内に存在し、1 つ以上のスタイル属性で構成されています。スタイル要素はユーザーが定義することも、SAS から提供されるものを使用することもできます。ユーザー定義のスタイル要素は “STYLE Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*)で作成できます。

   注: HTML およびマークアップ言語とその継承に使用されるデフォルトのスタイル要素のリストについては、“Style Elements” (*SAS Output Delivery System: User's Guide*)を参照してください。

3 *スタイル属性*の例として、BORDERCOLOR=、BACKGROUNDCOLOR=、COLOR=があります。スタイル属性では、スタイル要素が適用される出力の領域の 1 つの側面の値を指定します。たとえば、COLOR=属性では、フォントの色に **`cx112277`** の値を指定します。SAS で提供されるスタイル属性のリストについては、“Style Attributes” (*SAS Output Delivery System: User's Guide*)を参照してください。

   スタイル属性はスタイル参照を使用して参照できます。スタイル参照の詳細については、“style-reference” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

次の表に、PROC PRINT、PROC TABULATE および PROC REPORT.ステートメントの STYLE=オプションで設定できる一般的に使用されているスタイル属性を示します。これらの属性のうちほとんどは、セル以外のテーブルの各部分(テーブルの罫線、列と行の間の罫線など)に適用されます。すべての属性がすべての出力先で有効であるわけではありません。これらのスタイル属性、有効な値および適用可能な出力先に関する詳細は、“Style Attributes Tables ” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。

表 45.2 PROC REPORT、PROC TABULATE および PROC PRINT のスタイル属性

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
ASIS=	X	X		X		X
BACKGOUNDCOLOR=	X	X	X	X	X	X
BACKGOUNDIMAGE=	X	X	X	X	X	X
BORDERBOTTOMCOLOR=	X	X		X		
BORDERBOTTOMSTYLE=	X	X	X	X		
BORDERBOTTOMWIDTH=	X	X	X	X		
BORDERLEFTCOLOR=	X	X		X		
BORDERLEFTSTYLE=	X	X	X	X		
BORDERLEFTWIDTH=	X	X	X	X		
BORDERCOLOR=	X	X		X	X	X
BORDERCOLORDARK=	X	X	X	X	X	X
BORDERCOLORLIGHT=	X	X	X	X	X	X
BORDERRIGHTCOLOR=	X	X		X		
BORDERRIGHTSTYLE=	X	X	X	X		
BORDERRIGHTWIDTH=	X	X	X	X		
BORDERTOPCOLOR=	X	X		X		
BORDERTOPSTYLE=	X	X	X	X		
BORDERTOPWIDTH=	X	X	X	X		
BORDERWIDTH=	X	X	X	X	X	X

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
CELLPADDING=	X		X		X	
CELLSPACING=	X		X		X	
CELLWIDTH=	X	X	X	X		X
CLASS=	X	X	X	X	X	X
COLOR=	X	X	X			
FLYOVER=	X	X		X		X
FONT=	X	X	X	X	X	X
FONTFAMILY=	X	X	X	X	X	X
FONTSIZE=	X	X	X	X	X	X
FONTSTYLE=	X	X	X	X	X	X
FONTWEIGHT=	X	X	X	X	X	X
FONTWIDTH=	X	X	X	X		X
FRAME=	X		X		X	
HEIGHT=	X	X		X	X	X
HREFTARGET=		X		X		X
HTMLSTYLE=	X	X	X	X	X	
NOBREAKSPACE=	X	X		X		X
OUTPUTWIDTH=	X	X	X	X	X	
POSTHTML=*	X	X	X	X	X	X
POSTIMAGE=	X	X	X	X	X	X
POSTTEXT=*	X	X	X	X	X	X
PREHTML=*	X	X	X	X	X	X
PREIMAGE=	X	X	X	X	X	X
PRETEXT=*	X	X	X	X	X	X

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
PROTECTSPECIALCHARS=		X		X	X	X
RULES=	X		X		X	
TAGATTR=	X	X	X	X	X	X
TEXTALIGN=	X	X	X	X	X	X
URL=		X		X		X
VERTICALALIGN=		X		X		X
WIDTH=	X	X	X	X	X	

* これらの属性をこの場所で使用する場合には、属性 PRETEXT=、POSTTEXT=、PREHTML=、POSTHTML=で指定されるテキストにのみ影響します。表に表示されるテキストの前景色またはフォントを変更するには、表ではなくセルに影響する場所で対応する属性を設定する必要があります。スタイル属性とその値の詳細な説明については、“Style Attributes” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

テーブル部分のデフォルトのスタイル要素およびスタイル属性

次のテーブルには PROC PRINT 出力のさまざまな場所におけるデフォルトのスタイル要素とスタイル属性がリストされます。このテーブルの場所は表 45.1 (1351 ページ)の場所に対応します。テーブルには最も一般的に使用される ODS 出力先のデフォルトである HTML、PDF および RTF のデフォルトがリストされています。それぞれの出力先には、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルテンプレートが含まれます。

- HTML 出力のデフォルトスタイルは HTMLBlue です。
- PRINTER の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

ODS 出力先とそのデフォルトのスタイルに関する詳細なドキュメントは、“Style Templates” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

表 45.3 レポート部分のデフォルトのスタイル要素およびスタイル属性

場所	スタイル要素	HTML スタイル属性	PDF スタイル属性	RTF スタイル属性
BYLABEL	Byline	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BORDERWIDTH = NaN
GRANDTOTAL OBSHEADER HEADER TOTAL	Header	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb
N	Linecontent	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxfafbfe	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT =medium FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = roman COLOR =cx000000
OBS	Rowheader	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb

場所	スタイル要素	HTML スタイル属性	PDF スタイル属性	RTF スタイル属性
DATA	Data	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 8pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000 BORDERWIDTH = NaN COLOR = cx000000	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000
Titles	SystemTitle	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 3 FONTWEIGHT = bold FONTSTYLE = roman BACKGROUNDCOLOR = cxfafbfe	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx000000 BACKGROUNDCOLOR = cxffffff	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 13pt FONTWEIGHT = bold FONTSTYLE = italic COLOR = cx000000
Footnotes	Footers	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx000000 BACKGROUNDCOLOR = cxffffff	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 13pt FONTWEIGHT = bold FONTSTYLE = italic COLOR = cx000000

PRINT プロシジャの出力でのエラー処理

PRINT プロシジャでエラーが発生した場合、または PRINT プロシジャが停止した場合は、エラー発生時までに処理されたオブザベーションに対して出力が作成される可能性があります。SAS はメッセージを SAS ログに書き込み、PRINT プロシジャを終了します。

LISTING 出力について、ページサイズが小さすぎる値に設定されると、データとタイトルまたはフットノートを同じページに印刷できません。この場合、データだけが LISTING 出力先に印刷され、SAS は警告メッセージをログに書き込みます。データとタイトルまたはフットノートを同じページに書き込むには、ページサイズが十分であることを確認してください。

例: PRINT プロシジャ

例 1: 印刷する変数の選択

要素: PROC PRINT ステートメントオプション
- BLANKLINE
- DOUBLE
- STYLE

VAR ステートメント

他の要素: DATA ステップ
FOOTNOTE statement
ODS HTML ステートメント
OPTIONS ステートメント
TITLE statement

データセット: EXPREV

ODS 出力先: HTML, LISTING

詳細

この例では、次のタスクについて説明します。

- レポートに 3 つの変数を選択する
- 変数ラベルを列ヘッダーとして使用する
- LISTING 出力のレポートの行をダブルスペースにする
- デフォルト HTML 出力先と LISTING 出力先に同時にレポートを作成する
- 様式的 HTML レポートを作成します

プログラム:HTML レポートの作成

```
options obs=10;

ods listing;

proc print data=exprev;

   var country price sale_type;
   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

プログラムの説明

HTML は、ウィンドウ環境で SAS が開くときのデフォルトの出力先です。

OBS=システムオプションを設定して、10 のオブザベーションを処理します。

```
options obs=10;
```

LISTING 出力先を開きます。 ウィンドウ環境でのデフォルトでは、HTML 出力先が開いています。HTML と LISTING 出力を同時に作成するために、ODS LISTING ステートメントにより LISTING 出力先が開かれます。

```
ods listing;
```

出力を印刷します VAR ステートメントは、ランク付けする変数を指定します。

```
proc print data=exprev;
   var country price sale_type;
   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

出力:HTML および LISTING

アウトプット 45.1 *変数の選択: デフォルト HTML 出力*

Monthly Price Per Unit and Sale Type for Each Country

Obs	Country	Price	Sale_Type
1	Antarctica	92.6	Internet
2	Puerto Rico	51.2	Catalog
3	Virgin Islands (U.S.)	31.1	In Store
4	Aruba	123.7	Catalog
5	Bahamas	113.4	Catalog
6	Bermuda	41.0	Catalog
7	Belize	146.4	In Store
8	British Virgin Islands	40.2	Catalog
9	Canada	11.8	Catalog
10	Cayman Islands	71.0	In Store

*prices in USD

アウトプット 45.2 *変数の選択:LISTING 出力*

```
                    Monthly Price Per Unit and Sale Type for Each Country 1
Sale_ Obs Country Price Type 1 Antarctica 92.6 Internet 2 Puerto Rico 51.2
Catalog 3 Virgin Islands (U.S.)      31.1 In Store 4 Aruba 123.7 Catalog 5
Bahamas 113.4 Catalog 6 Bermuda 41.0 Catalog 7 Belize 146.4 In Store 8 British
Virgin Islands 40.2 Catalog 9 Canada 11.8 Catalog 10 Cayman Islands 71.0 In
Store *prices in USD
```

プログラム:STYLE および BLANKLINE オプションを使用した HTML レポートの作成

```
options obs=5;

ods html file='your_file_styles.html';


proc print data=exprev
   style(header)={fontstyle=italic color= green}
   style(obs)={backgroundcolor=#a8a44ff8a color=blue}
   blankline=(count= 1 style={backgroundcolor=cx456789});

   var country price sale_type;

   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

プログラムの説明

さらに、追加の形式設定を HTML 出力に追加できます。次の例では STYLE オプションを使用して、シェーディングとスペースを HTML レポートに追加できます。

```
options obs=5;

ods html file='your_file_styles.html';
```

スタイルを指定した HTML 出力を作成します。 最初の STYLE オプションは、列ヘッダーが緑色の斜体で書き込まれるように指定します。2 番目の STYLE オプションは、オブザベーション番号列の背景色が RGB カラー a8a44ff8a に、テキストの色が青になるように指定します。BLANKLINE オプションは、各オブザベーション間に黒線を追加し、背景色に CMYK カラー cx456789 を使用するように指定します。OBSHEADER 位置にスタイルが定義されていないため、出力の Obs 列ヘッダーには緑ではなくデフォルトのスタイルカラーが使用されます。

```
proc print data=exprev
   style(header)={fontstyle=italic color= green}
   style(obs)={backgroundcolor=#a8a44ff8a color=blue}
   blankline=(count= 1 style={backgroundcolor=cx456789});

   var country price sale_type;

   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

出力:スタイルを使用した HTML 出力

アウトプット 45.3 *変数の選択:スタイルを使用した HTML 出力*

Monthly Price Per Unit and Sale Type for Each Country

Obs	Country	Price	Sale_Type
1	Antarctica	92.6	Internet
2	Puerto Rico	51.2	Catalog
3	Virgin Islands (U.S.)	31.1	In Store
4	Aruba	123.7	Catalog
5	Bahamas	113.4	Catalog

*prices in USD

プログラム:LISTING レポートの作成

```
options nodate pageno=1 linesize=80 pagesize=30 obs=10;

ods html close;
ods listing;

proc print data=exprev double;

   var country price sale_type;

   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;

ods listing close;
ods html;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=オプションは開始ページ番号を指定します。LINESIZE=オプションで出力行長さを指定し、PAGESIZE=オプションで出力ページの行数を指定します。OBS=オプションは、表示するオブザベーション数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=30 obs=10;
```

HTML 出力先を閉じて、LISTING 出力先を開きます。 HTML は、AS 開始時のデフォルトの出力先です。LISTING レポートのみ作成するには、HTML 出力先を閉じて、LISTING 出力先を開きます。

```
ods html close;
ods listing;
```

データセット EXPREV を印刷します。 EXPREV には、2 ヶ月間の企業の製品の発注の種類と単位価格に関する情報が含まれています。DOUBLE は、オブザベーション間にブランクを挿入します。DOUBLE オプションは、HTML 出力に影響しません。

```
proc print data=exprev double;
```

レポートに含める変数を選択します。 VAR ステートメントは、Country、Price、Sale_Type に対する列をこの順序で作成します。

```
   var country price sale_type;
```

タイトルとフットノートを指定します。 TITLE ステートメントは、レポートのタイトルを指定します。FOOTNOTE ステートメントは、レポートのフットノートを指定します。

```
   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

LISTING 出力先を閉じて、HTML 出力先を再度開きます。 HTML 出力先を閉じてから再度開くと、HTML 出力は Work ライブラリではなく、現在のディレクトリに保存されます。

```
ods listing close;
ods html;
```

出力:LISTING

デフォルトでは、PROC PRINT は次の列ヘッダー下の番号別に各オブザベーションを識別します。**Obs**

アウトプット 45.4 *変数の選択:LISTING 出力*

```
Monthly Price Per Unit and Sale Type for Each Country 1 Sale_ Obs Country Price
Type 1 Antarctica 92.6 Internet 2 Puerto Rico 51.2 Catalog 3 Virgin Islands
(U.S.)      31.1 In Store 4 Aruba 123.7 Catalog 5 Bahamas 113.4 Catalog 6
Bermuda 41.0 Catalog 7 Belize 146.4 In Store 8 British Virgin Islands 40.2
Catalog 9 Canada 11.8 Catalog 10 Cayman Islands 71.0 In Store *prices in USD
```

例 2: 列ヘッダーテキストのカスタマイズ

要素: PROC PRINT ステートメントオプション
- N
- OBS=
- SPLIT=
- STYLE

VAR ステートメントオプション:
- STYLE

他の要素: LABEL statement

ODS PDF ステートメント

FORMAT statement

TITLE statement

データセット: EXPREV

ODS 出力先: LISTING, PDF

詳細

この例では、次のタスクについて説明します。

- LISTING 出力の変数の列ヘッダーのテキストに下線を引く
- PDF 出力の変数の列ヘッダーに背景色を追加する
- 番号別にオブザベーションを識別する列の列ヘッダーをカスタマイズする
- レポートにオブザベーション数を表示する
- 変数 Price の値をドル記号とピリオドとともに書き込む

プログラム:LISTING レポートの作成

```
options nodate pageno=1 linesize=80 pagesize=30 obs=10;

ods html close;
ods listing;

proc print data=exprev split='*' n
obs='Observation*Number*===========';

   var country sale_type price;

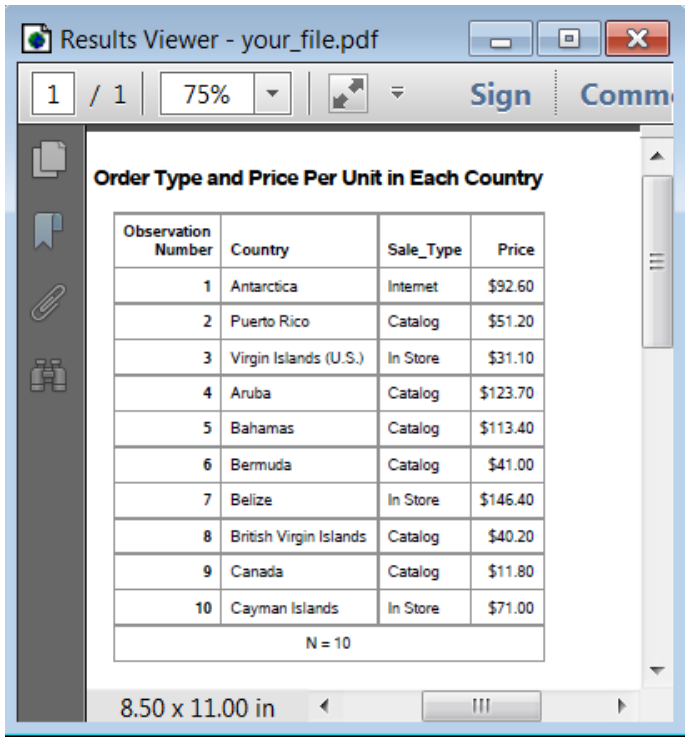
   label country='Country Name**============'
         sale_type='Order Type**==========='
         price='Price Per Unit*in USD*==============';

   format price dollar10.2;
   title 'Order Type and Price Per Unit in Each Country';
run;

ods listing close;
ods html;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=オプションは開始ページ番号を指定します。LINESIZE=オプションで出力行長さを指定し、PAGESIZE=オプションで出力ページの行数を指定します。OBS=オプションは、表示するオブザベーション数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=30 obs=10;
```

HTML 出力先を閉じて、LISTING 出力先を開きます。 デフォルトでは、HTML 出力先が開いています。

```
ods html close;
ods listing;
```

レポートを印刷し、列ヘッダーを定義します。 SPLIT=は、列ヘッダーで新しい行を開始する文字としてアスタリスクを識別します。N オプションは、オブザベーション数をレポートの最後に印刷します。OBS=は、各オブザベーションを番号別に識別する列の列ヘッダーを指定します。区切り文字(*)は、列ヘッダーで新しい行を開始します。OBS=の値の等号記号(=)は、列ヘッダーに下線を引きます。

```
proc print data=exprev split='*' n
```

```
obs='Observation*Number*===========';
```

レポートに含める変数を選択します。 VAR ステートメントは、Country、Sale_Type、Price に対する列をこの順序で作成します。

```
var country sale_type price;
```

変数のラベルを列ヘッダーとして割り当てます。 LABEL ステートメントは、ラベルを ROC PRINT ステップの期間の各変数と連付けます。PROC PRINT ステートメントで SPLIT=オプションを使用すると、プロシジャは列ヘッダーにラベルを使用します。区切り文字(*)は、列ヘッダーで新しい行を開始します。ラベルの等号記号(=)は、列ヘッダーに下線を引きます。

```
label country='Country Name**============'
      sale_type='Order Type**==========='
      price='Price Per Unit*in USD*==============';
```

レポートのタイトルを指定し、番号を含む変数をフォーマットします。 FORMAT ステートメントは、DOLLAR10.2 出力形式をレポートの変数 Price に割り当てます。TITLE ステートメントでタイトルを指定します。

```
format price dollar10.2;
title 'Order Type and Price Per Unit in Each Country';
run;
```

LISTING 出力先を閉じて、HTML 出力先を再度開きます。

```
ods listing close;
ods html;
```

出力:LISTING

アウトプット 45.5 *列ヘッダーのカスタマイズ:LISTING 出力*

```
Order Type and Price Per Unit in Each Country 1 Observation Country Name Order
Type Price Per Unit Number in USD ========== ============ ==========
============== 1 Antarctica Internet $92.60 2 Puerto Rico Catalog $51.20 3
Virgin Islands (U.S.)      In Store $31.10 4 Aruba Catalog $123.70 5 Bahamas
Catalog $113.40 6 Bermuda Catalog $41.00 7 Belize In Store $146.40 8 British
Virgin Islands Catalog $40.20 9 Canada Catalog $11.80 10 Cayman Islands In
Store $71.00 N = 10
```

プログラム:PDF レポートの作成

```
options obs=10;

ods pdf file='your_file.pdf';

proc print data=exprev n obs='Observation Number';


   var country sale_type price;
         label country='Country Name'
               sale_type='Order Type'
               price='Price Per Unit in USD';
         format price dollar10.2;
         title 'Order Type and Price Per Unit in Each Country';
run;
```

```
ods pdf close;
```

プログラムの説明

ODS ステートメントをいくつか追加して、PDF 出力を簡単に作成できます。次の例では、ODS ステートメントが PDF 出力の作成のために追加されました。

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;
```

PDF 出力を作成し、出力を格納するファイルを指定します。 ODS PDF ステートメントは PDF 出力先を開き、PDF 出力を作成します。FILE=ステートメントで、PDF 出力を含む外部ファイルを指定します。

```
ods pdf file='your_file.pdf';
```

プロシジャオプションを設定します。 N オプションは、オブザベーション数をレポートの最後に印刷します。OBS=は、各オブザベーションを番号別に識別する列の列ヘッダーを指定します。

```
proc print data=exprev n obs='Observation Number';
```

データセットの変数を処理します。 VAR ステートメントは、ランク付けする変数を指定します。LABEL ステートメントは、変数名のかわりに印刷するテキストを作成します。FORMAT ステートメントは、DOLLARw.フォーマットを使用して価格変数をフォーマットするように指定します。TITLE ステートメントは、レポートのタイトルを作成します。

```
   var country sale_type price;
         label country='Country Name'
               sale_type='Order Type'
               price='Price Per Unit in USD';
         format price dollar10.2;
         title 'Order Type and Price Per Unit in Each Country';
run;
```

PDF 出力先を閉じます。 ODS PDF CLOSE ステートメントで、PDF 出力先を閉じます。

```
ods pdf close;
```

出力:PDF

アウトプット 45.6 *列ヘッダーのカスタマイズ:デフォルト PDF 出力*



Order Type and Price Per Unit in Each Country

Observation Number	Country	Sale_Type	Price
1	Antarctica	Internet	$92.60
2	Puerto Rico	Catalog	$51.20
3	Virgin Islands (U.S.)	In Store	$31.10
4	Aruba	Catalog	$123.70
5	Bahamas	Catalog	$113.40
6	Bermuda	Catalog	$41.00
7	Belize	In Store	$146.40
8	British Virgin Islands	Catalog	$40.20
9	Canada	Catalog	$11.80
10	Cayman Islands	In Store	$71.00
N = 10			

プログラム:STYLE オプションを使用した PDF レポートの作成

```
options obs=10;

ods pdf file='your_file.pdf';

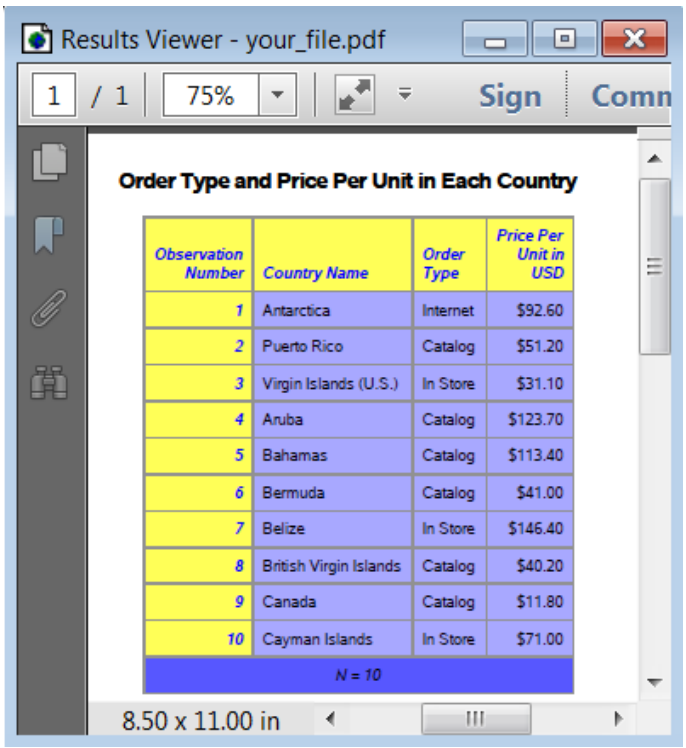
proc print data=exprev n obs='Observation Number'
     style(n)={backgroundcolor=light blue fontstyle=italic}
     style(header obs obsheader)={backgroundcolor=light yellow color=blue
      fontstyle=italic};
     style(data)={backgroundcolor=very light blue}

   var country sale_type price / style(data)=[backgroundcolor=very light blue];
   label country='Country Name'
         sale_type='Order Type'
         price='Price Per Unit in USD';
   format price dollar10.2;
run;

title 'Order Type and Price Per Unit in Each Country';

ods pdf close;
```

プログラムの説明

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;

ods pdf file='your_file.pdf';
```

様式的 PDF 出力を作成します。最初の STYLE オプションは、N の値を含むセルの背景色が薄青色に、フォントスタイルが斜体にそれぞれ変更されるように指定します。2 番目の STYLE オプションは、オブザベーション列、オブザベーションヘッダー、その他の変数のヘッダーの背景色が薄黄色に、テキストの色が青色に、フォントスタイルが斜体にそれぞれ変更されるように指定します。

```
proc print data=exprev n obs='Observation Number'
     style(n)={backgroundcolor=light blue fontstyle=italic}
     style(header obs obsheader)={backgroundcolor=light yellow color=blue
      fontstyle=italic};
     style(data)={backgroundcolor=very light blue}
```

様式的 PDF 出力を作成します。STYLE オプションは、データを含むセルの色が非常に薄い青色に変更するものです。

```
   var country sale_type price / style(data)=[backgroundcolor=very light blue];
   label country='Country Name'
         sale_type='Order Type'
         price='Price Per Unit in USD';
   format price dollar10.2;
run;

title 'Order Type and Price Per Unit in Each Country';
```

PDF 出力先を閉じます。ODS PDF CLOSE ステートメントで、PDF 出力先を閉じます。

```
ods pdf close;
```

出力:スタイルを使用した PDF レポート

アウトプット 45.7 *列ヘッダーのカスタマイズ:スタイルを使用した PDF*



Observation Number	Country Name	Order Type	Price Per Unit in USD
1	Antarctica	Internet	$92.60
2	Puerto Rico	Catalog	$51.20
3	Virgin Islands (U.S.)	In Store	$31.10
4	Aruba	Catalog	$123.70
5	Bahamas	Catalog	$113.40
6	Bermuda	Catalog	$41.00
7	Belize	In Store	$146.40
8	British Virgin Islands	Catalog	$40.20
9	Canada	Catalog	$11.80
10	Cayman Islands	In Store	$71.00
N = 10			

例 3: オブザベーショングループごとのレポートの別のセクションの作成

要素: PROC PRINT ステートメントオプション
- LABEL
- N=
- NOOBS
- STYLE

BY ステートメント

PAGEBY statement

他の要素: SORT プロシジャ

FORMAT statement

LABEL statement

ODS RTF ステートメント

TITLE statement

データセット: EXPREV

ODS 出力先: HTML, RTF

詳細

この例では、次のタスクについて説明します。

- 各行の始めのオブザベーション番号の印刷を抑制する
- 種類ごとに販売データをレポートの別のセクションに表示する
- デフォルトの HTML レポートを作成する
- デフォルトの RTF レポートと様式的 RTF レポートを作成する

プログラム:HTML レポートの作成

```
options obs=10;

proc sort data=exprev;
   by sale_type order_date quantity;
run;

proc print data=exprev n='Number of observations for the month: '
           noobs label;

   var quantity cost price;

   by sale_type order_date;
   pageby order_date;

   label sale_type='Order Type' order_date='Order Date';

   format price dollar7.2 cost dollar7.2;
   title 'Prices and Cost Grouped by Date and Order Type';
   title2 'in USD';
run;

proc options option=bufno define;
run;
```

プログラムの説明

デフォルトでは、HTML 出力先は開いています。ODS HTML ステートメントは不要です。

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;
```

EXPREV データセットを並べ替えます。 PROC SORT は、オブザベーションを Sale_Type、Order_Date、Quantity 別に並べ替えます。

```
proc sort data=exprev;
   by sale_type order_date quantity;
run;
```

レポートを印刷し、各 BY グループのオブザベーション数の合計を指定し、オブザベーション番号の印刷を抑制します。 N=は、BY グループのオブザベーション数を BY グループの最後に印刷します。N=オプションが提供する説明テキストは、その数字の前に表示されます。NOOBS は、行の始めのオブザベーション番号の印刷を抑制します。LABEL は、変数のラベルを列ヘッダーとして使用します。

```
proc print data=exprev n='Number of observations for the month: '
           noobs label;
```

レポートに含める変数を指定します。 VAR ステートメントは、Quantity、Cost、Price に対する列をこの順序で作成します。

```
   var quantity cost price;
```

発注の種類ごとに別のセクションを作成し、Order_Date の BY グループごとに改ページを指定します。 BY ステートメントは、BY グループごとにレポートの別のセクションを生成し、それぞれの上にヘッダーを印刷します。PAGEBY ステートメントは、Order_Date の値が変わるたびに新しいページを開始します。

```
   by sale_type order_date;
   pageby order_date;
```

列ヘッダーを作成します。 LABEL ステートメントは、ラベルを PROC PRINT ステップの期間に対する変数 Sale_Type と Order_Date と関連付けます。PROC PRINT ステートメントで LABEL オプションを使用すると、プロシジャは列ヘッダーにラベルを使用します。

```
   label sale_type='Order Type' order_date='Order Date';
```

数字を含む列をフォーマットして、タイトルとフットノートを指定します。 FORMAT ステートメントは、このレポートの Price と Cost に出力形式を割り当てます。TITLE ステートメントでタイトルを指定します。TITLE2 ステートメントは、2 番目のタイトルを指定します。

```
   format price dollar7.2 cost dollar7.2;
   title 'Prices and Cost Grouped by Date and Order Type';
   title2 'in USD';
run;

proc options option=bufno define;
run;
```

出力:出力:HTML

アウトプット 45.8 *オブザベーショングループ用レポートの各セクションの作成:HTML 出力*

Prices and Cost Grouped by Date and Order Type in USD

Order Type=Catalog Order Date=1/1/12

Quantity	Cost	Price
7	$9.25	$41.00
8	$28.45	$113.40
14	$12.10	$51.20
30	$59.00	$123.70
Number of observations for the month: 4		

Prices and Cost Grouped by Date and Order Type in USD

Order Type=Catalog Order Date=1/2/12

Quantity	Cost	Price
11	$20.20	$40.20
100	$5.00	$11.80
Number of observations for the month: 2		

Prices and Cost Grouped by Date and Order Type in USD

Order Type=In Store Order Date=1/1/12

Quantity	Cost	Price
25	$15.65	$31.10
Number of observations for the month: 1		

Prices and Cost Grouped by Date and Order Type in USD

Order Type=In Store Order Date=1/2/12

Quantity	Cost	Price
2	$36.70	$146.40
20	$32.30	$71.00
Number of observations for the month: 2		

Prices and Cost Grouped by Date and Order Type in USD

Order Type=Internet Order Date=1/1/12

Quantity	Cost	Price
2	$20.70	$92.60
Number of observations for the month: 1		

プログラム:HTML レポートの作成

```
options obs=10;

ods rtf file='your_file.rtf' startpage=no;

proc sort data=exprev;
   by sale_type order_date quantity;
run;

proc print data=exprev n='Number of observations for each order type:'
           noobs label;
   var quantity cost price;
   by sale_type order_date;
   pageby order_date;
   label sale_type='Order Type' order_date='Order Date';
   format price dollar7.2 cost dollar7.2;
   title 'Price and Cost Grouped by Date and Order Type';
```

```
   title2 'in USD';
run;

ods rtf close;
```

プログラムの説明

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;
```

Microsoft Word に対する出力を作成し、出力を格納するファイルを指定します。 ODS RTF ステートメントは RTF 出力先を開いて、Microsoft Word に対しフォーマットされた出力を作成します。FILE=オプションは、RTF 出力を含む外部ファイルを指定します。STARTPAGE=NO オプションは、新しいページが各 BY グループの始めに明示的に挿入されないように指定します。

```
ods rtf file='your_file.rtf' startpage=no;

proc sort data=exprev;
   by sale_type order_date quantity;
run;

proc print data=exprev n='Number of observations for each order type:'
           noobs label;
   var quantity cost price;
   by sale_type order_date;
   pageby order_date;
   label sale_type='Order Type' order_date='Order Date';
   format price dollar7.2 cost dollar7.2;
   title 'Price and Cost Grouped by Date and Order Type';
   title2 'in USD';
run;
```

RTF 出力先を閉じます。 ODS RTF CLOSE ステートメントは、RTF 出力先を閉じます。

```
ods rtf close;
```

出力:RTF

アウトプット 45.9 *オブザベーショングループ用レポートの各セクションの作成: デフォルト RTF 出力*

Price and Cost Grouped by Date and Order Type in USD

Order Type=Catalog Order Date=1/1/12

Quantity	Cost	Price
7	$9.25	$41.00
8	$28.45	$113.40
14	$12.10	$51.20
30	$59.00	$123.70
Number of observations for each order type:4		

Order Type=Catalog Order Date=1/2/12

Quantity	Cost	Price
11	$20.20	$40.20
100	$5.00	$11.80
Number of observations for each order type:2		

Order Type=In Store Order Date=1/1/12

Quantity	Cost	Price
25	$15.65	$31.10
Number of observations for each order type:1		

Order Type=In Store Order Date=1/2/12

Quantity	Cost	Price
2	$36.70	$146.40
20	$32.30	$71.00
Number of observations for each order type:2		

Order Type=Internet Order Date=1/1/12

Quantity	Cost	Price
2	$20.70	$92.60
Number of observations for each order type:1		

プログラム:STYLE オプションを使用して RTF レポートを作成する

```
options obs=10;

ods rtf file='your_file.rtf' startpage=no;

proc sort data=exprev;
   by sale_type order_date quantity;
run;

proc print data=exprev n='Number of observations for the month: '
           noobs label style(N)={backgroundcolor=very light gray};
   var quantity / style(header)=[backgroundcolor=light yellow];
   var cost / style(header)=[backgroundcolor=light blue foreground =
white];
   var price / style(header)=[backgroundcolor=light green];
   by sale_type order_date;
   pageby order_date;
   label sale_type='Order Type' order_date='Order Date';
   format price dollar7.2 cost dollar7.2;

title 'Prices and Cost Grouped by Date and Order Type';
title2 '*prices in USD';
run;

ods rtf close;
```

プログラムの説明

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;

ods rtf file='your_file.rtf' startpage=no;

proc sort data=exprev;
   by sale_type order_date quantity;
run;
```

様式的 RTF レポートを作成します。最初の STYLE オプションは、オブザベーションの数を含むセルの背景色が薄灰色に変更されるように指定します。2 番目の STYLE オプションは、変数 Quantity の列ヘッダーの背景色が薄黄色に変更されるように指定します。3 番目の STYLE オプションは、変数 Cost の列ヘッダーの背景色が薄青色に、フォントの色が白色にそれぞれ変更されるように指定します。4 番目の STYLE オプションは、変数 Price の列ヘッダーの背景色が薄緑色に変更されるように指定します。

```
proc print data=exprev n='Number of observations for the month: '
           noobs label style(N)={backgroundcolor=very light gray};
   var quantity / style(header)=[backgroundcolor=light yellow];
   var cost / style(header)=[backgroundcolor=light blue foreground =
white];
   var price / style(header)=[backgroundcolor=light green];
   by sale_type order_date;
   pageby order_date;
   label sale_type='Order Type' order_date='Order Date';
   format price dollar7.2 cost dollar7.2;

title 'Prices and Cost Grouped by Date and Order Type';
title2 '*prices in USD';
```

```
run;

ods rtf close;
```

出力:スタイルを使用した RTF

アウトプット 45.10 *オブザベーショングループ用レポートの各セクションの作成:スタイルを使用した RTF 出力*

Prices and Cost Grouped by Date and Order Type
**prices in USD*

Order Type=Catalog Order Date=1/1/12

Quantity	Cost	Price
7	$9.25	$41.00
8	$28.45	$113.40
14	$12.10	$51.20
30	$59.00	$123.70
Number of observations for the month: 4		

Order Type=Catalog Order Date=1/2/12

Quantity	Cost	Price
11	$20.20	$40.20
100	$5.00	$11.80
Number of observations for the month: 2		

Order Type=In Store Order Date=1/1/12

Quantity	Cost	Price
25	$15.65	$31.10
Number of observations for the month: 1		

Order Type=In Store Order Date=1/2/12

Quantity	Cost	Price
2	$36.70	$146.40
20	$32.30	$71.00
Number of observations for the month: 2		

Order Type=Internet Order Date=1/1/12

Quantity	Cost	Price
2	$20.70	$92.60
Number of observations for the month: 1		

例 4: 1 つの BY グループを使用し、数値変数を合計する

要素: PROC PRINT ステートメントオプション
N=
SUMLABEL
BY ステートメント
SUM statement

他の要素: ODS CSVALL ステートメント
SORT プロシジャ
TITLE statement
#BYVAL 指定
SAS システムオプション:
BYLINE
NOBYLINE

データセット: EXPREV

ODS 出力先: HTML, CSV

詳細

この例では、次のタスクについて説明します。

- 地域ごと、およびすべての地域に対する経費と収益を合計する
- 各 BY グループおよびレポート全体のオブザベーション数を表示する
- 地域名を含む、カスタマイズされたタイトルを作成するこのタイトルは、BY グループごとのデフォルトの BY 行となります。
- デフォルト HTML ファイルを作成する
- CSV ファイルを作成する

プログラム:HTML レポートの作成

```
options obs=10 nobyline;

proc sort data=exprev;
   by sale_type;
run;

proc print data=exprev noobs label sumlabel
           n='Number of observations for the order type: '
           'Number of observations for the data set: ';

   var country order_date quantity price;

   label  sale_type='Sale Type'
          price='Total Retail Price* in USD'
          country='Country' order_date='Date' quantity='Quantity';

   sum price quantity;
   by sale_type;

   format price dollar7.2;

   title 'Retail and Quantity Totals for #byval(sale_type) Sales';
```

```
run;

options byline;
```

プログラムの説明

デフォルトでは、HTML 出力先は開いています。このプログラムでは、HTML 出力にデフォルトのファイル名が使用されます。ODS HTML ステートメントは不要です。

新しいページで BY グループをそれぞれ開始し、デフォルトの BY 行の印刷を抑制します。 SAS システムオプション NOBYLINE は、デフォルトの BY 行の印刷を抑制します。PROC PRINT を NOBYLINE オプションと使用すると、BY グループがそれぞれ新しいページを開始します。OBS=オプションは、処理するオブザベーション数を指定します。

```
options obs=10 nobyline;
```

データセットを並べ替えます。 PROC SORT は、オブザベーションを Sale_Type 別に並べ替えます。

```
proc sort data=exprev;
   by sale_type;
run;
```

レポートを印刷し、オブザベーション数の印刷を抑制し、選択した変数に対するオブザベーションの合計数を印刷します。 NOOBS は、行の始めのオブザベーション番号の印刷を抑制します。SUMLABEL は、それぞれの要約行に BY 変数ラベルを印刷します。N=は BY グループの最後に BY グループのオブザベーション数を印刷し、(SUM ステートメントにより)レポートの最後にデータセットのオブザベーション数を印刷します。N=が提供する最初の説明テキストが BY グループごとの数より前に置かれます。N=が提供する 2 番目の説明テキストがデータセット全体の数より前に置かれます。

```
proc print data=exprev noobs label sumlabel
           n='Number of observations for the order type: '
           'Number of observations for the data set: ';
```

レポートに含める変数を選択します。 VAR ステートメントは、Country、Order_Date、Quantity、Price に対する列をこの順序で作成します。

```
    var country order_date quantity price;
```

変数のラベルを列ヘッダーとして割り当てます。 LABEL ステートメントは、ラベルを ROC PRINT ステップの期間の各変数と連付けます。

```
    label  sale_type='Sale Type'
           price='Total Retail Price* in USD'
           country='Country' order_date='Date' quantity='Quantity';
```

選択されている変数の値を合計します。 SUM ステートメントは、それだけでデータセット全体の Price と Quantity の値を合計します。PROC PRINT ステップに BY ステートメントが含まれているため、SUM ステートメントも複数のオブザベーションを含む販売の種類ごとに Price と Quantity の値を合計します。

```
    sum price quantity;
    by sale_type;
```

指定列の数値をフォーマットします。 FORMAT ステートメントは、DOLLAR7.2.出力形式をこのレポートの Price に割り当てます。

```
    format price dollar7.2;
```

動的な(または現在の)タイトルを指定して、フォーマットします。 TITLE ステートメントでタイトルを指定します。#BYVAL 指定は、タイトルに BY 変数 Sale_Type の現在の値を配置します。NOBYLINE が有効であるため、BY グループはそれぞれ新しいページで開始し、タイトルは BY 行として機能します。

```
   title 'Retail and Quantity Totals for #byval(sale_type) Sales';
run;
```

デフォルトの BY 行を生成します。 SAS システムオプション BYLINE は、デフォルトの BY 行の印刷をリセットします。

```
options byline;
```

出力:出力:HTML

アウトプット 45.11 *1 つの BY グループ HTML 出力を使用し、数値変数を合計する*

Retail and Quantity Totals for Catalog Sales

Country	Date	Quantity	Total Retail Price* in USD
Bermuda	1/1/12	7	$41.00
Bahamas	1/1/12	8	$113.40
Puerto Rico	1/1/12	14	$51.20
Aruba	1/1/12	30	$123.70
British Virgin Islands	1/2/12	11	$40.20
Canada	1/2/12	100	$11.80
Sale Type		**170**	**$381.30**

Number of observations for the order type: 6

Retail and Quantity Totals for In Store Sales

Country	Date	Quantity	Total Retail Price* in USD
Virgin Islands (U.S.)	1/1/12	25	$31.10
Belize	1/2/12	2	$146.40
Cayman Islands	1/2/12	20	$71.00
Sale Type		**47**	**$248.50**

Number of observations for the order type: 3

Retail and Quantity Totals for Internet Sales

Country	Date	Quantity	Total Retail Price* in USD
Antarctica	1/1/12	2	$92.60
		219	$722.40

Number of observations for the order type: 1
Number of observations for the data set: 10

プログラム:CSV ファイルを作成する

```
options obs=10 nobyline;

ods csvall file='your_file.csv';

proc sort data=exprev;
   by sale_type;
run;

proc print data=exprev noobs label sumlabel
           n='Number of observations for the order type: '
           'Number of observations for the data set: ';

var country order_date quantity price;

   label  price='Total Retail Price* in USD'
          country='Country' order_date='Date' quantity='Quantity';

   sum price quantity;
   by sale_type;

   format price dollar7.2;

   title 'Retail and Quantity Totals for #byval(sale_type) Sales';
run;

options byline;

ods csvall close;
```

プログラムの説明

```
options obs=10 nobyline;
```

CSV 形式の出力を生成し、格納するファイルを指定します。 ODS CSVALL ステートメントは CSVALL 出力先を開き、表形式出力を含むファイルをタイトル、説明、BY 行で作成します。FILE=引数は、CSV 出力を含む外部ファイルを指定します。

```
ods csvall file='your_file.csv';

proc sort data=exprev;
   by sale_type;
run;

proc print data=exprev noobs label sumlabel
           n='Number of observations for the order type: '
           'Number of observations for the data set: ';

var country order_date quantity price;
```

```
   label  price='Total Retail Price* in USD'
          country='Country' order_date='Date' quantity='Quantity';

   sum price quantity;
   by sale_type;

   format price dollar7.2;

   title 'Retail and Quantity Totals for #byval(sale_type) Sales';
run;

options byline;
```

CSVALL 出力先を閉じます。 ODS CSVALL CLOSE ステートメントは、CSVALL 出力先を閉じます。

```
ods csvall close;
```

出力:CSV ファイル

アウトプット 45.12 *1 つの BY グループを使用し、数値変数を合計する:1 つの BY グループを使用し、数値変数を合計する:*

your_file.csv

	A	B	C	D	E	F
1	Retail and Quantity Totals for Catalog Sales					
2						
3	Country	Date	Quantity	Total Retail Price* in USD		
4	Bermuda	1/1/12	7	$41.00		
5	Bahamas	1/1/12	8	$113.40		
6	Puerto Ric	1/1/12	14	$51.20		
7	Aruba	1/1/12	30	$123.70		
8	British Vir	1/2/12	11	$40.20		
9	Canada	1/2/12	100	$11.80		
10	Sale_Type		170	$381.30		
11	Number of observations for the order type: 6					
12						
13	Retail and Quantity Totals for In Store Sales					
14						
15	Country	Date	Quantity	Total Retail Price* in USD		
16	Virgin Isla	1/1/12	25	$31.10		
17	Belize	1/2/12	2	$146.40		
18	Cayman Is	1/2/12	20	$71.00		
19	Sale_Type		47	$248.50		
20	Number of observations for the order type: 3					
21						
22	Retail and Quantity Totals for Internet Sales					
23						
24	Country	Date	Quantity	Total Retail Price* in USD		
25	Antarctica	1/1/12	2	$92.60		
26			219	$722.40		
27	Number of observations for the order type: 1					
28	Number of observations for the data set: 10					

例 5: 複数の BY 変数を使用し、数値変数を合計する

要素: PROC PRINT ステートメントオプション
GRANDTOTAL_LABEL=
N=
NOOBS
STYLE
SUMLABEL=
BY ステートメント
SUM statement

他の要素: ODS HTML ステートメント
LABEL statement
FORMAT statement
SORT プロシジャ
TITLE statement

データセット: EXPREV

ODS 出力先: HTML, LISTING

詳細

この例では、次のタスクについて説明します。

- 次の項目に対する数量と販売価格を合計する:
  - 各注文日
  - レポートに複数行が存在する販売の酒類
  - レポートのすべての行
- 各 BY グループおよびレポート全体のオブザベーション数を表示します
- 要約行の BY グループ変数名のかわりにカスタマイズしたラベルを表示します
- 総計行にカスタマイズしたラベルを表示します
- デフォルトの HTML レポートを作成します
- 様式的 HTML レポートを作成します

プログラム:HTML レポートの作成

```
options obs=10;

proc sort data=exprev;
   by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel='Totals' grandtotal_label='Grand Total';
   by sale_type order_date;
   sum price quantity cost;

   label  sale_type='Sale Type' order_date='Sale Date';
   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
```

```
run;
```

プログラムの説明

```
options obs=10;
```

HTML 出力を生成し、出力を格納するファイルを指定します。 デフォルトでは、HTML 出力先は開いています。ODS HTML FILE=ステートメントは、HTML 出力を含むファイルを作成します。FILE=引数は、HTML 出力を含む外部ファイルを指定します。

```
proc sort data=exprev;
   by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel='Totals' grandtotal_label='Grand Total';
   by sale_type order_date;
   sum price quantity cost;

   label  sale_type='Sale Type' order_date='Sale Date';
   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;
```

出力:出力:HTML

アウトプット 45.13 *複数の BY 変数を使用し、数値変数を合計する店舗販売: デフォルト HTML 出力*

Retail and Quantity Totals for Each Sale Date and Sale Type

Sale Type=Catalog Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Puerto Rico	99999999	1/5/12	14	$51.20	$12.10
Aruba	99999999	1/4/12	30	$123.70	$59.00
Bahamas	99999999	1/4/12	8	$113.40	$28.45
Bermuda	99999999	1/4/12	7	$41.00	$9.25
Totals			**59**	**$329.30**	**$108.80**
N = 4					

Sale Type=Catalog Sale Date=1/2/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
British Virgin Islands	99999999	1/5/12	11	$40.20	$20.20
Canada	99999999	1/5/12	100	$11.80	$5.00
Totals			**111**	**$52.00**	**$25.20**
Totals			**170**	**$381.30**	**$134.00**
N = 2					

Sale Type=In Store Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Virgin Islands (U.S.)	99999999	1/4/12	25	$31.10	$15.65
N = 1					

Sale Type=In Store Sale Date=1/2/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Belize	120458	1/2/12	2	$146.40	$36.70
Cayman Islands	120454	1/2/12	20	$71.00	$32.30
Totals			**22**	**$217.40**	**$69.00**
Totals			**47**	**$248.50**	**$84.65**
N = 2					

Sale Type=Internet Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Antarctica	99999999	1/7/12	2	$92.60	$20.70
Grand Total			**219**	**$722.40**	**$239.35**
N = 1 Total N = 10					

プログラム:STYLE オプションを使用した HTML レポートの作成

```
options obs=10;

proc sort data=exprev;
   by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel='Totals' grandtotal_label='Grand Total';

   by sale_type order_date;
sum price / style(GRANDTOTAL)=[backgroundcolor=white color=blue];
sum quantity / style(TOTAL)=[backgroundcolor=dark blue color=white];

   label  sale_type='Sale Type' order_date='Sale Date';
   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;
```

プログラムの説明

```
options obs=10;
```

```
proc sort data=exprev;
   by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel='Totals' grandtotal_label='Grand Total';
```

スタイルを指定した HTML 出力を作成します。 最初の SUM ステートメントの STYLE オプションは、変数 Price の総計を含むセルの背景色が白に、フォントの色が青に変更されるように指定します。2 番目の SUM ステートメントの STYLE オプションは、変数 Quantity の合計を含むセルの背景色が紺青に、フォントの色が白に変更されるように指定します。

```
   by sale_type order_date;
sum price / style(GRANDTOTAL)=[backgroundcolor=white color=blue];
sum quantity / style(TOTAL)=[backgroundcolor=dark blue color=white];

   label  sale_type='Sale Type' order_date='Sale Date';
   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;
```

出力:スタイルを使用した HTML

アウトプット 45.14 *複数の BY 変数を使用し、数値変数を合計するカタログ販売:スタイルを使用した HTML 出力*

Retail and Quantity Totals for Each Sale Date and Sale Type

Sale Type=Catalog Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Puerto Rico	99999999	1/5/12	14	$51.20	$12.10
Aruba	99999999	1/4/12	30	$123.70	$59.00
Bahamas	99999999	1/4/12	8	$113.40	$28.45
Bermuda	99999999	1/4/12	7	$41.00	$9.25
Totals			59	$329.30	
N = 4					

Sale Type=Catalog Sale Date=1/2/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
British Virgin Islands	99999999	1/5/12	11	$40.20	$20.20
Canada	99999999	1/5/12	100	$11.80	$5.00
Totals			111	$52.00	
Totals			170	$381.30	
N = 2					

Sale Type=In Store Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Virgin Islands (U.S.)	99999999	1/4/12	25	$31.10	$15.65
N = 1					

Sale Type=In Store Sale Date=1/2/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Belize	120458	1/2/12	2	$146.40	$36.70
Cayman Islands	120454	1/2/12	20	$71.00	$32.30
Totals			22	$217.40	
Totals			47	$248.50	
N = 2					

Sale Type=Internet Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Antarctica	99999999	1/7/12	2	$92.60	$20.70
Grand Total			219	$722.40	
N = 1 Total N = 10					

プログラム:LISTING レポートの作成

```
options nodate pageno=1 linesize=80 pagesize=40 obs=15;

ods html close;
ods listing;


proc sort data=exprev;
   by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel=Totals' grandtotal_label='Grand Total';

   by sale_type order_date;
   sum price quantity;

   label  sale_type='Sale Type'
          order_date='Sale Date';
   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;

ods listing close;
ods html;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=オプションは開始ページ番号を指定します。LINESIZE=オプションで出力行長さを指定し、PAGESIZE=オプションで出力ページの行数を指定します。OBS=はオブザベーション 15 の後に設定されたデータセットでのプロセスを停止するよう指定します。

```
options nodate pageno=1 linesize=80 pagesize=40 obs=15;
```

HTML 出力先を閉じて、LISTING 出力先を開きます。 デフォルトでは、HTML 出力先は開いています。

```
ods html close;
ods listing;
```

データセットを並べ替えます。 PROC SORT は、オブザベーションを Sale_Type と Order_Date 別に並べ替えます。

```
proc sort data=exprev;
   by sale_type order_date;
run;
```

レポートを印刷し、オブザベーション番号の印刷を抑制し、選択した変数のオブザベーションの合計数を印刷し、要約行で BY 変数名の代わりに BY 変数ラベルを使用します。 N オプションは BY グループの最後に BY グループのオブザベーション数を印刷し、レポートの下部にレポートで使用されるオブザベーションの合計数を印刷します。NOOBS は、行の始めのオブザベーション番号の印刷を抑制します。SUMLABEL オプションは、BY 変数ラベルのかわりに要約行の‘Totals’を印刷します。

```
proc print data=exprev n noobs sumlabel=Totals' grandtotal_label='Grand Total';
```

BY グループごとにレポートの別のセクションを作成し、選択した変数の値を合計します。 BY ステートメントは、BY グループごとにレポートの別のセクションを生成します。SUM ステートメントは、それだけでデータセット全体の Price と Quantity の値を合計します。プログラムに BY ステートメントが含まれているため、SUM ステートメントも複数のオブザベーションを含む BY グループごとに Price と Quantity の値を合計します。

```
   by sale_type order_date;
   sum price quantity;
```

選択した変数のラベルを作成し、指定した変数の値をフォーマットし、タイトルを作成します。 LABEL ステートメントは、ラベルを ROC PRINT ステップの期間に対する変数 Sale_Type と Order_Date と連付けます。ラベルは、各 BY グループの始めの BY 行、BY 変数の代わりに要約行でそれぞれ使用されます。FORMAT ステートメントは、このレポートの変数 Price と Cost に出力形式を割り当てます。TITLE ステートメントでタイトルを指定します。

```
   label  sale_type='Sale Type'
          order_date='Sale Date';
   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;
```

LISTING 出力先を閉じて、HTML 出力先を再度開きます。

```
ods listing close;
ods html;
```

出力:LISTING

SPLIT=も LABEL オプションも使用されないため、レポートではデフォルトの列ヘッダー(変数名)が使用されます。それにもかかわらず、レポートの各セクションの上部の BY 行には、BY 変数のラベルとその値が表示されます。BY 変数のラベルは、レポート要約行の小計を識別します。

PROC PRINT は、複数のオブザベーションを含む BY グループごとに Price と Quantity を合計します。ただし、値が BY グループ間で変わる BY 変数に対してのみ合計が表示されます。たとえば最初の BY グループで、販売の種類が **`Catalog Sale`** で販売日が**`>1/1/12`** であれば、この販売日に対してのみ Quantity と Price が合計されます。これは次の BY グループも販売の種類が同じためです。

アウトプット 45.15 *Price と Quantity の総価格を示す PROC PRINT の LISTING 出力*

```
          Retail and Quantity Totals for Each Sale Date and Sale Type 1
---------------------- Sale Type=Catalog Sale Date=1/1/12 ----------------------
Ship_ Country Emp_ID Date Quantity Price Cost Puerto Rico 99999999 1/5/12 14
$51.20 $12.10 Aruba 99999999 1/4/12 30 $123.70 $59.00 Bahamas 99999999 1/4/12 8
$113.40 $28.45 Bermuda 99999999 1/4/12 7 $41.00 $9.25 ----------- --------
---------- Totals 59 $329.30 N = 4 ---------------------- Sale Type=Catalog Sale
Date=1/2/12 ---------------------- Ship_ Country Emp_ID Date Quantity Price Cost
British Virgin Islands 99999999 1/5/12 11 $40.20 $20.20 Canada 99999999 1/5/12
100 $11.80 $5.00 El Salvador 99999999 1/6/12 21 $266.40 $66.70
---------------------- -------- ---------- Totals 132 $318.40 Totals 191 $647.70
N = 3
```

```
 Retail and Quantity Totals for Each Sale Date and Sale Type 2
---------------------- Sale Type=In Store Sale Date=1/1/12 ----------------------
Ship_ Country Emp_ID Date Quantity Price Cost Virgin Islands (U.S.)   99999999
1/4/12      25          $31.10       $15.65 N = 1 --------------------- Sale
Type=In Store Sale Date=1/2/12 ---------------------- Ship_ Country Emp_ID Date
Quantity Price Cost Belize 120458 1/2/12 2 $146.40 $36.70 Cayman Islands 120454
1/2/12 20 $71.00 $32.30 Guatemala 120931 1/2/12 13 $144.40 $65.70 --------------
-------- ---------- Totals  35 $361.80 Totals 60 $392.90 N = 3
---------------------- Sale Type=Internet Sale Date=1/1/12 ----------------------
Ship_ Country Emp_ID Date Quantity Price Cost Antarctica 99999999 1/7/12 2 $92.60
$20.70 N = 1
```

```
Retail and Quantity Totals for Each Sale Date and Sale Type 3
---------------------- Sale Type=Internet Sale Date=1/2/12 ----------------------
Ship_ Country Emp_ID Date Quantity Price Cost Costa Rica 99999999 1/6/12 31
$53.00 $26.60 Cuba 121044 1/2/12 12 $42.40 $19.35 Dominican Republic 121040
1/2/12 13 $48.00 $23.95 ------------------- -------- ---------- Totals 56 $143.40
Totals 58 $236.00 ================== ======== ========== Grand Total 309
$1,276.60 N = 3 Total N = 15
```

例 6: レポート内の合計数を制限する

要素: BY ステートメント
SUM statement
SUMBY statement

他の要素: FORMAT statement
LABEL statement
ODS PDF ステートメント
SORT プロシジャ

TITLE statement

データセット: EXPREV

ODS 出力先: PDF

詳細

この例では、次のタスクについて説明します。

- 販売の種類と販売日の組み合わせごとにレポートの別のセクションを作成する
- 個別の日付ではなく、販売の種類別およびすべての販売の種類に対する数量と販売価格のみを合計する
- PDF ファイルを作成する

プログラム:PDF ファイルを作成する

```
options obs=10;

ods html close;
ods pdf file='your_file.pdf';

proc sort data=exprev;
   by sale_type order_date;
run;

proc print data=exprev noobs sumlabel='Total' grand_total='Grand Total';

   by sale_type order_date;
   sum price quantity;
   sumby sale_type;

   label sale_type='Sale Type' order_date='Sale Date';

   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Type';
run;

ods pdf close;
```

プログラムの説明

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;
```

PDF 出力を作成し、出力を格納するファイルを指定します。 ODS HTML CLOSE ステートメントはデフォルトの出力先を閉じます。ODS PDF ステートメントは PDF 出力先を開いて、PDF 出力を含むファイルを作成します。FILE=ステートメントで、PDF 出力を含む外部ファイルを指定します。

```
ods html close;
ods pdf file='your_file.pdf';
```

データセットを並べ替えます。 PROC SORT は、オブザベーションを Sales_Type と Order_Date 別に並べ替えます。

```
proc sort data=exprev;
   by sale_type order_date;
```

```
run;
```

レポートを印刷し、オブザベーション番号を削除します。 NOOBS は、行の始めのオブザベーション番号の印刷を抑制します。SUMLABEL は、各 BY グループの要約行の BY 変数のラベルを使用します。

```
proc print data=exprev noobs sumlabel='Total' grand_total='Grand Total';
```

地域ごとの値を合計します。 SUM ステートメントと BY ステートメントを使用して、BY グループごと、レポート全体の Price と Quantity の値を合計します。SUMBY ステートメントは、小計を販売の種類ごとに 1 つに制限します。

```
   by sale_type order_date;
   sum price quantity;
   sumby sale_type;
```

ラベルを特定の変数に割り当てます。 LABEL ステートメントは、ラベルを ROC PRINT ステップの期間に対する変数 Sale_Type と Order_Date と連付けます。これらのラベルは、BY グループタイトルまたは要約行で使用されます。

```
   label sale_type='Sale Type' order_date='Sale Date';
```

出力形式を必要な変数に割り当て、タイトルを指定します。 FORMAT ステートメントは、このレポートの Cost と Price に COMMA10.出力形式を割り当てます。TITLE ステートメントでタイトルを指定します。

```
   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Type';
run;
```

PDF 出力先を閉じます。 ODS PDF CLOSE ステートメントで、PDF 出力先を閉じます。

```
ods pdf close;
```

出力:PDF

アウトプット 45.16 *レポート内の合計数を制限する:PDF 出力*

Retail and Quantity Totals for Each Sale Type

Sale Type=Catalog Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Puerto Rico	99999999	1/5/12	14	$51.20	$12.10
Aruba	99999999	1/4/12	30	$123.70	$59.00
Bahamas	99999999	1/4/12	8	$113.40	$28.45
Bermuda	99999999	1/4/12	7	$41.00	$9.25

Sale Type=Catalog Sale Date=1/2/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
British Virgin Islands	99999999	1/5/12	11	$40.20	$20.20
Canada	99999999	1/5/12	100	$11.80	$5.00
Total			170	$381.30	

Sale Type=In Store Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Virgin Islands (U.S.)	99999999	1/4/12	25	$31.10	$15.65

Sale Type=In Store Sale Date=1/2/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Belize	120458	1/2/12	2	$146.40	$36.70
Cayman Islands	120454	1/2/12	20	$71.00	$32.30
Total			47	$248.50	

Sale Type=Internet Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Antarctica	99999999	1/7/12	2	$92.60	$20.70
Grand Total			219	$722.40	

プログラム:STYLE オプションを使用した PDF レポートの作成

```
options obs=10;

ods pdf file='your_file.pdf';

proc sort data=exprev;
   by sale_type order_date;
run;

proc print data=exprev noobs sumlabel='Total' grandtotal_label='Grand Total';

   by sale_type order_date;


   sum quantity price / style(TOTAL)=[backgroundcolor=light blue color=white];
```

```
   sum quantity price / style(GRANDTOTAL)=[backgroundcolor=green color=white];
   sumby sale_type;

   label sale_type='Sale Type' order_date='Sale Date';

   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Type';
run;

ods pdf close;
```

プログラムの説明

```
options obs=10;

ods pdf file='your_file.pdf';

proc sort data=exprev;
   by sale_type order_date;
run;

proc print data=exprev noobs sumlabel='Total' grandtotal_label='Grand Total';

   by sale_type order_date;
```

様式的 PDF 出力を作成します。 最初の SUM ステートメントの STYLE オプションは、変数 Price の合計を含むセルの背景色が薄青色に、フォントの色が白色に変更されるように指定します。2 番目の SUM ステートメントの STYLE オプションは、Quantity 変数の総計を含むセルの背景色が黄色、フォントの色が赤に変更されるように指定します。

```
   sum quantity price / style(TOTAL)=[backgroundcolor=light blue color=white];
   sum quantity price / style(GRANDTOTAL)=[backgroundcolor=green color=white];
   sumby sale_type;

   label sale_type='Sale Type' order_date='Sale Date';

   format price dollar10.2 cost dollar10.2;
   title 'Retail and Quantity Totals for Each Sale Type';
run;

ods pdf close;
```

出力:スタイルを使用した PDF

アウトプット 45.17 *レポート内の合計数を制限する:スタイルを使用した PostScript 出力*

Retail and Quantity Totals for Each Sale Type

Sale Type=Catalog Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Puerto Rico	99999999	1/5/12	14	$51.20	$12.10
Aruba	99999999	1/4/12	30	$123.70	$59.00
Bahamas	99999999	1/4/12	8	$113.40	$28.45
Bermuda	99999999	1/4/12	7	$41.00	$9.25

Sale Type=Catalog Sale Date=1/2/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
British Virgin Islands	99999999	1/5/12	11	$40.20	$20.20
Canada	99999999	1/5/12	100	$11.80	$5.00
Total			170	$381.30	

Sale Type=In Store Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Virgin Islands (U.S.)	99999999	1/4/12	25	$31.10	$15.65

Sale Type=In Store Sale Date=1/2/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Belize	120458	1/2/12	2	$146.40	$36.70
Cayman Islands	120454	1/2/12	20	$71.00	$32.30
Total			47	$248.50	

Sale Type=Internet Sale Date=1/1/12

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Antarctica	99999999	1/7/12	2	$92.60	$20.70
Grand Total			219	$722.40	

例 7: 多数の変数を使用し、レポートのレイアウトを制御する

要素: PROC PRINT ステートメントオプション
ROWS=
ID ステートメントオプション
STYLE

他の要素: ODS RTF ステートメント
SAS データセットオプション
OBS=

データセット: EMPDATA

ODS 出力先: LISTING, RTF

詳細

この例では、多くの変数を含むデータセットを印刷するための 2 つの方法を示します。1 つはデフォルトで、多くの変数がある場合に複数行を印刷します。もう 1 つは ROWS=オプションを使用して 1 行を印刷します。ROWS=オプションは、LISTING 出力先にのみ有効です。これらの 2 つのレポートのレイアウトに関する詳細説明については、オプション ROWS= (1350 ページ)と “ページのサイズが制限されている場合のページレイアウト” (1341 ページ)を参照してください。

これらのレポートには、異なるレイアウトの表示に役立つページサイズ 24、ページサイズ 64 が使用されます。

プログラム:LISTING レポートの作成

```
data empdata;
   input IdNumber $ 1-4 LastName $ 8-18 FirstName $ 19-28
      City $ 29-41 State $ 42-43
      Gender $ 45 JobCode $ 49-51 Salary 55-60 @63 Birth date7.
      @73 Hired date7. HomePhone $ 83-95;
   format birth hired date9.;
   datalines;
1919   Adams      Gerald    Stamford     CT M   TA2   34376   15SEP70   07JUN05   203/781-1255
1653   Alexander  Susan     Bridgeport   CT F   ME2   35108   18OCT72   12AUG98   203/675-7715

. . . more lines of data . . .

1407   Grant      Daniel    Mt. Vernon   NY M   PT1   68096   26MAR77   21MAR98   914/468-1616
1114   Green      Janice    New York     NY F   TA2   32928   21SEP77   30JUN06   212/588-1092
;

proc print data=empdata(obs=12);
   id idnumber;
   title 'Personnel Data';
run;

proc print data=empdata(obs=12) rows=page;
   id idnumber;
   title 'Personnel Data';
run;
```

プログラムの説明

EMPDATA データセットを作成します。 データセット EMPDATA には、会社の従業員に関する個人情報および業務上の情報が含まれます。DATA ステップでこのデータセットが作成されます。

```
data empdata;
   input IdNumber $ 1-4 LastName $ 8-18 FirstName $ 19-28
      City $ 29-41 State $ 42-43
      Gender $ 45 JobCode $ 49-51 Salary 55-60 @63 Birth date7.
      @73 Hired date7. HomePhone $ 83-95;
   format birth hired date9.;
   datalines;
1919   Adams      Gerald    Stamford     CT M   TA2   34376   15SEP70   07JUN05   203/781-1255
1653   Alexander  Susan     Bridgeport   CT F   ME2   35108   18OCT72   12AUG98   203/675-7715

. . . more lines of data . . .
```

```
1407   Grant      Daniel    Mt. Vernon   NY M   PT1   68096   26MAR77   21MAR98   914/468-1616
1114   Green      Janice    New York     NY F   TA2   32928   21SEP77   30JUN06   212/588-1092
;
```

データセットの最初の 12 オブザベーションのみ印刷します。 OBS=データセットオプションは、最初の 12 オブザベーションのみ使用して、レポートを作成します。(ここでスペースを節約するため)。) ID ステートメントは、オブザベーション番号ではなく、IdNumber のフォーマットされた値を含むオブザベーションを識別します。このレポートは、“出力:LISTING” (1413 ページ)で表示されます。

```
proc print data=empdata(obs=12);
   id idnumber;
   title 'Personnel Data';
run;
```

各ページに 1 変数行のみを含むレポートを印刷します。 ROWS=PAGE は、ページのオブザベーションごとに 1 変数行のみを印刷します。このレポートは、アウトプット 45.19 (1414 ページ)で表示されます。

```
proc print data=empdata(obs=12) rows=page;
   id idnumber;
   title 'Personnel Data';
run;
```

出力:LISTING

従来のプロシジャ出力では、このレポートの各ページに各オブザベーションのすべての変数の値が含まれています。HTML 出力では、このレポートは ROWS=PAGE を使用するレポートと同じです。

PROC PRINT は、全体の名前が列に合わない場合、大文字表記の変更時に列ヘッダーとして使用される変数名を自動的に分割します。LastName (列に適合)と FirstName (列に適合しない)の列ヘッダーを比較します。

アウトプット 45.18 *多数の変数を使用したレポートのデフォルトレイアウト:LISTING 出力*

```
                Personnel Data 1 Id First Number LastName Name City State Gender
1919 Adams Gerald Stamford CT M 1653 Alexander Susan Bridgeport CT F 1400 Apple
Troy New York NY M 1350 Arthur Barbara New York NY F 1401 Avery Jerry Paterson NJ
M 1499 Barefoot Joseph Princeton NJ M 1101 Baucom Walter New York NY M Id Job
Number Code Salary Birth Hired HomePhone 1919 TA2 34376 15SEP70 07JUN05
203/781-1255 1653 ME2 35108 18OCT72 12AUG98 203/675-7715 1400 ME1 29769 08NOV85
19OCT06 212/586-0808 1350 FA3 32886 03SEP63 01AUG00 718/383-1549 1401 TA3 38822
16DEC68 20NOV93 201/732-8787 1499 ME3 43025 29APR62 10JUN95 201/812-5665 1101 SCP
18723 09JUN80 04OCT98 212/586-8060
```

```
                Personnel Data 2 Id First Number LastName Name City State Gender
1333 Blair Justin Stamford CT M 1402 Blalock Ralph New York NY M 1479 Bostic
Marie New York NY F 1403 Bowden Earl Bridgeport CT M 1739 Boyce Jonathan New York
NY M Id Job Number Code Salary Birth Hired HomePhone 1333 PT2 88606 02APR79
13FEB03 203/781-1777 1402 TA2 32615 20JAN71 05DEC98 718/384-2849 1479 TA3 38785
25DEC66 08OCT03 718/384-8816 1403 ME1 28072 31JAN79 24DEC99 203/675-3434 1739 PT1
66517 28DEC82 30JAN00 212/587-1247
```

このレポートの各ページには、各オブザベーションの一部の変数の値のみ含まれています。ただし、各ページにはデフォルトのレポートよりも多くのオブザベーションの値が含まれています。

アウトプット 45.19 *ROWS=PAGE オプションによって生成されるレイアウト:LISTING 出力*

```
                        Personnel Data 3 Id First Number LastName Name City
State Gender 1919 Adams Gerald Stamford CT M 1653 Alexander Susan Bridgeport CT F
1400 Apple Troy New York NY M 1350 Arthur Barbara New York NY F 1401 Avery Jerry
Paterson NJ M 1499 Barefoot Joseph Princeton NJ M 1101 Baucom Walter New York NY
M 1333 Blair Justin Stamford CT M 1402 Blalock Ralph New York NY M 1479 Bostic
Marie New York NY F 1403 Bowden Earl Bridgeport CT M 1739 Boyce Jonathan New York
NY M
```

```
                        Personnel Data 4 Id Job Number Code Salary Birth Hired
HomePhone 1919 TA2 34376 15SEP70 07JUN05 203/781-1255 1653 ME2 35108 18OCT72
12AUG98 203/675-7715 1400 ME1 29769 08NOV85 19OCT06 212/586-0808 1350 FA3 32886
03SEP63 01AUG00 718/383-1549 1401 TA3 38822 16DEC68 20NOV93 201/732-8787 1499 ME3
43025 29APR62 10JUN95 201/812-5665 1101 SCP 18723 09JUN80 04OCT98 212/586-8060
1333 PT2 88606 02APR79 13FEB03 203/781-1777 1402 TA2 32615 20JAN71 05DEC98
718/384-2849 1479 TA3 38785 25DEC66 08OCT03 718/384-8816 1403 ME1 28072 31JAN79
24DEC99 203/675-3434 1739 PT1 66517 28DEC82 30JAN00 212/587-1247
```

プログラム:HTML レポートの作成

```
options pageno=1;

ods rtf file='your_file.rtf';

proc print data=empdata(obs=12);
   id idnumber;
   title 'Personnel Data';
run;

ods rtf close;
```

プログラムの説明

RTF 出力には、すべてのデータが 1 行に表示されます。ROWS=オプションは、LISTING 出力先にのみ有効です。

```
options pageno=1;
```

Microsoft Word に対する出力を作成し、出力を格納するファイルを指定します。 ODS RTF ステートメントは RTF 出力先を開いて、Microsoft Word に対しフォーマットされた出力を作成します。FILE=引数は、RTF 出力を含む外部ファイルを指定します。

```
ods rtf file='your_file.rtf';

proc print data=empdata(obs=12);
   id idnumber;
   title 'Personnel Data';
run;
```

RTF 出力先を閉じます。 ODS RTF CLOSE ステートメントは、RTF 出力先を閉じます。

```
ods rtf close;
```

出力:RTF

アウトプット 45.20 *多数の変数を使用したレポートのレイアウト:RTF 出力*

1

Personnel Data

IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary	Birth	Hired	HomePhone
1919	Adams	Gerald	Stamford	CT	M	TA2	34376	15SEP70	07JUN05	203/781-1255
1653	Alexander	Susan	Bridgeport	CT	F	ME2	35108	18OCT72	12AUG98	203/675-7715
1400	Apple	Troy	New York	NY	M	ME1	29769	08NOV85	19OCT06	212/586-0808
1350	Arthur	Barbara	New York	NY	F	FA3	32886	03SEP63	01AUG00	718/383-1549
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822	16DEC68	20NOV93	201/732-8787
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025	29APR62	10JUN95	201/812-5665
1101	Baucom	Walter	New York	NY	M	SCP	18723	09JUN80	04OCT98	212/586-8060
1333	Blair	Justin	Stamford	CT	M	PT2	88606	02APR79	13FEB03	203/781-1777
1402	Blalock	Ralph	New York	NY	M	TA2	32615	20JAN71	05DEC98	718/384-2849
1479	Bostic	Marie	New York	NY	F	TA3	38785	25DEC66	08OCT03	718/384-8816
1403	Bowden	Earl	Bridgeport	CT	M	ME1	28072	31JAN79	24DEC99	203/675-3434
1739	Boyce	Jonathan	New York	NY	M	PT1	66517	28DEC82	30JAN00	212/587-1247

例 8: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成

要素: PROC PRINT ステートメントオプション
- SUMLABEL
- GRANDTOTAL_LABEL

BY ステートメント

ID ステートメント

SUM statement

VAR ステートメント

他の要素: SORT プロシジャ

データセット: EMPDATA

ODS 出力先: LISTING, HTML

詳細

このカスタマイズされたレポートでは、次のタスクについて説明します。

- レポートに含める変数と、表示される順序を選択する
- レポートに含めるオブザベーションを選択する
- 選択したオブザベーションを JobCode 別にグループ化する
- 給与をジョブコードごと、およびすべてのジョブコードに対して合計する
- 数値データをカンマとドル記号とともに表示する

プログラム:LISTING レポートの作成

```
options nodate pageno=1 linesize=64 pagesize=60;
proc sort data=empdata out=tempemp;
   by jobcode gender;
run;

proc print data=tempemp split='*' sumlabel='Total' grandtotal_label='Grand Total';

   id jobcode;
   by jobcode;
   var gender salary;

   sum salary;

   label jobcode='Job Code*========'
         gender='Gender*======'
         salary='Annual Salary*=============';

   format salary dollar11.2;
   where jobcode contains 'FA' or jobcode contains 'ME';
   title 'Salary Expenses';
run;
```

プログラムの説明

一時データセットを作成して、並べ替えます。 PROC SORT は、オブザベーションが JobCode と Gender 別に並べ替えられる一時データセットを作成します。

```
options nodate pageno=1 linesize=64 pagesize=60;
proc sort data=empdata out=tempemp;
   by jobcode gender;
run;
```

列ヘッダーに新しい行を開始する文字を識別します。 SPLIT=は、列ヘッダーで新しい行を開始する文字としてアスタリスクを識別します。

```
proc print data=tempemp split='*' sumlabel='Total' grandtotal_label='Grand Total';
```

レポートに含める変数を指定します。 VAR ステートメントと ID ステートメントを使用して、レポートに含める変数を選択します。ID ステートメントと BY ステートメントは、特別な出力形式を生成します。

```
   id jobcode;
   by jobcode;
   var gender salary;
```

BY グループごとの合計値を計算します。 SUM ステートメントは BY グループごと、およびレポート全体の Salary の値を総計します。

```
   sum salary;
```

ラベルを適切な変数に割り当てます。 LABEL ステートメントは、ラベルを ROC PRINT ステップの期間の各変数と連付けます。PROC PRINT ステートメントで SPLIT=を使用すると、プロシジャは列ヘッダーのラベルを使用します。

```
   label jobcode='Job Code*========'
         gender='Gender*======'
         salary='Annual Salary*=============';
```

フォーマットされた列を作成します。 FORMAT ステートメントは、このレポートの Salary に出力形式を割り当てます。WHERE ステートメントは、文字'FA'または'ME'を含むジョブコードに対するオブザベーションのみレポートに選択します。TITLE ステートメントはレポートタイトルを指定します。

```
      format salary dollar11.2;
      where jobcode contains 'FA' or jobcode contains 'ME';
      title 'Salary Expenses';
   run;
```

出力:LISTING

ID ステートメントと BY ステートメントを使用して、このレイアウトを生成します。ID 変数は、BY グループごとに一度だけリストされます。BY 行は非表示にされます。代わりに、ID 変数、JobCode の値が各 BY グループを識別します。

アウトプット 45.21 *BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成 LISTING 出力*

```
           Salary Expenses 1 Job Code Gender Annual Salary ======== ======
============= FA1 F $23,177.00 F $22,454.00 M $22,268.00 -----------
------------- Total $67,899.00 FA2 F $28,888.00 F $27,787.00 M $28,572.00
----------- ------------- Total $85,247.00 FA3 F $32,886.00 F $33,419.00 M
$32,217.00 ----------- ------------- Total $98,522.00 ME1 M $29,769.00 M
$28,072.00 M $28,619.00 ----------- ------------- Total $86,460.00 ME2 F
$35,108.00 F $34,929.00 M $35,345.00 M $36,925.00 M $35,090.00 M $35,185.00
----------- ------------- Total $212,582.00 ME3 M $43,025.00 ===========
============= Grand Total $593,735.00
```

プログラム:HTML レポートの作成

```
proc sort data=empdata out=tempemp;
   by jobcode gender;
run;

ods html file='your_file.html';

proc print data=tempemp (obs=10) sumlabel='Total' grandtotal_lable='Grand Total';

 id jobcode;
   by jobcode;
   var gender salary;

   sum salary;

   label jobcode='Job Code'
         gender='Gender'
         salary='Annual Salary';

   format salary dollar11.2;
   where jobcode contains 'FA' or jobcode contains 'ME';
   title 'Salary Expenses';
run;
```

プログラムの説明

```
proc sort data=empdata out=tempemp;
   by jobcode gender;
run;
```

HTML 出力を生成し、出力を格納するファイルを指定します。 HTML 出力先は、デフォルトの ODS 出力先です。ODS HTML ステートメント FILE=オプションは、HTML 出力を含む外部ファイルを指定します。

```
ods html file='your_file.html';
```

プロシジャのオプションを定義してください。 (obs=10)データセットのオプションは、処理するオブザベーション数を指定します。SUMLABEL オプションは、各 BY グループの要約行でラベル'Total'を使用することを示します。GRANDTOTAL_LABEL オプションは、レポート内のすべての BY グループの後、総計行でラベル'Grand Total'を使用することを示します。

```
proc print data=tempemp (obs=10) sumlabel='Total' grandtotal_lable='Grand Total';

 id jobcode;
   by jobcode;
   var gender salary;

   sum salary;

   label jobcode='Job Code'
         gender='Gender'
         salary='Annual Salary';

   format salary dollar11.2;
   where jobcode contains 'FA' or jobcode contains 'ME';
   title 'Salary Expenses';
run;
```

出力:出力:HTML

アウトプット 45.22 *BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成:デフォルト HTML 出力*

Salary Expenses

JobCode	Gender	Salary
FA1	F	$23,177.00
	F	$22,454.00
	M	$22,268.00
Total		$67,899.00

JobCode	Gender	Salary
FA2	F	$28,888.00
	F	$27,787.00
	M	$28,572.00
Total		$85,247.00

JobCode	Gender	Salary
FA3	F	$32,886.00
	F	$33,419.00
	M	$32,217.00
Total		$98,522.00

JobCode	Gender	Salary
ME1	M	$29,769.00
Grand Total		$281,437.00

プログラム:STYLE オプションを使用した HTML レポートの作成

```
proc sort data=empdata out=tempemp;
   by jobcode gender;
run;

proc print data=tempemp (obs=10)sumlabel='Total' grandtotal_label='Grand Total'
   style(HEADER)={fontstyle=italic}
   style(DATA)={backgroundcolor=blue foreground=white};

 id jobcode;
   by jobcode;
   var gender salary;
```

```
   sum salary  / style(total)={color=red};

   label jobcode='Job Code'
         gender='Gender'
         salary='Annual Salary';

   format salary dollar11.2;
   where jobcode contains 'FA' or jobcode contains 'ME';
   title 'Expenses Incurred for';
   title2 'Salaries for Flight Attendants and Mechanics';
run;
```

プログラムの説明

```
proc sort data=empdata out=tempemp;
   by jobcode gender;
run;
```

スタイルを指定した HTML 出力を作成します。 最初の STYLE オプションは、ヘッダーのフォントが斜体に変更されるように指定します。2 番目の STYLE オプションは、データを含むセルの背景が青色に、これらのセルの前景が白色に変更されるように指定します。SUMLABEL オプションは要約行で、GRANDTOTAL_LABEL オプションは総計行で、それぞれ変数名のかわりにラベルを使用します。

```
proc print data=tempemp (obs=10)sumlabel='Total' grandtotal_label='Grand Total'
   style(HEADER)={fontstyle=italic}
   style(DATA)={backgroundcolor=blue foreground=white};

 id jobcode;
   by jobcode;
   var gender salary;
```

赤で書き込まれる合計値を作成します。 STYLE オプションは、合計を含むセルの前景色が赤に変更されるように指定します。

```
   sum salary  / style(total)={color=red};

   label jobcode='Job Code'
         gender='Gender'
         salary='Annual Salary';

   format salary dollar11.2;
   where jobcode contains 'FA' or jobcode contains 'ME';
   title 'Expenses Incurred for';
   title2 'Salaries for Flight Attendants and Mechanics';
run;
```

出力:スタイルを使用した HTML

アウトプット 45.23 *BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成:スタイルを使用した HTML 出力*

Expenses Incurred for
Salaries for Flight Attendants and Mechanics

Job Code	Gender	Annual Salary
FA1	F	$23,177.00
	F	$22,454.00
	M	$22,268.00
Total		$67,899.00

Job Code	Gender	Annual Salary
FA2	F	$28,888.00
	F	$27,787.00
	M	$28,572.00
Total		$85,247.00

Job Code	Gender	Annual Salary
FA3	F	$32,886.00
	F	$33,419.00
	M	$32,217.00
Total		$98,522.00

Job Code	Gender	Annual Salary
ME1	M	$29,769.00
Grand Total		$281,437.00

例 9: SAS ライブラリのすべてのデータセットを印刷する

要素: マクロ機能
DATASETS プロシジャ
PRINT プロシジャ

データセット: PROCLIB.DELAY と
PROCLIB.INTERNAT(生データと DATA ステップ付録)

ODS 出力先: HTML

詳細

この例では、SAS ライブラリのすべてのデータセットを印刷します。同じプログラミング論理をどのプロシジャとも使用できます。例の最後辺りに PROC PRINT ステップと実行するプロシジャステップを置き換えるだけです。例では、マクロ言語を使用します。マクロ言語の詳細については、*SAS マクロ言語: リファレンス*を参照します。

プログラム:ライブラリのすべてのデータセットを印刷する

```
libname printlib 'SAS-data-library';
libname proclib 'SAS-data-library';
options nodate pageno=1;

proc datasets library=proclib memtype=data nolist;
   copy out=printlib;
      select delay internat;
run;

%macro printall(libname,worklib=work);

   %local num i;

   proc datasets library=&libname memtype=data nodetails;
      contents out=&worklib..temp1(keep=memname) data=_all_ noprint;
   run;

   data _null_;
      set &worklib..temp1 end=final;
      by memname notsorted;
      if last.memname;
      n+1;
      call symput('ds'||left(put(n,8.)),trim(memname));

      if final then call symput('num',put(n,8.));

   run;

   %do i=1 %to &num;
      proc print data=&libname..&&ds&i noobs;
         title "Data Set &libname..&&ds&i";
      run;
   %end;
%mend printall;

%printall(printlib)
```

プログラムの説明

```
libname printlib 'SAS-data-library';
libname proclib 'SAS-data-library';
options nodate pageno=1;
```

必要なデータセットを WORK ライブラリから永久ライブラリにコピーします。 PROC DATASETS は 2 つのデータセットを WORK ライブラリから PRINTLIB ライブラリにコピーし、例に利用可能なデータセット数を制限します。

```
proc datasets library=proclib memtype=data nolist;
   copy out=printlib;
      select delay internat;
```

```
run;
```

マクロを作成して、パラメータを指定します。 %MACRO ステートメントは、マクロ PRINTALL を作成します。マクロの呼び出し時には、1 つまたは 2 つのパラメータを渡すことができます。最初のパラメータは、データセットを印刷するライブラリの名前です。2 番目のパラメータは、マクロによって使用されるライブラリです。このパラメータを指定しない場合、WORK ライブラリがデフォルトとなります。

```
%macro printall(libname,worklib=work);
```

ローカルマクロ変数を作成します。 %LOCAL ステートメントは、ループで使用する 2 つのローカルマクロ変数、NUM と I を作成します。

```
%local num i;
```

出力データセットを生成します。 この PROC DATASETS ステップは、マクロの起動時にパラメータとして指定するライブラリを読み込みます。CONTENTS ステートメントは、WORKLIB に TEMP1 という出力データセットを生成します。このデータセットには、ライブラリ LIBNAME の各データセットの変数ごとに 1 つのオブザベーションが含まれています。デフォルトでは、オブザベーションにはそれぞれ、変数が含まれるデータセット名および変数に関する情報が含まれます。ただし、KEEP=データセットオプションは、データセット名のみ TEMP1 に書き込みます。

```
proc datasets library=&libname memtype=data nodetails;
   contents out=&worklib..temp1(keep=memname) data=_all_ noprint;
run;
```

データセットの一意の値を指定し、それぞれにマクロ変数を割り当て、マクロ変数に DATA ステップ情報を割り当てます。 この DATA ステップは、最後のデータセット名を読み込むたびに N の値を増分します。(IF LAST.MEMNAME が True の場合) CALL SYMPUT ステートメントは N の現在の値を使用して、データセット TEMP1 の一意の値 MEMNAME に対してそれぞれマクロ変数を作成します。TRIM 関数は、後に続く PROC PRINT ステップの TITLE ステートメントの余分なブランクを削除します。

```
data _null_;
   set &worklib..temp1 end=final;
   by memname notsorted;
   if last.memname;
   n+1;
   call symput('ds'||left(put(n,8.)),trim(memname));
```

DATA ステップでオブザベーション数を決定します。 データセットの最後のオブザベーションの読み込み時に(FINAL が Ttrue のとき)、DATA ステップは N の値をマクロ変数 NUM に割り当てます。プログラムのこの時点で、N の値はデータセットのオブザベーション数です。

```
   if final then call symput('num',put(n,8.));
```

DATA ステップを実行します。 RUN ステートメントは重要です。これにより DATA ステップが実行されるため、マクロ変数の作成が実行されます。このマクロ変数は、それらを使用する%DO ループより前に CALL SYMPUT ステートメントで使用されます。

```
run;
```

データセットを印刷し、マクロを終了します。 %DO ループは、データセットごとに PROC PRINT ステップを発行します。%MEND ステートメントはマクロを終了します。

```
%do i=1 %to &num;
   proc print data=&libname..&&ds&i noobs;
```

```
            title "Data Set &libname..&&ds&i";
          run;
       %end;
    %mend printall;
```

PRINTLIB ライブラリのすべてのデータセットを印刷します。PRINTALL マクロを起動することによって、ライブラリ PRINTLIB のすべてのデータセットが印刷されます。

```
%printall(printlib)
```

出力:出力:HTML

アウトプット 45.24 *データセット PRINTLIB.DELAY*

Data Set printlib.DELAY

flight	date	orig	dest	delaycat	destype	delay
114	01MAR12	LGA	LAX	1-10 Minutes	Domestic	8
202	01MAR12	LGA	ORD	No Delay	Domestic	-5
219	01MAR12	LGA	LON	11+ Minutes	International	18
622	01MAR12	LGA	FRA	No Delay	International	-5
132	01MAR12	LGA	YYZ	11+ Minutes	International	14
271	01MAR12	LGA	PAR	1-10 Minutes	International	5
302	01MAR12	LGA	WAS	No Delay	Domestic	-2
114	02MAR12	LGA	LAX	No Delay	Domestic	0
202	02MAR12	LGA	ORD	1-10 Minutes	Domestic	5
219	02MAR12	LGA	LON	11+ Minutes	International	18
622	02MAR12	LGA	FRA	No Delay	International	0
132	02MAR12	LGA	YYZ	1-10 Minutes	International	5
271	02MAR12	LGA	PAR	1-10 Minutes	International	4
302	02MAR12	LGA	WAS	No Delay	Domestic	0
114	03MAR12	LGA	LAX	No Delay	Domestic	-1

アウトプット 45.25 データセット PRINTLIB.INTERNAT

Data Set printlib.INTERNAT

flight	date	dest	boarded
219	01MAR12	LON	198
622	01MAR12	FRA	207
132	01MAR12	YYZ	115
271	01MAR12	PAR	138
219	02MAR12	LON	147
622	02MAR12	FRA	176
132	02MAR12	YYZ	106
271	02MAR12	PAR	172
219	03MAR12	LON	197
622	03MAR12	FRA	180
132	03MAR12	YYZ	75
271	03MAR12	PAR	147
219	04MAR12	LON	232
622	04MAR12	FRA	137
132	04MAR12	YYZ	117

46 章

PRINTTO プロシジャ

概要: PRINTTO プロシジャ

PRINTTO プロシジャでは、SAS プロシジャ出力と SAS ログの出力先(ODS 出力先以外)が定義されます。デフォルトでは、SAS プロシジャ出力と SAS ログは、操作方法に応じたデフォルトプロシジャ出力ファイルとデフォルト SAS ログファイルに送られます。PRINTTO プロシジャでは、ODS 出力先は定義されません。SAS ログとプロシジャ出力のデフォルト出力先については、次の表を参照してください。

SAS ログやプロシジャ出力は、外部ファイルや SAS カタログエントリに保存できます。ファイルやカタログエントリに SAS 出力を書き込むには、ODS LISTING 出力先を開く必要があります。追加プログラミングによって、同じジョブ内で SAS 出力を入力データとして使用できます。

表 46.1 *SAS ログとプロシジャ出力のデフォルト出力先*

SAS 実行方法	SAS ログ出力先	プロシジャ出力先
ウィンドウ環境	**LOG** ウィンドウ	**結果ビューア**ウィンドウ
対話型ラインモード	ディスプレイモニタ(ステートメントの入力時)	ディスプレイモニタ(各ステップの実行時)

SAS 実行方法	SAS ログ出力先	プロシジャ出力先
非対話型モードまたはバッチモード	ホスト OS に依存	動作環境に依存

動作環境の情報
OS または環境固有の情報や例については、使用する動作環境のドキュメントを参照してください。

構文: PRINTTO プロシジャ

参照項目: “PRINTTO Procedure: UNIX” (*SAS Companion for UNIX Environments*)
“PRINTTO Procedure: Windows” (*SAS Companion for Windows*)
“PRINTTO Procedure: z/OS” (*SAS Companion for z/OS*)

PROC PRINTTO <*option(s)*>;

ステートメント	タスク	例
“PROC PRINTTO ステートメント”	SAS プロシジャ出力と SAS ログの出力先(ODS 出力先以外)が定義されます。	Ex. 1, Ex. 2, Ex. 3, Ex. 4

PROC PRINTTO ステートメント

SAS プロシジャ出力と SAS ログの出力先(ODS 出力先以外)が定義されます。

制限事項: SAS ログとプロシジャ出力をプリンタに直接送るには、FILENAME ステートメントを PROC PRINTTO ステートメントとあわせて使用する必要があります。“SAS ログまたはプロシジャの出力を直接にプリンタに送る” (1433 ページ) および “例 4: 出力先をプリンタに指定する” (1445 ページ)を参照してください。

PRINTTO プロシジャでは、ODS 出力先は定義されません。

SAS がオブジェクトサーバーモードで起動されている場合、PRINTTO プロシジャはログメッセージを ALTLOG=システムオプションで指定したログに送りません。

ヒント: SAS ログとプロシジャ出力の出力先をデフォルトにリセットするには、PROC PRINTTO ステートメントをオプションなしで使用します。

SAS ログとプロシジャ出力を同じファイルに送るには、LOG=と PRINT=の両オプションで同じファイルを指定します。

例: “例 1: 出力先を外部ファイルに指定する” (1434 ページ)
“例 2: 出力先を SAS カタログエントリに指定する” (1437 ページ)
“例 3: プロシジャ出力を入力ファイルとして使用する” (1441 ページ)
“例 4: 出力先をプリンタに指定する” (1445 ページ)

構文

PROC PRINTTO <*option(s)*>;

オプション引数の要約

LABEL='*description*'
: SAS カタログエントリに格納された SAS ログまたはプロシジャ出力の説明が提供されます。

LOG=LOG | *file-specification* | *SAS-catalog-entry*
: SAS ログを永久外部ファイルまたは SAS カタログエントリに送ります。

NEW
: ファイルを追加せずに置き換えます。

PRINT= PRINT | *file-specification* | *SAS-catalog-entry*
: プロシジャ出力を、永久外部ファイルか SAS カタログエントリかプリンタに送ります。

UNIT=*nn*
: ファイル参照名で識別されるファイルに出力を送ります。

引数なし

オプションを指定しなかった場合、PROC PRINTTO ステートメントでは次が実行されます。

- PROC PRINTTO ステートメントで開いたファイルをすべて閉じます。
- SAS ログと SAS プロシジャ出力の両方にデフォルト出力先を示します。
- LISTING 出力先を閉じます。

操作: 適切なファイルを閉じて、SAS ログまたはプロシジャ出力のみをデフォルト出力先に戻すには、LOG=LOG または PRINT=PRINT を使用します。

例:

“例 1: 出力先を外部ファイルに指定する” (1434 ページ)
“例 2: 出力先を SAS カタログエントリに指定する” (1437 ページ)

オプション引数

LABEL='*description*'
SAS ログまたはプロシジャ出力を含むカタログエントリの説明が提供されます。

範囲 1–256 文字

操作 カタログエントリを LOG=または PRINT=オプションの値として指定する場合に限り、LABEL=オプションを使用します。

例 “例 2: 出力先を SAS カタログエントリに指定する” (1437 ページ)

LOG=LOG | *file-specification* | *SAS-catalog-entry*
SAS ログを 3 つのうちいずれかの場所に送ります。

LOG
SAS ログをデフォルト出力先に送ります。

file-specification
SAS ログを外部ファイルに送ります。*file-specification* として次のいずれかを指定できます。

'external-file'
引用符で囲んで指定する外部ファイル名。

制限事項 *external-file* は 1024 文字以内にしてください。

log-filename
引用符なしの英数字のテキスト文字列です。SAS で、*log-filename*.log をログファイル名として使用するログが作成されます。

動作環境 *log-filename* の詳細については、使用する動作環境のドキュメントを参照してください。

fileref
事前に外部ファイルに割り当てられたファイル参照名。

SAS-catalog-entry
SAS ログを SAS カタログエントリに送ります。デフォルトでは、*libref* が SASUSER、*catalog* が PROFILE、*type* が LOG です。次のいずれかの方法で *SAS-catalog-entry* を示します。

libref.catalog.entry<.LOG>
指定された SAS ライブラリおよび SAS カタログに格納された SAS カタログエントリ。

catalog.entry<.LOG>
デフォルト SAS ライブラリ SASUSER の指定 SAS カタログに格納された SAS カタログエントリ。

entry.LOG
デフォルトの SAS ライブラリおよび SAS カタログに格納された SAS カタログエントリ SASUSER.PROFILE.

fileref
SAS カタログエントリに事前に割り当てられたファイル参照名。SAS オンラインドキュメントの"FILENAME, CATALOG Access Method"を検索してください。

デフォルト LOG

操作 SAS ログとプロシジャ出力を、同時に同じカタログエントリに送ることはできません。

NEW オプションで、ファイルの既存コンテンツが新しいログに置き換えられます。そうでなければ、新しいログはファイルに追加されます。

SAS ログとプロシジャ出力を同じファイルに送るには、LOG=と PRINT=の両オプションで同じファイルを指定します。

ログを SAS カタログエントリに送る際、LABEL オプションを使用すると、カタログディレクトリのエントリの説明を提供できます。

ログがデフォルトログファイル以外のファイルに送られ、複数のソースからプログラムがサブミットされると、実時間と CPU 時間を含む最後の SAS システムメッセージがデフォルト SAS ログに書き込まれます。

ヒント 外部ファイルまたはカタログエントリにログを送った後で、SAS ログをデフォルト出力先に送り直すには、LOG を指定します。

SAS ログを送る際は、PROC PRINTTO ステートメントに RUN ステートメントを含めます。RUN ステートメントを省略した場合、その後に続く DATA または PROC ステップの最初の行は新しいファイルに送られません。(このような事象が発生するのは、ステップの境界を越えるまではステートメントが実行されないためです。

パスワードを含むマクロを作成して、パスワードを SAS ログに表示したくない場合、LOG=*file-specification* オプションを使用してログを外部ファイルにリダイレクトします。

LOG=を指定すると、SAS では&SYSPRINTTOLOG 自動マクロ変数に SAS ログファイルのパスが保存されます。このマクロ変数を使用して前の SAS ログファイルの場所を復元できます。詳細については、“前の SAS ログまたは LISTING 出力ファイルの場所の復元” (1434 ページ)を参照してください。

例 “例 1: 出力先を外部ファイルに指定する” (1434 ページ)

“例 2: 出力先を SAS カタログエントリに指定する” (1437 ページ)

“例 3: プロシジャ出力を入力ファイルとして使用する” (1441 ページ)

NEW
ファイルに存在する情報をすべてクリアし、ファイルが SAS ログまたはプロシジャ出力を受信する準備をします。

デフォルト NEW を省略した場合、新しい情報は既存のファイルに追加されます。

操作 LOG=と PRINT=の両方を指定した場合、両方に NEW が適用されます。

例 “例 1: 出力先を外部ファイルに指定する” (1434 ページ)

“例 2: 出力先を SAS カタログエントリに指定する” (1437 ページ)

“例 3: プロシジャ出力を入力ファイルとして使用する” (1441 ページ)

PRINT= PRINT | *file-specification* | *SAS-catalog-entry*
プロシジャ出力を 3 つのうちいずれかの場所に送ります。

PRINT
プロシジャ出力をデフォルト出力先に送ります。

ヒント 外部ファイルまたはカタログエントリに送った後で、後続のプロシジャ出力をデフォルト出力先に送るには、PRINT を指定します。

file-specification
プロシジャ出力を外部ファイルに送ります。*file-specification* として次のいずれかを指定できます。

'*external-file*'
引用符で囲んで指定する外部ファイル名。

制限事項 *external-file* は 1024 文字以内にしてください。

print-filename
引用符なしの英数字のテキスト文字列です。SAS で、*print-filename* を印刷ファイル名として使用する印刷ファイルが作成されます。

動作環境の情報
print-filename の使用法の詳細については、使用する動作環境のドキュメントを参照してください。

fileref
事前に外部ファイルに割り当てられたファイル参照名。

動作環境の情報
PRINT オプションの *file-specification* の追加情報については、使用する動作環境のドキュメントを参照してください。

SAS-catalog-entry
プロシジャ出力を SAS カタログエントリに送ります。デフォルトでは、*libref* が SASUSER、*catalog* が PROFILE、*type* が OUTPUT です。次のいずれかの方法で *SAS-catalog-entry* を示します。

***libref.catalog.entry*<.OUTPUT>**
指定された SAS ライブラリおよび SAS カタログに格納された SAS カタログエントリ。

***catalog.entry*<.OUTPUT>**
デフォルト SAS ライブラリ SASUSER の指定 SAS カタログに格納された SAS カタログエントリ。

***entry*.OUTPUT**
デフォルトの SAS ライブラリおよび SAS カタログに格納された SAS カタログエントリ SASUSER.PROFILE.

fileref
SAS カタログエントリに事前に割り当てられたファイル参照名。SAS オンラインドキュメントの"FILENAME, CATALOG Access Method"を検索してください。

別名 FILE=, NAME=

デフォルト PRINT

操作 PRINT または FILE=または NAME=を指定して、LISTING 出力先が開いていないときは、PRINTTO プロシジャ出力の送信中に、このプロシジャにより LISTING 出力先が開きます。PRINT または FILE=または NAME=が指定される前に LISTING 出力先が開いた場合は、出力が LISTING 出力先に送られた後も、LISTING 出力先が開いたままになります。

プロシジャ出力と SAS ログを、同時に同じカタログエントリに送ることはできません。

NEW オプションで、ファイルの既存コンテンツが新しいプロシジャ出力に置き換えられます。NEW を省略した場合、新しい出力は既存のファイルに追加されます。

SAS ログとプロシジャ出力を同じファイルに送るには、LOG=と PRINT=の両オプションで同じファイルを指定します。

プロシジャ出力を SAS カタログエントリに送る際、LABEL オプションを使用すると、カタログディレクトリのエントリの説明を提供できます。

ヒント PRINT=を指定すると、SAS では&SYSPRINTTOLIST 自動マクロ変数に LISTING 出力ファイルのパスが保存されます。このマクロ変数を使用して前の LISTING 出力ファイルの場所を復元できます。詳細については、“前の SAS ログまたは LISTING 出力ファイルの場所の復元” (1434 ページ)を参照してください。

例 “例 3: プロシジャ出力を入力ファイルとして使用する” (1441 ページ)

UNIT=*nn*
ファイル参照名 FT*nn*F001 で識別されるファイルに出力を送ります。このとき、*nn* は 1 から 99 までの整数です。

範囲 1–99、整数のみ

ヒント このファイル参照名は各自で定義できます。ただし、一部の OS では、このフォームで特定のファイル参照名が事前定義されます。

UNIT=を指定すると、SAS では&SYSPRINTTOLIST 自動マクロ変数に LISTING 出力ファイルのパスが保存されます。このマクロ変数を使用して前の LISTING 出力ファイルの場所を復元できます。詳細については、“前の SAS ログまたは LISTING 出力ファイルの場所の復元” (1434 ページ)を参照してください。

SAS システムオプションを使用してページ番号を設定する

NUMBER SAS システムオプションが有効な場合、現在のジョブまたはセッションのすべての出力に対する単一のページ採番が行われます。NONUMBER が有効な場合、出力ページは採番されません。

OPTIONS ステートメントの PAGENO=を使用すると、現在作成している出力の開始ページ番号を指定できます。

SAS ログまたはプロシジャの出力を直接にプリンタに送る

SAS ログまたはプロシジャ出力を直接プリンタに送るには、FILENAME ステートメントでファイル参照名とプリンタ名を関連付けて、そのファイル参照名を LOG=または PRINT=オプションで使用します。例については、“例 4: 出力先をプリンタに指定する” (1445 ページ)を参照してください。

詳細については、“FILENAME Statement” (*SAS Statements: Reference*)を参照してください。

動作環境の情報
プリンタ名の例については、使用する OS のドキュメントを参照してください。

PRINTTO プロシジャでは、COLORPRINTING システムオプションはサポートされません。SAS ログまたはプロシジャ出力をカラープリンタに送っても、出力はカラー印刷されません。

PROC PRINTTO および LISTING 出力先

LISTING 出力先を開いて、プロシジャ出力を外部ファイルに送る必要があります。PRINT=オプションの指定時に、LISTING 出力先が開いていない場合は SAS により開かれます。プロシジャ出力が外部ファイルに送られた後は、SAS により LISTING 出力先が閉じられます。**proc printto;**を使用して出力の送信先をリセットするときに LISTING 出力先が開いている場合も、閉じられます。LOG=オプションを指定すると、LISTING 出力先は SAS により開かれません。

PROC PRINTTO PRINT=オプションの実行前に LISTING 出力先が開かれる場合は、出力が外部ファイルに送られた後、LISTING 出力先が開いたままになります。

前の SAS ログまたは LISTING 出力ファイルの場所の復元

PROC PRINTTO ステートメントで LOG=、PRINT=または UNIT=オプションを指定すると、SAS では自動マクロ変数に該当するファイルの場所が次のように保存されます。

SYSPRINTTOLOG には、PRINTTO プロシジャによるリダイレクション前の SAS ログファイルの場所のパスが保存されます

SYSPRINTTOLIST には、PRINTTO プロシジャによるリダイレクション前の LISTING 出力ファイルの場所のパスが保存されます。

前のファイルの場所を復元するには、LOG=、PRINT=または UNIT=オプションの値として適切な自動マクロ変数を指定します。次に、一部の例を示します。

```
/* Restore the previous log and the listing file locations. */
proc printto log=&sysprinttolog print=&sysprinttolist;
   run;

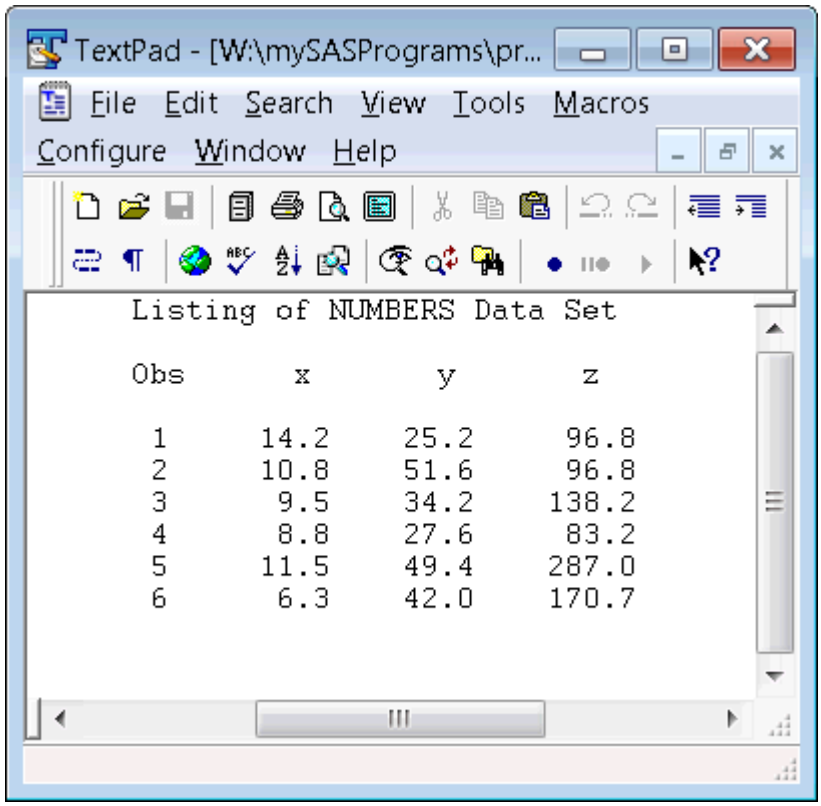
/* Restore the previous listing file location. */
proc printto unit=&sysprinttolist;
   run;
```

例: PRINTTO プロシジャ

例 1: 出力先を外部ファイルに指定する

要素: オプションなしの PRINTTO ステートメント

PRINTTO ステートメントオプション
LOG=
NEW
PRINT=

詳細

この例では、PROC PRINTTO を使用してログとプロシジャ出力を外部ファイルに送ってから、出力先を両方ともデフォルトにリセットします。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60 source;

proc printto log='log-file';
   run;

data numbers;
   input x y z;
   datalines;
 14.2   25.2   96.8
 10.8   51.6   96.8
  9.5   34.2  138.2
  8.8   27.6   83.2
 11.5   49.4  287.0
  6.3   42.0  170.7
;

proc printto print='output-file'
new;
run;

proc print data=numbers;
   title 'Listing of NUMBERS Data Set';
run;

proc printto;
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。SOURCE オプションはソースコード行を SAS ログのデフォルト出力先に書き込みます。

```
options nodate pageno=1 linesize=80 pagesize=60 source;
```

SAS ログを外部ファイルに送ります。 PROC PRINTTO では、LOG=オプションによって SAS ログが外部ファイルに送られます。デフォルトでは、このログは *log-file* の現在のコンテンツに追加されます。

```
proc printto log='log-file';
   run;
```

NUMBERS データセットを作成します。 DATA ステップでは、リスト入力によって NUMBERS データセットが作成されます。

```
data numbers;
   input x y z;
   datalines;
 14.2   25.2   96.8
 10.8   51.6   96.8
  9.5   34.2  138.2
  8.8   27.6   83.2
 11.5   49.4  287.0
  6.3   42.0  170.7
;
```

プロシジャ出力を外部ファイルに送ります。 PROC PRINTTO で、出力が外部ファイルに送られます。SAS 出力を外部ファイルに送るために LISTING 出力先を開く必要があるため、LISTING 出力先が開かれていなければ SAS により開かれます。ODS LISTING ステートメントを含める必要はありません。NEW を指定しているため、*output-file* に出力が書き込まれると、ファイルの現在のコンテンツが上書きされます。PROC PRINTTO 出力を処理するために SAS により LISTING 出力先が開かれると、出力が *output-file* に書き込まれた後で LITING 出力先は SAS により閉じられます。

```
proc printto print='output-file'
new;
run;
```

NUMBERS データセットを印刷します。 PROC PRINT 出力が、指定した外部ファイルに書き込まれます。

```
proc print data=numbers;
   title 'Listing of NUMBERS Data Set';
run;
```

SAS ログとプロシジャ出力の出力先をデフォルトにリセットします。 PROC PRINTTO で、後続のログとプロシジャ出力がデフォルト出力先に送られ、現在のファイルが両方とも閉じられます。

```
proc printto;
run;
```

ログ

ログ 46.1 *デフォルト出力先に送られたログの一部*

```
01   options nodate pageno=1 linesize=80 pagesize=60 source; 02 03   proc
printto log='c:\em\log1.log'; 04   run; NOTE:PROCEDURE PRINTTO used (Total
process time): real time           0.01 seconds cpu time            0.00 seconds
```

ログ 46.2 外部ファイルに送られたログの一部

```
NOTE:PROCEDURE PRINTTO used (Total process time): real time           0.00
seconds cpu time            0.00 seconds 5 6    data numbers; 7       input x y
z; 8       datalines; NOTE:The data set WORK.NUMBERS has 6 observations and 3
variables.NOTE:DATA statement used (Total process time): real time
0.01 seconds cpu time            0.01 seconds 15    ; 16 17    proc printto
print='print1.out' new; 18    run; NOTE:Writing HTML Body file: sashtml.htm
NOTE:PROCEDURE PRINTTO used (Total process time): real time           4.04
seconds cpu time            0.62 seconds 19 20    proc print data=numbers;
21      title 'Listing of NUMBERS Data Set'; 22    run; NOTE:There were 6
observations read from the data set WORK.NUMBERS.NOTE:PROCEDURE PRINT used
(Total process time): real time           0.56 seconds cpu time            0.09
seconds 23 24    proc printto; 25    run;
```

出力

アウトプット 46.1 外部ファイルに送られたプロシジャ出力



例 2: 出力先を SAS カタログエントリに指定する

要素: オプションなしの PRINTTO ステートメント

PRINTTO ステートメントオプション

- LABEL=
- LOG=
- NEW
- PRINT=

詳細

この例では、PROC PRINTTO を使用して SAS ログとプロシジャ出力を SAS カタログエントリに送ってから、出力先を両方ともデフォルトにリセットします。

プログラム

```
options source;

libname lib1 'SAS-library';

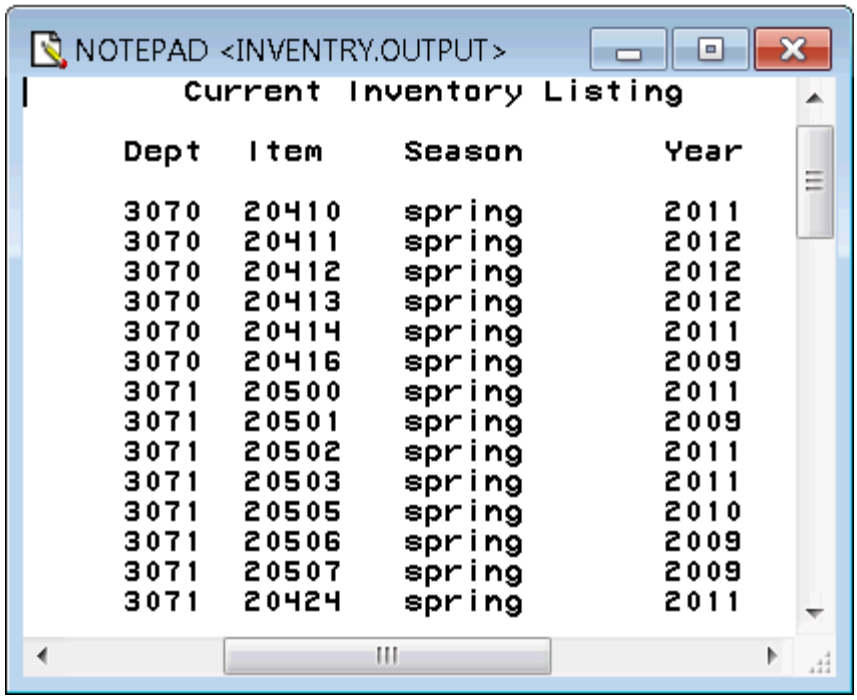
proc printto log=test.log label='Inventory program' new;
run;

data lib1.inventry;
   length Dept $ 4 Item $ 6 Season $ 6 Year 4;
   input dept item season year @@;
   datalines;
3070 20410  spring 2011 3070 20411  spring 2012
3070 20412  spring 2012 3070 20413  spring 2012
3070 20414  spring 2011 3070 20416  spring 2009
3071 20500  spring 2011 3071 20501  spring 2009
3071 20502  spring 2011 3071 20503  spring 2011
3071 20505  spring 2010 3071 20506  spring 2009
3071 20507  spring 2009 3071 20424  spring 2011
;

proc printto print=lib1.cat1.inventry.output
             label='Inventory program' new;
run;

proc report data=lib1.inventry nowindows headskip;
   column dept item season year;
   title 'Current Inventory Listing';
run;

proc printto;
run;
```

プログラムの説明

SAS システムオプションを設定します。 SOURCE オプションで、ソースステートメントを SAS ログに書き込むことが指定されます。

```
options source;
```

ライブラリ参照名を割り当てます。

```
libname lib1 'SAS-library';
```

SAS ログを SAS カタログエントリに送ります。 PROC PRINTTO で、SAS ログが、SASUSER.PROFILE.TEST.LOG という名前の SAS カタログエントリに送られます。エントリの名前と種類のみ指定したので、PRINTTO プロシジャでは、デフォルトのライブラリ参照名およびカタログ SASUSER.PROFILE が使用されます。LABEL=で、カタログエントリの説明が割り当てられます。

```
proc printto log=test.log label='Inventory program' new;
run;
```

LIB1.INVENTORY データセットを作成します。 DATA ステップでは、永久 SAS データセットが作成されます。

```
data lib1.inventry;
   length Dept $ 4 Item $ 6 Season $ 6 Year 4;
   input dept item season year @@;
   datalines;
3070 20410  spring 2011 3070 20411  spring 2012
3070 20412  spring 2012 3070 20413  spring 2012
3070 20414  spring 2011 3070 20416  spring 2009
3071 20500  spring 2011 3071 20501  spring 2009
3071 20502  spring 2011 3071 20503  spring 2011
3071 20505  spring 2010 3071 20506  spring 2009
3071 20507  spring 2009 3071 20424  spring 2011
;
```

プロシジャ出力を SAS カタログエントリに送ります。 後続の PROC REPORT ステップから SAS カタログエントリ LIB1.CAT1.INVENTRY.OUTPUT まで、プロシジャ出力を送るために、PROC PRINTTO ルートにより LISTING 出力先が開かれます。LABEL=で、カタログエントリの説明が割り当てられます。プロシジャ出力が SAS カタログに送られた後、PROC PRINTTO により LISTING 出力先が閉じられます。

```
proc printto print=lib1.cat1.inventry.output
             label='Inventory program' new;
run;

proc report data=lib1.inventry nowindows headskip;
   column dept item season year;
   title 'Current Inventory Listing';
run;
```

SAS ログとプロシジャ出力をリセットしてデフォルトに戻し、ファイルを閉じます。 PROC PRINTTO によって、前の PROC PRINTTO ステップで開かれた現在のファイルが閉じられ、後続の SAS ログとプロシジャ出力がデフォルト出力先に送り直されます。

```
proc printto;
run;
```

ログ

SAS エクスプローラを使用してこのログを表示するには、**Sasuser** ⇨ **Profile** を選択します。**Test** をダブルクリックします。NOTEPAD でログが開きます。

ログ 46.3 *SAS カタログエントリ SASUSER.PROFILE.TEST.LOG に送られた SAS ログ*

```
NOTE:PROCEDURE PRINTTO used (Total process time): real time           0.00
seconds cpu time            0.00 seconds 49 50   data lib1.inventry; 51
length Dept $ 4 Item $ 6 Season $ 6 Year 4; 52      input dept item season year
@@; 53      datalines; NOTE:SAS went to a new line when INPUT statement reached
past the end of a line.NOTE:The data set LIB1.INVENTRY has 14 observations and 4
variables.NOTE:DATA statement used (Total process time): real time
0.03 seconds cpu time            0.00 seconds 61    ; 62   ods listing; 63   proc
printto print=lib1.cat1.inventry.output 64                label='Inventory
program' new; 65   run; NOTE:PROCEDURE PRINTTO used (Total process time): real
time           0.00 seconds cpu time            0.00 seconds 66 67   proc report
data=lib1.inventry nowindows headskip; 68      column dept item season year;
69      title 'Current Inventory Listing'; 70   run; NOTE:There were 14
observations read from the data set LIB1.INVENTRY.NOTE:PROCEDURE REPORT used
(Total process time): real time           0.09 seconds cpu time            0.04
seconds 71 72   proc printto; 73   run; NOTE:PROCEDURE PRINTTO used (Total
process time): real time           0.00 seconds cpu time            0.00 seconds
74   ods listing close;
```

出力

SAS エクスプローラを使用してこのログを表示するには、**Lib1** ⇨ **Cat1** を選択します。**Inventry** をダブルクリックします。NOTEPAD で出力が開きます。

アウトプット 46.2 *SAS カタログエントリ LIB1.CAT1.INVENTRY.OUTPUT に送られたプロシジャ出力*



例 3: プロシジャ出力を入力ファイルとして使用する

要素: オプションなしの PRINTTO ステートメント

PRINTTO ステートメントオプション
- LOG=
- NEW
- PRINT=

詳細

この例では、PROC PRINTTO を使用してプロシジャ出力を外部ファイルに送り、そのファイルを DATA ステップへの入力として使用します。

プログラム

```
data test;
   do n=1 to 1000;
      x=int(ranuni(77777)*7);
      y=int(ranuni(77777)*5);
      output;
   end;
run;

filename routed 'output-filename';

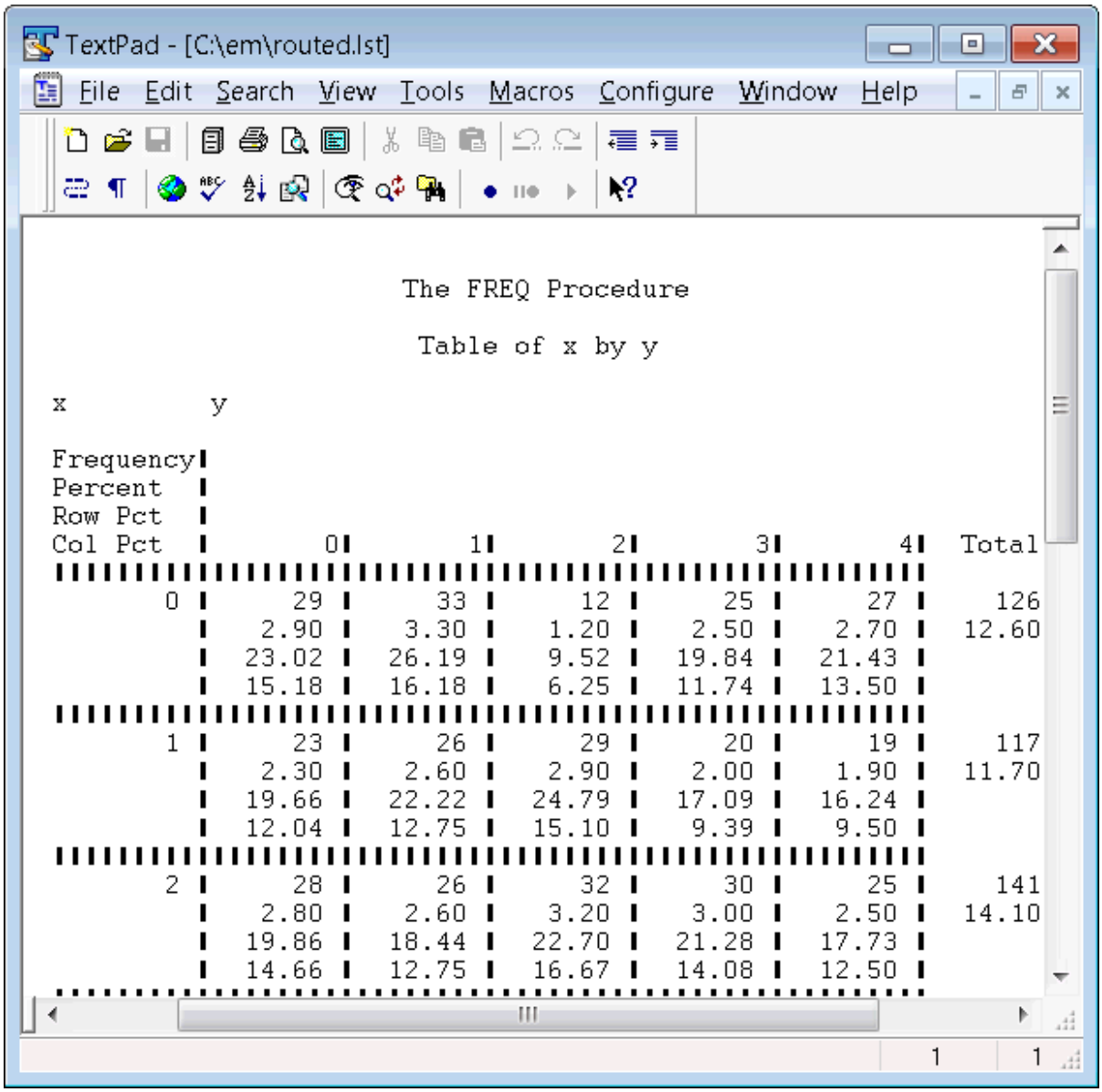
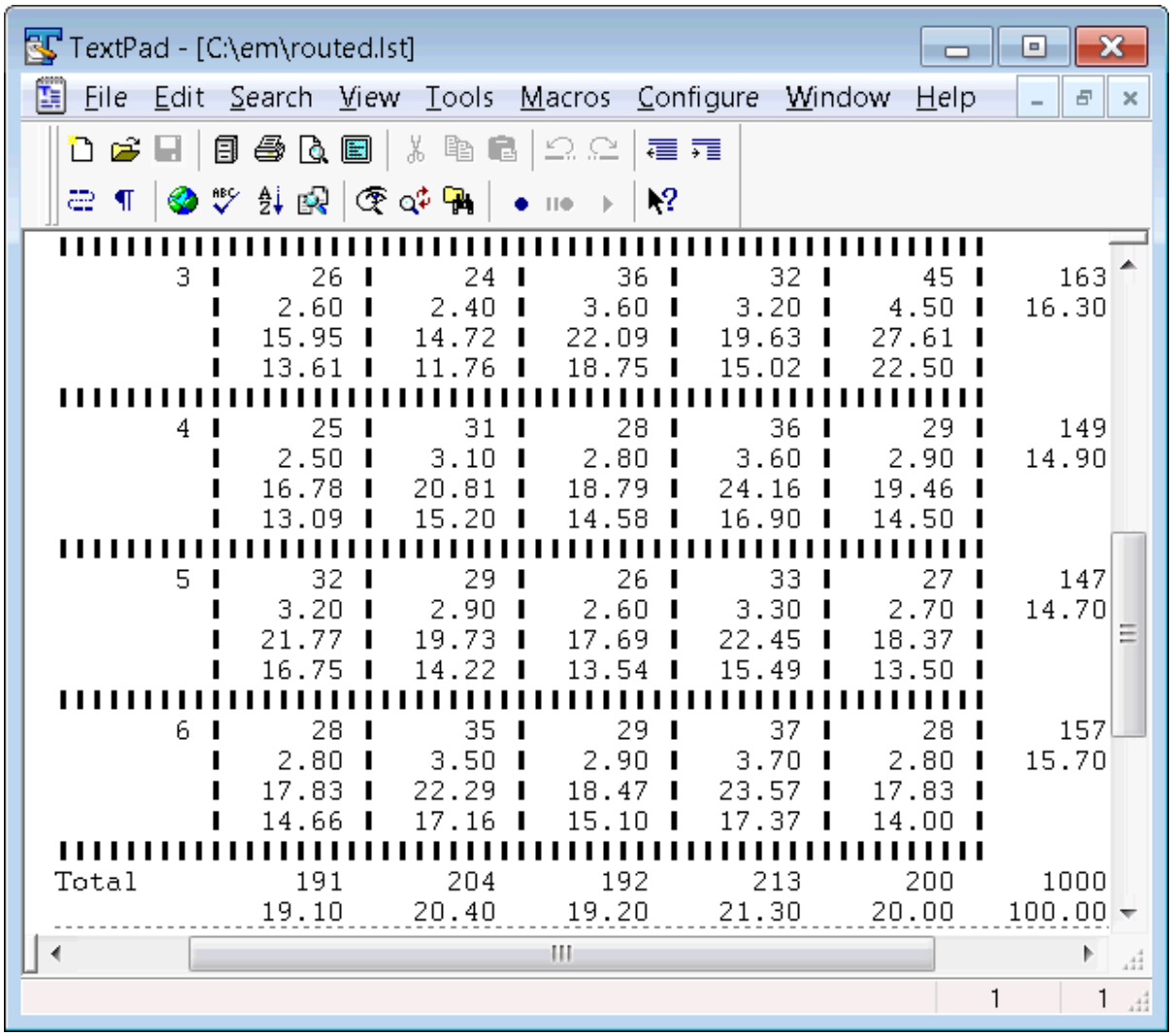
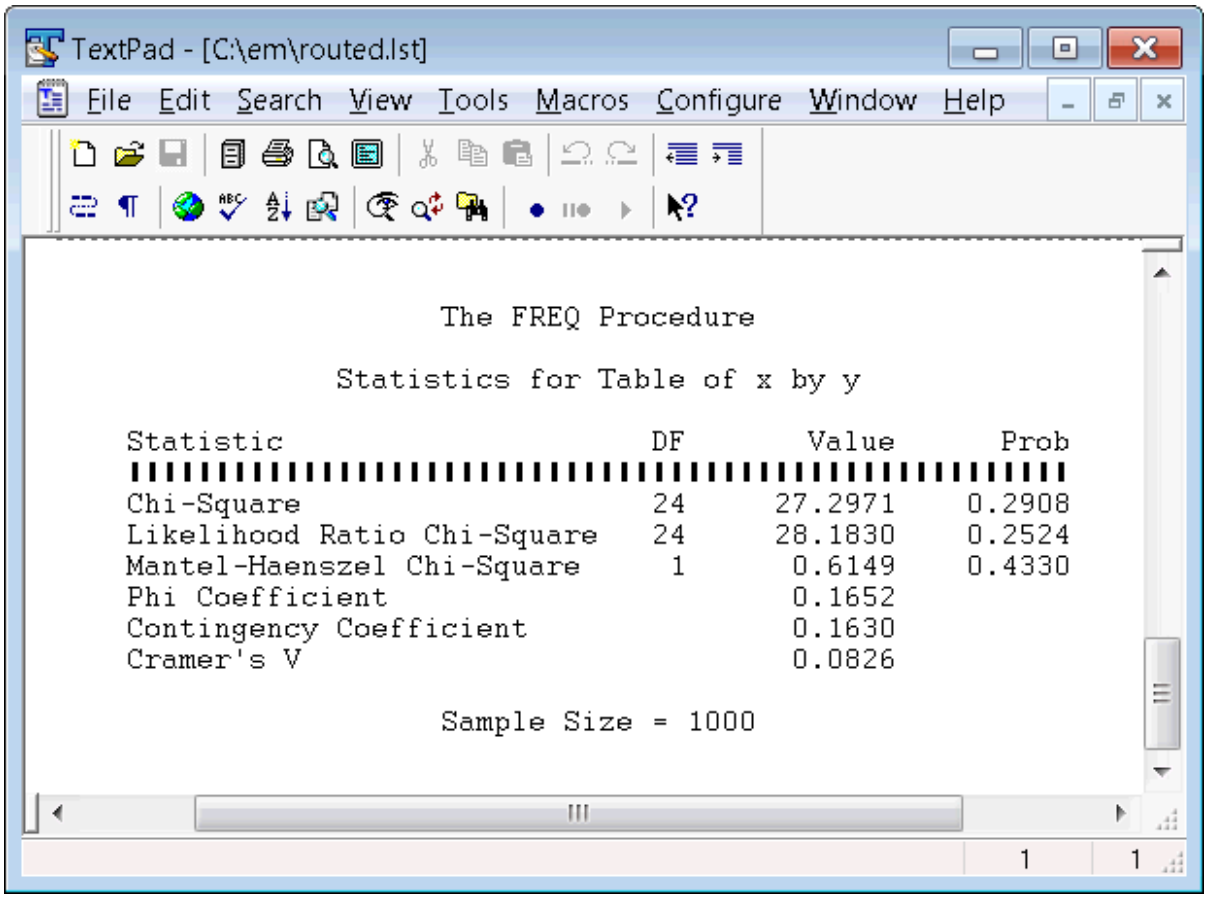
proc printto print=routed new;
   run;

   proc freq data=test;
      tables x*y / chisq;
   run;

   proc printto print=print;
   run;

   data probtest;
      infile routed;
      input word1 $ @;
      if word1='Chi-Squa' then
         do;
            input df chisq prob;
            keep chisq prob;
            output;
         end;
   run;

   proc print data=probtest;
      title 'Chi-Square Analysis for Table of X by Y';
   run;
```

プログラムの説明

変数に対してランダム値を生成します。 DATA ステップで、RANUNI 関数を使用して、データセット A の変数 X と Y に対する値がランダム生成されます。

```
data test;
   do n=1 to 1000;
      x=int(ranuni(77777)*7);
      y=int(ranuni(77777)*5);
      output;
   end;
run;
```

ファイル参照名を割り当てて、プロシジャ出力を参照先ファイルに送ります。 FILENAME ステートメントで、ファイル参照名が外部ファイルに割り当てられます。PROC PRINTTO で、後続のプロシジャ出力が、ファイル参照名 ROUTED で参照されるファイルに送られます。このプロシジャオプションを送っている間に、PROC PRINTTO により LISTING 出力先が開きます。後述の"参照名 ROUTED の外部ファイルに送られた PROC FREQ 出力"を参照してください。

```
filename routed 'output-filename';

proc printto print=routed new;
   run;
```

度数を作成します。 PROC FREQ で、データセット TEST の変数 X と Y の度数とカイ 2 乗分析が計算されます。この出力は ROUTED として参照されるファイルに送られます。

```
proc freq data=test;
   tables x*y / chisq;
run;
```

ファイルを閉じます。 次の DATA ステップでファイルの読み取りを可能にするために、もう 1 度 PROC PRINTTO を使用して、ファイル参照名 ROUTED で参照されるファイルを閉じる必要があります。また、このステップでは、後続のプロシジャ出力がデフォルト出力先に送られます。PRINT=を使用すると、ステップはプロシジャ出力にのみ影響し、SAS ログには影響しません。

```
proc printto print=print;
run;
```

データセット PROBTEST を作成します。 DATA ステップでは、ROUTED(PROC FREQ 出力を含むファイル)が入力ファイルとして使用され、データセット PROBTEST が作成されます。DATA ステップでは、ROUTED のレコードがすべて読み取られますが、**Chi-Squa** で始まるレコードからのみオブザベーションが作成されます。

```
data probtest;
   infile routed;
   input word1 $ @;
   if word1='Chi-Squa' then
      do;
         input df chisq prob;
         keep chisq prob;
         output;
      end;
run;
```

PROBTEST データセットを印刷します。 PROC PRINT で、データセット PROBTEST の簡単なリストが作成されます。この出力がデフォルト出力先に送られます。出力セクションの"デフォルト出力先に送られるデータセット PROBTEST の PROC PRINT 出力"を参照してください。

```
proc print data=probtest;
```

```
    title 'Chi-Square Analysis for Table of X by Y';
run;
```

出力

アウトプット 46.3 *参照名 ROUTED の外部ファイルに送られた PROC FREQ 出力*





3	26	24	36	32	45	163
	2.60	2.40	3.60	3.20	4.50	16.30
	15.95	14.72	22.09	19.63	27.61	
	13.61	11.76	18.75	15.02	22.50	
4	25	31	28	36	29	149
	2.50	3.10	2.80	3.60	2.90	14.90
	16.78	20.81	18.79	24.16	19.46	
	13.09	15.20	14.58	16.90	14.50	
5	32	29	26	33	27	147
	3.20	2.90	2.60	3.30	2.70	14.70
	21.77	19.73	17.69	22.45	18.37	
	16.75	14.22	13.54	15.49	13.50	
6	28	35	29	37	28	157
	2.80	3.50	2.90	3.70	2.80	15.70
	17.83	22.29	18.47	23.57	17.83	
	14.66	17.16	15.10	17.37	14.00	
Total	191	204	192	213	200	1000
	19.10	20.40	19.20	21.30	20.00	100.00



アウトプット 46.4 *デフォルト出力先に送られるデータセット PROBTEST の PROC PRINT 出力*

Chi-Square Analysis for Table of X by Y

Obs	chisq	prob
1	27.2971	0.2908

例 4: 出力先をプリンタに指定する

要素: PRINTTO ステートメントオプション
PRINT=

詳細

この例では、PROC PRINTTO によって、プロシジャ出力が直接プリンタに送られます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

filename your_fileref printer
```

```
'printer-name';

proc printto print=your_fileref;
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

ファイル参照名をプリンタ名に関連付けます。 FILENAME ステートメントで、ファイル参照名が、指定したプリンタ名に関連付けられます。ファイル参照名をデフォルトプリンタに関連付ける場合は、'*printer-name*'を省略します。

```
filename your_fileref printer
'printer-name';
```

プリンタに送るファイルを指定します。 PRINT=オプションで、PROC PRINTTO によってプリンタに送られるファイルが指定されます。

```
proc printto print=your_fileref;
run;
```

47 章

PRODUCT_STATUS プロシジャ

概要: PRODUCT_STATUS プロシジャ

PROC PRODUCT_STATUS は、システムにインストールされている SAS Foundation 製品の一覧を、それらの製品のバージョン番号とともに返します。PROC PRODUCT_STATUS を使用すると、該当の SAS 製品を使用できるかどうかすばやく確認できます。PROC PRODUCT_STATUS の結果は SAS ログに書き込まれます。

PROC PRODUCT_STATUS では、Web アプリケーションまたはその他の Java ベースの製品に関する情報は返されません。

サイトに SAS Metadata Server がインストールされている場合は、PROC PRODUCT_STATUS ではなく SAS ViewRegistry ユーティリティを使用します。

SYSVLONG および SYSVLONG4 自動マクロ変数は、サイトにインストールされている SAS ホストイメージのバージョン情報のみを返します。これらのマクロ変数では、サイトにインストールされているすべての SAS Foundation 製品の情報は返されません。詳細については、“SYSVLONG Automatic Macro Variable” (SAS Macro Language: Reference)および “SYSVLONG4 Automatic Macro Variable” (SAS Macro Language: Reference)を参照してください。

構文: PRODUCT_STATUS プロシジャ

PROC PRODUCT_STATUS;

ステートメント	タスク
“例: PROC PRODUCT_STATUS の結果”	オペレーティングシステムにインストールされている SAS Foundation 製品の名前とバージョンを指定します。

PRODUCT_STATUS ステートメント

オペレーティングシステムにインストールされている SAS Foundation 製品の名前とバージョンを返します。

構文

PROC PRODUCT_STATUS;

詳細

PROC PRODUCT STATUS ステートメントに引数はありません。

例: PROC PRODUCT_STATUS の結果

```
proc product_status;
run;
```

PROC PRODUCT_STATUS で生成される結果の例を含む部分出力を次に示します。

```
For Base SAS Software ...Custom version information:9.4_M3 Image version
information:9.04.01M3D041815 For SAS/STAT ...Custom version information:14.1
For SAS/GRAPH ...Custom version information:9.4_M3 For SAS/ETS ...Custom
version information:14.1 For SAS/FSP ...Custom version information:9.4_M3 For
SAS/OR ...Custom version information:14.1 For SAS/AF ...Custom version
information:9.4_M3 For SAS/IML ...Custom version information:14.1 For SAS/
QC ...Custom version information:14.1 For SAS/ASSIST ...Custom version
information:9.4 For SAS/CONNECT ...Custom version information:9.4_M3 For SAS/
TOOLKIT ...Custom version information:9.4 For SAS/GIS ...Custom version
information:9.4_M3 For SAS Table Server ...Custom version information:9.4 For
SAS/ACCESS Interface to Netezza ...Custom version information:9.4_M3
```

48 章

PROTO プロシジャ

概要: PROTO プロシジャ

PROTO プロシジャを使用すると、C または C++プログラミング言語で記述される外部関数をバッチモードで登録できます。これらの関数は、SAS、C-language 構造および種類で使用できます。C 言語関数は PROC PROTO に登録された後、FCMP プロシジャで宣言される SAS 関数またはサブルーチン、COMPILE プロシジャで宣言される SAS 関数、サブルーチン、メソッドブロックから呼び出し可能です。

概念: PROTO プロシジャ

関数プロトタイプの登録

関数プロトタイプは PROTO プロシジャで登録(宣言)されます。次の形式を使用します。

return-type function-name (*argument-type* <*argument-name*> / <*iotype*> <*argument-label*>, ...)<*option(s)*>;

return-type
戻り値の C 言語の種類を指定します。

ヒント *return-type* の前に UNSIGNED か EXCELDATE のどちらかの修飾子を付けられます。戻り値の種類が Microsoft Excel 日付の場合は、EXCELDATE を使用する必要があります。

参照項目 “サポート対象の C 言語の戻り値の種類” (1451 ページ)

function-name
登録する関数の名前を指定します。

ヒント 指定パッケージ内の関数名は、最初の 32 文字を一意にする必要があります。

argument-type
関数の引数に対して C 言語の型を指定します。

関数の引数リストで、引数ごとに *argument-type* を指定する必要があります。引数リストは、必ずかっこで囲んでください。引数が配列の場合は、配列サイズを含む角かっこが後ろに付いた引数を指定する必要があります(例:`double A[10]`)。サイズが不明な場合や、長さの検証を無効化する場合は、かわりに型**name* を使用します(例:`double*A`)。

ヒント *Argument-type* の前に UNSIGNED、CONST、EXCELDATE のいずれかの修飾子を付けられます。戻り値の種類が Microsoft Excel 日付の場合は、EXCELDATE を使用する必要があります。

参照項目 “サポート対象の C 引数の種類” (1452 ページ)

argument-name
引数の名前を指定します。

iotype
引数の入出力の種類を指定します。入力(input)の場合は I、出力(output)の場合は O、更新(update)の場合は U を使用します。

別名 IO

ヒント プログラムコードでは、IOTYPE=I | O | U を使用します。

デフォルトでは、ポインタとなるパラメータではすべて入力の種類が U と見なされます。ポインタ以外の値ではすべて入力の種類が I と見なされます。この動作は、C 言語のパラメータ渡しスキームに対応しています。

参照項目 “例: 分割関数の例” (1469 ページ)</*iotype* の使用例

argument-label
引数のラベルや説明を指定します。

次のオプションは PROTO プロシジャとともに使用できます。

LABEL="*text-string*"
関数の説明やラベルを指定します。テキスト文字列を引用符で囲みます。

KIND | GROUP=*group-type*
関数の属するグループを指定します。KIND=または GROUP=オプションを使用すると、パッケージの関数を簡単にグループ化できます。

引用符内の任意の文字列(40 文字以内)を使用して、類似する関数をグループ化できます。

ヒント 次はリスクディメンション用に提供された特殊なケースで、引用符は必要ありません。INPUT(金融商品の入力)、TRANS(リスク ファクタの変換)、PRICING(金融商品の評価)、PROJECT.デフォルトは PRICING です。

サポート対象の C 言語の戻り値の種類

PROTO プロシジャでは、次の C 言語の戻り値の種類がサポートされています。

表 48.1 *サポート対象の C 言語の戻り値の種類*

関数プロトタイプ	SAS 変数の種類	C 変数の種類
short	numeric	short, short *, short **
short *	numeric, array	short, short *, short **
int	numeric	int, int *, int **
int *	numeric, array	int, int *, int **
long	numeric	long, long *, long **
long *	numeric, array	long, long *, long **
double	numeric	double, double *, double **
double *	numeric, array	double, double *, double **
char *	character	char *, char **
struct *	struct	struct *, struct **

関数プロトタイプ	SAS 変数の種類	C 変数の種類
void		void

サポート対象の C 引数の種類

PROTO プロシジャでは、次の C 引数の種類がサポートされています。

表 48.2 サポート対象の C 引数の種類

関数プロトタイプ	SAS 変数の種類	C 変数の種類
short	numeric	short, short *, short **
short *	numeric, array	short, short *, short **
short **	array	short *, short **
int	numeric	int, int *, int **
int *	numeric, array	int, int *, int **
int **	array	int *, int **
long	numeric	long, long *, long **
long *	numeric, array	long, long *, long **
long **	array	long *, long **
double	numeric	double, double *, double **
double *	numeric, array	double, double *, double **
double **	array	double *, double **
char *	character	char *, char **
char **	character	char *, char **
struct *	structure	struct *, struct **
struct **	structure	struct *, struct **

SAS の C 言語構造

基本概念

多くの C 言語ライブラリには、引数として構造体ポインタを有する関数が含まれています。SAS では、PROC PROTO でのみ構造を定義できます。定義後は、PROC PROTO に対応した多くのプロシジャ(PROC COMPILE など)内で、宣言、インスタンス化ができます。

C 構造体は、連続するメモリに適用されるテンプレートです。テンプレートの各エントリでは、名前と種類が指定されます。各要素の種類によって、各エントリに関連付けられるバイト数、および各エントリの使用法が決定されます。調整ルールとベースの種類のサイズがさまざまなため、SAS では、構造体のメモリの各エントリの場所決定は、現在のマシンコンパイラに依存しています。

SAS での構造体の宣言と参照

SAS における構造宣言の構文は、C 非ポインタ構造宣言と同じです。構造宣言は、次の形式になります。

struct *structure_name structure_instance*;

各構造体は、宣言時にゼロ値に設定されます。構造体では、前のパスから、データによって次のパスが開始されるまで、値が保持されます。

構造要素は、C の静的ピリオド(.)表記を使用して参照されます。SAS にはポインタ構文がありません。ある構造体が別の構造体を指す場合、指されている構造体を参照する唯一の方法は、同じ型の宣言された構造体にポインタを割り当てることです。要素にアクセスするには、その宣言された構造体を使用します。

構造エントリが short、int または long 型で、なおかつ式で参照されている場合は、まず double 型にキャストしてから計算に使用します。構造エントリがベースの種類へのポインタの場合は、ポインタが間接参照され、値が返されます。ポインタが null の場合は、欠損値が返されます。変換に失敗した場合や、欠損値が二重構造以外のエンティティに割り当てられた場合は、PROC PROTO コードで行われた欠損値の割り当てが使用されます。

SAS で配列の長さがわかっている必要があります。これは、構造でディメンションが宣言されている限り、構造の配列エントリを SAS の配列と同様に使用できるようにするためです。この要件には、short、int および long 型の配列も含まれます。エントリが実際に double 型の配列へのポインタである場合は、SAS 配列にそのポインタを割り当てると、配列要素にアクセスできます。配列構文を使用すると、その他の型の配列へのポインタにはアクセスできません。

構造体の例

```
proc proto package =
sasuser.mylib.struct
   label = "package of structures";

   #define MAX_IN 20;

   typedef char * ptr;
   struct foo {
      double hi;
      int mid;
      ptr buf1;
      long * low;
```

```
      struct {
         short ans[MAX_IN + 1];
         struct {     /* inner */
            int inner;
         } n2;
         short outer;
      } n;
   };

   typedef struct foo *str;

   struct foo2 {
      str tom;
   };

   str get_record(char *name, int userid);
run;

proc fcmp library = sasuser.mylib;
   struct foo result;
   result = get_record("Mary", 32);
   put result=;
run;
```

SAS での列挙

列挙は、整数値のニーモニックです。列挙によって、リテラルな名前を特定の数字として設定できるので、C プラグラムの読みやすさやサポートしやすさが促進されます。列挙は、C 言語ライブラリで、リターンコードを簡略化するために使用されます。C プログラムをコンパイルした後は、列挙名にアクセスできなくなります。

列挙の種類の例

次の例では、PROC PROTO の 2 つの列挙値の種類 YesNoMaybeType と Tens の設定方法を示します。両方とも構造 ESTRUCTURE で参照されます。

```
proc proto package=sasuser.mylib.str2
   label="package of structures";

   #define E_ROW 52;
   #define L_ROW 124;
   #define S_ROW 15;

   typedef double ExerciseArray[S_ROW][2];
   typedef double LadderArray[L_ROW];
   typedef double SamplingArray[S_Row];

   typedef enum
   {
      True, False, Maybe
   }  YesNoMaybeType;

   typedef enum {
      Ten=10, Twenty=20, Thirty=30, Forty=40, Fifty=50
      } Tens;

   typedef struct {
```

```
      short           rows;
      short           cols;
      YesNoMaybeType  type;
      Tens            dollar;
      ExerciseArray   dates;
    } EStructure;
run;
```

次の PROC FCMP の例では、これらの列挙の種類にアクセスする方法を示します。この例では、PROC PROTO で設定される列挙値が、SAS でマクロ変数として実装されます。したがって、&記号を使用してアクセスする必要があります。

```
proc fcmp library=sasuser.mylib;
   struct EStructure mystruct;

   mystruct.type=&True;
   mystruct.dollar=&Twenty;
run;
```

SAS の C ソースコード

外部 C 関数をコンパイルするには、PROC PROTO を限定的な方法で使用します。次のように、PROC PROTO でソースコードを指定できます。

```
externc function-name;
   ... C-source-statements ...
externcend;
```

関数名によって、PROC PROTO で、EXTERNC ステートメントと EXTERNCEND ステートメントの間にどの関数のソースコードが指定されているかがわかります。PROC PROTO でソースコードをコンパイルする際には、現在宣言されているすべての構造定義と C 関数プロトタイプが含まれます。ただし、typedef と#define は含まれません。

この機能は、先に存在する外部 C ライブラリへのインターフェイスを促進する単純な“Helper”関数の作成を可能にするために提供されます。有効な C ステートメントはすべて、#include ステートメントを除いて許可されます。C-stdlib 関数の限定サブセットのみ使用可能です。ただし、現在の PROC PROTO ステップ内ですでに宣言されているその他の C 関数を呼び出せます。

次の C-stdlib 関数がサポートされています。

表 48.3 サポート対象の stdlib 関数

関数	説明
double sin(double x)	x の正弦を返します(ラジアン)。
double cos(double x)	x の余弦を返します(ラジアン)。
double tan(double x)	x の正接を返します(ラジアン)。
double asin(double x)	x の逆正弦を返します(-pi/2 から pi/2 ラジアンまで)。
double acos(double x)	x の逆余弦を返します(0 から pi ラジアンまで)。
double atan(double x)	x の逆正接を返します(-pi/2 から pi/2 ラジアンまで)。

関数	説明
double atan2(double x, double y)	y/x の逆正接を返します(-pi から pi ラジアンまで)。
double sinh(double x)	x の双曲線正弦を返します(ラジアン)。
double cosh(double x)	x の双曲線余弦を返します(ラジアン)。
double tanh(double x)	x の双曲線正接を返します(ラジアン)。
double exp(double x)	x の指数値を返します。
double log(double x)	x の対数を返します。
double log2(double x)	底を 2 とする x の対数を返します。
double log10(double x)	底を 10 とする x の対数を返します。
double pow(double x, double y)	x の y 乗(x**y)を返します。
double sqrt(double x)	x の平方根を返します。
double ceil(double x)	x 以上の最小の整数を返します。
double fmod(double x, double y)	(x/y)の余りを返します。
double floor(double x)	x 以下の最大の整数を返します。
int abs(int x)	x の絶対値を返します。
double fabs(double)	x の絶対値を返します。
int min(int x, int y)	x と y の最小値を返します。
double fmin(double x, double y)	x と y の最小値を返します。
int max(int x, int y)	x と y の最大値を返します。
double fmax(double x, double y)	x と y の最大値を返します。
char* malloc(int x)	サイズ x のメモリを割り当てます。
void free(char*)	malloc で割り当てたメモリを解放します。

次の例は、PROC PROTO に直接書き込まれる単純な C 関数を示しています。

```
proc proto
package=sasuser.mylib.foo;
   struct mystruct {
      short a;
      long b;
   };
```

```
   int fillMyStruct(short a, short b,
struct mystruct * s);
   externc  fillMyStruct;
      int fillMyStruct(short a, short b,
struct mystruct * s) {
         s ->a = a;
         s ->b = b;
         return(0);
      }
   externcend;
run;
```

C 言語指定の制限

PROTO プロシジャでの C 言語指定の制限は次のとおりです。

- #define ステートメントは、後ろにセミコロン(;)を指定し、値を数値にする必要があります。
- #define ステートメントの機能は、単純な置き換えと、ネストされていない式に制限されています。影響を受ける記号は、配列ディメンション参照のみです。
- C プリプロセッサステートメントの#include と#if はサポートされていません。SAS マクロ%INC を#include のかわりに使用できます。
- 構造要素では最大 2 レベルの間接指定が使用できます。"double ***"などの要素は使用できません。これらの要素の種類が構造体で必要にもかかわらず、SAS でアクセスされない場合は、プレースホルダを使用できます。
- float 型はサポートされません。
- 現在サポートされている型指定子は Unsigned のみです。
- 構造変数の指定ビットサイズはサポートされません。
- 関数ポインタと関数ポインタの定義はサポートされません。
- union 型はサポートされません。ただし、union の 1 要素のみを使用する予定であれば、union の変数をその要素の型として宣言できます。
- 他の構造体へのポインタ以外の参照は、使用前に定義する必要があります。
- 構造体では ENUM キーワードは使用できません。構造体で ENUM を指定するには、TYPEDEF キーワードを使用します。
- 同じ英数字の名前を持つ構造要素でも、大文字/小文字の表記が異なる場合(ALPHA、Alpha および alpha など)はサポートされません。SAS では、大文字と小文字は区別されません。したがって、大文字と個別の区別がないプログラムで比較する場合、すべての構造要素を一意にする必要があります。

構文: PROTO プロシジャ

PROC PROTO PACKAGE=*entry* <*options*>;
MAPMISS *type1=value1 type2=value2 ...*;
LINK *load-module* <NOUNLOAD>;
function–prototype–1 <function-prototype-n …;>

ステートメント	タスク	例
“PROC PROTO ステートメント”	C または C++プログラミング言語で記述される外部関数をバッチモードで登録します。	Ex. 1
“LINK ステートメント”	名前、パス、および関数を含むロードモジュールを指定します。	
“MAPMISS ステートメント”	値が欠損している場合に関数に渡す代替値を型によって指定します。	

PROC PROTO ステートメント

C または C++プログラミング言語で記述される外部関数をバッチモードで登録します。

例: “例: 分割関数の例” (1469 ページ)

構文

PROC PROTO PACKAGE=*entry* <*options*>;

オプション引数の要約

ENCRYPT | HIDE
: XML データベースの場合のみ、データセット内でコードのエンコードを可能にします。

LABEL=*package-label*
: パッケージの説明やラベル付けのためのテキスト文字列を指定します。

NOSIGNALS
: パッケージ内の関数で例外を作成しないことを指定します。

STDCALL
: Windows PC プラットフォームでのみ、"_ stdcall"規則を使用して関数を呼び出すことを指定します。

STRUCTPACK*n* | PACK*n*
: Windows PC プラットフォームでのみ、パッケージ内のすべての構造を、特定の N-BYTE パッキングプラグマを使用してコンパイルすることを指定します。

必須引数

PACKAGE=*entry*
: プロトタイプ情報が保存されている SAS エントリを指定します。*Entry* は、***library.dataset.package*** というフォームがある 3 レベルの名前です。*Package* により、GUI でのグループ化を指定できます。

function-prototype-1
function-prototype-n
: 関数プロトタイプの C コードが含まれます。

オプション引数

ENCRYPT | HIDE
データベース内のエンコードの許可を指定します。

制限事項 このオプションは XML データベースにのみ使用可です。

LABEL=*package-label*
パッケージの説明やラベル付けのためのテキスト文字列を指定します。ラベルの最大長は 256 文字です。

NOSIGNALS
パッケージ内の関数で例外やシグナルを作成しないことを指定します。

STDCALL
Windows PC プラットフォームでのみ、パッケージ内のすべての関数を"_stdcall"規則を使用して呼び出すことを指定します。

STRUCTPACK*n* | PACK*n*
Windows PC プラットフォームでのみ、このパッケージ内のすべての構造を、指定 N-BYTE パッキングプラグマを使用してコンパイルすることを指定します。したがって、STRUCTURE4 では、パッケージ内のすべての構造を“#pragma pack(4)”オプションによりコンパイルすることを指定します。

LINK ステートメント

関数を含むロードモジュールの名前および(任意で)パスを指定します。

構文

LINK *load-module* <NOUNLOAD>;

必須引数

load-module
関数を含むロードモジュールを指定します。LINK ステートメントをさらに追加すると、プロトタイプに対して必要な数のライブラリを含められます。*Load-module* は動作環境に応じて次の形式を取ります。

```
'c:\mylibs\xxx.dll';

'c:\mylibs\xxx';

'/users/me/mylibs/xxx';
```

ヒント PROTO プロシジャによってモジュールをリンクするには、フルパス名の指定が最も安全かつ望ましい方法です。

オプション引数

NOUNLOAD
SAS セッション終了時に、選択したライブラリをロードしたままにしておくことを指定します。

詳細

モジュールの拡張子を指定する必要はありません。SAS では、動作環境固有の拡張子が付いたモジュールがロードされます。

SAS での検索を可能にするために、ロードモジュールですべての関数を外部宣言する必要があります。ほとんどのプラットフォームでは、外部宣言がコンパイラのデフォルト動作です。ただし、多くの C コンパイラでは、デフォルトでは関数名がエクスポートされません。次の例では、ほとんどの PC コンパイラに対して外部ロードのために関数を宣言する方法を示します。

```
_declspec(dllexport) int myfunc(int, double);
_declspec(dllexport) int price2(int a, double foo);
```

MAPMISS ステートメント

値が欠損している場合に関数に渡す代替値を型によって指定します。

構文

MAPMISS <POINTER=*pointer-value* > <INT=*integer-value* > <DOUBLE=*double-value*> < LONG=*long-value* > <SHORT=*short-value*>;

オプション引数

POINTER=*pointer-value*
欠損しているポインタ値のかわりに関数に渡すポインタ値を指定します。

デフォルト null

INT=*integer-value*
欠損している整数値のかわりに関数に渡す整数値を指定します。

DOUBLE=*double-value*
欠損している double 値のかわりに関数に渡す double 値を指定します。

LONG=*long-value*
欠損している long 値のかわりに関数に渡す long 値を指定します。

SHORT=*short-value*
欠損している short 値のかわりに関数に渡す short 値を指定します。

詳細

MAPMISS ステートメントを使用して、データの種類またはポインタ値によって、代替値を指定します。値が欠損している場合、これらの値が関数に渡されます。値は MAPMISS ステートメントの引数としてして指定されます。

POINTER=NULL を設定した場合、欠損しているポインタ変数のかわりに NULL 値ポインタが関数に渡されます。引数として使用される種類に対して関数へのマッピングを指定しない場合は、その種類の引数が欠損していても、関数は呼び出されません。

パラメータとして C 関数に渡されるときに、配列要素の欠損値チェックが行われないので、MAPMISS 値は配列に影響を及ぼしません。

基本的な C 言語の種類

SAS 言語では、文字と数値という 2 つのデータの種類がサポートされます。これらのデータの種類は、C プログラミング言語での character および double (倍精度浮動小数点)データ型の配列に対応しています。SAS 変数は、外部 C 関数に対する引き数として使用される場合、適切な型に変換(キャスト)されます。

文字変数の操作

文字変数は、“char *”値を必要とする引数に対してのみ使用できます。渡される文字列は、現在の文字列長で終了する null 文字列です。現在の文字列長は、文字列の割り当て長と、文字列に格納された最終値の長さの最小値です。文字列の割り当て長(デフォルトでは、32 バイト)は、LENGTH ステートメントを使用すると指定できます。"char *"を返す関数では、SAS 変数にコピーされる、null またはゼロ区切りの文字列が返される場合があります。現在の文字列長が割り当て長よりも短い場合、文字列にはブランクが埋め込まれます。

次の例では、*str* の割り当て長は 10 ですが、現在の長さは 5 です。文字列が割り当て長で NULL で終了する場合、"hello "が関数 xxx に渡されます。

```
length str $ 10;
str = "hello";
call xxx(str);
```

ブランクの埋め込みを回避するには、関数呼び出し内のパラメータで SAS 関数 TRIM を使用します。

```
length str $ 10;
str = "hello";
call xxx(trim(str));
```

この場合、値"hello"が関数 xxx に渡されます。

数値変数の操作

short、int、long、double データ型を必要とする引数、ならびにそのデータ型へのポインタとして、数値変数を使用できます。数値変数は、自動的に必要な型に変換されます。変換に失敗すると、関数は呼び出されず、関数に対する出力は欠損に設定されます。これらの型へのポインタが要求された場合は、変換値のアドレスが渡されます。値は、呼び出し先から返される際、double 型に変換し直され、SAS 変数に格納されます。SAS スカラ変数は、2 レベル以上の間接指定を要する引数として渡すことはできません。たとえば、SAS 変数は、"long **"型へのキャストを要する引数として渡すことはできません。

欠損値の操作

欠損値を含む SAS 変数は、PROTO プロシジャ使用時の呼び出し関数での欠損値のマッピング方法に従って変換されます。関数から返されたすべての値に対して、マップされた欠損値のチェックが行われ、SAS 欠損値に変換されます。

たとえば、関数に対する引数が欠損しており、なおかつその引数が整数への変換対象の場合、整数は-99 にマップされ、-99 が関数に渡されます。同じ関数で値-99 を含む整数が返された場合、この値の返し先の変数には欠損値が含まれます。

関数名

外部関数および FCMP 関数は、異なるパッケージに保存されている限り、同じ名前を保持できます。これらのパッケージのロード時には、ログの警告メッセージに、一定の関数に対するデフォルト定義を含むパッケージが表示されます。デフォルト定義以外の、パッケージの関数定義を使用するには、*package-name.function-name* を使用して関数を呼び出します。

function-name 複数の外部関数パッケージをロードするときは、すべての関数名が固有である必要があります。同一名の外部関数を 2 つ以上ロードする場合は、最初にロードされる関数を使用します。重複する外部関数は無視します。警告メッセージに、使用される関数を含むパッケージと、破棄された定義を含んでいたパッケージが表示されます。

外部 C 関数との連携

外部 C 関数との連携を容易にするために、PROTO に対応した多くのプロシジャが拡張され、その C 型のほとんどがサポートされるようになりました。

配列での作業時には、SAS 配列が EXTERNC ルーチンに一次元配列として受け渡されるため注意が必要です。SAS 配列は、FCMP 関数に戻るときには一次元配列として読み込まれます。

次の例は、配列とダブル**変数タイプの使用時に生成されるログ出力を示しています。

```
proc proto package=work.proto_ds.test;
   void idmat(double** a,int m,int n);
   externc idmat;
   void idmat(double** a,int m,int n)
   {
      int i=0,j=0;
      for(i=0;i<m;i++)
      {
         for(j=0;j<n;j++)
         {
          if(i==j)
             a[0][i*m+j]=1;
          else
```

```
            a[0][i*m+j]=0;
         }
      }
};
externcend;
run;

proc fcmp outlib=work.proto_ds.test inlib=work.proto_ds;
   subroutine sas_idmat(b[*,*]);
      outargs b;
      m=dim(b,1);
      n=dim(b,2);
      call idmat(b,m,n);
      put b=;
      put ' ';
   endsub;
quit;
run;

options cmplib=work.proto_ds;

data cmat;
   array ctmp[3,3] _temporary_;
   array c[3,3];
   output;
   call sas_idmat(ctmp);
   do i=1 to dim(c,1);
      do j=1 to dim(c,2);
         c[i,j]=ctmp[i,j];
      end;
   end;
   output;
   drop i j;
run;
```

SAS では、次の結果がログに書き込まれます。

```
b[1, 1]=1 b[1, 2]=0 b[1, 3]=0 b[2, 1]=0 b[2, 2]=1 b[2, 3]=0 b[3, 1]=0 b[3, 2]=0
b[3, 3]=1
```

SAS 変数でいずれかの種類にポインタを返して保存する方法はありません。ポインタは常に間接参照され、そのコンテンツが変換されて SAS 変数にコピーされます。

PROTO に対応したプロシジャで C 変数を指定するには、EXTERNC ステートメントを使用します。EXTERNC ステートメントの構文は、次の形式を取ります。

EXTERNC DOUBLE | INT | LONG | SHORT | CHAR <[*][*]> *var-1 <var-2 ... var-n>*;

次の表(表 48.4 (1464 ページ))は、これらの変数が式の左側に位置する場合の処理を示しています。この表では、割り当ての右側で short 型に対して実行される自動キャストも示されます。(キャストと呼ばれる単項演算子によって、いずれの式でも明示的な型の変換が強制実行される場合があります。)表には、SAS 変数に関連付けられた short 型の使用可能な組み合わせがすべてリストされています。

注: int、long および double 型の表を作成するには、任意の型をこの表の"short"と置き換えます。

ポインタのいずれかが null で、間接参照を要する場合は、結果変数に対して設定された欠損値があれば、結果は欠損に設定されます。詳細については、“MAPMISS ステートメント” (1460 ページ)を参照してください。

表 48.4 *割り当てステートメントでの short データ型の自動型キャスト*

割り当ての左側の型	割り当ての右側の型	実行されるキャスト
short	SAS 数値	y = (short) x
short	short	y = x
short	short *	y = * x
short	short **	y = ** x
short *	SAS 数値	* y = (short) x
short *	short	y = & x
short *	short *	y = x
short *	short **	y = * x
short **	SAS 数値	**y = (short) x
short **	short *	y = & x
short **	short **	y = x
SAS 数値	short	y = (double) x
SAS 数値	short *	y = (double) * x
SAS 数値	short **	y = (double) ** x

次の表は、これらの変数が引数として外部 C 関数に渡されるときの処理を示しています。

表 48.5 *外部 C 引数に使用できる型*

関数プロトタイプ	SAS 変数の種類	C 変数の種類
short	numeric	short, short *, short **
short *	numeric, array	short, short *, short **
short **	array	short *, short **
int	numeric	int, int *, int **

関数プロトタイプ	SAS 変数の種類	C 変数の種類
int *	numeric, array	int, int *, int **
int **	array	int *, int **
long	numeric	long, long *, long **
long *	numeric, array	long, long *, long **
long **	array	long *, long **
double	numeric	double, double *, double **
double *	numeric, array	double, double *, double **
double **	array	double *, double **
char *	character	char *, char **
char **	character	char *, char **
struct *	structure	struct *, struct **
struct **	structure	struct *, struct **

注: 2 つの異なる C 型間の自動変換は実行されません。

PROC PROTO のパッケージのスコープ

PROC PROTO パッケージは CMPLIB システムオプションに指定された順序でロードされ、コンテンツがグローバルに使用されます。PROC ステートメントオプションによってロードされたパッケージは、スコープにおいてはローカルと見なされます。ローカル定義は最後にロードされ、いかなる場合でも、ローカルスコープによってグローバルスコープが上書きされます。

名前が一意でも重複していても、定義がロードされます。特定の PROC PROTO 要素の複数定義(列挙名や関数プロトタイプなど)によって、名前が競合し、エラーが発生する可能性があります。PROC PROTO パッケージ間の名前の競合を防ぐには、列挙される型や関数定義などの要素の名前を一意にするようにします。

次の例では、3 つの PROC PROTO パッケージをロードし、typedef および#define ステートメントの相互上書き順序が示されます。

例のこの部分では、最初の 2 つの PROC PROTO パッケージがロードされます。

```
proc proto package = work.p1.test1;
    typedef struct { int a; int b; } AB_t;
    #define NUM 1;
```

```
    int p1(void);
    externc p1;
        int p1(void)
        {
            return NUM;
        }
    externcend;
run;

proc proto package = work.p2.test2;
    typedef struct { int a; int b; } AB_t;
    #define NUM 2;
    int p2(void);
    externc p2;
        int p2(void)
        {
            return NUM;
        }
    externcend;
run;

options CMPLIB = (work.p1 work.p2);

proc fcmp;
   x = p1();
   put "Should be 2: " x=;
run;
```

パッケージは順序どおりにロードされるので、前述のプログラムの実行結果は 2 です。

次の例では、前述の CMPLIB=システムオプション設定を保持したまま、PROC PROTO で 3 番目のパッケージを追加し、PROC FCMP でローカルに含めます。

```
proc proto package = work.p3.test3;
    typedef struct { int a; int b; } AB_t;
    #define NUM 3;
    int p3(void);
    externc p3;
        int p3(void)
        {
            return NUM;
        }
    externcend;
run;

proc fcmp libname = work.p3;
   x = p1();
   put "Should be 3: " x=;
run;
```

この例では、*work.p3* の NUM のローカル定義が、*work.p1* と *work.p2* によってロードされるグローバル定義のかわりに使用されます。

C Helper 関数と CALL ルーチン

C Helper 関数と CALL ルーチンについて

PROC FCMP で C 言語構造体を処理するために、パッケージとあわせて複数の Helper 関数と CALL ルーチンが提供されます。ほとんどの C 言語構造体は、参照したり PROC FCMP で使用する前に、PROC PROTO によって作成されるカタログパッケージで定義しておく必要があります。ISNULL 関数ならびに STRUCTINDEX および SETNULL CALL ルーチンが追加されて、SAS 言語が拡張され、そのままでは SAS 言語に適応しない C 言語構造体を処理できるようになりました。

次の C Helper 関数と CALL ルーチンが使用可能です。

表 48.6 C Helper 関数と CALL ルーチン

C Helper 関数または CALL ルーチン	説明
“ISNULL C Helper 関数” (1467 ページ)	構造体のポインタ要素が NULL かどうかを決定します。
“SETNULL C Helper CALL ルーチン” (1468 ページ)	構造体のポインタ要素を NULL に設定します。
“STRUCTINDEX C Helper CALL ルーチン” (1468 ページ)	構造体の配列の各構造要素にアクセスできます。

ISNULL C Helper 関数

ISNULL 関数では、構造体のポインタ要素が NULL かどうかが決定されます。関数は次の形式を取ります。

double **ISNULL** (*pointer-element*);

Pointer-element はポインタ要素を指します。

次の例では、PROC PROTO を使用して、LINKLIST 構造と GET_LIST 関数が定義されます。GET_LIST 関数は、要求される数の要素を含むリンクリストを生成する外部 C ルーチンです。

```
struct linklist{
   double value;
   struct linklist * next;
};

struct linklist * get_list(int);
```

The following example shows how to use the ISNULL helper function to loop over the linked list that is created by the GET_LIST function:

```
struct linklist list;

list = get_list(3);
```

```
put list.value=;

do while (^isnull(list.next));
   list = list.next;
   put list.value=;
end;
```

ログ 48.1 *リンクされたリストのループ化*

```
LIST.value=0 LIST.value=1 LIST.value=2
```

SETNULL C Helper CALL ルーチン

SETNULL CALL ルーチンで、構造体のポインタ要素が null に設定されます。これは次の形式を取ります。

CALL SETNULL(*pointer-element*);

Pointer-element は構造へのポインタです。

ポインタ値のある変数(構造体エントリ)を指定すると、SETNULL でポインタが null に設定されます。

```
call setnull(12.next);
```

次の例では、PROC PROTO を使用して、“ISNULL C Helper 関数” (1467 ページ) で説明されているものと同じ LINKLIST 構造が定義されると仮定しています。SETNULL CALL ルーチンを使用すると、次の要素を null に設定できます。

```
proc proto;
   struct linklist list;
   call setnull(list.next);
run;
```

STRUCTINDEX C Helper CALL ルーチン

STRUCTINDEX CALL ルーチンを使用すると、構造体の配列の各構造要素にアクセスできます。構造体に構造体の配列が含まれている場合、STRUCTINDEX CALL ルーチンを使用すると、配列の各構造要素にアクセスできます。STRUCTINDEX CALL は次の形式を取ります。

CALL STRUCTINDEX(*struct_array*, *index*, *struct_element*);

Struct_array は配列を指定します。*index* は、SAS 配列で使用される、1 つから始まるインデックスです。また、*struct_element* は、配列の要素を指します。

次の例は 2 つの部分で構成されています。例の 2 つの部分をコピーして SAS エディタに貼り付け、それらを 1 つの SAS プログラムとして実行します。

この例の最初の部分では、PROC PROTO を使用して、次の構造体と関数が定義されます。

```
options cmplib=(work.proto_ds work.fcmp_ds);
proc proto package=work.proto_ds.cfcns;
     struct POINT {
           short s;
           int i;
           long l;
```

```
      double d;
};
struct POINT_ARRAY {
      int length;
      struct POINT * p;
      char name[32];
};
struct POINT * struct_array( int );

externc struct_array;
    struct POINT * struct_array( int num ) {
         return(malloc(sizeof(struct POINT) * num));
    }
externcend;
run;
```

この例の 2 番目の部分の PROC FCMP コードセグメントでは、STRUCTINDEX CALL ルーチンを使用して、POINT_ARRAY 構造で P と呼ばれる配列の各 POINT 構造要素を取得、設定する方法が示されます。

```
proc fcmp;
struct point_array pntarray;
struct point pnt;

   /* Call struct_array to allocate an array of 2 POINT structures. */
pntarray.p = struct_array(2);
pntarray.plen = 2;
pntarray.name = "My funny structure";

   /* Get each element using the STRUCTINDEX CALL routine and set values. */
do i = 1 to 2;
   call structindex(pntarray.p, i, pnt);
   put "Before setting the" i "element: " pnt=;
   pnt.s = 1;
   pnt.i = 2;
   pnt.l = 3;
   pnt.d = 4.5;
   put "After setting the" i "element: " pnt=;
end;

run;
```

ログ 48.2 *STRUCTINDEX CALL ルーチンの結果*

```
Before setting the 1 element:PNT {s=0, i=0, l=0, d=0} After setting the 1
element:PNT {s=1, i=2, l=3, d=4.5} Before setting the 2 element:PNT {s=0, i=0,
l=0, d=0} After setting the 2 element:PNT {s=1, i=2, l=3, d=4.5}
```

例: 分割関数の例

要素: INT ステートメント
KIND=プロトタイプ引数

他の要素: PROC FCMP

詳細

この例では、PROC PROTO を使用して、SPLIT と CASHFLOW という 2 つの外部 C 言語関数のプロトタイプを作成する方法を示します。これらの関数は、LINK ステートメントで指定された 2 つの共有ライブラリに含まれます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=40;

proc proto package =
sasuser.myfuncs.mathfun
           label = "package of math functions";


           link "link-library";
           link "link-library";

   int split(int x "number to split")
       label = "splitter function" kind=PRICING;

   int cashflow(double amt, double rate, int periods,
                double * flows / iotype=O)
       label = "cash flow function" kind=PRICING;

run;

proc fcmp libname=sasuser.myfuncs;
   array flows[20];
   a = split(32);
   put a;
   b = cashflow(1000, .07, 20, flows);
   put b;
   put flows;
run;

run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=は開始ページ番号を指定します。LINESIZE=は出力行の長さを指定します。PAGESIZE=は出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

関数パッケージ情報を保存するカタログエントリを指定します。 カタログエントリは 3 レベルの名前です。

```
proc proto package =
sasuser.myfuncs.mathfun
           label = "package of math functions";
```

SPLIT および CASHFLOW 関数を含むライブラリを指定します。 LINK ステートメントをさらに追加すると、プロトタイプに対して必要な数のライブラリを含められます。

```
            link "link-library";
            link "link-library";
```

SPLIT 関数のプロトタイプを作成します。 INT ステートメントで、SPLIT 関数のプロトタイプが作成され、その関数にラベルが割り当てられます。

```
      int split(int x "number to split")
          label = "splitter function" kind=PRICING;
```

CASHFLOW 関数のプロトタイプを作成します。 INT ステートメントで、CASHFLOW 関数のプロトタイプが作成され、その関数にラベルが割り当てられます。

```
      int cashflow(double amt, double rate, int periods,
                   double * flows / iotype=0)
          label = "cash flow function" kind=PRICING;
```

PROTO プロシジャを実行します。 RUN ステートメントで、PROTO プロシジャが実行されます。

```
   run;
```

SPLIT および CASHFLOW 関数を呼び出します。 PROC FCMP で、SPLIT および CASHFLOW 関数が呼び出されます。PROC FCMP からの出力が作成されます。

```
   proc fcmp libname=sasuser.myfuncs;
      array flows[20];
      a = split(32);
      put a;
      b = cashflow(1000, .07, 20, flows);
      put b;
      put flows;
   run;
```

FCMP プロシジャを実行します。 RUN ステートメントは、FCMP プロシジャを実行します。

```
   run;
```

出力:関数のプロトタイプ作成

アウトプット 48.1 *SPLIT 関数および CASHFLOW のプロトタイプ作成の結果*

```
                                          The SAS System 1 FCMP プロシジャ 16 12
70 105 128.33333333 145.83333333 159.83333333 171.5 181.5 190.25 198.02777778
205.02777778 211.39141414 217.22474747 222.60936286 227.60936286 232.27602953
236.65102953 240.76867658 244.65756547 248.341776 251.841776
```

49 章

PRTDEF プロシジャ

概要: PRTDEF プロシジャ

PRTDEF プロシジャでは、個々のユーザーかサイトのすべての SAS ユーザーのどちらかに対して、バッチモードでプリンタ定義が作成されます。システム管理者は、PROC PRTDEF で USESASHELP オプションを使用して、SAS レジストリにプリンタ定義を作成し、サイトのすべての SAS ユーザーがそのプリンタを使用できるようにします。個々のユーザーは、PROC PRTDEF を使用して、SAS レジストリに個人用プリンタ定義を作成します。

関連項目:

50 章, “PRTEXP プロシジャ,” (1489 ページ)

構文: PRTDEF プロシジャ

PROC PRTDEF <*option(s)*>;

ステートメント	タスク	例
“PROC PRTDEF ステートメント”	個々のユーザーかサイトのすべての SAS ユーザーのどちらかに対して、バッチモードでプリンタ定義を作成します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5

PROC PRTDEF ステートメント

バッチモードでプリンタ定義を作成します。

例: “例 1: 複数プリンタ定義の定義” (1480 ページ)
“例 2: SASUSER の PostScript プリンタ出力をプレビューするために、SASUSER に Ghostview プリンタを作成する” (1481 ページ)
“例 3: すべてのユーザーが使用可能な単一プリンタ定義の作成” (1483 ページ)
“例 4: プリンタ定義の追加、変更、削除” (1484 ページ)
“例 5: 単一プリンタ定義の削除” (1486 ページ)

構文

PROC PRTDEF <*option(s)*>;

オプション引数

DATA=*SAS-data-set*
プリンタ属性を含む SAS データセットを指定します。

要件 指定の必要なプリンタ属性変数は、DEST、DEVICE、MODEL、NAME ですが、変数 OPCODE の値が DELETE の場合は例外です。その場合は、NAME 変数のみ必須になります。

参照項目 “入力データセット:PRTDEF プロシジャ” (1475 ページ)

DELETE
デフォルト操作としてレジストリからプリンタ定義を削除することを指定します。

操作 DELETE と REPLACE の両方を指定すると、DELETE がデフォルト操作になります。

ヒント ユーザー定義のプリンタ定義を削除しても、SASHELP カタログに管理者定義のプリンタが存在する場合は、そのプリンタを表示できます。

例 “例 5: 単一プリンタ定義の削除” (1486 ページ)

FOREIGN
異なるホストにエクスポートするためにレジストリエントリを作成することを指定します。その結果として、いずれのホスト依存項目(TRANTAB など)のテストもスキップされます。

LIST
作成または置換するプリンタのリストをログに書き込むことを指定します。

例 “例 2: SASUSER の PostScript プリンタ出力をプレビューするために、SASUSER に Ghostview プリンタを作成する” (1481 ページ)

“例 4: プリンタ定義の追加、変更、削除” (1484 ページ)

REPLACE
デフォルト操作として既存のプリンタ定義を変更することを指定します。すでに存在するプリンタ名はすべて、プリンタ属性データセットの情報を使用して変更されます。存在しないプリンタ名はすべて追加されます。

操作 REPLACE と DELETE の両方を指定すると、DELETE が実行されます。

例 “例 2: SASUSER の PostScript プリンタ出力をプレビューするために、SASUSER に Ghostview プリンタを作成する” (1481 ページ)

USESASHELP
プリンタ定義を SASHELP ライブラリに置くことを指定します。ここでは、すべてのユーザーが定義を使用できます。USESASHELP オプションを指定しなければ、プリンタ定義は現在の SASUSER ライブラリに置かれます。ここでは、ローカルユーザーのみ定義を使用できます。

Windows 固有
Windows 動作環境で PROC PRTDEF を使用して、プリンタ定義を作成できます。ただし、Windows ではデフォルトでユニバーサル印刷が無効になっているため、これらのプリンタ定義は**印刷**ウィンドウに表示されません。ユニバーサル印刷が無効なときにプリンタ定義を使用する場合は、PRINTERPATH システムオプションの一部としてプリンタ定義を指定するか、または Output Delivery System (ODS)から、次のコードを発行します。

```
ODS PRINTER SAS PRINTER=myprinter;
```

この場合、*myprinter* は使用するプリンタ定義の名前になります。

制限事項 USESASHELP オプションを使用するには、SASHELP カタログへの書き込み権限が必要です。

例 “例 3: すべてのユーザーが使用可能な単一プリンタ定義の作成” (1483 ページ)

入力データセット:PRTDEF プロシジャ

有効変数の要約

プリンタ定義を作成するには、適切なプリンタ属性を含む変数を持つ SAS データセットを作成する必要があります。次の表では、このデータセットの必須変数とオプション変数の両方をリストにして説明を加えています。

表 49.1 *プリンタ定義レコード作成用の必須変数とオプション変数*

変数名	変数説明
必須	

変数名	変数説明
DEST	出力先
DEVICE	デバイス
MODEL	プロトタイプ
NAME	プリンタ名
任意	
BOTTOM	デフォルト下余白
CHARSET	デフォルトフォント文字セット
DESC	説明
FONTSIZE	デフォルトのフォントサイズ
HOSTOPT	ホストオプション
LEFT	デフォルト左余白
LRECL	出力バッファサイズ
OPCODE	操作コード
PAPERIN	用紙トレイまたは入力トレイ
PAPEROUT	給紙先または出力トレイ
PAPERSIZ	用紙のサイズ
PAPERTYP	用紙の種類
PREVIEW	プレビュー
PROTOCOL	プロトコル
RES	デフォルトプリンタ解像度
RIGHT	デフォルト右余白
STYLE	デフォルトのフォントスタイル
TOP	デフォルト上余白
TRANTAB	変換テーブル
TYPEFACE	デフォルトのフォント

変数名	変数説明
UNITS	CM 単位または IN 単位
VIEWER	ビューア
WEIGHT	デフォルトのフォントの太さ

必須変数

プリンタを作成または変更するには、NAME、MODEL、DEVICE、DEST 変数を指定する必要があります。その他のすべての変数では、MODEL 変数で指定されたプリンタプロトタイプのデフォルト値が使用されます。

プリンタを削除するには、必須の NAME 変数のみ指定します。

入力データセットでは、次の変数が必須です。

DEST
プリンタの出力先を指定します。

動作環境の情報
DEST では、一部のデバイスで大文字と小文字が区別されます。

制限事項 DEST は 1023 文字に制限されます。

DEVICE
プリンタへの出力送信時に使用する I/O デバイスの種類を指定します。有効なデバイスは、**プリンタ定義**ウィザードと SAS レジストリエディタでリストされます。

制限事項 DEVICE は 31 文字までにしてください。

MODEL
プリンタの定義時に使用するプリンタプロトタイプを指定します。

プロトタイプまたはモデルの説明の有効なリストについては、SAS レジストリエディタで CORE\PRINTING\PROTOTYPES の下を参照してください。

制限事項 MODEL は 127 文字までにしてください。

ヒント 対話モード中は、REGEDIT コマンドでレジストリを呼び出せます。

NAME
プリンタ定義のその他の属性に関連付けられるプリンタ定義名を指定します。

名前は、指定レジストリ内では一意です。新しいプリンタ定義に既存の名前が含まれている場合は、REPLACE オプションを指定しておくか、OPCODE 変数の値が `Modify` でなければ、レコードは処理されません。

制限事項 NAME は 127 文字までにしてください。また、ブランクではない文字が少なくとも 1 つ必要で、バックスラッシュは含められません。先頭と末尾のブランクは名前から切り捨てられます。

オプション変数

入力データセットでは、次の変数がオプションです。

BOTTOM
UNITS 変数で指定した単位でデフォルト下余白を指定します。

CHARSET
デフォルトフォント文字セットを指定します。

制限事項 値は、TYPEFACE 変数で指定したタイプフェイスの文字セット名のいずれかである必要があります。

DESC
プリンタの説明を指定します。

デフォルト DESC は、デフォルトで、プリンタの作成に使用するプロトタイプになります。

制限事項 説明には最大で 1023 文字まで使用できます。

FONTSIZE
デフォルトフォントのポイントサイズを指定します。

HOSTOPT
出力先の任意のホストオプションを指定します。ホストオプションの大文字と小文字は区別されません。

制限事項 ホストオプションには最大で 1023 文字まで使用できます。

LEFT
UNITS 変数で指定した単位でデフォルト左余白を指定します。

LRECL
プリンタへの出力送信時に使用するバッファサイズとレコード長を指定します。

デフォルト 既存プリンタの変更時に LRECL がゼロより小さい場合、プリンタのバッファサイズは、プリンタプロトタイプで指定するサイズにリセットされます。

OPCODE
プリンタ定義に対して実行するアクション(追加、削除または変更)を指定する文字変数です。

追加
レジストリに新しいプリンタ定義を作成します。REPLACE オプションを指定した場合、この操作でも既存のプリンタ定義が変更されます。

削除
既存のプリンタ定義をレジストリから削除します。

制限事項 この操作では、NAME 変数のみ定義が必要です。その他の変数は無視されます。

変更
レジストリの既存プリンタ定義を変更するか、新しく追加します。

制限事項 OPTCODE は 8 文字までにしてください。

ヒント ユーザーが SASHELP ライブラリのプリンタの新しい属性を変更、保存すると、その変更は SASUSER ライブラリに格納されます。ユーザーの指定した値が、管理者の指定した値より優先されますが、置き換えられることはありません。

PAPERIN
デフォルトの用紙トレイまたは入力トレイを指定します。

制限事項 PAPERIN の値は、MODEL 変数で指定したプリンタプロトタイプの用紙トレイ名のいずれかである必要があります。

PAPEROUT
デフォルトの給紙先または出力トレイを指定します。

制限事項 PAPEROUT の値は、MODEL 変数で指定したプリンタプロトタイプの給紙先名のいずれかである必要があります。

PAPERSIZ
デフォルトの用紙トレイまたは入力トレイを指定します。

制限事項 PAPERSIZ の値は、MODEL 変数で指定するプリンタプロトタイプにリストされた用紙のサイズ名のいずれかである必要があります。

PAPERTYP
デフォルトの用紙の種類を指定します。

制限事項 PAPERTYP の値は、MODEL 変数で指定するプリンタプロトタイプにリストされた用紙トレイ名のいずれかである必要があります。

PREVIEW
印刷プレビューに使用するプリンタアプリケーションを指定します。

制限事項 PREVIEW は 127 文字までにしてください。

PROTOCOL
プリンタへの出力送信時に使用する I/O プロトコルを指定します。

動作環境の情報
メインフレームシステムでは、プロトコルによって、メインフレームと ASCII デバイスを接続するプロトコルコンバータで処理可能な出力形式に出力を変換する方法が記述されます。

制限事項 PROTOCOL は 31 文字までにしてください。

RES
デフォルトプリンタ解像度を指定します。

制限事項 RES の値は、MODEL 変数で指定するプリンタプロトタイプで使用可能な解像度値のいずれかである必要があります。

RIGHT
UNITS 変数で指定した単位でデフォルト右余白を指定します。

STYLE
デフォルトのフォントのスタイルを指定します。

制限事項 STYLE の値は、TYPEFACE 変数で指定したタイプフェイスで使用可能なスタイルのいずれかである必要があります。

TOP
UNITS 変数で指定した単位でデフォルト上余白を指定します。

TRANTAB
プリンタへの出力送信時に使用する変換テーブルを指定します。

動作環境の情報
変換テーブルは、EBCDIC ホストでデータが ASCII デバイスに送信される際に必要です。

制限事項 TRANTAB は 8 文字までにしてください。

TYPEFACE
デフォルトフォントのタイプフェイスを指定します。

制限事項 タイプフェイスは、MODEL 変数で指定したプリンタプロトタイプで使用可能なタイプフェイス名のいずれかである必要があります。

UNITS
余白変数で指定した単位 CM または IN を指定します。

VIEWER
印刷プレビュー中に使用するホストシステムコマンドを指定します。その結果として、PROC PRTDEF によりプレビュープリンタが作成されます。

プレビュープリンタは、印刷前の画面でプリンタ出力を表示するために使用する特別プリンタです。

制限事項 VIEWER は 127 文字までにしてください。

ヒント VIEWER の値が指定されていると、PREVIEW、PROTOCOL、DEST、HOSTOPT 変数の値は無視されます。ビューアコマンドで、通常は入力ファイル名を指定する場所に*%s* を置きます。*%s* は必要な数だけ使用できます。

WEIGHT
デフォルトのフォントの太さを指定します。

制限事項 値は、TYPEFACE 変数で指定したタイプフェイスの有効な太さのいずれかである必要があります。

例: PRTDEF プロシジャ

例 1: 複数プリンタ定義の定義

要素: PROC PRTDEF ステートメントオプション
DATA=

詳細

この例では、各種プリンタの設定方法を示します。

プログラム

```
data printers;
input name $ 1-14 model $ 16-42 device $ 46-53 dest $ 57-70;
datalines;
Myprinter      PostScript Level 1 (Color)    PRINTER   printer1
Laserjet       PCL 5 (DeltaRow)              PIPE      lp -dprinter5
Color LaserJet PostScript Level 2 (Color)    PIPE      lp -dprinter2
;

proc prtdef data=printers;
run;
```

プログラムの説明

PRINTERS データセットを作成します。 INPUT ステートメントには、4 つの必須変数の名前が含まれます。各データ行に、1 つのプリンタ定義を作成するのに必要な情報が含まれます。

```
data printers;
input name $ 1-14 model $ 16-42 device $ 46-53 dest $ 57-70;
datalines;
Myprinter      PostScript Level 1 (Color)    PRINTER   printer1
Laserjet       PCL 5 (DeltaRow)              PIPE      lp -dprinter5
Color LaserJet PostScript Level 2 (Color)    PIPE      lp -dprinter2
;
```

プリンタ属性を含む入力データセットを指定し、プリンタ定義を作成します。 PROC PRTDEF で、SAS レジストリに対してプリンタ定義が作成され、DATA=オプションで、プリンタ属性を含む入力データセットとして PRINTERS が指定されます。

```
proc prtdef data=printers;
run;
```

ログ

ログ 49.1 *プリンタ定義後の SAS ログ*

```
1    data printers; 2    input name $ 1-14 model $ 16-42 device $ 46-53 dest
$ 57-70; 3    datalines; NOTE:The data set WORK.PRINTERS has 3 observations and
4 variables.NOTE:DATA statement used (Total process time): real time
0.03 seconds cpu time            0.03 seconds 7    ; 8    proc prtdef
data=printers; 9    run; NOTE:3 printer definitions added to the registry.NOTE:0
printer definitions modified in the registry.NOTE:0 printer definitions deleted
from the registry.NOTE:PROCEDURE PRTDEF used (Total process time): real
time           0.15 seconds cpu time            0.01 seconds
```

例 2: SASUSER の PostScript プリンタ出力をプレビューするために、SASUSER に Ghostview プリンタを作成する

要素: PROC PRTDEF ステートメントオプション
- DATA=
- LIST
- REPLACE

詳細

この例では、PostScript 出力をプレビューするために、SASUSER ライブラリに Ghostview プリンタ定義を作成します。

プログラム

```
data gsview;
   name = "Ghostview";
   desc = "Print Preview with Ghostview";
   model= "PostScript Level 2 (Color)";
   viewer = 'ghostview %s';
   device = "Dummy";
   dest = " ";
run;

proc prtdef data=gsview list replace;
run;
```

プログラムの説明

GSVIEW データセットを作成し、プリンタ名、プリンタ説明、プリンタプロトタイプ、印刷プレビューに使用するコマンドを指定します。 GSVIEW データセットに含まれる変数の値には、プリンタ定義の作成に必要な情報が含まれます。NAME 変数では、プリンタ定義データレコードのその他の属性に関連付けられるプリンタ名が指定されます。DESC 変数では、プリンタの説明が指定されます。MODEL 変数では、このプリンタの定義時に使用するプリンタプロトタイプが指定されます。VIEWER 変数では、印刷プレビューに使用するホストシステムコマンドが指定されます。GSVIEW をシステムにインストールして、VIEWER の値にそれを見つけるパスを含める必要があります。`%s` なので、値を単一引用符で囲む必要があります。二重引用符を使用した場合、SAS では、`%s` がマクロ変数と見なされます。DEVICE と DEST は必須変数ですが、この例では値は必要ありません。したがって、“dummy”またはブランク値を割り当ててください。

```
data gsview;
   name = "Ghostview";
   desc = "Print Preview with Ghostview";
   model= "PostScript Level 2 (Color)";
   viewer = 'ghostview %s';
   device = "Dummy";
   dest = " ";
run;
```

プリンタ属性を含む入力データセットを指定し、プリンタ定義を作成し、そのプリンタ定義を SAS ログに書き込み、SAS レジストリの印刷定義を置き換えます。 DATA=オプションでは、プリンタ属性を含む入力データセットとして GSVIEW が指定されます。PROC PRTDEF では、プリンタ定義が作成されます。LIST オプションでは、作成または置換されるプリンタのリストを SAS ログに書き込むことが指定されます。REPLACE オプションでは、プリンタ定義名がすでにレジストリにある名前と一致する場合、そのプリンタ定義とレジストリのプリンタ定義を置き換えることが指定されます。プリンタ定義名が一致しない場合は、新しいプリンタ定義がレジストリに追加されます。

```
proc prtdef data=gsview list replace;
run;
```

ログ

ログ 49.2 GhostView プリンタ定義後の SAS ログ

```
10   data gsview; 11   name = "Ghostview"; 12   desc = "Print Preview with
Ghostview"; 13   model= "PostScript Level 2 (Color)"; 14   viewer = 'ghostview
%s'; 15   device = "Dummy"; 16   dest = " "; NOTE:The data set WORK.GSVIEW has 1
observations and 6 variables.NOTE:DATA statement used (Total process time): real
time           0.00 seconds cpu time            0.00 seconds 17   proc prtdef
data=gsview list replace; 18   run; NOTE:Printer Ghostview created.NOTE:1
printer definitions added to the registry.NOTE:0 printer definitions modified in
the registry.NOTE:0 printer definitions deleted from the registry.NOTE:PROCEDURE
PRTDEF used (Total process time): real time           0.01 seconds cpu
time            0.01 seconds
```

例 3: すべてのユーザーが使用可能な単一プリンタ定義の作成

要素: PROC PRTDEF ステートメントオプション
DATA=
USESASHELP

制限事項: USESASHELP オプションを使用するには、SASHELP カタログへの書き込み権限が必要です。

詳細

この例では、次を指定して、Tektronix Phaser 780 プリンタおよび Ghostview 印刷プレビューアの定義を作成します。

- 下余白を 1 インチに設定
- フォントサイズを 14 ポイントに設定
- 用紙のサイズを A4 に設定

プログラム

```
data tek780;
   name = "Tek780";
   desc = "Test Lab Phaser 780P";
   model = "Tek Phaser 780 Plus";
   device = "PRINTER";
   dest = "testlab3";
   preview = "Ghostview";
   units = "cm";
   bottom = 2.5;
   fontsize = 14;
   papersiz = "ISO A4";
run;

proc prtdef data=tek780 usesashelp;
run;
```

プログラムの説明

TEK780 データセットを作成し、プリンタ出力先について適切な情報を指定します。 TEK780 データセットに含まれる変数の値には、プリンタ定義の作成に必要な情報が含まれます。この例では、割り当てステートメントを使用してこれらの変数を割り当てます。NAME 変数では、プリンタ定義データレコードのその他の属性に関連付けられるプリンタ名が指定されます。DESC 変数では、プリンタの説明が指定されます。MODEL 変数では、このプリンタの定義時に使用するプリンタプロトタイプが指定されます。DEVICE 変数では、プリンタへの出力送信時に使用する I/O デバイスの種類が指定されます。DEST 変数では、プリンタの出力先が指定されます。PREVIEW 変数では、印刷プレビューに使用するプリンタが指定されます。UNITS 変数では、余白変数をセンチとインチのどちらで測定するかが指定されます。BOTTOM 変数では、UNITS 変数で指定した単位でデフォルト下余白が指定されます。FONTSIZE 変数では、デフォルトフォントのポイントサイズが指定されます。PAPERSIZ 変数では、デフォルトの用紙のサイズが指定されます。

```
data tek780;
   name = "Tek780";
   desc = "Test Lab Phaser 780P";
   model = "Tek Phaser 780 Plus";
   device = "PRINTER";
   dest = "testlab3";
   preview = "Ghostview";
   units = "cm";
   bottom = 2.5;
   fontsize = 14;
   papersiz = "ISO A4";
run;
```

TEK780 プリンタ定義を作成し、すべてのユーザーが定義を使用できるようにします。 DATA= オプションでは、TEK780 が入力データセットとして指定されます。USESASHELP オプションでは、すべてのユーザーがプリンタ定義を使用できるようにすることが指定されます。

```
proc prtdef data=tek780 usesashelp;
run;
```

例 4: プリンタ定義の追加、変更、削除

要素: PROC PRTDEF ステートメントオプション
DATA=
LIST

詳細

この例では、次を行います。

- プリンタ定義を 2 つ追加
- プリンタ定義を変更
- プリンタ定義を 2 つ削除

プログラム

```
data printers;
length name   $ 80
   model  $ 80
   device $ 8
   dest   $ 80
   opcode $ 3
   ;
input opcode $& name $& model $& device $& dest $&;
datalines;
add  Color PostScript   PostScript Level 2 (Color)       DISK    sasprt.ps
mod  LaserJet 5         PCL 5 (DeltaRow)                DISK    sasprt.pcl
del  Gray Postscript    PostScript Level 1 (Gray Scale)  DISK    sasprt.ps
del  test               PostScript Level 2 (Color)       DISK    sasprt.ps
add  ColorPS            PostScript Level 2 (Color)       DISK    sasprt.ps
;

proc prtdef data=printers replace list;
run;
```

プログラムの説明

PRINTERS データセットを作成し、プリンタ定義に対して実行するアクションを指定します。 PRINTERS データセットに含まれる変数の値には、プリンタ定義の作成に必要な情報が含まれます。MODEL 変数では、このプリンタの定義時に使用するプリンタプロトタイプが指定されます。DEVICE 変数では、プリンタへの出力送信時に使用する I/O デバイスの種類が指定されます。DEST 変数では、プリンタの出力先が指定されます。OPCODE 変数では、プリンタ定義に対して実行するアクション(追加、削除または変更)が指定されます。最初の Add 操作では、SAS レジストリの Color PostScript に対して新しいプリンタ定義が作成されます。2 番目の Add 操作では、SAS レジストリの ColorPS に対して新しいプリンタ定義が作成されます。Mod 操作では、レジストリの LaserJet 5 に対する既存プリンタ定義が変更されます。Del 操作では、プリンタ定義をレジストリから削除します。&では、2 つ以上のブランクで文字値を区切ることが指定されます。これにより、name と model の値にブランクを含めることが可能になります。

```
data printers;
length name   $ 80
   model  $ 80
   device $ 8
   dest   $ 80
   opcode $ 3
   ;
input opcode $& name $& model $& device $& dest $&;
datalines;
add  Color PostScript   PostScript Level 2 (Color)       DISK    sasprt.ps
mod  LaserJet 5         PCL 5 (DeltaRow)                DISK    sasprt.pcl
del  Gray Postscript    PostScript Level 1 (Gray Scale)  DISK    sasprt.ps
del  test               PostScript Level 2 (Color)       DISK    sasprt.ps
add  ColorPS            PostScript Level 2 (Color)       DISK    sasprt.ps
;
```

複数のプリンタ定義を作成し、SAS ログに書き込みます。 DATA=オプションでは、プリンタ属性を含む入力データセット PRINTERS が指定されます。PROC PRTDEF では、5 つのプリンタ定義が作成され、そのうち 2 つが削除されました。LIST オプションでは、作成または置換されるプリンタのリストをログに書き込むことが指定されます。

```
proc prtdef data=printers replace list;
run;
```

LOG

ログ 49.3 *プリンタ変更および削除後の SAS ログ*

```
15   data printers; 16   length name   $ 80 17      model  $ 80 18      device
$ 8 19      dest   $ 80 20      opcode $ 3 21      ; 22   input opcode $& name
$& model $& device $& dest $&; 23   datalines; NOTE:The data set WORK.PRINTERS
has 5 observations and 5 variables.NOTE:DATA statement used (Total process
time): real time           0.01 seconds cpu time            0.01 seconds 29   ;
30   proc prtdef data=printers list replace; 31   run; NOTE:Printer Color
PostScript modified.NOTE:Printer LaserJet 5 modified.NOTE:Printer Gray
Postscript deleted.NOTE:Printer test deleted.NOTE:Printer ColorPS
modified.NOTE:PROCEDURE PRTDEF used (Total process time): real time
0.04 seconds cpu time            0.03 seconds
```

例 5: 単一プリンタ定義の削除

要素: PROC PRTDEF ステートメントオプション
DELETE

詳細

この例では、レジストリからプリンタを削除する方法を示します。

プログラム

```
data deleteprt;
name='printer1';
run;

proc prtdef data=deleteprt delete list;
run;
```

プログラムの説明

DELETEPRT データセットを作成します。 NAME 変数には、削除するプリンタの名前が含まれます。

```
data deleteprt;
name='printer1';
run;
```

レジストリからプリンタ定義を削除し、削除したプリンタをログに書き込みます。 DATA=オプションで、DELETEPRT が入力データセットとして指定されます。PROC PRTDEF で、SAS レジストリに対してプリンタ定義が作成されます。DELETE で、プリンタの削除が指定されます。LIST で、削除したプリンタをログに書き込むことが指定されます。

```
proc prtdef data=deleteprt delete list;
run;
```

ログ

ログ 49.4 *単一プリンタ削除後の SAS ログ*

```
45   data deleteprt; 46      name='printer1'; 47      run; NOTE:The data set
WORK.DELETEPRT has 1 observations and 1 variables.NOTE:DATA statement used
(Total process time): real time           0.01 seconds cpu time            0.00
seconds 48      proc prtdef data=deleteprt delete list; 49      run;
NOTE:Printer printer1 deleted.NOTE:0 printer definitions added to the
registry.NOTE:0 printer definitions modified in the registry.NOTE:1 printer
definitions deleted from the registry.NOTE:PROCEDURE PRTDEF used (Total process
time): real time           0.00 seconds cpu time            0.00 seconds
```

50 章

PRTEXP プロシジャ

概要: PRTEXP プロシジャ

PRTEXP プロシジャを使用すると、複製および変更のために SAS レジストリからプリンタ属性を抽出できます。次に、PROC PRTEXP で、その属性が SAS ログか SAS データセットに書き込まれます。PROC PRTEXP によってレジストリの SASHELP 部分または SAS レジストリ全体でこれらの属性を検索することを指定できます。

関連項目:

49 章, “PRTDEF プロシジャ,” (1473 ページ)

概念 PRTEXP プロシジャ

PRTEXP プロシジャは、PRTDEF プロシジャとあわせて使用すると、個々のユーザーかサイトのすべての SAS ユーザーのどちらかに対するプリンタ定義を複製、変更、作成できます。PROC PRTEXP では、プリンタ定義の作成に使用する属性のみをレジストリから抽出できます。これを SAS データセットに書き込むと、後で複製、変更できます。次に、PROC PRTDEF を使用して、入力データセットから、SAS レジストリにプリンタ定義を作成できます。PROC PRTDEF ならびにプリンタ定義の作成に使用する変数および属性の完全な説明については、“入力データセット:PRTDEF プロシジャ” (1475 ページ)を参照してください。

構文: PRTEXP プロシジャ

ヒント: SELECT ステートメントも EXCLUDE ステートメントも使用しなければ、すべてのプリンタが出力に含まれます。

PROC PRTEXP <*option(s)*>;
SELECT *printer(s)*;
EXCLUDE *printer(s)*

ステートメント	タスク	例
“PROC PRTEXP ステートメント”	SAS レジストリからのプリンタ属性の取得	Ex. 1, Ex. 2
“EXCLUDE ステートメント”	指定プリンタ以外のすべてのプリンタのプリンタ属性の取得	
“SELECT ステートメント”	指定プリンタのプリンタ属性の取得	Ex. 1, Ex. 2

PROC PRTEXP ステートメント

プリンタ定義を複製、変更、作成します。

例: “例 1: SAS ログへの属性の書き込み” (1491 ページ)
“例 2: SAS データセットへの属性の書き込み” (1492 ページ)

構文

PROC PRTEXP <*option(s)*>;

オプション引数

USESASHELP
SAS でレジストリの SASHELP 部分でのみプリンタ定義を検索することを指定します。

デフォルト デフォルトでは、レジストリの SASUSER と SASHELP の両部分でプリンタ定義が検索されます。

OUT=*SAS-data-set*
プリンタ定義を含む SAS データセットを指定します。

OUT=*SAS-data-set* オプションで指定したデータセットは、各プリンタを定義するために PROC PRTDEF の DATA=*SAS-data-set* オプションで指定したデータセットと同じ種類です。

デフォルト OUT=*SAS-data-set* を指定しなければ、各プリンタの定義に必要なデータは SAS ログに書き込まれます。

EXCLUDE ステートメント

出力に情報を表示しないプリンタの名前を指定します。

構文

EXCLUDE *printer(s)*;

必須引数

printer(s)
出力に関連情報を含めないプリンタを 1 つ以上指定します。

SELECT ステートメント

出力に情報を含めるプリンタの名前を指定します。

例: “例 1: SAS ログへの属性の書き込み” (1491 ページ)
“例 2: SAS データセットへの属性の書き込み” (1492 ページ)

構文

SELECT *printer(s)*;

必須引数

printer(s)
出力に関連情報を含めるプリンタを 1 つ以上指定します。

例: PRTEXP プロシジャ

例 1: SAS ログへの属性の書き込み

要素: PROC PRTEXP ステートメントオプション
USESASHELP
SELECT ステートメント

詳細

この例では、プリンタの定義に使用する属性を SAS ログに書き込む方法を示します。

プログラム

```
proc prtexp usesashelp;
select postscript;
run;
```

プログラムの説明

関連情報が必要なプリンタを指定し、レジストリの SASHELP 部分のみ検索することを指定して、SAS ログに情報を書き込みます。 SELECT ステートメントで、プリンタ PostScript の定義に使用する属性情報を出力に含めることが指定されます。USESASHELP オプションで、SASHELP レジストリでのみ PostScript のプリンタ定義を検索することが指定されます。OUT=オプションによる SAS データセットの指定を行わなかったので、各プリンタの定義に必要なデータは SAS ログに書き込まれます。

```
proc prtexp usesashelp;
select postscript;
run;
```

LOG

ログ 50.1 *レジストリの SASHELP 部分からプリンタ情報抽出後の SAS ログ*

```
379  proc prtexp usesashelp; 380  select postscript; 381  run; NAME:
PostScript MODEL:    PostScript Level 1 (Color) DEVICE:   DISK DEST:
sasprt.ps HOSTOPT:PROTOCOL:TRANTAB:DESC:     Generic PostScript Level 1 Printer
PREVIEW:Adobe Reader VIEWER:PAPERSIZ:PAPERTYP:PAPERIN:PAPEROUT:RES:      300 DPI
TOP:      0.50 LEFT:     0.50 RIGHT:    0.50 BOTTOM:   0.50 UNITS:    IN
TYPEFACE:<MTmonospace> WEIGHT:   Normal STYLE:    Regular CHARSET:Western
FONTSIZE:8.00 LRECL:    .
```

例 2: SAS データセットへの属性の書き込み

要素: PROC PRTEXP ステートメントオプション
OUT=
SELECT ステートメント

詳細

この例では、PROC PRTDEF がプリンタ PCL4、PCL5、PCL5E、PCLC の定義に使用するデータを含む SAS データセットを作成する方法を示します。

プログラム

```
proc prtexp out=PRDVTER;
   select pcl4 pcl5 pcl5e pcl5c;
run;

proc print data=prdvter;
run;
```

プログラムの説明

関連情報が必要なプリンタを指定し、PRDVTER データセットを作成します。 SELECT ステートメントで、プリンタ PCL4、PCL5、PCL5E、PCLC が指定されます。OUT=オプションで、SAS データセット PRDVTER が作成されます。これには、PROC PRTDEF がプリンタ PCL4、PCL5、PCL5E、PCLC の定義に使用するものと同じ属性が含まれます。USESASHELP を指定しなかったので、SAS では SASUSER と SASHELP の両レジストリが検索されます。

```
proc prtexp out=PRDVTER;
   select pcl4 pcl5 pcl5e pcl5c;
run;

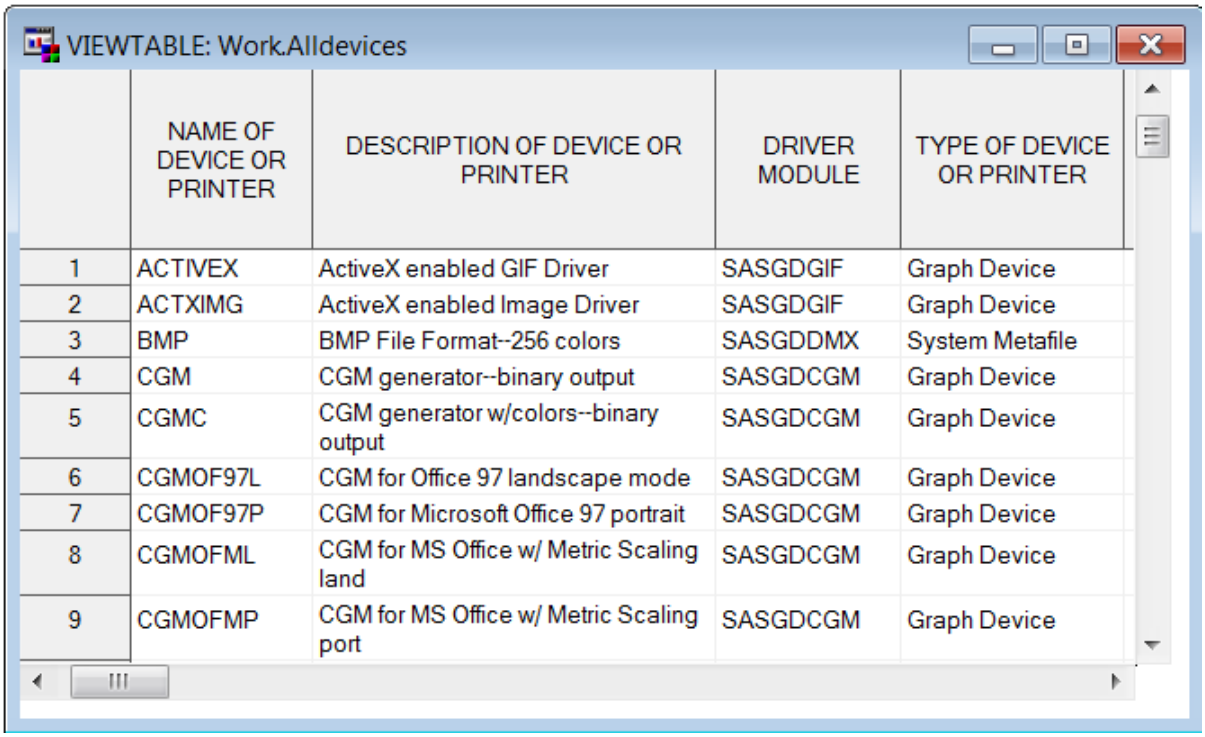
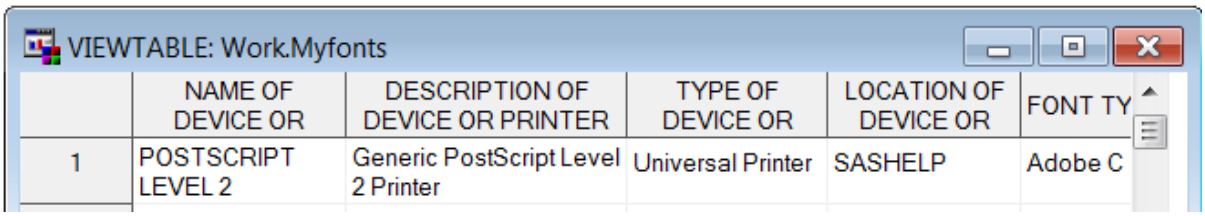
proc print data=prdvter;
run;
```

出力

次のデータセットは 26 の変数および 4 つのオブザベーションを含む Prdter データの表示の一部です。

アウトプット 50.1 *Prdvter の出力データセット*

The SAS System

Obs	DEST	HOSTOPT	DESC	VIEWER	NAME	MODEL	PREVIEW	TYPEFACE	I
1	sasprt.pcl		Generic PCL 4 Printer		PCL4	PCL 4	Adobe Reader		
2	sasprt.pcl		Generic PCL 5 Printer		PCL5	PCL 5 (DeltaRow)	Adobe Reader		
3	sasprt.pcl		Generic PCL 5 RGB Color Printer with Alpha Blending		PCL5c	PCL 5rgba (DeltaRow)	Adobe Reader		
4	sasprt.pcl		Generic PCL 5e Printer		PCL5e	PCL 5e (DeltaRow)	Adobe Reader		

51 章

PWENCODE プロシジャ

概要: PWENCODE プロシジャ

PWENCODE プロシジャを使用すると、パスワードをエンコードできます。エンコードされたパスワードは、リレーショナルデータベース管理システム(RDBMS)および各種サーバーにアクセスする SAS プログラムで、プレーンテキストパスワードのかわりに使用できます。各種サーバーとは、SAS/CONNECT サーバー、SAS/SHARE サーバー、SAS 統合オブジェクトモデル (IOM)サーバー(SAS メタデータサーバーなど)などです。

概念: PWENCODE プロシジャ

SAS プログラムでエンコードされたパスワードを使用する

PROC PWENCODE でパスワードをエンコードすると、文字列がエンコードされていることを示すタグが出力文字列に挿入されます。タグの例は{`sas001`}です。タグでは、エンコーディング方式が示されます。SAS サーバーと SAS/ACCESS エンジンでタグが認識され、使用前に文字列がデコードされます。パスワードをエンコードすると、プレーンテキストでパスワードを指定しなくても、SAS プログラムを書けます。

注: これには、英数字、スペースおよび特殊文字が含まれます。PROC PWENCODE パスワードには最大で 512 文字まで使用できます。ただし、データセットパスワード

は SAS 命名規則に従うものとします。SAS 命名規則の詳細については、“Rules for Most SAS Names” (*SAS Language Reference: Concepts*)を参照してください。

エンコードされたパスワードが SAS ログにプレーンテキストで書き込まれることはありません。かわりに、SAS ログではパスワードの各文字が X におき換えられます。

エンコーディングと暗号化

PROC PWENCODE では、エンコードによってパスワードが隠されます。エンコードを行うと、一部のテーブルルックアップ形式によって、ある文字セットが別の文字セットに変換されます。それとは対照的に、暗号化では、演算処理および(通常は)“key”値の使用による、ある形式から別の形式へのデータ変換が必要になります。一般に、暗号化はエンコードよりも解読が困難です。PROC PWENCODE は、悪意のない偶然のパスワード参照を防止するためのものです。PROC PWENCODE に頼っていては、データセキュリティのニーズをすべて満たすことはできません。覚悟と知識のある攻撃者であれば、エンコードされたパスワードのデコードが可能です。

構文: PWENCODE プロシジャ

PROC PWENCODE IN='*password*' <OUT=*fileref*> <METHOD=*encoding-method*>;

ステートメント	タスク	例
“PROC PWENCODE ステートメント”	パスワードのエンコード	Ex. 1, Ex. 2, Ex. 3, Ex. 4

PROC PWENCODE ステートメント

パスワードをエンコードします。

例:
“例 1: パスワードのエンコード” (1498 ページ)
“例 2: SAS プログラムでエンコードされたパスワードを使用する” (1499 ページ)
“例 3: エンコードされたパスワードのペーストバッファへの保存” (1500 ページ)
“例 4: パスワードのエンコードに Method=SAS003 を指定” (1501 ページ)

構文

PROC PWENCODE IN='*password*' <OUT=*fileref*> <METHOD=*encoding-method*>;

必須引数

IN='*password*'
エンコードするパスワードを指定します。パスワードには最大で 512 文字まで使用できます。これには、英数字、スペースおよび特殊文字が含まれます。

注: データセットパスワードは SAS 命名規則に従うものとします。SAS 命名規則を順守するものであれば、IN=*password* は SAS データセットにも使用できま

す。SAS 命名規則の詳細については、“Rules for Most SAS Names” (*SAS Language Reference: Concepts*)を参照してください。

パスワードに埋め込み単一引用符または二重引用符が含まれている場合は、文字定数引用の標準 SAS ルールを使用します。このルールについては、*SAS 言語リファレンス: 解説編*の式の SAS 定数の章を参照してください。

注: エンコードされたパスワードの各文字は、SAS ログへの書き込み時に、X に置き換えられます。

参照項目
“例 1: パスワードのエンコード” (1498 ページ)

“例 2: SAS プログラムでエンコードされたパスワードを使用する” (1499 ページ)

“例 3: エンコードされたパスワードのペーストバッファへの保存” (1500 ページ)

オプション引数

OUT=*fileref*
出力文字列の書き込み先のファイル参照名を指定します。OUT=オプションを指定しなければ、出力文字列は SAS ログに書き込まれます。

注: グローバルマクロ変数

`_PWENCODE`

は、OUT=ファイル参照名に書き込まれた値、または SAS ログに表示された値に設定されます。

参照項目 “例 2: SAS プログラムでエンコードされたパスワードを使用する” (1499 ページ)

METHOD=*encoding-method*
エンコーディング方式を指定します。*encoding-method* としてサポートされている値は次のとおりです。

表 51.1 *サポートされているエンコーディングおよび暗号化方式*

エンコーディング方式	説明	サポートされているデータ暗号化アルゴリズム
`sas001`	Base64 を使用してパスワードをエンコードします。	なし
`sas002`、(`sasenc` でも指定可)	32 ビットキーを使用してパスワードをエンコードします。	SAS ソフトウェアに含まれている SAS プロパティ。
`sas003`	256 ビットキー＋ 16 ビットソルトを使用してパスワードをエンコードします。	SAS/SECURE でサポートされている AES (Advanced Encryption Standard)。
`sas004`	256 ビットキー 64 ビットソルトを使用してパスワードをエンコードします。	SAS/SECURE でサポートされている AES (Advanced Encryption Standard)。

注: SAS/SECURE は、業界標準暗号化やハッシング法の使用を通じたデータ保護を可能にする製品です。詳細については、*SAS の暗号化*の"SAS/SECURE Software Availability"を参照してください。

METHOD=オプションを指定しない場合、デフォルトのエンコーディング方式が使用されます。FIPS 140-2 準拠オプションである-encryptfips を指定する場合、デフォルトのエンコーディング方式は **sas003** です。その他の場合はすべて、エンコーディング方式 **sas002** がデフォルト方式として使用されます。

注: METHOD=オプションは、SAS/SECURE がある場合に限り、値 SAS003 と値 SAS004 をサポートします。

SAS003 および SAS004 のエンコードされたパスワードは、256 ビットキー 64 ビットソルト値で構成されています。ソルト値はランダムです。そのため、PROC PWENCODE を使用して同じパスワードをエンコードするたびにソルト値は異なり、エンコードされたパスワードも異なります。

例: PWENCODE プロシジャ

例 1: パスワードのエンコード

要素: IN=引数

詳細

この例では、パスワードをエンコードし、エンコードされたパスワードを SAS ログに書き込む単純なケースを示します。

プログラム

```
proc pwencode in='my password';
run;
```

プログラムの説明

パスワードをエンコードします。

```
proc pwencode in='my password';
run;
```

ログ

SAS ログではパスワードの各文字が X に置き換えられることに注意してください。

```
19   proc pwencode in=XXXXXXXXXXXXX; 20   run;
{SAS002}DBCC571245AD0B31433834F80BD2B99E16B3C969 NOTE:PROCEDURE PWENCODE used
(Total process time): real time           0.01 seconds cpu time            0.01
seconds
```

例 2: SAS プログラムでエンコードされたパスワードを使用する

要素: IN=引数
OUT=オプション

詳細

この例では、次について説明します。

- パスワードのエンコードと外部ファイルへの保存
- DATA ステップによるエンコードされたパスワードの読み取り、マクロ変数での格納、および SAS/ACCESS LIBNAME ステートメントでの使用

プログラム 1:パスワードをエンコード

```
filename pwfile
'external-filename';

proc pwencode in='mypass1' out=pwfile;
run;
```

プログラムの説明

ファイル参照名を宣言します。

```
filename pwfile
'external-filename';
```

パスワードをエンコードし、外部ファイルに書き込みます。 OUT=オプションでは、エンコードされたパスワードの書き込み先の外部ファイル参照名が指定されます。

```
proc pwencode in='mypass1' out=pwfile;
run;
```

プログラム 2:エンコードされたパスワードの使用

```
filename pwfile
'external-filename';

options symbolgen;

data _null_;
infile pwfile truncover;
input line :$50.;
call symputx('dbpass',line);
run;

libname x odbc dsn=SQLServer user=testuser password="&dbpass";
```

プログラムの説明

エンコードパスワードファイルのファイル参照名を宣言します。

```
filename pwfile
```

```
'external-filename';
```

SYMBOLGEN SAS システムオプションを設定します。 このステップから、エンコードされたパスワードを含むマクロ変数が SAS ログで解決されても、実際のパスワードを明示できないことがわかります。このステップは、プログラムを適切に機能させるために必要というわけではありません。

```
options symbolgen;
```

ファイルを読み取り、エンコードされたパスワードをマクロ変数に格納します。 DATA ステップでは、エンコードされたパスワードがマクロ変数 DBPASS に格納されます。

```
data _null_;
infile pwfile truncover;
input line :$50.;
call symputx('dbpass',line);
run;
```

エンコードされたパスワードを使用して DBMS にアクセスします。 マクロ変数が適切に解決するように、二重引用符(“ ”)を使用してください。

```
libname x odbc dsn=SQLServer user=testuser password="&dbpass";
```

ログ

```
1    filename pwfile 'external-filename'; 2    options symbolgen; 3    data
_null_; 4    infile pwfile truncover; 5    input line :$50.; 6    call
symputx('dbpass',line); 7    run; NOTE:The infile PWFILE is:Filename=external-
filename RECFM=V,LRECL=256,File Size (bytes)=4, Last
Modified=12Apr2012:13:23:49, Create Time=12Apr2012:13:23:39 NOTE:1 record was
read from the infile PWFILE.The minimum record length was 4.The maximum record
length was 4.NOTE:DATA statement used (Total process time): real time
0.57 seconds cpu time            0.04 seconds 8 9   libname x odbc
SYMBOLGEN:Macro variable DBPASS resolves to {sas002}bXlwYXNzMQ==
9 !                dsn=SQLServer user=testuser password="&dbpass"; NOTE:Libref X
was successfully assigned as follows:Engine:        ODBC Physical Name:SQLServer
```

例 3: エンコードされたパスワードのペーストバッファへの保存

要素: IN=引数
OUT=オプション

他の要素: CLIPBRD アクセスメソッドを指定した FILENAME ステートメント

DETAILS

この例では、エンコードされたパスワードをペーストバッファに保存します。その場合は、エンコードされたパスワードを、別の SAS プログラムや、認証ダイアログボックスのパスワードフィールドに貼り付けられます。

プログラム

```
filename clip clipbrd;

proc pwencode in='my password' out=clip;
```

```
run;
```

プログラムの説明

CLIPBRD アクセスメソッドを使用してファイル参照名を宣言します。

```
filename clip clipbrd;
```

パスワードをエンコードし、ペーストバッファに保存します。 OUT=オプションによって、エンコードされたパスワードが、前のステートメントで宣言したファイル参照名に保存されます。

```
proc pwencode in='my password' out=clip;
run;
```

ログ

SAS ログではパスワードの各文字が X に置き換えられることに注意してください。

```
24 25   filename clip clipbrd; 26     proc pwencode in=XXXXXXXXXXXXX out=clip;
27   run; NOTE:PROCEDURE PWENCODE used (Total process time): real time
0.00 seconds cpu time            0.00 seconds
```

例 4: パスワードのエンコードに Method=SAS003 を指定

要素: METHOD=引数

詳細

この例では、**sas003** エンコーディング方式を使用してパスワードをエンコードし、エンコードされたパスワードを SAS ログに書き込む単純なケースを示します。

プログラム

```
proc pwencode in='my password' method=sas003;
run;
```

プログラムの説明

SAS003 を使用してパスワードをエンコードします。

```
proc pwencode in='my password' method=sas003;
run;
```

ログ

SAS ログではパスワードの各文字が X に置き換えられることに注意してください。SAS003 は AES 暗号化+16 ビットソルトを使用します。SAS003 はランダムなソル

ティングを使用するため、次のコードを実行するたびに異なるパスワードが生成されます。

```
8   proc pwencode in=XXXXXXXXXXXXX method=sas003; 29   run;
{SAS003}08D7B93810D390916F615117D71B2639B4BE NOTE:PROCEDURE PWENCODE used (Total
process time): real time           0.00 seconds cpu time            0.00 seconds
```

52 章

QDEVICE プロシジャ

概要: QDEVICE プロシジャ

QDEVICE プロシジャでは、グラフィックデバイスおよびユニバーサルプリンタについてのレポートが作成されます。このレポートの情報を使用すると、特定のアプリケーション向けの使用に最適のデバイスやプリンタを決定できます。

異なる 6 つのレポートが使用可能です。これらのレポートでは、カラーサポート、デフォルトの出力サイズ、余白量、解像度、サポートされているフォント、ハードウェアシンボル、ハードウェアの塗りつぶしの種類、ハードウェアの線スタイル、デバイスオプションなどの情報が要約されます。

このプロシジャの出力は、SAS ログまたは出力 SAS データセットに送ることができます。

概念: QDEVICE プロシジャ

Windows 動作環境のレポート

デフォルトでは、SAS は、Windows の印刷を使用するために、NOUNIVERSALPRINT (NOUPRINT)システムオプションを使用して Windows 上で起動します。SYSPRINT=システムオプションによって、デフォルトの Windows プリンタが決定されます。Windows のプリンタは SAS プリンタインターフェイスデバイスに関連付けられているので、デフォルトの Windows プリンタ用に作成したレポートには、次の SAS プリンタインターフェイスデバイスのいずれかのデバイス名が含まれます

- WINPRTC (カラー)
- WINPRTG (グレースケール)
- WINPRTM (モノクローム)

Windows 上でユニバーサル印刷をアクティブにして SAS を起動した場合、デフォルトプリンタレポートは、Windows プリンタではなくデフォルト SAS ユニバーサルプリンタ用になります。

レポート例については、“例 4: デフォルトのプリンタのレポートの生成” (1534 ページ)を参照してください。

構文: QDEVICE プロシジャ

デフォルト: 作成するレポート、プリンタまたはデバイスを指定しなかった場合、プロシジャでは、ウィンドウ環境を使用して SAS を実行するとデフォルトディスプレイデバイス用に、その他のモードではデフォルトユニバーサルプリンタ用に、GENERAL レポートが生成されます。

注: DEVICE、PRINTER または VAR ステートメントは複数指定できます。ステートメントは指定された順番で処理されます。

PROC QDEVICE
<REPORT=GENERAL | FONT | DEVOPTION | LINESTYLE | RECTANGLE | SYMBOL>
<OUT=*SAS-data-set*>
<CATALOG=*catalog-name*>
<DEVLOC=GDEVICE*n* | SASHELP | _ALL_ | *libref*>
<REGISTRY=SASHELP | SASUSER>
<SUPPORT=YES | NO | ALL>
<UNITS=IN | CM;>
DEVICE <*device-name(s)*> <_ALL_> <_HTML_> <_LISTING_> <_RTF_>;
PRINTER <*printer-name(s)*> <_ALL_> <_PCL_> <_PDF_> <_PRINTER_> <_PS_>;
VAR *variable-1* <*variable-2* …>;

ステートメント	タスク	例
“PROC QDEVICE ステートメント”	(任意)出力データセット、生成するレポート、サポート機能または未サポート機能のリストの有無、サイズ情報の指定	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7
“DEVICE ステートメント”	レポート生成対象の SAS/GRAPH デバイスの指定	Ex. 2, Ex. 7
“PRINTER ステートメント”	レポート生成対象のユニバーサルプリンタの指定	Ex. 6, Ex. 3
“VAR ステートメント”	生成レポートに含める情報(変数)の指定	Ex. 5

PROC QDEVICE ステートメント

検証された入力データ、レポートの内容、作成する出力の種類を制御します。

構文

PROC QDEVICE
<REPORT=GENERAL | FONT | DEVOPTION | LINESTYLE | RECTANGLE | SYMBOL>
<OUT=*SAS-data-set*>
<CATALOG=*catalog-name*>
<DEVLOC=GDEVICE*n* | SASHELP | _ALL_ | *libref*>
<REGISTRY=SASHELP | SASUSER>
<SUPPORT=YES | NO | ALL>
<UNITS=IN | CM>;

オプション引数の要約

CATALOG=*catalog-name*
SAS デバイスカタログの名前を指定してデバイスを検索します。

DEVLOC = GDEVICE*n* | SASHELP | _ALL_ | *libref*
デバイスカタログが位置する単数または複数のライブラリを指定します。

OUT=*SAS-data-set*
レポートの出力 SAS データセットを指定します。

REGISTRY = SASHELP | SASUSER
ユニバーサルプリンタのクエリ時に SAS レジストリのどの部分を検索するかを指定します。

REPORT = DEVOPTION | FONT | GENERAL | LINESTYLE | RECTANGLE | SYMBOL
生成するレポートの種類を指定します。

SUPPORT= YES | NO | ALL
サポート機能のみ、未サポート機能のみ、または全機能のいずれをレポートするのかを指定します。

UNITS = IN | CM
GENERAL レポートにおける特定の変数の値の単位をインチまたは cm に指定します。

オプション引数

CATALOG=*catalog-name*
SAS デバイスカタログの名前を指定してデバイスを検索します。

別名 C=, CAT=

デフォルト DEVICES

操作 CATALOG=オプションは DEVLOC=オプションとともに機能します。CATALOG=オプションを指定すると、DEVLOC=オプションにより指定されたライブラリ(sashelp.mycatalog など)で SAS が検索します。

例 “例 7: レポートに対するユーザーライブラリおよびユーザーカタログの指定” (1541 ページ)

DEVLOC = GDEVICE*n* | SASHELP | _ALL_ | *libref*
デバイスカタログが位置する単数または複数のライブラリを指定します。指定できるライブラリは次のとおりです。

GDEVICE*n*
SAS/GRAPH デバイスライブラリの 1 つでデバイスを検索するように指定します。*n* は 0～9 となります。

SASHELP
Sashelp ライブラリでデバイスを検索するよう指定します。

ALL
ライブラリ Gdevice0～Gdevice9、次に Sashelp ライブラリという順序でデバイスを検索するよう指定します。これらライブラリからのデバイス検索はすべてレポートされます。

libref
検索する有効な SAS ライブラリを指定します。

デフォルト DEVLOC=オプションを指定しなかった場合、ライブラリは次の順序で検索されます。

1. Gdevice0–Gdevice9
2. Sashelp

DEVLOC=_ALL_を指定しなければ、指定したデバイスの検索の初回がレポートされます。

操作 DEVLOC=オプションは CATALOG=オプションとともに機能します。CATALOG=オプションを指定すると、DEVLOC=オプションにより指定されたライブラリ(sashelp.mycatalog など)で SAS が検索します。

例 “例 7: レポートに対するユーザーライブラリおよびユーザーカタログの指定” (1541 ページ)

OUT=*SAS-data-set*
レポートの出力 SAS データセットを指定します。

デフォルト SAS ログ

REGISTRY = SASHELP | SASUSER
ユニバーサルプリンタのクエリ時に SAS レジストリのどの部分を検索するかを指定します。REGISTRY=オプションを指定しなかった場合、Sasuser と Sashelp の両方が検索されます。

別名 REG

REPORT = DEVOPTION | FONT | GENERAL | LINESTYLE | RECTANGLE | SYMBOL
生成するレポートの種類を指定します。要求できるレポートの種類は 1 つだけです。各レポートに含まれる変数の説明については、“すべてのレポートに有効な変数” (1511 ページ)を参照してください。

DEVOPTION
指定デバイスでサポートされるハードウェアデバイスオプションのレポートを作成します。

制限事項 このレポートは、ユニバーサルプリンタでは使用できません。

FONT
指定したデバイスまたはプリンタでサポートされるすべてのシステムフォントとデバイス常駐フォントのレポートを作成します。

参照項目 フォントカテゴリの説明については、“SAS/GRAPH, System, and Device-Resident Fonts” (*SAS/GRAPH: Reference*)を参照してください。

GENERAL
指定したデバイスまたはプリンタに関する一般情報のレポートを作成します。このレポートには、出力先、余白サイズ、デフォルトフォント情報、解像度、色情報、ピクセル単位サイズなどの情報が含まれます。これはデフォルトレポートです。

LINESTYLE
指定デバイスでサポートされるハードウェアの線スタイルのレポートを作成します。

制限事項 このレポートは、ユニバーサルプリンタでは使用できません。

RECTANGLE
指定デバイスでサポートされるハードウェアの塗りつぶしの種類のレポートを作成します。

制限事項 このレポートは、ユニバーサルプリンタでは使用できません。

SYMBOL
指定デバイスでサポートされるハードウェアシンボルのレポートを作成します。

制限事項 このレポートは、ユニバーサルプリンタでは使用できません。

デフォルト GENERAL

参照項目 各レポートに含まれる変数の説明については、“すべてのレポートに有効な変数” (1511 ページ)を参照してください。

SUPPORT= YES | NO | ALL
サポート機能のみ、未サポート機能のみ、または全機能のいずれをレポートするのかを指定します。

YES サポートされているハードウェア機能とオプションのみをレポートします。

NO サポートされていないハードウェア機能とオプションのみをレポートします。

ALL ハードウェア機能とオプションについて、サポートされているものとされていないものの両方をレポートします。

デフォルト YES

制限事項 このオプションは、DEVOPTION、LINESTYLE、RECTANGLE、SYMBOL レポートのいずれかの作成時にのみデバイスに適用されます。

UNITS = IN | CM
HEIGHT、WIDTH、LEFT、LMIN、RIGHT、RMIN、BOTTOM、BMIN、TOP、TMIN、HRES、VRES 変数の値を、インチとセンチのどちらでレポートするのかを指定します。HRES と VRES の値は、ピクセルパーインチかピクセルパーセンチでレポートされます。

デフォルト IN

制限事項 このオプションは、GENERAL レポートの作成時のみ適用されます。

DEVICE ステートメント

どの SAS/GRAPH デバイス用にレポートを作成するかを指定します。

要件 デバイス名_ALL_、_HTML_、_LISTING_、_RTF_のうちから少なくとも 1 つ指定する必要があります。

構文

DEVICE *<device-name(s)>* <_ALL_> <_HTML_> <_LISTING_> <_RTF_>;

オプション引数

device-name(s)
レポート生成対象のデバイスを指定します。デバイス名はブランクスペースで区切ります。スペースを含むデバイス名を引用符で囲みます。

類似名のデバイスをすべてレポートするには、* (アスタリスク)または?(疑問符)をワイルドカード文字として使用できます。

*
対象デバイス名のこの位置に任意の数の文字を表示します。

要件 ワイルドカード文字を含むデバイス名は、必ず引用符で囲んでください。

注 同一のデバイス名に含まれる*と?は指定できます。

例 'svg*'

例 “例 3: SAS/GRAPH デバイスドライバとユニバーサルプリンタのレポートの生成” (1533 ページ)

?
対象デバイス名の中の 1 文字を表示します。対象デバイス名の中の複数の連続する?文字を指定できます。

要件 ワイルドカード文字を含むデバイス名は、必ず引用符で囲んでください。

注 同一のデバイス名に含まれる*と?は指定できます。

例 'tiff?300'

ALL
全デバイスのレポートを生成します。

HTML
ODS HTML 出力先で使用するデフォルトデバイスを決定し、そのデバイスに対するレポートを生成します。デバイスは、SAS レジストリの ODS キーで割り当てられるデフォルト HTML バージョンに基づいています。

LISTING
ODS リスト出力先で使用するデフォルトデバイスを決定し、そのデバイスに対するレポートを生成します。デフォルト値はホスト固有のディスプレイデバイスです。

RTF
ODS RTF 出力先で使用するデフォルトデバイスを決定し、そのデバイスに対するレポートを生成します。

PRINTER ステートメント

どのユニバーサルプリンタ用にレポートを生成するかを指定します。

要件 プリンタ名_ALL_、_PCL_、_PDF_、_PRINTER_、_PS_のうちから少なくとも 1 つ指定する必要があります。

構文

PRINTER *<printer-name(s)>* <_ALL_> <_PCL_> <_PDF_> <_PRINTER_> <_PS_>;

オプション引数

printer-name(s)

レポート生成対象のユニバーサルプリンタを指定します。プリンタ名にスペースが含まれる場合は、そのプリンタ名を引用符で囲みます。プリンタ名はブランクスペースで区切ります。

類似名のプリンタをすべてレポートするには、* (アスタリスク)または?(疑問符)をワイルドカード文字として使用できます。

*

対象プリンタ名の*の位置にある任意の数の文字と適合するプリンタに関するレポートを意味します。

要件 ワイルドカード文字を含むプリンタ名は、必ず引用符で囲んでください。

注 同一のデバイス名に含まれる*と?は指定できます。

例 'pcl*'はプリンタ pcl4、pcl5、pcl5c、pcl5e をレポートします

?

対象プリント名に適合する全プリンタのレポートを意味します。?位置の文字は任意です。対象プリンタ名の同一位置に同じ数の文字を表示するには、*printer-name* で複数の?文字を使用できます。

要件 ワイルドカード文字を含むプリンタ名は、必ず引用符で囲んでください。

注 同一のデバイス名に含まれる*と?は指定できます。

例 'tiff?'は、プリンター tiffa と tiffk をレポートします。プリンタ tiff は 4 文字だけなのでレポートされません。

例 “例 3: SAS/GRAPH デバイスドライバとユニバーサルプリンタのレポートの生成” (1533 ページ)

ALL

全ユニバーサルプリンタのレポートを生成します。

PCL

ODS PCL 出力先で使用するデフォルトプリンタを決定し、そのプリンタに対するレポートを生成します。

PDF

ODS PDF 出力先で使用するデフォルトプリンタを決定し、そのプリンタに対するレポートを生成します。

PRINTER

ODS PRINTER 出力先で使用するデフォルトプリンタを決定し、そのプリンタに対するレポートを生成します。

Windows 固有 SAS では、ユニバーサル印刷ではなく Windows 印刷がデフォルトで使用されます。SAS で Windows 印刷を使用する場合、_PRINTER_引数指定時に生成されるレポートには、デフォルトの Windows プリンタに関連付けられた SAS プリンタインターフェイスデバイスの情報が含まれます。デフォルトの Windows プリンタは、SYSPRINT=システムオプションによって指定されます。SAS プリンタインターフェイスデバイスは、WINPRTC (カラー)、WINPRTG (グレースケール)または WINPRTM (モノクローム)です。レポートでは、プリンタインターフェイスデバイスは名前フィールドに、プリンタ名は説明フィールドに表示されます。

_PS_
ODS PS 出力先で使用するデフォルトプリンタを決定し、そのプリンタに対するレポートを生成します。

VAR ステートメント

レポートにどの変数を含めるかを指定します。レポートでの変数の順序は、VAR ステートメントで指定した順序によって決定されます。

デフォルト: VAR ステートメントを指定しない場合は、レポートに対する変数はすべて、デフォルトの順序でレポートに含められます。

ヒント: VAR ステートメントを指定する場合は、変数を少なくとも 1 つ指定する必要があります。そうでなければ、ステートメントは無視されます。

構文

VAR *variable-1* <*variable-2* …>;

すべてのレポートに有効な変数

DESC
デバイスまたはプリンタのデフォルト説明を表示します。

LOCATION
検出されたデバイスエントリについて、Gdevice0-Gdevice9 ライブラリ、または Sashelp ライブラリ、または DEVLOC=オプションにより指定されているライブラリの物理的位置を表示します。ユニバーサルプリンタの場合、この変数では、プリンタが見つかった SAS レジストリ(SASHELP または SASUSER)が表示されます。

NAME
デバイスまたはプリンタの名前を表示します。

TYPE
デバイスまたはプリンタの種類を表示します。

- グラフデバイス
- プリンタインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル
- ユニバーサルプレビューア
- ユニバーサルプリンタ

参照項目 “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

“Managing Universal Printers Using the PRTDEF Procedure” (*SAS Language Reference: Concepts*)

DEVOPTION レポート変数

詳細については、“DEVOPTION レポートの作成” (1526 ページ)を参照してください。

BIT
対応するデバイスオプションに対する DEVOPTS 文字列のビット位置を表示します。

参照項目 “DEVOPTS” (*SAS/GRAPH: Reference*)

BITSTRING
対応するデバイスオプションのビットパターンを表示します。

参照項目 “DEVOPTS” (*SAS/GRAPH: Reference*)

DESC
デバイスまたはプリンタのデフォルト説明を表示します。

LOCATION
検出されたデバイスエントリについて、Gdevice0-Gdevice9 ライブラリ、または Sashelp ライブラリ、または DEVLOC=オプションにより指定されているライブラリの物理的位置を表示します。

NAME
デバイスまたはプリンタの名前を表示します。

ODESC
デバイスに対する有効なハードウェアオプションの説明を表示します。

OPTION
デバイスに対する有効なハードウェアオプションの名前を表示します。

SUPPORT
デバイスオプションを表示します。

操作 SUPPORT 変数の値は、PROC QDEVICE ステートメントの SUPPORT=オプションに影響されます。SUPPORT=YES の場合、レポートにはサポートされているデバイスオプションが表示されます。SUPPORT=NO の場合、レポートにはサポートされていないデバイスオプションが表示されます。SUPPORT=ALL の場合、レポートには、サポートされているデバイスオプションもサポートされていないデバイスオプションもすべて表示されます。

TYPE
デバイスまたはプリンタの種類を表示します。

- グラフデバイス
- プリンタインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル

参照項目 “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

FONT レポート変数

詳細については、“FONT レポートの作成” (1524 ページ)を参照してください。

ALIAS
プロシジャにより登録されているフォントの別名をレポートします。

DESC
デバイスまたはプリンタのデフォルト説明を表示します。

FONT
デフォルトフォントの名前を表示します。

参照項目 “Default Fonts” (*SAS/GRAPH: Reference*)

“FONT レポートの変数ラベル” (1525 ページ)

FSTYLE
出力データセットに、各フォントのフォントスタイル(Roman や Italic など)およびフォントの太さを表示します。

制限事項 レポート出力が SAS ログに送られた場合、FONT レポートには、フォントファミリ名(Courier、Helvetica、Times など)のみ表示されます。特定のフォントスタイルはレポートされません。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、スタイルは使用できません。詳細については、“CHARREC” (*SAS/GRAPH: Reference*)を参照してください。

参照項目 “FONT レポートの変数ラベル” (1525 ページ)

FTYPE
フォントの種類(Printer Resident、System、Software など)を表示します。

注 出力データセットの FTYPE 変数の値は、Printer Resident、System または Software です。値 Software は、ハードウェアフォントサポートが無効な SAS/GRAPH デバイスの FONT レポートにのみ表示されます。

FWEIGHT
出力データセットに、各フォントのフォントの太さ(Normal や Bold など)およびフォントスタイルを表示します。

制限事項 レポート出力が SAS ログに送られた場合、FONT レポートには、フォントファミリ名(Courier、Helvetica、Times など)のみ表示されます。特定のフォントの太さはレポートされません。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、太さは使用できません。詳細については、“CHARREC” (*SAS/GRAPH: Reference*)を参照してください。

FVERSION
フォントのバージョンを指定します。

制限事項 レポート出力が SAS ログに送られた場合、FONT レポートには、フォントファミリ名(Courier、Helvetica、Times など)のみ表示されます。特定のフォントのバージョンはレポートされません。

LOCATION
検出されたデバイスエントリについて、Gdevice0-Gdevice9 ライブラリ、または Sashelp ライブラリ、または DEVLOC=オプションにより指定されているライブラリの物理的位置を表示します。ユニバーサルプリンタの場合、この変数では、プリンタが見つかった SAS レジストリ(SASHELP または SASUSER)が表示されます。

NAME
デバイスまたはプリンタの名前を表示します。

TYPE
デバイスまたはプリンタの種類を表示します。

- グラフデバイス
- プリンタインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル
- ユニバーサルプレビューア
- ユニバーサルプリンタ

参照項目 “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

“Managing Universal Printers Using the PRTDEF Procedure” (*SAS Language Reference: Concepts*)

GENERAL レポート変数

詳細については、“GENERAL レポートの作成” (1521 ページ)を参照してください。

ALIAS
プロシジャにより登録されているフォントの別名をレポートします。

ANIMATION
アニメーションのアクティブ化、有効化、無効化、ユニバーサルプリンタへの非対応のいずれかを指定します。

Active	ODS HTML 出力のグラフィックがアニメーションで一緒にグループ化されていることを示します。Animate=Start と Animate=Stop は無視されます。
Enabled	アニメーションがサポートされていることを示します。アニメーションを開始するには Animate=Start を、停止するには Animate=Stop を指定してください。
Disabled	アニメーションはサポートされていますが、デバイスまたはプリンタには無効となっていることを示します。
Unsupported	アニメーションがデバイスまたはプリンタにサポートされないことを示します。

BMIN
下余白の最小サイズを表示します。

BOTTOM
下余白の現在のサイズを表示します。

CLRSPACE
カラーサポート(カラースペース)の種類(RGB、RGBA、CMYK、HLS など)を表示します。

参照項目 “Color-Naming Schemes” (SAS/GRAPH: Reference)

COLS
出力に横の列の数を表示します。

参照項目 “Cells” (*SAS/GRAPH: Reference*)

COMPRESSION
圧縮が使用される状況を示します。圧縮値は次のとおりです。

Always
圧縮が常に有効であることを示します。

オプション
圧縮が UPRINTCOMPRESSION システムオプションまたは ODS PRINTER ステートメントの COMPRESS=オプションにより指定されていることを示します。

Never
圧縮がデバイスまたはプリンタには絶対に使用されないよう指定します。

COMPMETHOD
デバイスまたはプリンタによって圧縮がサポートされる場合、使用する圧縮方法を示します。

注 Compression が Never でどの圧縮も使用できない場合は、SAS ログに Compression Method 値は表示されません。

DESC
デバイスまたはプリンタのデフォルト説明を表示します。

DEST
デバイスまたはプリンタが出力をプリンタまたはディスプレイデバイスに直接送信しない場合、デバイスまたはユニバーサルプリンタのデフォルト出力先を表示します。デバイスが出力をプリンタまたはディスプレイデバイスに直接送信する場合、DEST の値はブランクになります。

出力がモニタに送られる場合、または、Windows で、出力がプリンタに送られる場合、出力先がブランク値になることがあります。

EMBEDDING
フォントの埋め込みがサポートされるかどうかを示します。

ALWAYS	デバイスまたはプリンタのフォントの埋め込みが常に有効です。
OPTION	FONTEMBEDDING システムオプションで、フォントの埋め込みをサポートするかどうかが制御されます。
NEVER	フォントの埋め込みはサポートされません。

FHEIGHT
デフォルトフォントの高さを各単位で表示します。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、高さは使用できません。詳細については、“CHARREC” (*SAS/GRAPH: Reference*)を参照してください。

FONT
デフォルトフォントの名前を表示します。

参照項目 “Default Fonts” (*SAS/GRAPH: Reference*)

FORMAT
出力フォーマットタイプ(EMF、EMF Plus、EMF Dual、PostScript、GIF、Host Display など)を表示します。

参照項目 “Commonly Used Devices” (*SAS/GRAPH: Reference*)

FSTYLE
デフォルトのフォントスタイル(Roman、Regular など)を表示します。

操作 出力を SAS ログに送る GENERAL レポートで FSTYLE 変数を指定した結果は、FONTS レポートに対して FSTYLE 変数を指定したときに得られる結果とは異なります。GENERAL レポートでは、フォントスタイルが SAS ログにレポートされます。FONT レポートでは、SAS ログレポートにフォントファミリ名のみ表示されます。特定のフォントスタイルはレポートされません。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、スタイルは使用できません。詳細については、“CHARREC” (*SAS/GRAPH: Reference*)を参照してください。

FWEIGHT
デフォルトのフォントの太さ(Normal、Medium など)を表示します。

操作 出力を SAS ログに送る GENERAL レポートで FWEIGHT 変数を指定した結果は、FONTS レポートに対して FWEIGHT 変数を指定したときに得られる結果とは異なります。GENERAL レポートでは、フォントの太さが SAS ログにレポートされます。FONT レポートでは、SAS ログレポートにフォントファミリ名のみ表示されます。特定のフォントの太さはレポートされません。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、太さは使用できません。詳細については、“CHARREC” (*SAS/GRAPH: Reference*)を参照してください。

FVERSION
フォントのバージョンを指定します。

HEIGHT
デバイスまたはプリンタに送信される出力のデフォルトの縦の高さ(UNITS 単位)を表示します。

HRES
デバイスまたはプリンタに送信される出力の水平解像度(pixels per UNIT)を表示します。水平解像度は式 **`HRES=XPIXELS/WIDTH`** によって計算されます。

操作 VAR ステートメントで HRES 変数か VRES 変数のどちらかを指定した場合、SAS ログでは水平解像度と垂直解像度がラベル XxY Resolution を使用して一緒に表示されます。出力データセットでは、HRES と VRES は別々にレポートされます。

参照項目 “Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” (*SAS/GRAPH: Reference*)

IOTYPE
デバイスまたはプリンタにより使用される入力/出力のタイプ(DISK、PRINTER、PIPE、GTERM など)を表示します。

参照項目 “FILENAME Statement” (*SAS Statements: Reference*) および “DEVTYPE” (*SAS/GRAPH: Reference*)

LEFT
出力の左余白のサイズを表示します。

LMIN
最小左余白を表示します。

LOCATION
検出されたデバイスエントリについて、Gdevice0-Gdevice9 ライブラリ、または Sashelp ライブラリ、または DEVLOC=オプションにより指定されているライブラリの

物理的位置を表示します。ユニバーサルプリンタの場合、この変数では、プリンタが見つかった SAS レジストリ(SASHELP または SASUSER)が表示されます。

MAXCOLORS
デバイスまたはプリンタでサポートされる色の最大数を表示します。

参照項目 “Maximum Number of Colors Displayed on a Device” (*SAS/GRAPH: Reference*)

MODULE
デバイスドライバモジュールの名前を指定します。

NAME
デバイスまたはプリンタの名前を表示します。

PROTOTYPE
ユニバーサルプリンタの定義に使用するプロトタイプ(モデル)を表示します。

参照項目 “Universal Printing” (*SAS Language Reference: Concepts*) および“Define a New Printer” (*SAS Language Reference: Concepts*)

RIGHT
右余白のサイズを表示します。

RMIN
右余白の最小サイズを表示します。

ROWS
出力の縦の行の数を表示します。

参照項目 “Cells” (*SAS/GRAPH: Reference*)

TMIN
出力の最小上余白を表示します。

TOP
上余白のサイズを表示します。

TYPE
デバイスまたはプリンタの種類を表示します。

- グラフデバイス
- プリンタインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル
- ユニバーサルプレビューア
- ユニバーサルプリンタ

参照項目 “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

“Managing Universal Printers Using the PRTDEF Procedure” (*SAS Language Reference: Concepts*)

UNITS
サイズを表示する際の単位(インチを示す IN かセンチを示す CM)を表示します。SAS ログでは、UNITS の値はそれぞれ inches か centimeters で表示されます。出力データセットでは、UNITS の値は IN か CM で表示されます。

操作 VAR ステートメントで、サイズ、余白または解像度の変数を指定しない場合、SAS ログでは、サイズ、余白または解像度の測定に使用する単位が示されます。次に例を示します。

```
 Name: EMF
Units: inches
```

VAR ステートメントで、サイズ、余白または解像度の変数を指定した場合、SAS ログでは値とともに単位が表示されます。次に例を示します。

```
XxY Resolution: 96x96 pixels per inch
```

VISUAL
表示色の種類(Indexed Color、Direct Color、True Color、Monochrome、Gray Scale など)を表示します。

VRES
デバイスまたはプリンタに送信される出力の垂直解像度(pixels per UNIT)を表示します。垂直解像度は式 **`VRES=YPIXELS/HEIGHT`** で計算されます。

操作 VAR ステートメントで HRES 変数か VRES 変数のどちらかを指定した場合、SAS ログでは水平解像度と垂直解像度がラベル XxY Resolution を使用して一緒に表示されます。出力データセットでは、HRES と VRES は別々にレポートされます。

参照項目 “Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” (*SAS/GRAPH: Reference*)

WIDTH
デバイスまたはプリンタに送信される出力の幅(UNITS 単位)を表示します。

XPIXELS
出力の幅をピクセル単位で表示します。

参照項目 “XPIXELS” (*SAS/GRAPH: Reference*) および“Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” (*SAS/GRAPH: Reference*)

YPIXELS
出力の高さをピクセル単位で表示します。

参照項目 “YPIXELS” (*SAS/GRAPH: Reference*) および “Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” (*SAS/GRAPH: Reference*)

LINESTYLE レポート変数
詳細については、“LINESTYLE レポートの作成” (1528 ページ)を参照してください。

DESC
デバイスまたはプリンタのデフォルト説明を表示します。

LINE
デバイスまたはプリンタでサポートされる線のスタイルを表示します。

操作 SAS ログ LINESTYLE レポートでは、LINE 変数と SUPPORT 変数が一緒にレポートされます。VAR ステートメントで LINE 変数か SUPPORT 変数のどちらかを指定すると、線のスタイルは、Supported Line Styles または Unsupported Line Styles の変数ラベルを使用してレポートされます。

参照項目 “Line Types” (*SAS/GRAPH: Reference*)

LOCATION
Gdevice0-Device9 物理的位置、またはデバイスエントリが検出された Devices カタログを含む Sashelp ライブラリの物理的位置、または DEVLOC=オプションにより指定されるライブラリの 物理的位置を表示します。

NAME
デバイスの名前を表示します。

SUPPORT
デバイスラインのスタイルを表示します。

操作 SUPPORT 変数の値は、PROC QDEVICE ステートメントの SUPPORT=オプションに影響されます。SUPPORT=YES の場合、レポートにはサポートされている線のスタイルが表示されます。SUPPORT=NO の場合、レポートにはサポートされていない線のスタイルが表示されます。SUPPORT=ALL の場合、レポートには、サポートされている線のスタイルもサポートされていない線のスタイルもすべて表示されます。

TYPE
デバイスまたはプリンタの種類を表示します。

- グラフデバイス
- プリンタインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル

参照項目 “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

RECTANGLE レポート変数
詳細については、“RECTANGLE レポートの作成” (1529 ページ)を参照してください。

DESC
デバイスまたはプリンタのデフォルト説明を表示します。

FILL
デバイスでサポートされているハードウェアの塗りつぶしの種類を表示します。

操作 SAS ログ RECTANGLE レポートでは、FILL 変数と SUPPORT 変数が一緒にレポートされます。VAR ステートメントで FILL 変数または SUPPORT 変数を指定した場合、塗りつぶし名は、ラベル Supported Hardware Fills またはラベル Unsupported Hardware Fills のどちらかを使用してレポートされます。

参照項目 “PATTERN Statement” (*SAS/GRAPH: Reference*)

LOCATION
Gdevice0-Device9 物理的位置、またはデバイスエントリが検出された Devices カタログを含む Sashelp ライブラリの物理的位置、または DEVLOC=オプションにより指定されるライブラリの 物理的位置を表示します。

NAME
デバイスの名前を表示します。

SUPPORT
ハードウェア塗りつぶしを表示します。

操作 SUPPORT 変数の値は、PROC QDEVICE ステートメントの SUPPORT=オプションに影響されます。SUPPORT=YES の場合、レポートにはサポートされているハードウェア塗りつぶしが表示されます。SUPPORT=NO の場合、レポートにはサポートされていないハードウェア塗りつぶしが表示されます。SUPPORT=ALL の場合、レポートには、サポートされているハードウェア塗りつぶしもサポートされていないハードウェア塗りつぶしもすべて表示されます。

TYPE
デバイスまたはプリンタの種類を表示します。

- グラフデバイス
- プリンタインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル

参照項目 “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

SYMBOL レポート変数

詳細については、“SYMBOL レポートの作成” (1530 ページ)を参照してください。

DESC
デバイスまたはプリンタのデフォルト説明を表示します。

LOCATION
Gdevice0-Device9 物理的位置、またはデバイスエントリが検出された Devices カタログを含む Sashelp ライブラリの物理的位置、または DEVLOC=オプションにより指定されるライブラリの 物理的位置を表示します。

NAME
デバイスの名前を表示します。

SUPPORT
デバイスまたはプリンタのシンボルを表示します。

操作 SUPPORT 変数の値は、PROC QDEVICE ステートメントの SUPPORT=オプションに影響されます。SUPPORT=YES の場合、レポートにはサポートされているシンボルのスタイルが表示されます。SUPPORT=NO の場合、レポートにはサポートされていないシンボルが表示されます。SUPPORT=ALL の場合、

レポートには、サポートされているシンボルもサポートされていないシンボルもすべて表示されます。

SYMBOL
ハードウェアシンボルの名前を指定します。

操作 SAS ログ SYMBOL レポートでは、VAR ステートメントで SYMBOL 変数か SUPPORT 変数のどちらかを指定した場合、シンボル名は、Supported Hardware Symbols ラベルまたは Unsupported Hardware Symbols ラベルを使用してレポートされます。

参照項目 “SYMBOL” (*SAS/GRAPH: Reference*) および“SYMBOLS” (*SAS/GRAPH: Reference*)

TYPE
デバイスまたはプリンタの種類を表示します。

- グラフデバイス
- プリンタインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル

参照項目 “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

全レポート共通の変数

次の変数は、どのレポートでも使用できます。

- NAME
- DESC
- TYPE
- LOCATION

変数の説明については、“すべてのレポートに有効な変数” (1511 ページ)を参照してください。

GENERAL レポートの作成

GENERAL レポートでは、指定したデバイスまたはプリンタに関する一般情報のレポートが作成されます。この情報には、余白サイズ、デフォルトフォント情報、解像度、色情報が含まれます。

GENERAL レポート変数について

変数の説明については、“GENERAL レポート変数” (1514 ページ)を参照してください。

次の表に、GENERAL レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこの表の表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	Description	DESCRIPTION OF DEVICE OR PRINTER
MODULE	Module	DRIVER MODULE
TYPE	Type	TYPE OF DEVICE OR PRINTER
LOCATION	デバイスには"Device Catalog" プリンタには"Registry"	LOCATION OF DEVICE OR PRINTER DEFINITION
PROTOTYPE	プロトタイプ	PRINTER PROTOTYPE
FONT	Default Typeface	FONT TYPEFACE DEFAULT
ALIAS	Typeface Alias	FONT TYPEFACE ALIAS
FSTYLE	Font Style	FONT STYLE DEFAULT
FWEIGHT	Font Weight	FONT WEIGHT DEFAULT
FHEIGHT	Font Height	FONT HEIGHT DEFAULT
FVERSION	Font Version	FONT VERSION
MAXCOLORS	Maximum Colors	MAXIMUM NUMBER OF SUPPORTED COLORS
VISUAL	Visual Color	TYPE OF VISUAL COLOR
CLRSPACE	Color Support	TYPE OF COLOR SUPPORT
DEST	出力先	OUTPUT DESTINATION DEFAULT
IOTYPE	I/O Type	TYPE OF I/O DEFAULT
FORMAT	Data Format	OUTPUT DATA FORMAT
HEIGHT	Height	HEIGHT OF OUTPUT

変数	SAS ログラベル	出力データセットラベル
WIDTH	Width	WIDTH OF OUTPUT
UNITS	Units *	UNITS FOR SIZE, MARGINS AND RESOLUTION
YPIXELS	Ypixels	VERTICAL PIXELS
XPIXELS	Xpixels	HORIZONTAL PIXELS
ROWS	Rows (vpos)	ROWS
COLS	Columns (hpos)	COLUMNS
LEFT	Left Margin	LEFT MARGIN
LMIN	Minimum Left Margin	MINIMUM LEFT MARGIN
RIGHT	Right Margin	RIGHT MARGIN
RMIN	Minimum Right Margin	MINIMUM RIGHT MARGIN
BOTTOM	Bottom Margin	BOTTOM MARGIN
BMIN	Minimum Bottom Margin	MINIMUM BOTTOM MARGIN
TOP	Top Margin	TOP MARGIN
TMIN	Minimum Top Margin	MINIMUM TOP MARGIN
HRES	XxY Resolution	HORIZONTAL PIXELS PER UNIT
VRES	XxY Resolution	VERTICAL PIXELS PER UNIT
COMPRESSION	Compression Enabled	COMPRESSION ENABLED
COMPMETHOD	Compression Method	COMPRESSION METHOD
EMBEDDING	Font Embedding	FONT EMBEDDING SUPPORT
ANIMATION	Animation	ANIMATION SUPPORT

* 単位の種類が変数の値とともに表示される場合(左余白の 0 inches など)、SAS ログへの出力には Units ラベルが表示されません。

サイズ変数の値に影響を及ぼすシステムオプション

ユニバーサルプリンタの場合、HEIGHT、WIDTH、LEFT、RIGHT、BOTTOM、TOP、LMIN、RMIN、BMIN、TMIN 変数の値は、PAPERSIZE、LEFTMARGIN、RIGHTMARGIN、BOTTOMMARGIN、TOPMARGIN SAS システムオプションの設定に影響を受けます。デフォルトの用紙のサイズは、SAS ロケール(ユニバーサルプリンタのデフォルトサイズに影響を及ぼす)によって決定されます。SAS/GRAPH デバイスの場合、これらの変数値はシステムオプション設定の影響を受けません。

詳細については、*SAS システムオプション: リファレンス*を参照してください。

例:GENERAL レポート

次の QDEVICE プロシジャでは、SVG ユニバーサルプリンタ用の GENERAL レポートが作成されます。

```
proc qdevice;
   printer svg;
run;
```

SAS ログの GENERAL レポートを次に示します。

```
16   proc qdevice; 17   printer svg; 18    run; Name:SVG Description:Scalable
Vector Graphics 1.1 Module:SASPDSVG Type:Universal Printer Registry:SASHELP
Prototype:SVG 1.1 Default Typeface:Cumberland AMT Typeface Alias:Courier Font
Style:Regular Font Weight:Normal Font Height:8 points Font Version:Version 1.03
Maximum Colors:16777216 Visual Color:Direct Color Color Support:RGBA
Destination: sasprt.svg I/O Type:DISK Data Format:SVG Height:6.25 inches Width:
8.33 inches Ypixels:600 Xpixels:800 Rows(vpos):50 Columns(hpos):114 Left Margin:
0 inches Minimum Left Margin:0 inches Right Margin:0 inches Minimum Right Margin:
0 inches Bottom Margin:0 inches Minimum Bottom Margin:0 inches Top Margin:0
inches Minimum Top Margin:0 inches XxY Resolution:96x96 pixels per inch
Compression Enabled:Never Compression Method:Deflate Font Embedding:Option
Animation:Enabled
```

FONT レポートの作成

FONT レポートでは、指定したデバイスまたはプリンタでサポートされるすべてのシステムフォントおよびデバイス常駐フォントのレポートが作成されます。

FONT レポート変数について

変数の説明については、“FONT レポート変数” (1512 ページ)を参照してください。

次の表に、FONT レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこの表の表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER

変数	SAS ログラベル	出力データセットラベル
DESC	Description	DESCRIPTION OF DEVICE OR PRINTER
TYPE	Type	TYPE OF DEVICE OR PRINTER
LOCATION	デバイスには"Device Catalog" プリンタには"Registry"	LOCATION OF DEVICE OR PRINTER DEFINITION
FONT	フォントまたは種類に依存	FONT TYPEFACE
ALIAS	(alias:フォント名の右に *alias*) *	FONT TYPEFACE ALIAS
FTYPE	フォントの種類に依存	FONT TYPE
FSTYLE	なし	FONT STYLE
FWEIGHT	なし	FONT WEIGHT
FVERSION	なし	FONT VERSION

* 次に示すのは別名が SAS ログにどう表示されるのかについての例です。

```
Symbol MT (alias: Symbol)
Tahoma
Thorndale AMT (alias: Times)
```

FONT レポートの変数ラベル

FONT レポートで FONT、FTYPE、FSTYLE、FWEIGHT 変数のいずれかを指定した場合、SAS ログに表示される変数ラベルは変化します。出力データセットの変数ラベルは常に同じで、それぞれ、FONT TYPEFACE DEFAULT、FONT TYPE、FONT STYLE DEFAULT、および FONT WEIGHT DEFAULT です。

VAR ステートメントにより変数 FONT、FTYPE、FSTYLE、FWEIGHT、FVERSION のうち 1 つ以上が指定されている場合、SAS ログではフォントタイプラベルとフォントファミリ名のみがレポートされます。フォントのスタイル、太さ、バージョンは SAS ログにはレポートされません。表示されるフォントタイプラベルはフォントタイプに依存します。ラベルの例としては、Supported Font Typefaces、Supported Resident Typefaces、Supported TrueType Typefaces、Supported Type1 Typefaces などがあります。

例:例:FONT レポート

次の QDEVICE プロシジャでは、ACTIVEX グラフィックデバイス用の FONT レポートが作成されます。

```
proc qdevice report=font;
   device activex;
run;
```

SAS ログの FONT レポートの一部を次に示します。

```
41   proc qdevice report=font; 42   device activex; 43   run; Name:ACTIVEX
Description:ActiveX enabled GIF Driver Type:Graph Device Device Catalog:your-
font-catalog-path Supported Font Typefaces:System (7x16) 8pt System (9x20) 10pt
Terminal (8x12) 7pt Terminal (4x6) 4pt Terminal (5x12) 7pt Terminal (6x8) 5pt
Terminal (7x12) 7pt Terminal (10x18) 11pt Terminal (12x16) 10pt Fixedsys (8x15)
7pt Fixedsys (10x20) 11pt Modern Roman Script Courier (8x13) 8pt
```

DEVOPTION レポートの作成

DEVOPTION レポートでは、指定したデバイスでサポートされるハードウェアデバイスオプションのレポートが作成されます。このレポートは、ユニバーサルプリンタでは使用できません。

DEVOPTION レポート変数について

変数の説明については、“DEVOPTION レポート変数” (1511 ページ)を参照してください。

次の表に、DEVOPTION レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこの表の表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	Description	DESCRIPTION OF DEVICE OR PRINTER
TYPE	Type	TYPE OF DEVICE OR PRINTER
LOCATION	Device Catalog	LOCATION OF DEVICE OR PRINTER DEFINTION
BIT	Bit Position	DEVICE OPTION BIT POSITON
BITSTRING	Bit Pattern	DEVICE OPTION BIT PATTERN
OPTION	Device Option	DEVICE OPTION NAME
ODESC	Option Description	DEVICE OPTION DESCRIPTION
SUPPORT	Support	DEVICE OPTION SUPPORT

例:DEVOPTION レポート

次の QDEVICE プロシジャでは、SASEMF グラフィックデバイス用の DEVOPTION レポートが作成されます。SUPPORT のデフォルトが YES なので、サポートされているオプションもレポートされます。

```
proc qdevice report=devoption;
   device sasemf;
run;
```

SAS ログのレポートを次に示します。

```
104  proc qdevice report=devoption; 105     device sasemf; 106  run;
NOTE:Writing HTML Body file: sashtml.htm Name:SASEMF Description:Enhanced
Metafile Driver Type:Shortcut Device Device Catalog:your-sas-path\sashelp Bit
Pattern:8000000000000000 Device Option:GDCIRCLEARC Option Description:Hardware
is capable of drawing circles Support:Yes Bit Position:1 Bit Pattern:
4000000000000000 Device Option:GDPIEFILL Option Description:Device has hardware
pie-fill capability Support:Yes Bit Position:3 Bit Pattern:1000000000000000
Device Option:GDCRT Option Description:Hardware is a CRT or the device acts like
a CRT Support:Yes Bit Position:5 Bit Pattern:0400000000000000 Device
Option:GDPOLYGONFILL Option Description:Device has polygonfill capability
Support:Yes Bit Position:7 Bit Pattern:0100000000000000 Device Option:GDRGB
Option Description:Hardware is capable of defining colors in one or more color
spaces Support:Yes Bit Position:8 Bit Pattern:0080000000000000 Device
Option:GDMBPOLY Option Description:Hardware can draw polygons with multiple
boundaries Support:Yes Bit Position:9 Bit Pattern:0040000000000000 Device
Option:GDOPACITY Option Description:Hardware is capable of supporting opacity
Support:Yes Bit Position:11 Bit Pattern:0010000000000000 Device Option:GDLWIDTH
Option Description:Hardware can draw lines of varying widths Support:Yes Bit
Position:14 Bit Pattern:0002000000000000 Device Option:GDHRDCHR Option
Description:Hardware characters are supported by the device Support:Yes
```

```
Bit Position:15 Bit Pattern:0001000000000000 Device Option:GDXLIMIT Option
Description:There is no limit on max value allowed for x coordinate Support:Yes
Bit Position:16 Bit Pattern:0000800000000000 Device Option:GDYLIMIT Option
Description:There is no limit on max value allowed for y coordinate Support:Yes
Bit Position:18 Bit Pattern:0000200000000000 Device Option:GDTXJUSTIFY Option
Description:Hardware is capable of justifying proportional text Support:Yes Bit
Position:24 Bit Pattern:0000008000000000 Device Option:GDUNICODE Option
Description:Device supports the use of the Unicode font attribute Support:Yes
Bit Position:25 Bit Pattern:0000004000000000 Device Option:GDPOLYLINE Option
Description:Hardware is capable of supporting polylines Support:Yes Bit Position:
28 Bit Pattern:0000000800000000 Device Option:GDTRUETYPE Option
Description:Device supports the use of TrueType fonts Support:Yes Bit Position:
36 Bit Pattern:0000000008000000 Device Option:GDIMAGE Option Description:Device
is capable of drawing images Support:Yes Bit Position:39 Bit Pattern:
0000000001000000 Device Option:GDIMGROTATE Option Description:Device is
incapable of doing image rotation Support:Yes
```

```
      Bit Position:40 Bit Pattern:0000000000800000 Device Option:GDTRUECOLOR
Option Description:Hardware is a 24-bit true color device Support:Yes Bit
Position:44 Bit Pattern:0000000000080000 Device Option:GDTEXTCLIP Option
Description:Hardware will clip text at the device limits Support:Yes Bit
Position:50 Bit Pattern:0000000000002000 Device Option:GDAUTOSIZE Option
Description:Autosize text to fit rows and columns Support:Yes Bit Position:54
Bit Pattern:0000000000000200 Device Option:GDPOLYOUTLINE Option
Description:Device draws polygon outlines Support:Yes Bit Position:55 Bit
Pattern:0000000000000100 Device Option:GDPRINTERPATH Option Description:Device
temporarily sets printerpath to that of the device name Support:Yes Bit Position:
56 Bit Pattern:0000000000000080 Device Option:GDOPTPASSTHRU Option
Description:PAPERSIZE option sets default value of PAPERSIZE goption Support:Yes
Bit Position:57 Bit Pattern:0000000000000040 Device Option:GDPIEOUTLINE Option
Description:Driver draws pie slice outlines (empty pies) Support:Yes
```

LINESTYLE レポートの作成

LINESTYLE レポートでは、指定したデバイスでサポートされるハードウェアの(点)線のスタイルのレポートが作成されます。

LINESTYLE レポート変数について

変数の説明については、“LINESTYLE レポート変数” (1518 ページ)を参照してください。

次の表に、LINESTYLE レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこの表の表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	Description	DESCRIPTION OF DEVICE OR PRINTER
TYPE	Type	TYPE OF DEVICE OR PRINTER
LOCATION	Device Catalog	LOCATION OF DEVICE OR PRINTER DEFINTION
LINE	Supported Line Styles Unsupported Line Styles	HARDWARE DASHED LINE NUMBER
SUPPORT	Supported Line Styles Unsupported Line Styles	HARDWARE DASHED LINE SUPPORT

例:例:LINESTYLE レポート

次の QDEVICE プロシジャでは、LJ5PS デバイス用の LINESTYLE レポートが作成されます。SUPPORT のデフォルトが YES なので、サポートされている線のスタイルもレポートされます。

```
proc qdevice report=linestyle;
   device lj5ps;
run;
```

SAS ログの LINESTYLE レポートを次に示します。

```
202  proc qdevice report=linestyle; 203     device lj5ps; 204  run; Name:LJ5PS
Description:LaserJet 5P -- 600 dpi -- PostScript Type:Graph Device Device
Catalog:your-sas-path\sashelp Supported Line Styles:1-44
```

RECTANGLE レポートの作成

RECTANGLE レポートでは、指定したデバイスでサポートされるハードウェアの塗りつぶしの種類のレポートが作成されます。

RECTANGLE レポート変数について

変数の説明については、“RECTANGLE レポート変数” (1519 ページ)を参照してください。

次の表に、RECTANGLE レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこの表の表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	Description	DESCRIPTION OF DEVICE or PRINTER
TYPE	Type	TYPE OF DEVICE OR PRINTER
LOCATION	Device Catalog	LOCATION OF DEVICE OR PRINTER DEFINTION
FILL	Supported Hardware Fills	HARDWARE RECTANGLE FILL NAME
SUPPORT	Supported Hardware Fills Unsupported Hardware Fills	HARDWARE RECTANGLE FILL SUPPORT

例:例:RECTANGLE レポート

次の QDEVICE プロシジャでは、SASPRTG ユニバーサルプリンタ用の RECTANGLE レポートが作成されます。SUPPORT のデフォルトが YES なので、サポートされているハードウェア塗りつぶしもレポートされます。

```
proc qdevice report=rectangle;
   device sasprtg;
run;
```

SAS ログの RECTANGLE レポートを次に示します。

```
205  proc qdevice report=rectangle; 206     device sasprtg; 207  run;
Name:SASPRTG Description:POSTSCRIPT LEVEL 1 Type:Printer Interface Device Device
Catalog:your-sas-path\sashelp Supported Hardware Fills:Empty,Solid
```

SASPRTG はプリンタインターフェイスデバイスです。ユニバーサル印刷がアクティブなので、SASPRTG はデフォルトユニバーサルプリンタとインターフェイスでつながっています。したがって、レポートにはユニバーサルプリンタとして PostScript Level 1 プリンタの情報が表示されます。

SYMBOL レポートの作成

SYMBOL レポートでは、指定したデバイスでサポートされるハードウェアシンボルのレポートが作成されます。

SYMBOL レポート変数について

変数の説明については、“SYMBOL レポート変数” (1520 ページ)を参照してください。

次の表に、SYMBOL レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこの表の表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	Description	DESCRIPTION OF DEVICE OR PRINTER
TYPE	Type	TYPE OF DEVICE OR PRINTER
LOCATION	Device Catalog	LOCATION OF DEVICE OR PRINTER DEFINTION
SYMBOL	Supported Hardware Symbols Unsupported Hardware Symbols	HARDWARE SYMBOL NAME

変数	SAS ログラベル	出力データセットラベル
SUPPORT	Support	DEVICE OPTION SUPPORT

例:SYMBOL レポート

次の QDEVICE プロシジャでは、CGM デバイス用の SYMBOL レポートが作成されます。SUPPORT のデフォルトが YES なので、サポートされているハードウェアシンボルもレポートされます。

```
proc qdevice report=symbol;
   device cgm;
run;
```

SAS ログの SYMBOL レポートを次に示します。

```
235  proc qdevice report=symbol; 236     device cgm; 237  run; Name:CGM
Description:CGM generator--binary output Type:Graph Device Device Catalog:your-
sas-path\sashelp Supported Hardware Symbols:Plus,X,Star
```

例: QDEVICE プロシジャ

例 1: デフォルトのディスプレイデバイスのレポートの生成

要素: PROC QDEVICE

詳細

次の例では、デフォルトディスプレイデバイス用の一般レポートを作成します。この例では、Windows 上で対話モードで実行しているとします。

WIN デバイスの場合、色数は Windows ディスプレイ設定によって制御されます。サイズはモニタと解像度の設定によって制御されます。

プログラム

```
proc qdevice;
run;
```

ログ

OUT=オプションを指定しなかった場合、QDEVICE プロシジャでは、SAS ログに出力が送信されます。Windows 動作環境の出力の SAS ログでの表示を次に示します。

ログ 52.1 *PROC QDEVICE 実行後の SAS ログ*

```
Name:WIN Description:Microsoft Windows Display Type:System Display Device
Catalog:your-device-catalog Default Typeface:Sasfont Font Style:Roman Font
Weight:Normal Font Height:7 points Maximum Colors:2147483647 Visual Color:True
Color Color Support:RGB I/O Type:GTERM Data Format:Host Display Height:5.75
inches Width:9.25 inches Ypixels:690 Xpixels:1110 Rows(vpos):46 Columns(hpos):
111 Left Margin:0 inches Minimum Left Margin:0 inches Right Margin:0 inches
Minimum Right Margin:0 inches Bottom Margin:0 inches Minimum Bottom Margin:0
inches Top Margin:0 inches Minimum Top Margin:0 inches XxY Resolution:120x120
pixels per inch Compression Enabled:Never Font Embedding:Never
Animation:Unsupported
```

例 2: 全デバイスの一般レポートの生成

要素: PROC QDEVICE ステートメントオプション:OUT=
DEVICE statement

詳細

次の例では、全デバイスの一般レポートを作成し、その結果を WORK.ALLDEVICES に書き込みます。

全デバイスのレポートを生成するには、_ALL_キーワードを使用します。

プログラム

```
proc qdevice out=allDevices;
   device _all_;
run;
```

出力

次の図は、Viewtable ウィンドウに表示されるレポートの一部です。

アウトプット 52.1 *全デバイスに対する出力データセットレポート*



	NAME OF DEVICE OR PRINTER	DESCRIPTION OF DEVICE OR PRINTER	DRIVER MODULE	TYPE OF DEVICE OR PRINTER
1	ACTIVEX	ActiveX enabled GIF Driver	SASGDGIF	Graph Device
2	ACTXIMG	ActiveX enabled Image Driver	SASGDGIF	Graph Device
3	BMP	BMP File Format--256 colors	SASGDDMX	System Metafile
4	CGM	CGM generator--binary output	SASGDCGM	Graph Device
5	CGMC	CGM generator w/colors--binary output	SASGDCGM	Graph Device
6	CGMOF97L	CGM for Office 97 landscape mode	SASGDCGM	Graph Device
7	CGMOF97P	CGM for Microsoft Office 97 portrait	SASGDCGM	Graph Device
8	CGMOFML	CGM for MS Office w/ Metric Scaling land	SASGDCGM	Graph Device
9	CGMOFMP	CGM for MS Office w/ Metric Scaling port	SASGDCGM	Graph Device

例 3: SAS/GRAPH デバイスドライバとユニバーサルプリンタのレポートの生成

要素: PROC QDEVICE ステートメントオプション:OUT=
DEVICE statement
PRINTER statement

詳細

次の例では、EMF に落ち着くすべてのデバイス、PDF および SVG?ユニバーサルプリンタ向けの一般レポートを作成します。結果は WORK.MYREPORT データセットに書き込まれます。REPORT=オプションを指定しなかった場合、QDEVICE プロシジャでは、一般レポートが生成されます。

プログラム

```
proc qdevice out=myreport;
  device '*emf';
  printer pdf 'svg?';
run;
```

出力

次の図は、Viewtable ウィンドウに表示されるレポートの一部です。

アウトプット 52.2 *EMF デバイス、PDF プリンタ、SVG プリンタ向け出力データセットレポート*

VIEWTABLE: Work.Myreport

	NAME OF DEVICE OR PRINTER	DESCRIPTION OF DEVICE OR PRINTER	DRIVER MODULE	TYPE OF DEVICE OR PRINT
1	EMF	Enhanced Metafile Format Plus Extensions	SASGDDM	Shortcut Device
2	SASEMF	Enhanced Metafile Format	SASGDDM	Shortcut Device
3	PDF	Portable Document Format Version 1.4	SASPDPD	Universal Printer
4	SVGT	Transparent SVG	SASPDSV	Universal Printer
5	SVGZ	Compressed Scalable Vector Graphics 1.1	SASPDSV	Universal Printer

例 4: デフォルトのプリンタのレポートの生成

要素: PROC QDEVICE ステートメント
PRINTER statement

詳細

デフォルトでは、Windows 環境の SAS での印刷は、ユニバーサル印刷ではなく、デフォルトの Windows プリンタによって実行されます。したがって、QDEVICE プロシジャで **printer _PRINTER_** ステートメントを使用した場合は、異なる結果が出ます。Windows 環境では、NOUPRINT システムオプションがデフォルトで、レポートはデフォルトの Windows プリンタとつながるプリンタインターフェイスデバイスに基づきます。UNIX 環境では、UPRINT システムオプションが設定されているため、レポートはデフォルトの SAS ユニバーサルプリンタに基づきます。

REPORT=オプションを指定しないので、QDEVICE プロシジャでは、一般レポートが生成されます。OUT=オプションを指定しなければ、結果は SAS ログに書き込まれます。_PRINTER_キーワードによって、レポート対象のデフォルトプリンタが決定され、そのプリンタのレポートが生成されます。

詳細については、次のトピックを参照してください。

- “Printing” (*SAS Companion for Windows*)
- “UNIVERSALPRINT System Option” (*SAS Companion for Windows*)
- “Universal Printing” (*SAS Language Reference: Concepts*)

プログラム:Windows

```
proc qdevice;
  printer _PRINTER_;
run;
```

ログ:ログ:デフォルト Windows プリンタレポート

ログ 52.2 *デフォルト Windows プリンタ向け SAS ログ出力レポート*

```
1    proc qdevice; 2       printer _PRINTER_; 3       run; NOTE:The "\
\\wprt02nc0\clxmfpj21" printer will be used by default with the ODS PRINTER
destination.Name:WINPRTC Description:\\WPRT02NC0\CLXMFPJ21 Module:SASGDDMX
Type:Printer Interface Device Device Catalog:your-sas-path\sashelp Default
Typeface:SAS Monospace Font Style:Roman Font Weight:Normal Font Height:10 points
Font Version: mfgpctt-v4.4 Thu Sep 16 14:30:47 EDT 1999 Maximum Colors:2097152
Visual Color:True Color Color Support:RGB I/O Type:PRINTER Data Format:Host
Printer Height:10.67 inches Width:8.15 inches Ypixels:6392 Xpixels:4892
Rows(vpos):55 Columns(hpos):97 Left Margin:0.18 inches Minimum Left Margin:0.18
inches Right Margin:0.17 inches Minimum Right Margin:0.17 inches Bottom Margin:
0.18 inches Minimum Bottom Margin:0.18 inches Top Margin:0.17 inches Minimum Top
Margin:0.17 inches XxY Resolution:600x600 pixels per inch Compression
Enabled:Never Font Embedding:Never Animation:Unsupported
```

プログラム:UNIX

```
proc qdevice;
   printer _PRINTER_;
run;
```

ログ:UNIX 環境のデフォルトユニバーサルプリンタレポート

ログ 52.3 *UNIX 環境でのデフォルトユニバーサルプリンタ向け SAS ログ出力レポート*

```
1    proc qdevice; 2     printer _PRINTER_; 3     run; NOTE:The "PostScript
Level 1" printer will be used by default with the ODS PRINTER
destination.Name:PostScript Level 1 Description:Generic PostScript Level 1
Printer Module:SASPDPSL Type:Universal Printer Registry:SASHELP
Prototype:PostScript Level 1 (Color) Default Typeface:Cumberland AMT Typeface
Alias:Courier Font Style:Regular Font Weight:Normal Font Height:8 points Font
Version:Version 1.03 Maximum Colors:16777216 Visual Color:Direct Color Color
Support:CMYK Destination: sasprt.ps I/O Type:DISK Data Format:PostScript Height:
10 inches Width:7.5 inches Ypixels:3000 Xpixels:2250 Rows(vpos):81 Columns(hpos):
112 Left Margin:0.5 inches Minimum Left Margin:0 inches Right Margin:0.5 inches
Minimum Right Margin:0 inches Bottom Margin:0.5 inches Minimum Bottom Margin:0
inches Top Margin:0.5 inches Minimum Top Margin:0 inches XxY Resolution:300x300
pixels per inch Compression Enabled:Never Font Embedding:Option
Animation:Unsupported
```

例 5: FONT レポートの生成

要素: PROC QDEVICE ステートメントオプション
REPORT=
OUT=
PRINTER statement

詳細

最初の例では、プリンタで使用可能なすべてのプリンタ常駐フォントおよびシステムフォントのレポートが生成されます。結果は WORK.MYFONTS データセットに書き込まれます。

2 番目の例では、SAS プログラムで、マクロ、DATA ステップおよび PRINT プロシジャを使用して、デバイスのフォントリストが作成されます。

プログラム

```
proc qdevice report=font out=myfonts;
  printer 'postscript level 2';
run;
```

出力

次の出力は、**Viewtable** ウィンドウに表示されるレポートを示しています。

アウトプット 52.3 *フォントレポート向け出力データセット*



	NAME OF DEVICE OR	DESCRIPTION OF DEVICE OR PRINTER	TYPE OF DEVICE OR	LOCATION OF DEVICE OR	FONT TY
1	POSTSCRIPT LEVEL 2	Generic PostScript Level 2 Printer	Universal Printer	SASHELP	Adobe C

プログラム

```
/* Macro FONTLIST - Report fonts supported by a device */

%macro fontlist(type, name);

proc qdevice report=font out=fonts;
   &type &name;
   var font ftype fstyle fweight;
run;

data;
   set fonts;
   drop ftype;
   length type $16;
   if ftype = "System"
     then do;
        if substr(font,2,3) = "ttf" then type = "TrueType";
          else if substr(font,2,3) = "at1" then type = "Adobe Type1";
          else if substr(font,2,3) = "cff" then type = "Adobe CFF/Type2";
          else if substr(font,2,3) = "pfr" then type = "Bitstream PFR";
          else type = "System";
        if type ^= "System" then font = substr(font,7,length(font)-6);
          else if substr(font,1,1) = "@"
               then font = substr(font, 2,length(font)-1);
       end;
       else type = "Printer Resident";
run;

proc sort;
   by font;
run;

title "Fonts Supported by the %upcase(&name) &type";
```

```
proc print label;
   label fstyle="Style" fweight="Weight" font="Font" type="Type";
run;


%mend fontlist;



%fontlist(device, pcl5c)
```

プログラムの説明

マクロフォントリストを作成します。 %macro ステートメントで、マクロが開始されます。マクロに対する入力は、(デバイスでもプリンタでも)種類と、デバイスまたはプリンタの名前です。

```
/* Macro FONTLIST - Report fonts supported by a device */

%macro fontlist(type, name);
```

デバイスに対するデータセット fonts を作成します。 マクロ入力変数 type と name を使用して、QDEVICE プロシジャでフォントレポートを作成します。出力はデータセット fonts に書き込まれます。

```
proc qdevice report=font out=fonts;
   &type &name;
   var font ftype fstyle fweight;
run;
```

フォントの種類を分類します。 フォントは、System、TrueType、Adobe Type1、Adobe CFF/Type2、Bitstream PFR、または Printer Resident のいずれかの種類になります。

```
data;
   set fonts;
   drop ftype;
   length type $16;
   if ftype = "System"
     then do;
        if substr(font,2,3) = "ttf" then type = "TrueType";
          else if substr(font,2,3) = "at1" then type = "Adobe Type1";
          else if substr(font,2,3) = "cff" then type = "Adobe CFF/Type2";
          else if substr(font,2,3) = "pfr" then type = "Bitstream PFR";
          else type = "System";
        if type ^= "System" then font = substr(font,7,length(font)-6);
          else if substr(font,1,1) = "@"
               then font = substr(font, 2,length(font)-1);
       end;
       else type = "Printer Resident";
run;
```

フォントデータセットをフォント名順に並べ替えます。

```
proc sort;
   by font;
run;
```

デバイスまたはプリンタのフォントを印刷します。

```
   title "Fonts Supported by the %upcase(&name) &type";

   proc print label;
      label fstyle="Style" fweight="Weight" font="Font" type="Type";
   run;
```

マクロを終了します。

```
   %mend fontlist;
```

マクロ&fontlist を使用して、PCL5c デバイスのデータセットを作成、出力します。

```
   %fontlist(device, pcl5c)
```

出力

アウトプット 52.4 *PCL5c デバイスでサポートされるフォントの表示の一部*

Fonts Supported by the PCL5C device

Obs	Font	Style	Weight	Type
1	CG Courier	Italic	Bold	Printer Resident
2	CG Courier	Italic	Normal	Printer Resident
3	CG Courier	Roman	Bold	Printer Resident
4	CG Courier	Roman	Normal	Printer Resident
5	CG Letter Gothic	Italic	Bold	Printer Resident
6	CG Letter Gothic	Italic	Normal	Printer Resident
7	CG Letter Gothic	Roman	Bold	Printer Resident
8	CG Letter Gothic	Roman	Normal	Printer Resident
9	CG Marigold	Roman	Normal	Printer Resident
10	CG Times	Italic	Bold	Printer Resident
11	CG Times	Italic	Normal	Printer Resident
12	CG Times	Roman	Bold	Printer Resident
13	CG Times	Roman	Normal	Printer Resident
14	EW Arial	Italic	Bold	Printer Resident
15	EW Arial	Italic	Normal	Printer Resident
16	EW Arial	Roman	Bold	Printer Resident
17	EW Arial	Roman	Normal	Printer Resident
18	EW CG Times	Italic	Bold	Printer Resident
19	EW CG Times	Italic	Normal	Printer Resident
20	EW CG Times	Roman	Bold	Printer Resident

例 6: デバイスオプションレポートの生成

要素: PROC QDEVICE ステートメントオプション
- REPORT=
- SUPPORT=
- OUT=

DEVICE statement

詳細

次の例では、PNG デバイス向けデバイスオプション(DEVOPTIONS)レポートを作成します。レポートは出力データセットに書き込まれます。

プログラム

```
proc qdevice report=devoption support=all out=devop;
  device png;
run;

proc print data=devop;
   var bit bitstring option odesc;
   where Support="Yes";
   title "Supported PNG Device Options";
run;
```

プログラムの説明

PNG デバイスのオプションをレポートします。 REPORT=DEVOPTION オプションは、デバイスオプションレポートの作成を指定するものです。SUPPORT=ALL は、すべてのデバイス機能をレポートするよう指定するものです。オプション OUT=DEVOP は、データセット WORK.DEVOP を作成するものです。DEVICE PNG ステートメントは、PNG デバイスに関してレポートするよう指定するものです。

```
proc qdevice report=devoption support=all out=devop;
  device png;
run;
```

デバイスオプションレポートを印刷します。 WORK.DEVOP データセットを印刷すると、印刷されたレポートには、Support="Yes"であるデバイスオプションの BIT 変数、BITSTRING 変数、OPTION 変数、ODESC 変数が表示されます。

```
proc print data=devop;
   var bit bitstring option odesc;
   where Support="Yes";
   title "Supported PNG Device Options";
run;
```

出力

アウトプット 52.5 *PNG デバイス向け出力データセット*

Supported PNG Device Options

Obs	BIT	BITSTRING	OPTION	ODESC
1	0	8000000000000000	GDCIRCLEARC	Hardware is capable of drawing circles
2	1	4000000000000000	GDPIEFILL	Device has hardware pie-fill capability
4	3	1000000000000000	GDCRT	Hardware is a CRT or the device acts like a CRT
5	4	0800000000000000	GDTRANSPARENCY	Device supports transparency
6	5	0400000000000000	GDPOLYGONFILL	Device has polygonfill capability
8	7	0100000000000000	GDRGB	Hardware is capable of defining colors in one or more color spaces
9	8	0080000000000000	GDMBPOLY	Hardware can draw polygons with multiple boundaries
10	9	0040000000000000	GDOPACITY	Hardware is capable of supporting opacity
13	14	0002000000000000	GDHRDCHR	Hardware characters are supported by the device
14	15	0001000000000000	GDXLIMIT	There is no limit on max value allowed for x coordinate
15	16	0000800000000000	GDYLIMIT	There is no limit on max value allowed for y coordinate
17	18	0000200000000000	GDTXJUSTIFY	Hardware is capable of justifying proportional text
20	24	0000008000000000	GDUNICODE	Device supports the use of the Unicode font attribute
21	25	0000004000000000	GDPOLYLINE	Hardware is capable of supporting polylines
24	28	0000000800000000	GDTRUETYPE	Device supports the use of TrueType fonts
32	36	0000000008000000	GDIMAGE	Device is capable of drawing images
35	39	0000000001000000	GDIMGROTATE	Device is incapable of doing image rotation
36	40	0000000000800000	GDTRUECOLOR	Hardware is a 24-bit true color device
37	41	0000000000400000	GDFONTATTR	Device supports setting font attributes
38	42	0000000000200000	GDSCANLNFNT	Device will use scan line font rendering
40	44	0000000000080000	GDTEXTCLIP	Hardware will clip text at the device limits
46	50	0000000000002000	GDAUTOSIZE	Autosize text to fit rows and columns
50	54	0000000000000200	GDPOLYOUTLINE	Device draws polygon outlines
51	55	0000000000000100	GDPRINTERPATH	Device temporarily sets printerpath to that of the device name
52	56	0000000000000080	GDOPTPASSTHRU	PAPERSIZE option sets default value of PAPERSIZE goption
53	57	0000000000000040	GDPIEOUTLINE	Driver draws pie slice outlines (empty pies)
54	60	0000000000000008	GDNOROTATE	Force the graphics sublib not to rotate the graph

例 7: レポートに対するユーザーライブラリおよびユーザーカタログの指定

要素: PROC QDEVICE ステートメントオプション
CATALOG=
DEVLOC=
PRINTER statement

詳細

この例では、PROC QDEVICES ステートメントの DEVLOC=オプションと CATALOG=オプションを使用して、ユーザー指定のライブラリおよびカタログにおける GIF プリンタ用の一般レポートを作成します。結果は SAS ログに書き込まれます。

プログラム

```
libname devlib 'c:\em';
proc qdevice report=general devloc=devlib cat=mydevices;
   device gif;
run;
```

プログラムの説明

デバイスのライブラリおよびカタログの割り当て

```
libname devlib 'c:\em';
proc qdevice report=general devloc=devlib cat=mydevices;
   device gif;
run;
```

ログ

ログ 52.4 *特定のデバイスライブラリおよびカタログに対する SAS ログレポート*

```
144  libname devlib 'c:\em'; NOTE:Libref DEVLIB was successfully assigned as
follows:Engine:        V9 Physical Name:C:\em 145  proc qdevice report=general
devloc=devlib cat=mydevices; 146     device gif; 147  run; Name:GIF
Description:Graphics Interchange Format RGB Color/Alpha Blending Module:SASGDDMX
Type:Shortcut Device Device Catalog:C:\em Prototype:GIF Default
Typeface:Cumberland AMT Typeface Alias:Courier Font Style:Regular Font
Weight:Normal Font Height:8 points Font Version:Version 1.03 Maximum Colors:
16777216 Visual Color:True Color Color Support:RGBA Destination: sasprt.gif I/O
Type:DISK Data Format:GIF Height:6.25 inches Width:8.33 inches Ypixels:600
Xpixels:800 Rows(vpos):50 Columns(hpos):114 Left Margin:0 inches Minimum Left
Margin:0 inches Right Margin:0 inches Minimum Right Margin:0 inches Bottom
Margin:0 inches Minimum Bottom Margin:0 inches Top Margin:0 inches Minimum Top
Margin:0 inches XxY Resolution:96x96 pixels per inch Compression Enabled:Always
Compression Method:LZW Font Embedding:Never Animation:Enabled
```

53 章

RANK プロシジャ

概要: RANK プロシジャ

RANK プロシジャの動作について

RANK プロシジャでは、SAS データセットの全オブザベーションにわたって 1 つ以上の数値変数のランクが計算され、そのランクが新しい SAS データセットに出力されます。PROC RANK 単独では、印刷出力は作成されません。

データのランク付け

次の出力は、単純な PROC RANK ステップによる 1 変数の値のランク付け結果を示しています。この例では、新しいランク付け変数が、4 日間の試合での 5 人のゴルファーの最終順位を示しています。最も打数の少ない選手が 1 位になります。次のステートメントでは、出力が生成されます。

```
proc rank data=golf out=rankings;
   var strokes;
   ranks Finish;
run;

proc print data=rankings;
run;
```

アウトプット 53.1 *最小ランク値の最小変数値への割り当て*

```
The SAS System                                1 Obs    Player    Strokes    Finish 1     Jack
279         2 2     Jerry       283         3 3     Mike        274         1 4     Randy
296         4 5     Tito        302         5
```

次の出力では、市議会議員候補者が地区別に選挙の得票数によってランク付けされています。候補者は在職期間によってもランク付けされています。

この例では、PROC RANK で次のタスクを実行できることが示されます。

- ランキングの順序を逆にして、最大値のランクを 1 に、その次に大きな値のランクを 2 に、というようにランク付けします。
- オブザベーションを、複数の変数値順に、別々にランク付けします。
- BY グループ内でオブザベーションをランク付けします。
- タイ値を処理します。

このレポートを生成するプログラムの説明については、“例 2: BY グループ内で値をランク付けする” (1557 ページ)を参照してください。

アウトプット 53.2 *各 BY グループ内での最小ランク値の最大変数値への割り当て*

```
Results of City Council Election                    1 ---------------------------------
District=1 ---------------------------------- Vote    Years Obs    Candidate    Vote    Years
Rank     Rank 1     Cardella     1689      8        1       1 2     Latham       1005      2
3        2 3     Smith        1406      0        2       3 4     Walker        846      0
4        3 N = 4 ---------------------------------- District=2 ---------------------------------
Vote     Years Obs    Candidate      Vote    Years    Rank     Rank 5     Hinkley         912
0         3       3 6     Kreitemeyer    1198      0        2       3 7     Lundell         2447
6         1       1 8     Thrash          912      2        3       2 N = 4
```

概念:RANK プロシジャ

コンピュータリソース

PROC RANK では、ランク付けするすべての変数について、すべてのオブザベーションの該当変数の値がメモリに格納されます。

統計アプリケーション

ランクは変数値の分布の調査に役立ちます。ランクを *n* または *n*+1 で除算して 0 から 1 までの範囲の値を得て、その値によって累積分布関数を推定します。これらの分数ランクに逆累積分布関数を適用すると、確率四分位スコアを得られます。これらのスコアを元の値と比較すると、分布に対する適合度を判断できます。たとえば、データのセットが正規分布している場合、正規スコアは元の値の線形関数になり、元の値に対するスコアのプロットは直線になります。

多くのノンパラメトリックな手法は、変数のランクの分析に基づいています。

- ランクに適用される 2 標本の *t* 検定は、その有意水準の *t* 近似を使用した Wilcoxon のランク和検定に相当します。ランクではなく正規スコアに *t* 検定を適用した場合、その検定は Van der Waerden 検定に相当します。*t* 検定を中央値スコア (GROUPS=2)に適用した場合、その検定は中央値検定に相当します。
- ランクに適用される一元配置分散分析は、Kruskal-Wallis の *k* 標本検定に相当します。また、多くの場合、パラメトリックなプロシジャで生成される F 検定をランクに適用した結果は、Kruskal-Wallis で使用されるX^2近似より優れています。この検定は、範囲を広げて他のランクスコアに使用できます(Quade 1966)。
- ブロック計画に対する Friedman の二元分析は、BY グループ内のランク付けを行い、次に、そのランクについて分散の主効果分析を実行することによって得られます(Conover 1998)。
- Iman と Conover (1979)の提唱した手法でランクの変換を行うと、回帰関係を調べられます。

タイ値の処理

PROC RANK で値をランク付けする際、BY グループ内で値の等しい分析変数が 2 つ以上ある場合は、データのタイ値が存在します。値の区別ができず、また、通常はランクの適切な基準となり得る明白な情報がこれ以上ないため、PROC RANK では、その値に異なるランクは割り当てられません。タイ値には、任意に異なるランクを割り当てられます。ただし、ランクを使用したノンパラメトリックな統計検定などの統計アプリケーションでは、通常はタイ値と同じランクを割り当てます。

これらの統計検定では通例、データ元は連続分布と見なされ、タイの確率は理論的にはゼロです。実際には、測定の不正確さや、デジタルコンピュータ内の表現の正確性の限界などの理由により、しばしばタイ値が発生します。また、これらの統計検定では、通常、タイ値のグループに平均ランクを割り当てます。ランク合計が保持され、したがって、累積分布関数の推定値が歪められないため、平均ランクの割り当てをお勧めします。

統計量内外の適用のために、RANK プロシジャでは、タイ値の処理を制御する TIES= オプションが提供されます。このオプションのデフォルト値は、指定したランク付け方法

またはスコアリング方法によって決定されるもので、PROC RANK ステートメントのオプションで指定できます。ランク付け方法とスコアリング方法については、TIES=LOW、TIES=HIGH または TIES=MEAN の場合、最初はタイ値が区別可能な場合と同様に処理されます。これらの方法はすべて、BY グループ内の分析変数値の並べ替えから始まり、次に、数列内の位置を示す序数が各非欠損値に割り当てられます。

続いて、非スコアリング方法の場合、PROC RANK では、TIES=LOW により最小値を選択するか、TIES=HIGH により最大値を選択するか、または TIES=MEAN によりタイ値グループの序数の平均値を計算することによって、タイ値が解決されます。この後、PROC RANK では、さらに 1 つ以上の変換(スケーリング、変換、切り捨てなど)を加えることにより、この値からランクを得られます。

スコアリング方法には、正規スコアリングと Savage スコアリングが含まれます。非スコアリング方法には、順序尺度ランク付け、デフォルト、および FRACTION、NPLUS1、GROUPS=、PERCENT オプションによって要求される方法が含まれます。スコアリング方法 NORMAL=および SAVAGE では、PROC RANK によって、タイ値がデータ内に存在しない場合と同じように、適切な式によって確率分位点スコアが得られます。次に、PROC RANK では、タイ値を解決するために、タイグループ内の全スコアの最小値が選択されるか、最大値が選択されるか、または平均が計算されます。

TIES=DENSE の場合、すべてのランク付けおよびスコアリング方法において、タイ値は区別できないものとして処理され、タイグループ内の各値に同じ序数が割り当てられます。他の TIES=解決方法と同様に、すべてのランク付けおよびスコアリング方法は、分析変数値の並べ替えから始まり、次に、序数が割り当てられます。ただし、タイ値のグループは単一値として処理されます。グループに割り当てる序数は、そのグループの前の値に序数が割り当てられている場合、その序数と+1 だけ異なります。グループの直後の値に序数が割り当てられている場合は、その序数と-1 だけ異なります。したがって、BY グループ内の最小序数は 1 で、最大序数は BY グループの一意の非欠損値数です。

序数を割り当てた後、PROC RANK は、スケーリングのために、非欠損値数のかわりに一意の非欠損値数を使用してランクとスコアを計算します。累積分布関数推定値がひずむ傾向があるため、一般に、密度の高いランク付けでは、ノンパラメトリックな統計検定での使用は受け入れられません。

PROC RANK が、分析変数の内部数値に関する計算の基礎を成すことに注意してください。プロシジャでは、分析前にこれらの値のフォーマットや丸めが行われることはありません。値の内部表現が異なる場合は、たとえわずかな違いであっても、PROC RANK でタイ値として処理されることはありません。これが各自のデータに関係する場合は、PROC RANK を呼び出す前に、適切な数量で分析変数を丸めます。ROUND 関数の詳細については、“ROUND Function” (*SAS Functions and CALL Routines: Reference*)を参照してください。

PROC RANK のデータベース内処理

データベース内処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。セキュリティの強化は、機密データを DBMS から抽出する必要がないため可能です。より迅速な処理が可能になる理由は次のとおりです。

- データが、比較的低速なネットワーク接続を介して移送されるかわりに、高速の二次記憶装置を使用して、DBMS 上でローカル操作されます。
- DBMS では、より多くのリソースが用意されている可能性があります。
- DBMS には、実行するクエリを高度にパラレルで、スケーラブルな方法で最適化できる能力が備えられている可能性があります。

PROC RANK のデータベース内処理では、次のデータベース管理システムがサポートされています。

- Aster
- DB2
- Greenplum
- HAWQ
- Impala
- Netezza
- Oracle
- SAP HANA
- Teradata

表統計量の存在は、RANK プロシジャのデータベース内処理のパフォーマンスに影響を及ぼします。DBMS が表統計量を自動生成するように構成されていない場合は、受け入れ可能なデータベース内パフォーマンスを達成するために表統計量の手動生成が必要になる可能性があります。

注: DB2 の場合、最小の入力テーブル以外についてはすべて、表統計量の生成(自動か手動のどちか)を強くお勧めします。

RANK プロシジャの入力データセットが、サポート DBMS との SAS/ACCESS インターフェイスによる通常の行の取得元であるデータベース内に存在するテーブルまたはビューの場合、PROC RANK では、DBMS 内の大部分またはすべての作業を実行できます。そのようなデータベース内処理が実行可能かどうかを決定する因子は、他にもいくつかあります。データベース内処理は、次の環境では発生しません。

- 入力データセットで RENAME=データセットオプションを指定した場合。
- WHERE ステートメントが RANK プロシジャのコンテキストで表示されているか、入力データセットで WHERE=データセットオプションが指定されており、WHERE ステートメントまたはオプションに、DBMS に相当するものがない SAS 関数、または DBMS 内で SAS による使用のためのインストールが未実行の出力形式への参照が含まれる場合。
- BY ステートメントで指定した変数のいずれかに、関連する出力形式がある場合。PROC RANK では、フォーマットされた BY 変数のデータベース内処理はサポートされていません。
- FORMAT ステートメントがプロシジャコンテキスト内に出現し、BY ステートメントで指定した変数に適用された場合、データベース内処理は実行できません。RANK では、フォーマットされた BY 変数のデータベース内処理はサポートされていません。DBMS では、FORMAT または ATTRIB ステートメントがプロシジャコンテキスト内に出現する場合のみ、出力形式を変数に関連付けられます。
- TIES=CONDENSE オプションでは、Oracle DBMS における RANK プロシジャのデータベース内処理がサポートされていません。このオプションを使用した場合、SQL 生成およびデータベース内処理実行が行われません。

PROC RANK による DBMS 内のデータ処理が可能な場合、SQL クエリが生成されます。PROC RANK のデータベース内呼び出し時に生成される SQL クエリの構造は、次のようないくつかの因子に依存しています。

- ターゲット DBMS
- 使用するランク付け方法
- ランク付けする変数の数

- BY ステートメントと WHERE ステートメントの包含
- 使用する PROC RANK オプション(TIES=や DESCENDING など)

SQL クエリでは、必要な計算が表現され、DBMS にサブミットされます。このクエリの結果は、RANK プロシジャの出力先がテーブルの場合は DBMS 内に新しいテーブルとして残され、そうでなければ、SAS に返されます。MSGLEVEL オプションと SQLGENERATION オプションの設定によって、SAS ログにメッセージを出力するかどうかが決定されます。生成された SQL を検証するには、SQL_IP_TRACE オプションまたは SASTRACE=オプションを設定します。SQL_IP_TRACE によって、PROC RANK で生成された SQL が示されます。詳細については、*SAS/ACCESS for Relational Databases: Reference* の SASTRACE=オプション、または *SAS(R) Analytics Accelerator 1.3 for Teradata: Guide* の SQL_IP_TRACE オプションを参照してください。

SAS プロシジャのデータベース内実行に影響を及ぼすシステムオプション、ライブラリオプション、データセットオプション、ステートメントオプションの設定の詳細については、*SAS/ACCESS for Relational Databases: Reference* の SQLGENERATION= LIBNAME オプションおよび SQLGENERATION=オプションを参照してください。

構文: RANK プロシジャ

ヒント: ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントは、RANK プロシジャと併用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

グローバルステートメントを使用することもできます。リストは、“グローバルステートメント” (24 ページ)および“Global Statements” (*SAS Statements: Reference*)を参照してください。

発生するデータベース内処理に対し、データは、SAS のデータベース内処理用として適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、“PROC RANK のデータベース内処理” (1546 ページ)を参照してください。

PROC RANK <*option(s)*>;
  BY <DESCENDING> *variable-1*
    <<DESCENDING> *variable-2* …>
    <NOTSORTED>;
  VAR *data-set-variables(s)*;
  RANKS *new-variables(s)*;

ステートメント	タスク	例
“PROC RANK ステートメント”	SAS データセットの 1 つ以上の数値変数のランク計算、および新しい SAS データセットへのランク出力	Ex. 1, Ex. 2, Ex. 3
“BY ステートメント”	BY グループごとの別々のランクセットの計算	Ex. 2, Ex. 3
“RANKS ステートメント”	ランク割り当て先の変数の識別	Ex. 1, Ex. 2
“VAR ステートメント”	ランク付けする変数の指定	Ex. 1, Ex. 2, Ex. 3

PROC RANK ステートメント

1 つ以上の数値変数のランクを計算します。

制限事項: 単一の PROC RANK ステップでは、ランク付けの方法を 1 つしか指定できません。

例: “例 1: 複数の変数の値をランク付けする” (1555 ページ)
“例 2: BY グループ内で値をランク付けする” (1557 ページ)
“例 3: ランクに基づいてオブザベーションをグループ分けする” (1559 ページ)

構文

PROC RANK <*option(s)*>;

オプション引数の要約

Compute fractional ranks

NPLUS1
: 各ランクを分母 *n*+1 で除算して、分数ランクを計算します。

Create an output data set

OUT=*SAS-data-set*
: 出力データセットを指定します。

Preserve values

PRESERVERAWBYVALUES
: すべての BY 変数の未加工値を保持します

Reverse the order of the rankings

DESCENDING
: ランクを逆順にします。

Specify how to rank tied values

TIES=HIGH | LOW | MEAN | DENSE
: タイデータ値の正規スコアやランクの計算方法を指定します。

Specify the input data set

DATA=*SAS-data-set*
: 入力 SAS データセットを指定します。

Specify the ranking method

FRACTION
: 各ランクを、ランク付け変数が非欠損値のオブザベーション数で除算し、分数ランクを計算します。

GROUPS=*number-of-groups*
: 0 から *number-of-groups* - 1 までの範囲のグループ値を割り当てます。

NORMAL=BLOM | TUKEY | VW
: ランクから正規スコアを計算します。

PERCENT
: ランクにおける非欠損値を含むオブザベーションの割合を計算します。

SAVAGE
ランクから Savage(または指数)スコアを計算します。

オプション引数

DATA=*SAS-data-set*
入力 SAS データセットを指定します。

制限事項 別のユーザーが同時にデータセットを更新している場合、同時アクセスをサポートするエンジンでは PROC RANK を使用できません。

データベース内処理を実行するためには、データセットの指定によって、サポートされている DBMS に存在するテーブルを参照する必要があります。

参照項目 “入力データセット” (25 ページ)

DESCENDING
ランクを逆順にします。DESCENDING を指定すると、最大値にランク 1、その次に大きな値にランク 2、というようにランク付けされます。指定しなければ、昇順にランク付けされます。

参照項目 “例 1: 複数の変数の値をランク付けする” (1555 ページ)

“例 2: BY グループ内で値をランク付けする” (1557 ページ)

FRACTION
各ランクを、ランク付け変数が非欠損値のオブザベーション数で除算し、分数ランクを計算します。

別名 F

操作 TIES=HIGH が、FRACTION オプション指定時のデフォルトです。TIES=HIGH を指定した場合、分数ランクは右連続な累積経験分布関数の値と見なされます。

参照項目 NPLUS1 オプション

GROUPS=*number-of-groups*
0 から *number-of-groups* - 1 までの範囲のグループ値を割り当てます。通常の指定は、パーセント点を示す GROUPS=100、十分位数を示す GROUPS=10、および四分位数を示す GROUPS=4 です。たとえば、GROUPS=4 により、元の値は 4 つのグループに分かれます。たとえば、GROUPS=4 を指定すると、元の値が 4 グループに分割され、最小値にはデフォルトで四分位数値 0、最大値には四分位数値 3 が割り当てられます。

グループ値の計算式は次の通りです。

$$\mathrm{FLOOR}(rank * k/(n+1))$$

FLOOR は FLOOR 関数、*rank* は値の順位、*k* は GROUPS=の値、*n* は TIES=LOW、TIES=MEAN および TIES=HIGH でランク付け変数が非欠損値のオブザベーション数です。TIES=DENSE の場合、*n* は一意の非欠損値を含むオブザベーション数です。

オブザベーション数がグループ数で均等に割り切れる場合、グループの境界にタイ値がなければ、各グループのオブザベーション数は同じになります。多くのタイ値を含む変数によってオブザベーションをグループ化すると、結果的にグループの

バランスが悪くなる可能性があります。これは、PROC RANK では常に、同じ値のオブザベーションは同じグループに割り当てられるためです。

ヒント　グループ値を逆順に並べ替えるには、DESCENDING を使用します。

参照項目　“例 3: ランクに基づいてオブザベーションをグループ分けする” (1559 ページ)

NORMAL=BLOM | TUKEY | VW
ランクから正規スコアを計算します。結果変数の表示は正規分布します。*n* は、TIES=LOW、TIES=MEAN および TIES=HIGH でランク付け変数が非欠損値のオブザベーション数です。TIES=DENSE の場合、*n* は一意の非欠損値を含むオブザベーション数です。式は次のとおりです。

BLOM
$y_i=\Phi^{-1}((r_i-3/8)/(n+1/4))$

TUKEY
$y_i=\Phi^{-1}((r_i-1/3)/(n+1/3))$

VW
$y_i=\Phi^{-1}((r_i)/(n+1))$

これらの式では、Φ^{-1} は正規累積の逆(PROBIT)関数、r_i は i 番目のオブザベーションのランク、n はランク付け変数の非欠損オブザベーション数です。

VW は van der Waerden を表します。 NORMAL=VW,を指定すると、スコアをノンパラメトリックな位置の検定に使用できます。3 つの正規スコアはすべて、正規分布の正確な期待順序統計の近似値です(*正規スコア*とも呼ばれます)。BLOM 版の表示は、他の方法よりわずかに適合度が高くなります(Blom 1958; Tukey 1962)。

制限事項　NORMAL=オプションを使用すると、データベース内処理は行われません。

操作　TIES=オプションを指定すると、PROC RANK では、非タイ値に基づいてランクから正規スコアが計算され、TIES=の指定が結果のスコアに適用されます。

NPLUS1
各ランクを分母 *n*+1 で除算して、分数ランクを計算します。このとき、*n* は TIES=LOW、TIES=MEAN および TIES=HIGH でランク付け変数が非欠損値のオブザベーション数です。TIES=DENSE の場合、*n* は一意の非欠損値を含むオブザベーション数です。

別名　FN1、N1

操作　TIES=HIGH が、NPLUS1 オプション指定時のデフォルトです。

参照項目　FRACTION オプション

OUT=*SAS-data-set*
出力データセットを指定します。*SAS-data-set* が存在しない場合は、PROC RANK で作成されます。OUT=を省略した場合、DATA*n* 命名規則を使用してデータセット名が指定されます。

操作　データベース内処理が実行され、OUT=もサポートされた DBMS テーブルを参照している場合、IN=と OUT=の両方で同じライブラリを参照すると、すべ

ての処理が DBMS で実行可能になり、結果的に出力テーブルが直接作成されます。この場合、SAS には結果が返されません。

PRESERVERAWBYVALUES
すべての BY 変数の未加工値を保持します (その変数の出力データセットへのプロパゲート時)。PRESERVERAWBYVALUES オプションを指定せずに、BY 変数を 1 つ指定した場合は、各 BY グループの代表値が出力データセットに書き込まれます。複数の BY 変数を指定した場合は、各 BY グループの代表値セットが出力データセットに書き込まれます。

PERCENT
各ランクを、変数が非欠損値のオブザベーション数で除算し、その結果に 100 を乗算して、百分率を求めます。*n* は、TIES=LOW、TIES=MEAN および TIES=HIGH でランク付け変数が非欠損値のオブザベーション数です。TIES=DENSE の場合、*n* は一意の非欠損値を含むオブザベーション数です。

別名 P

操作 TIES=HIGH が、PERCENT オプション指定時のデフォルトです。

ヒント 累積百分率を計算するには PERCENT を使用しますが、パーセント点を計算するには GROUPS=100 を使用します。

SAVAGE
次の式(Lehman 1998)によってランクから Savage(または指数)スコアを計算します。

$$y_i = \left[\sum_{j=n-r_i+1} \left(\frac{1}{j}\right) \right] - 1$$

操作 TIES=オプションを指定すると、PROC RANK では、非タイ値に基づいてランクから Savage スコアが計算され、TIES=の指定が結果のスコアに適用されます。

TIES=HIGH | LOW | MEAN | DENSE
タイデータ値の正規スコアやランクの計算方法を指定します。

HIGH
対応するランクのうちの最大値を割り当てます(NORMAL=指定時は正規スコアの最大値)。

LOW
対応するランクのうちの最小値を割り当てます(NORMAL=指定時は正規スコアの最小値)。

MEAN
対応するランクの平均値を割り当てます(NORMAL=指定時は正規スコアの平均値)。

DENSE
タイ値を単一の順序統計量として扱って、スコアとランクを計算します。デフォルトの方法では、ランクは、1 の数から始まり、ランク付けされている変数の一意な非欠損値数で終わる連続した整数です。タイ値には同じランクが割り当てられます。

注: CONDENSE は、DENSE の別名です。

デフォルト　MEAN (FRACTION オプションも PERCENT オプションも無効の場合)。

操作　NORMAL=オプションを指定すると、TIES=の指定が、正規スコアの計算に使用するランクではなく、正規スコアに適用されます。

参照項目　“タイ値の処理” (1545 ページ)

“例 1: 複数の変数の値をランク付けする” (1555 ページ)

“例 2: BY グループ内で値をランク付けする” (1557 ページ)

BY ステートメント

BY グループごとに別々のランクセットを作成します。

操作: BY ステートメントで NOTSORTED オプションを指定すると、データベース内処理を実行できません。

FORMAT ステートメントなどを使用して、入力データセットの BY 変数のいずれかにフォーマットすると、データベース内処理が実行されません。

参照項目: “BY” (68 ページ)

“例 3: ランクに基づいてオブザベーションをグループ分けする” (1559 ページ)

例: “例 2: BY グループ内で値をランク付けする” (1557 ページ)

“例 3: ランクに基づいてオブザベーションをグループ分けする” (1559 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* …>
<NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは指定するすべての変数によって並べ替えるか、適切にインデックスを付ける必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
オブザベーションが BY ステートメントの文字 DESCENDING の直後に続く変数で降順に並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。オブザベーションは、時系列などの別の方法でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定した場合は、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ別の BY グループとして扱います。

SAS/ACCESS エンジンを使用している場合は、BY ステートメントを指定すると、データが常に並べ替え後の順序で返されます。NOTSORTED オプションを指定すると、オプションは無視され、データベース内処理が実行されます。

RANKS ステートメント

ランク値に対する新しい変数を作成します。

デフォルト: RANKS ステートメントを省略した場合、出力データセットではランク値が元の変数値と置き換えられます。

要件 RANKS ステートメントを使用する場合は、VAR ステートメントも使用する必要があります。

例: “例 1: 複数の変数の値をランク付けする” (1555 ページ)
“例 2: BY グループ内で値をランク付けする” (1557 ページ)

構文

RANKS *new-variables(s)*;

必須引数

new-variable(s)
VAR ステートメントでリストされる変数のランクを含める新しい変数を 1 つ以上指定します。RANKS ステートメントで最初にリストされた変数には、VAR ステートメントで最初にリストされた変数のランクが含まれます。RANKS ステートメントで 2 番目にリストされた変数には、VAR ステートメントで 2 番目にリストされた変数のランクが含まれ、以下も同様に処理されます。

VAR ステートメント

入力変数を指定します。

デフォルト: VAR ステートメントを省略した場合、PROC RANK では、入力データセットの全数値変数のランクが計算されます。

例: “例 1: 複数の変数の値をランク付けする” (1555 ページ)
“例 2: BY グループ内で値をランク付けする” (1557 ページ)
“例 3: ランクに基づいてオブザベーションをグループ分けする” (1559 ページ)

構文

VAR *data-set-variables(s)*;

必須引数

data-set-variable(s)
ランク計算対象の変数を 1 つ以上指定します。

詳細

VAR ステートメントと RANKS ステートメントの使用

RANKS ステートメントの使用時には、VAR ステートメントが必須です。これらのステートメントをあわせて使用すると、RANK ステートメントで名前を指定したランク付け変数が作成されます。この変数は、VAR ステートメントで指定した入力変数に対応しています。RANKS ステートメントを省略した場合、出力データセットではランク値が元の値と置き換えられます。

結果:RANK プロシジャ

欠損値

ランクまたはランクスコアが出力データセットで元の値と置き換えられる際、欠損値はランク付けされず、欠損したままです。

出力データセット

RANK プロシジャでは、ランクまたはランクスコアを含む SAS データセットが作成されますが、印刷出力は作成されません。出力データセットを印刷するには、PROC PRINT、PROC REPORT または別の SAS レポートツールを使用します。

出力データセットには、入力データセットの全変数に加えて、RANK ステートメントで指定した変数が含まれます。RANKS ステートメントを省略した場合、出力データセットではランク値が元の変数値と置き換えられます。

数値精度

データベース内処理については、SQL で RANK プロシジャによって表現される演算処理とその実行順序は、SAS 内で実行される処理と基本的に同じです。ただし、データベース内処理の結果は、SAS での直接の結果と比較して、数値の差が小さくなる場合があります。

例: RANK プロシジャ

例 1: 複数の変数の値をランク付けする

要素: PROC RANK ステートメントオプション
DESCENDING
TIES=

RANKS statement
VAR ステートメント

他の要素: PRINT プロシジャ

詳細

この例では、次のアクションを実行します。

- ランクを逆順にして、最大値にランク 1 を指定
- 可能な限り最高のランクをタイ値に割り当て
- ランク付け変数を作成し、元の変数とともに印刷

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data cake;
   input Name $ 1-10 Present 12-13 Taste 15-16;
   datalines;
Davis      77 84
Orlando    93 80
Ramey      68 72
Roe        68 75
Sanders    56 79
Simms      68 77
Strickland 82 79
;

proc rank data=cake out=order descending ties=low;

   var present taste;
   ranks PresentRank TasteRank;
run;

proc print data=order;
   title "Rankings of Participants' Scores";
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、SAS ジョブが開始する日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

CAKE データセットを作成します。 このデータセットには、ケーキ作りコンテストにおける、各参加者の姓、出来ばえのスコア、および味のスコアが含まれます。

```
data cake;
   input Name $ 1-10 Present 12-13 Taste 15-16;
   datalines;
Davis      77 84
Orlando    93 80
Ramey      68 72
```

```
Roe         68 75
Sanders     56 79
Simms       68 77
Strickland 82 79
;
```

降順で数値変数のランクを生成し、出力データセット ORDER を作成します。 DESCENDING で、ランクが逆順になり、最高スコアにランク 1 が指定されます。TIES=LOW で、可能な限り最高のランクがタイ値に指定されます。OUT=で、出力データセット Order が作成されます。

```
proc rank data=cake out=order descending ties=low;
```

ランクを含む新しい変数を 2 つ作成します。 VAR ステートメントで、ランク付けする変数が指定されます。RANKS ステートメントで、2 つの新しい変数 PresentRank と TasteRank が作成され、それぞれに変数 Present と Taste のランクが含められます。

```
   var present taste;
   ranks PresentRank TasteRank;
run;
```

データセットを出力します。 PROC PRINT で、ORDER データセットが印刷されます。TITLE ステートメントでタイトルを指定します。

```
proc print data=order;
   title "Rankings of Participants' Scores";
run;
```

出力:リスト

アウトプット 53.3 *参加者のスコアのランキング*

```
Rankings of Participants' Scores                        1 Present    Taste Obs    Name
Present    Taste      Rank      Rank 1     Davis            77        84        3         1 2
Orlando          93        80         1         2 3     Ramey            68        72
4         7 4     Roe              68        75        4         6 5     Sanders          56
79        7         3 6     Simms            68        77        4         5 7     Strickland
82        79        2         3
```

例 2: BY グループ内で値をランク付けする

要素: PROC RANK ステートメントオプション
- DESCENDING
- TIES=

BY ステートメント

RANKS statement

VAR ステートメント

他の要素: PRINT プロシジャ

詳細

この例では、次のアクションを実行します。

- BY グループ内のオブザベーションを別々にランク付け
- ランクを逆順にして、最大値にランク 1 を指定
- 可能な限り最高のランクをタイ値に割り当て
- ランク付け変数を作成し、元の変数とともに印刷

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data elect;
   input Candidate $ 1-11 District 13 Vote 15-18 Years 20;
   datalines;
Cardella    1 1689 8
Latham      1 1005 2
Smith       1 1406 0
Walker      1  846 0
Hinkley     2  912 0
Kreitemeyer 2 1198 0
Lundell     2 2447 6
Thrash      2  912 2
;

proc rank data=elect out=results ties=low descending;

   by district;

   var vote years;
   ranks VoteRank YearsRank;
run;

proc print data=results n;
   by district;
   title 'Results of City Council Election';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、SAS ジョブが開始する日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

ELECT データセットを作成します。 このデータセットには、各候補の姓、選挙区番号、得票総数、市議会の経験年数が含まれます。

```
data elect;
   input Candidate $ 1-11 District 13 Vote 15-18 Years 20;
   datalines;
Cardella    1 1689 8
Latham      1 1005 2
Smith       1 1406 0
Walker      1  846 0
Hinkley     2  912 0
Kreitemeyer 2 1198 0
Lundell     2 2447 6
```

```
   Thrash       2  912 2
   ;
```

降順で数値変数のランクを生成し、出力データセット RESULTS を作成します。 DESCENDING で、ランクが逆順になり、最高得票総数にランク 1 が指定されます。TIES=LOW で、可能な限り最高のランクがタイ値に指定されます。OUT=で、出力データセット Results が作成されます。

```
   proc rank data=elect out=results ties=low descending;
```

BY グループごとに別々のランクセットを作成します。 BY ステートメントは、ランク付けを District の値別に分割します。

```
      by district;
```

ランクを含む新しい変数を 2 つ作成します。 VAR ステートメントで、ランク付けする変数が指定されます。RANKS ステートメントで、新しい変数 VoteRank と YearsRank が作成され、それぞれに変数 Vote と Years のランクが含められます。

```
      var vote years;
      ranks VoteRank YearsRank;
   run;
```

データセットを出力します。 PROC PRINT で、Results データセットが印刷されます。N オプションでは、各 BY グループのオブザベーション数が印刷されます。TITLE ステートメントでタイトルを指定します。

```
   proc print data=results n;
      by district;
      title 'Results of City Council Election';
   run;
```

出力:リスト

次の出力では、第 2 区で Hinkley と Thrash が 912 票でタイになっています。TIES=LOW なので、双方ともランクが 3 になります。

アウトプット 53.4 *市議選の結果*

```
Results of City Council Election                      1 ----------------------------------
District=1 ---------------------------------- Vote    Years Obs    Candidate    Vote    Years
Rank     Rank 1     Cardella     1689      8        1       1 2     Latham       1005      2
3       2 3     Smith        1406      0        2       3 4     Walker        846      0
4       3 N = 4 ---------------------------------- District=2 ----------------------------------
Vote    Years Obs    Candidate      Vote    Years    Rank     Rank 5     Hinkley         912
0        3       3 6     Kreitemeyer    1198      0        2       3 7     Lundell        2447
6        1       1 8     Thrash          912      2        3       2 N = 4
```

例 3: ランクに基づいてオブザベーションをグループ分けする

要素: PROC RANK ステートメントオプション
GROUPS=
BY ステートメント
VAR ステートメント

他の要素: PRINT プロシジャ

SORT プロシジャ

詳細

この例では、次のアクションを実行します。

- 2 つの入力変数の値に基づいてオブザベーションをグループ分け
- BY グループ内のオブザベーションを別々にグループ分け
- 元の変数値をグループ値に置換

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

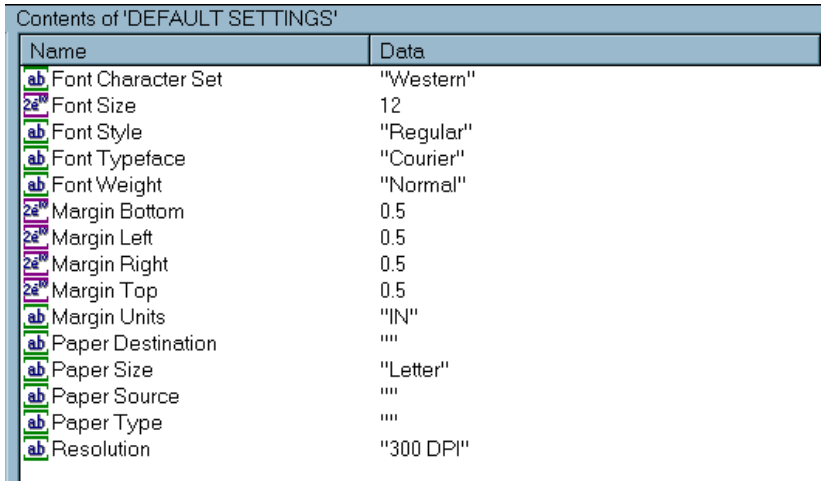
data swim;
   input Name $ 1-7 Gender $ 9 Back 11-14 Free 16-19;
   datalines;
Andrea  F 28.6 30.3
Carole  F 32.9 24.0
Clayton M 27.0 21.9
Curtis  M 29.0 22.6
Doug    M 27.3 22.4
Ellen   F 27.8 27.0
Jan     F 31.3 31.2
Jimmy   M 26.3 22.5
Karin   F 34.6 26.2
Mick    M 29.0 25.4
Richard M 29.7 30.2
Sam     M 27.2 24.1
Susan   F 35.1 36.1
;

proc sort data=swim out=pairs;
   by gender;
run;

proc rank data=pairs out=rankpair groups=3;

   by gender;

   var back free;
run;

proc print data=rankpair n;
   by gender;
   title 'Pairings of Swimmers for Backstroke and Freestyle';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションで、SAS ジョブの開始日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Swim データセットを作成します。 このデータセットには、水泳選手の名前、および背泳と自由型のタイム(秒)が含まれます。この例では、男子クラス内と女子クラス内で両泳法のタイムに基づいて水泳選手のペアを作り、全員が各泳法でタイムの似た相手とペアになれるようにします。

```
data swim;
   input Name $ 1-7 Gender $ 9 Back 11-14 Free 16-19;
   datalines;
Andrea  F 28.6 30.3
Carole  F 32.9 24.0
Clayton M 27.0 21.9
Curtis  M 29.0 22.6
Doug    M 27.3 22.4
Ellen   F 27.8 27.0
Jan     F 31.3 31.2
Jimmy   M 26.3 22.5
Karin   F 34.6 26.2
Mick    M 29.0 25.4
Richard M 29.7 30.2
Sam     M 27.2 24.1
Susan   F 35.1 36.1
;
```

Swim データセットを並べ替えて、出力データセット Pairs を作成します。 PROC SORT で、データセットが Gender 順に並べ替えられます。グループごとに別々のランクセットを取得するには、このアクションが必要です。OUT=で、出力データセット Pairs が作成されます。

```
proc sort data=swim out=pairs;
   by gender;
run;
```

3 つにグループ分けされたランクを生成し、出力データセットを作成します。 GROUPS=3 で、使用可能なグループ値(0、1、2)のいずれかが各泳法の各水泳選手に割り当てられます。OUT=で、出力データセット Rankpair が作成されます。

```
proc rank data=pairs out=rankpair groups=3;
```

BY グループごとに別々のランクセットを作成します。 BY ステートメントは、ランク付けを Gender 別に分割します。

```
   by gender;
```

変数の元の値をランク値に置き換えます。 VAR ステートメントで、Back と Free がランク付け対象の変数であることが指定されます。PROC RANK で RANKS ステートメントを指定しなかった場合、出力データセットでは元の変数値がグループ値に置き換えられます。

```
   var back free;
run;
```

データセットを出力します。 PROC PRINT で、Rankpair データセットが印刷されます。N オプションでは、各 BY グループのオブザベーション数が印刷されます。TITLE ステートメントでタイトルを指定します。

```
proc print data=rankpair n;
   by gender;
   title 'Pairings of Swimmers for Backstroke and Freestyle';
```

```
run;
```

出力:リスト

次の出力では、グループ値によって、各泳法でタイムの似た水泳選手にペアを組ませます。たとえば、Andrea と Ellen が背泳で一緒に組むのは、2 人のタイムが女子クラスで最速なためです。男子水泳選手のグループのバランスが取れていないのは、男子水泳選手が 7 人いるためです。泳法ごとに、水泳選手が 3 人いるグループが 1 つできます。

アウトプット 53.5 *背泳と自由型の水泳選手のペアリング*

```
Pairings of Swimmers for Backstroke and Freestyle               1
---------------------------------- Gender=F ---------------------------------- Obs    Name
Back    Free 1    Andrea      0       1 2    Carole      1       0 3    Ellen       0       1 4
Jan         1       2 5    Karin       2       0 6    Susan       2       2 N = 6
---------------------------------- Gender=M ---------------------------------- Obs    Name
Back    Free 7    Clayton     0       0 8    Curtis      2       1 9    Doug        1       0
10    Jimmy       0       1 11    Mick        2       2 12    Richard     2       2 13
Sam          1       1 N = 7
```

参考文献

Blom, G. 1958. *Statistical Estimates and Transformed Beta Variables*. New York, New York: John Wiley & Sons, Inc.

Conover, W.J. 1998. *Practical Nonparametric Statistics, Third Edition*. New York, New York: John Wiley & Sons, Inc.

Conover, W.J. and R.L. Iman. 1976. “On Some Alternative Procedures Using Ranks for the Analysis of Experimental Designs.” *Communications in Statistics* A5 (14): 1348–1368.

Conover, W.J. and R.L. Iman. 1981. “Rank Transformations as a Bridge between Parametric and Nonparametric Statistics.” *The American Statistician* 35: 124–129.

Iman, R.L. and W.J. Conover. 1979. “The Use of the Rank Transform in Regression.” *Technometrics* 21: 499–509.

Lehman, E.L. 1998. *Nonparametrics: Statistical Methods Based on Ranks*. Upper Saddle River, New Jersey: Prentice Hall.

Quade, D. 1966. “On Analysis of Variance for the K-Sample Problem.” *Annals of Mathematical Statistics* 37: 1747–1758.

Tukey, John W. 1962. “The Future of Data Analysis.” *Annals of Mathematical Statistics* 33: 22.

54 章

REGISTRY プロシジャ

概要: REGISTRY プロシジャ

REGISTRY プロシジャでは、SAS レジストリが管理されます。レジストリには 2 つのパーツがあります。1 つのパーツは Sashelp ライブラリに保存され、もう一つのパーツは Sasuser ライブラリに保存されます。

REGISTRY プロシジャでは、次を実行できます。

- レジストリファイルをインポートして Sashelp レジストリおよび Sasuser レジストリを作成します。
- レジストリの全部または一部を別のファイルにエクスポートします。
- SAS ログにレジストリの内容をリストします。
- レジストリとファイルの内容を比較します。
- レジストリファイルをアンインストールします。
- キーまたは値が上書きまたはアンインストールされた場合に、詳細ステータス情報を配信します。
- Sasuser レジストリのエントリをクリアします。
- レジストリが存在するかを検証します。

- 診断情報をリストします。

詳細については、“The SAS Registry” (*SAS Language Reference: Concepts*)を参照してください。

構文: REGISTRY プロシジャ

PROC REGISTRY <*options*>;

ステートメント	タスク	例
“PROC REGISTRY ステートメント”	レジストリファイルの管理	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6

PROC REGISTRY ステートメント

SAS レジストリを管理します。

例:
“例 1: SAS レジストリへのファイルのインポート” (1573 ページ)
“例 3: レジストリと外部ファイルの比較” (1574 ページ)
“例 2: レジストリファイルのリスト表示とエクスポート” (1574 ページ)
“例 4: レジストリファイルの比較” (1576 ページ)
“例 5: STARTAT=オプションを使用し、キーシーケンス全体を指定する” (1577 ページ)
“例 6: フォントリストの表示” (1577 ページ)

構文

PROC REGISTRY <*options*>;

オプション引数の要約

CLEARSASUSER
ユーザーが追加したキーを Sasuser レジストリから消去します。

COMPAREREG1='*libname.registry-name-1*'
2 つのレジストリファイルを比較します。

COMPAREREG2='*libname.registry-name-2*'
2 つのレジストリファイルを比較します。

COMPARETO=*file-specification*
レジストリとファイルの内容を比較します。

DEBUGOFF
レジストリのデバッグを無効化します。

DEBUGON
レジストリのデバッグを有効化します。

EXPORT=*file-specification*
レジストリの内容を指定ファイルに書き込みます。

FOLLOWLINKS

LIST コマンドの処理時に検出されたリンクを含めます。

FULLSTATUS
IMPORT=および UNINSTALL=オプションの結果について SAS ログで追加情報を提供します。

IMPORT=*file-specification*
指定ファイルをレジストリにインポートします。

KEYSONLY
LIST、LISTUSER、LISTHELP、LISTREG の各オプションの出力をキーのみ表示するように制限します。

LEVELS=*n*
LIST、LISTUSER、LISTHELP、LISTREG の各オプションの表示レベル数を制限します。

LIST
SAS ログにレジストリの内容を書き込みます。特定のキーをリストするには、このオプションを STARTAT=オプションと一緒に使用します。

LISTHELP
レジストリの Sashelp 部分の内容を SAS ログに書き込みます。

LISTREG='*libname.registry-name*'
レジストリの内容をログに送信します。

LISTUSER
レジストリの Sasuser 部分の内容を SAS ログに書き込みます。

STARTAT='*key-name*'
指定キーでレジストリの内容のエクスポートや書き込みや比較を開始します。

UNINSTALL=*file-specification*
指定ファイル内のすべてのキーと値を指定レジストリから削除します。

UPCASE
すべての内向きのキー名に大文字を使用します。

UPCASEALL
ファイルのインポート時に、すべてのキー、名前および項目値に大文字を使用します。

USESASHELP
SAS レジストリの Sashelp 部分に対する指定操作を実行します。

オプション引数

CLEARSASUSER
ユーザーが追加したキーを SAS レジストリの Sasuser 部分から消去します。

COMPAREREG1='*libname.registry-name-1*'
比較対象の 2 つのレジストリのうち 1 つを指定します。結果は SAS ログに表示されます。

libname
レジストリファイルが存在するライブラリの名前です。

registry-name-1
最初のレジストリの名前です。

要件 COMPAREREG1 は COMPAREREG2 と一緒に使用する必要があります。

操作 単一キーとそのサブキーすべてを指定するには、STARTAT=オプションを指定します。

例 “例 4: レジストリファイルの比較” (1576 ページ)

COMPAREREG2='*libname.registry-name-2*'
比較対象の 2 つのレジストリの 2 番目を指定します。結果は SAS ログに表示されます。

libname
レジストリファイルが存在するライブラリの名前です。

registry-name-2
2 番目のレジストリの名前です。

要件 COMPAREREG2 は COMPAREREG1 と一緒に使用する必要があります。

例 “例 4: レジストリファイルの比較” (1576 ページ)

COMPARETO=*file-specification*
レジストリ情報を含むファイルとレジストリの内容を比較します。キーと値がファイルにあってレジストリにはない場合、その情報が返されます。次の項目が差異としてレポートされます。

- 外部ファイルで定義されているが、レジストリでは定義されていないキー
- 外部ファイルにはあるがレジストリにはない指定キーの値名
- 似ている名前のキーの似ている名前の値の内容の差異

COMPARETO=では、レジストリにあるがファイルにないキーと値は差異としてレポートされません。レジストリはさまざまなファイルの断片から成ることが多いためです。

File-specification は次の値のうちのいずれかになります。

'*external-file*'
レジストリ情報を含む外部ファイルのパスと名前です。

fileref
外部ファイルに割り当てられたファイル参照名です。

要件 FILENAME ステートメント、FILENAME 関数、**エクスプローラ**ウィンドウ、適切な動作環境のコマンドのいずれかで、ファイル参照名と外部ファイルを事前に関連付けておく必要があります。

操作 デフォルトでは、PROC REGISTRY で *file-specification* とレジストリの Sasuser 部分が比較されます。*file-specification* とレジストリの Sashelp 部分を比較するには、オプション USESASHELP を指定します。

参照項目 “REGISTRY プロシジャを使用し、レジストリファイルを作成する ” (1570 ページ)レジストリ情報を含むファイルの構築方法の詳細については、

例 “例 3: レジストリと外部ファイルの比較” (1574 ページ)

DEBUGON
詳細なログエントリを提供して、レジストリのデバッグを有効化します。

DEBUGOFF
レジストリのデバッグを無効化します。

EXPORT=*file-specification*
レジストリの内容を指定ファイルに書き込みます。このとき、*file-specification* は次の値のいずれかになります。

'external-file'
レジストリ情報を含む外部ファイルの名前です。

fileref
外部ファイルに割り当てられたファイル参照名です。

要件 FILENAME ステートメント、FILENAME 関数、**エクスプローラ**ウィンドウ、適切な動作環境のコマンドのいずれかで、ファイル参照名と外部ファイルを事前に関連付けておく必要があります。

file-specification がすでに存在する場合は、PROC REGISTRY で上書きされます。その他の場合は、PROC REGISTRY でファイルが作成されます。

操作 デフォルトでは、EXPORT=でレジストリの Sasuser 部分が指定ファイルに書き込まれます。レジストリの Sashelp 部分を書き込むには、USESASHELP オプションを指定します。USESASHELP を使用するには、Sashelp ライブラリへの書き込み権限が必要です。

単一キーとそのサブキーすべてをエクスポートするには、STARTAT=オプションを指定します。

例 “例 2: レジストリファイルのリスト表示とエクスポート” (1574 ページ)

FOLLOWLINKS
LIST オプションの処理時に検出されたリンクを含めます。

通常、LIST オプションでは、リンク項目の値が表示されます。FOLLOWLINKS オプションを使用した場合、リンクはキーとして扱われ、リンクに含まれる項目が表示されます。

FULLSTATUS
IMPORT=および UNINSTALL オプションの実行結果として、追加または削除されたキー、サブキーおよび値をリストします。

IMPORT=*file-specification*
SAS レジストリにインポートするファイルを指定します。PROC REGISTRY では、既存のレジストリは上書きされません。そのかわりに、既存のレジストリを指定ファイルの内容で更新します。

注: ファイル拡張子.sasxreg は必要ありません。

File-specification は次の値のうちのいずれかになります。

'external-file'
レジストリ情報を含む外部ファイルのパスと名前です。

fileref
外部ファイルに割り当てられたファイル参照名です。

要件 FILENAME ステートメント、FILENAME 関数、**エクスプローラ**ウィンドウ、適切な動作環境のコマンドのいずれかで、ファイル参照名と外部ファイルを事前に関連付けておく必要があります。

操作 デフォルトでは、IMPORT=で SAS レジストリの Sasuser 部分にファイルがインポートされます。レジストリの Sashelp 部分にファイルをインポートするには、USESASHELP オプションを指定します。USESASHELP を使用するには、Sashelp への書き込み権限が必要です。

ファイルのインポート時に SAS ログで追加情報を取得するには、FULLSTATUS を使用します。

参照項目 “REGISTRY プロシジャを使用し、レジストリファイルを作成する ” (1570 ページ) レジストリ情報を含むファイルの構築方法の詳細については、

例 “例 1: SAS レジストリへのファイルのインポート” (1573 ページ)

KEYSONLY
LIST、LISTUSER、LISTHELP、LISTREG の各オプションの出力をキーのみ表示するように制限します。

LEVELS=*n*
LIST、LISTUSER、LISTHELP、LISTREG の各オプションの表示レベル数を制限します。

要件 LEVEL ≥ 1.LEVELS=0 は、LEVELS を指定しない場合と同様の動作になります。

LIST
SAS ログに SAS レジストリ全体の内容を書き込みます。

操作 単一キーとそのサブキーすべてを書き込むには、STARTAT=オプションを使用します。

LISTHELP
レジストリの Sashelp 部分の内容を SAS ログに書き込みます。

操作 単一キーとそのサブキーすべてを書き込むには、STARTAT=オプションを使用します。

LISTREG='*libname.registry-name*'
ログに指定レジストリの内容をリストします。

libname
レジストリファイルが存在するライブラリの名前です。

registry-name
レジストリの名前です。

例は、次のとおりです。

```
proc registry listreg='sashelp.regstry';
run;
```

操作 単一キーとそのサブキーすべてをリストするには、STARTAT=オプションを使用します。

LISTUSER
レジストリの Sasuser 部分の内容を SAS ログに書き込みます。

操作 単一キーとそのサブキーすべてを書き込むには、STARTAT=オプションを使用します。

例 “例 2: レジストリファイルのリスト表示とエクスポート” (1574 ページ)

STARTAT='*key-name*'
単一キーとそのサブキーすべての内容をエクスポートするか書き込みます。

ルートキーの下の任意のサブキーでリスト表示を開始する場合、キーシーケンス全体を指定する必要があります。

操作 STARTAT=は、EXPORT=、LIST、LISTHELP、LISTUSER、COMPAREREG1=、COMPAREREG2=、LISTREG オプションと一緒に使用します。

例 “例 4: レジストリファイルの比較” (1576 ページ)

UNINSTALL=*file-specification*
指定ファイル内のすべてのキーと値を指定レジストリから削除します。

File-specification は次の値のうちのいずれかになります。

'external-file'
削除するキーと値を含む外部ファイルの名前です。

fileref
外部ファイルに割り当てられたファイル参照名です。ファイル参照名を割り当てるには、次の操作を実行します。

- エクスプローラウィンドウを使用
- を使用 “FILENAME Statement” (*SAS Statements: Reference*)

操作 デフォルトでは、UNINSTALL で SAS レジストリの Sasuser 部分からキーと値が削除されます。レジストリの Sashelp 部分からキーと値を削除するには、USESASHELP オプションを指定します。このオプションを使用するには、Sashelp への書き込み権限が必要です。

レジストリのアンインストール時に SAS ログで追加情報を取得するには、FULLSTATUS を使用します。

参照項目 “REGISTRY プロシジャを使用し、レジストリファイルを作成する ” (1570 ページ) レジストリ情報を含むファイルの構築方法の詳細については、

UPCASE
すべての内向きのキー名に大文字を使用します。

UPCASEALL
ファイルのインポート時に、すべてのキー、名前および項目値に大文字を使用します。

USESASHELP
SAS レジストリの Sashelp 部分に対する指定操作を実行します。

操作 USESASHELP は、IMPORT=、EXPORT=、COMPARETO、UNINSTALL のいずれかのオプションと一緒に使用します。USESASHELP を IMPORT= または UNINSTALL と一緒に使用するには、Sashelp への書き込み権限が必要です。

REGISTRY プロシジャを使用し、レジストリファイルを作成する

レジストリファイルの構造

SAS レジストリエディタまたはテキストエディタを使用してレジストリファイルを作成できます。

レジストリファイルには特定の構造が必要です。レジストリファイルの各エントリは、キー名と、それに続く次行の 1 つ以上の値から成ります。キー名では、定義するキーまたはサブキーが識別されます。その後の値ではどれでも、キーに関連付ける名前またはデータが指定されます。

キー名の指定

キー名は、角かっこ([と])の間に 1 行で入力します。サブキーを指定するには、ルートキーで始まる複数のキー名をかっこの間に入力します。複数のキー名を並べる場合は、バックスラッシュ(\)で名前を区切ります。単一キー名またはキー名のシーケンスの長さは 255 文字(角かっことバックスラッシュを含む)を超えないようにしてください。キー名には、バックスラッシュ以外の任意の文字を含められます。

有効なキー名シーケンスの例は次のとおりです。これらのシーケンスは典型的な SAS レジストリです。

- [CORE\EXPLORER\MENUS\ENTRIES\CLASS]
- [CORE\EXPLORER\NEWMEMBER\CATALOG]
- [CORE\EXPLORER\NEWENTRY\CLASS]
- [CORE\EXPLORER\ICONS\ENTRIES\LOG]

キーの値の指定

キー名に続く行に、関連付ける各値を入力します。各キーに複数の値を指定できますが、各値は別々の行に入力します。

値の一般的な形式は次のとおりです。

value-name=value-content

value-name には、デフォルト値の名前を示すアットマーク(@)かまたは二重引用符で囲まれた任意のテキスト文字列を指定できます。テキスト文字列にアンパサンド(&)が含まれている場合、アンパサンドに続く文字(大文字か小文字のどちらか)は値名のショートカットです。詳細については、“サンプルレジストリエントリ” (1571 ページ)を参照してください。

文字列全体が 255 文字(引用符とアンパサンドを含む)を超えないようにしてください。これにはバックスラッシュ(\)以外の任意の文字を含められます。

Value-content は次のいずれかになります。

- 後に数値が続いている文字列 **double:**.

- 文字列。かっこ内に任意の文字を入れられます。何も入れず("")とすることも可能です。

  注: 引用符で囲まれている文字列にバックスラッシュを入力するには、隣接する 2 つのバックスラッシュを使用します。二重引用符を含めるには、隣接する 2 つの二重引用符を使用します。

- 文字列 **hex:**に続けて 255 字までの任意の 16 進文字の数字を指定し、カンマで区切ります。16 進文字が 1 行を超える場合は、行末をバックスラッシュにして、データが次行に続くことを示します。レジストリエディタでは、16 進値は"バイナリ値"とも呼ばれます。
- 文字列 **dword:**に続けて、符号なしの長い 16 進値を指定します。
- 文字列 **int:**に続けて、符号付きの長い整数値を指定します。
- 文字列 **uint:**に続けて、符号なしの長い整数値を指定します。

前述のさまざまな種類の値がレジストリエディタでどのように表示されるかを次に示します。

図 54.1 *レジストリ値の種類(レジストリエディタでの表示)*

Name	Data
A double value	2.4E-44
A string	"my data"
Binary data	01,00,76,63,62,6B
Dword	66051
Signed integer value	-123
Unsigned integer value (decimal)	123456

次のリストには、有効なレジストリ値のサンプルが含まれます。

- 倍精度値=double:2.4E-44
- 文字列="my data"
- バイナリデータ=hexadecimal:01,00,76,63,62,6B
- 倍長語=dword:00010203
- 符号付き整数値=int:-123
- 符号なし整数値(小数)=dword:0001E240

サンプルレジストリエントリ

レジストリエントリの内容と表示は、目的に応じて変化する場合があります。

デフォルトの PostScript プリンタ設定を含むレジストリエントリを次に示します。

図 54.2 *PostScript プリンタの設定を示すレジストリエディタ部分*



実際のレジストリテキストファイルがどのように表示されるかを確認するには、PROC REGISTRY で、LISTUSER および STARTAT=オプションを使用して、レジストリキーの内容を SAS ログに書き込みます。

次の例は、SASUSER レジストリエントリをログに送信する構文を示しています。

```
proc registry
     listuser
     startat='sasuser-registry-key-name';
run;
```

次の例は、STARTAT=オプションの値を示しています。

```
proc registry
     listuser
     startat='HKEY_SYSTEM_ROOT\CORE\PRINTING\PRINTERS\PostScript\DEFAULT
SETTINGS';
run;
```

次の例では、サブキーのリストは CORE\PRINTING\PRINTERS\PostScript\DEFAULT SETTINGS キーから始まります。

ログ 54.1 *PostScript プリンタのレジストリエントリ*

```
NOTE:Contents of SASUSER REGISTRY starting at subkey [CORE\ PRINTING\PRINTERS
\PostScript\DEFAULT SETTINGS key] Font Character Set="Western" Font Size=double:
12 Font Style="Regular" Font Typeface="Courier" Font Weight="Normal" Margin
Bottom=double:0.5 Margin Left=double:0.5 Margin Right=double:0.5 Margin
Top=double:0.5 Margin Units="IN" Paper Destination="" Paper Size="Letter" Paper
Source="" Paper Type="" Resolution="300 DPI"
```

例: REGISTRY プロシジャ

例 1: SAS レジストリへのファイルのインポート

要素: IMPORT=

他の要素: FILENAME ステートメント

詳細

この例では、SAS レジストリの Sasuser 部分にファイルがインポートされます。次のソースファイルには、レジストリファイルの有効なキー名シーケンスの例が含まれています。

```
[HKEY_USER_ROOT\AllGoodPeopleComeToTheAidOfTheirCountry]
 @="This is a string value"
  "Value2"=""
  "Value3"="C:\\This\\Is\\Another\\String\\Value"
```

プログラム

```
filename source 'external-file';

proc registry import=source;
run;
```

プログラムの説明

レジストリの有効テキストを含むファイルにファイル参照名を割り当てます。 FILENAME ステートメントで、ファイル参照名 SOURCE が、レジストリに読み込むテキストを含む外部ファイルに割り当てられます。

```
filename source 'external-file';
```

PROC REGISTRY を呼び出して、レジストリの入力を含むファイルをインポートします。 PROC REGISTRY では、ファイル参照名 SOURCE によって識別される入力ファイルが読み取られます。IMPORT=では、デフォルトで、SAS レジストリの Sasuser 部分に書き込まれます。

```
proc registry import=source;
run;
```

ログ

ログ 54.2 SAS レジストリへのファイルをインポートした結果

```
Parsing REG file and loading the registry please wait....Registry IMPORT is now
complete.
```

例 2: レジストリファイルのリスト表示とエクスポート

要素: EXPORT=
LISTUSER

詳細

通常、レジストリファイルは非常に大きいです。レジストリの一部をエクスポートするには、STARTAT=オプションを使用します。

この例では、SAS レジストリの Sasuser 部分がリスト表示され、外部ファイルにエクスポートされます。

プログラム

```
proc registry
   listuser

   export='external-file';
run;
```

プログラムの説明

レジストリの Sasuser 部分の内容を SAS ログに書き込みます。 LISTUSER オプションを使用すると、PROC REGISTRY では、レジストリの Sasuser 部分全体がログに書き込まれます。

```
proc registry
   listuser
```

レジストリを指定ファイルにエクスポートします。 EXPORT=オプションでは、SAS レジストリの Sasuser 部分のコピーが外部ファイルに書き込まれます。

```
   export='external-file';
run;
```

ログ

ログ 54.3 *SAS レジストリファイルをリスト表示しエクスポートした結果*

```
Starting to write out the registry file, please wait...The export to file
external-file is now complete.Contents of SASUSER REGISTRY.[  HKEY_USER_ROOT]
[    CORE] [      EXPLORER] [        CONFIGURATION] Initialized= "True"
[        FOLDERS] [          UNXHOST1] Closed= "658" Icon= "658" Name= "Home
Directory" Open= "658" Path= "~"
```

例 3: レジストリと外部ファイルの比較

要素: COMPARETO=オプション

他の要素: FILENAME ステートメント

詳細

この例では、SAS レジストリの Sasuser 部分と外部ファイルが比較されます。.txt ファイル拡張子で保存されたバックアップファイルと現在のレジストリファイルの差異を確認する場合、このような比較が有用です。

レジストリの Sashelp 部分と外部ファイルを比較するには、USESASHELP オプションを指定します。

この SAS ログには、レジストリの Sasuser 部分と指定外部ファイルの差異が 2 つ示されます。"Initialized"の値はレジストリでは"True"ですが、外部ファイルでは"False"です。“Icon”の値はレジストリでは“658”ですが、外部ファイルでは“343”です。

プログラム

```
filename testreg 'external-file';

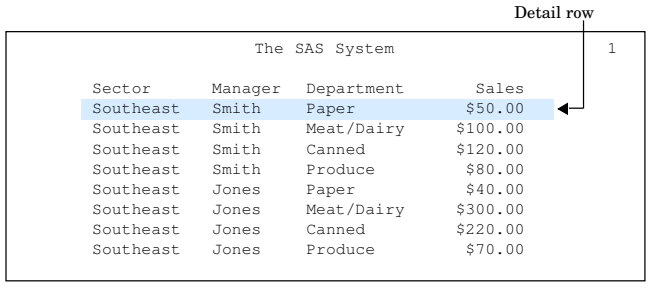
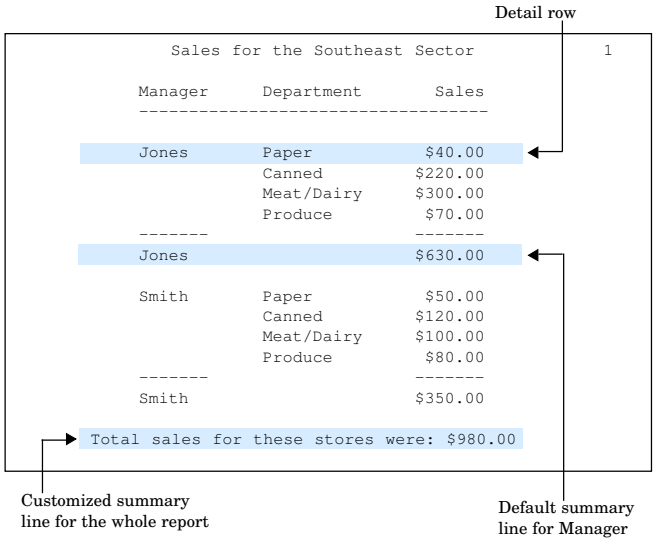
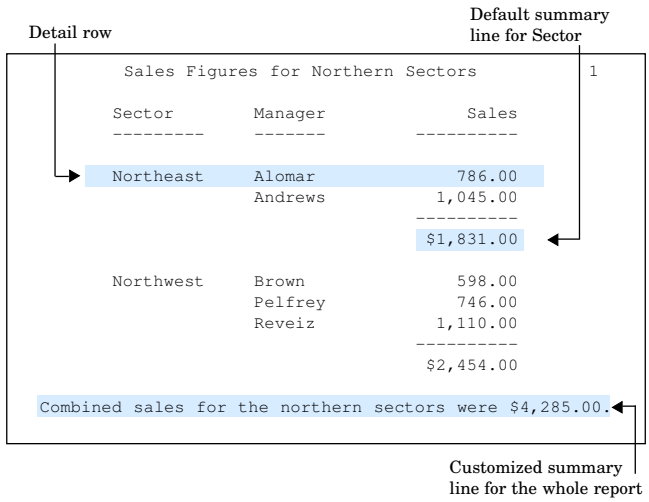
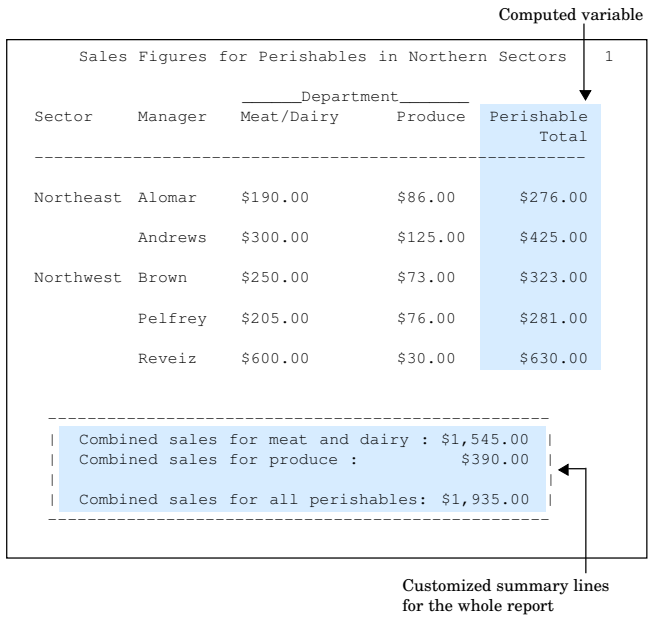
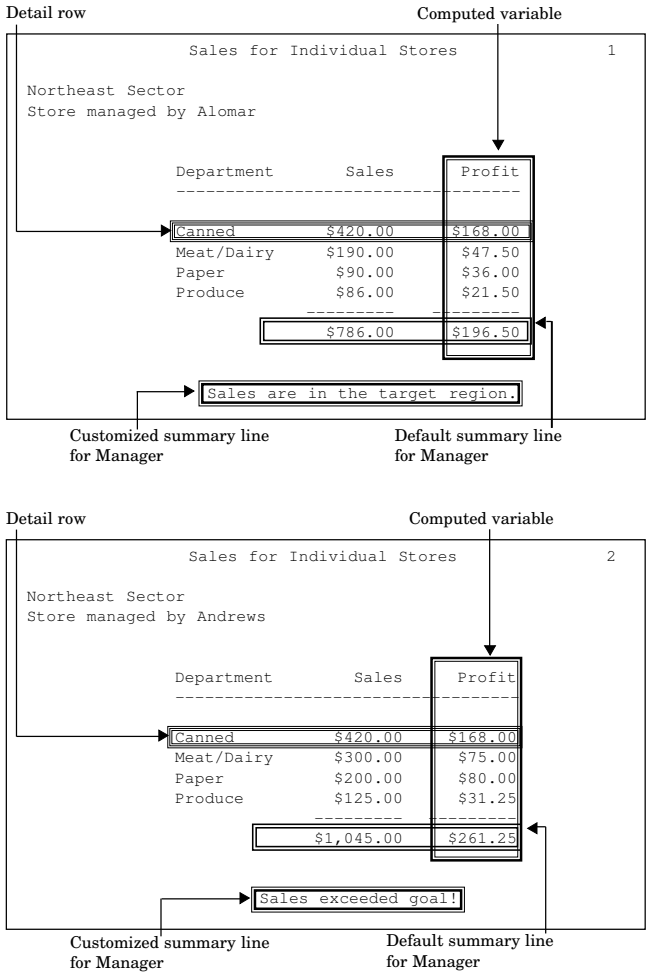
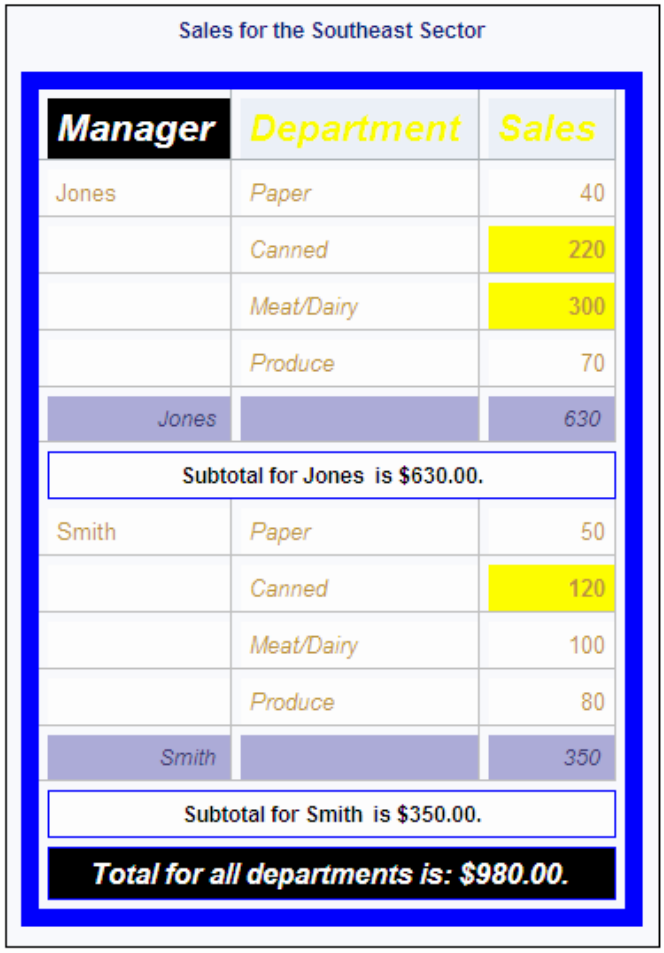
proc registry
   compareto=testreg;
run;
```

プログラムの説明

レジストリと比較するテキストを含む外部ファイルにファイル参照名を割り当てます。
FILENAME ステートメントで、ファイル参照名 TESTREG が外部ファイルに割り当てられます。

```
filename testreg 'external-file';
```

指定ファイルと SAS レジストリの Sasuser 部分を比較します。 COMPARETO オプションでは、ファイルとレジストリの内容が比較されます。キーと値がファイルにあってレジストリにはない場合、その情報が返されます。

```
proc registry
   compareto=testreg;
run;
```

ログ

ログ 54.4 *レジストリと外部ファイルを比較した結果*

```
Parsing REG file and comparing the registry please wait....COMPARE DIFF:Value
"Initialized" in [HKEY_USER_ROOT\CORE\EXPLORER\CONFIGURATION]:REGISTRY
TYPE=STRING, CURRENT VALUE="True" COMPARE DIFF:Value "Initialized" in
[HKEY_USER_ROOT\CORE\EXPLORER\CONFIGURATION]:FILE TYPE=STRING, FILE
VALUE="False" COMPARE DIFF:Value "Icon" in [HKEY_USER_ROOT\CORE\EXPLORER\FOLDERS
\UNXHOST1]:REGISTRY TYPE=STRING, CURRENT VALUE="658" COMPARE DIFF:Value "Icon"
in [HKEY_USER_ROOT\CORE\EXPLORER\FOLDERS\UNXHOST1]:FILE TYPE=STRING, FILE
VALUE="343" Registry COMPARE is now complete.COMPARE:There were differences
between the registry and the file.
```

例 4: レジストリファイルの比較

要素: COMPAREREG1=および COMPAREREG2=オプション
STARTAT=オプション

詳細

この例では、REGISTRY プロシジャオプション COMPAREREG1=および COMPAREREG2=を使用して、比較する 2 つのレジストリファイルを指定します。

プログラム

```
libname proclib 'SAS-library';

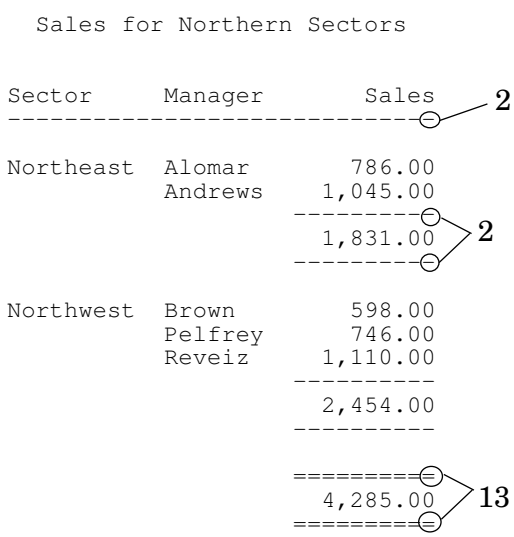
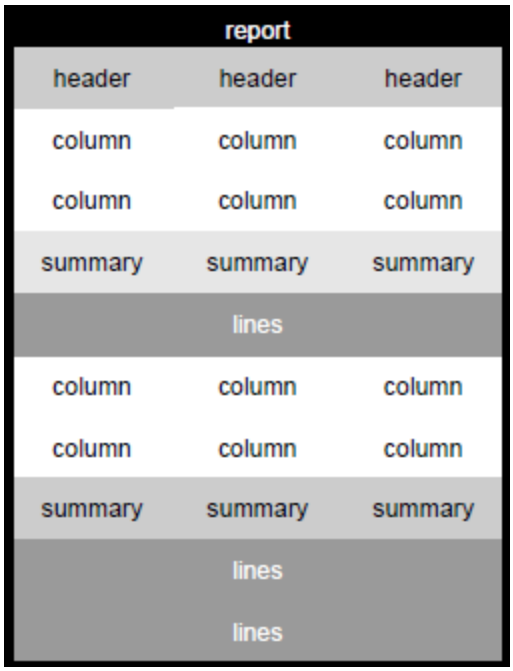
proc registry comparereg1='sasuser.regstry'

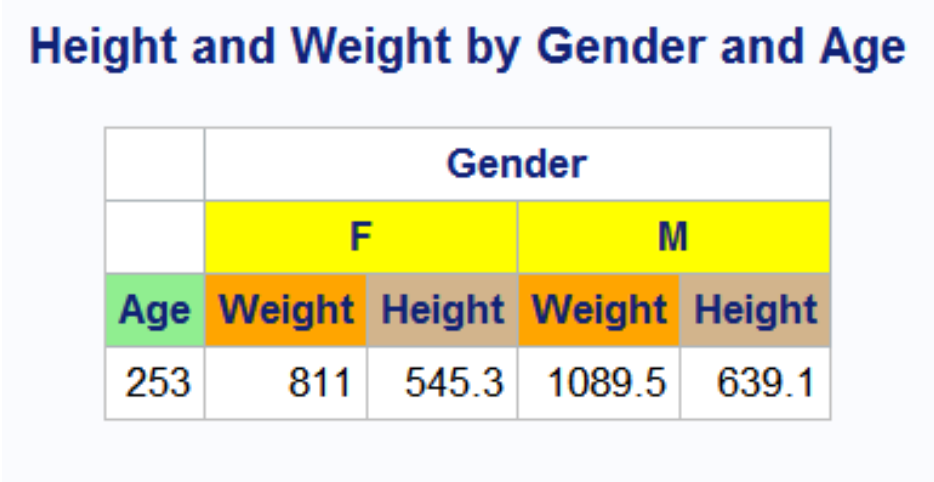
   startat='CORE\EXPLORER'

   comparereg2='proclib.regstry';
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリにはレジストリファイルが含まれます。

```
libname proclib 'SAS-library';
```

PROC REGISTRY を開始し、比較に使用する最初のレジストリファイルを指定します。

```
proc registry comparereg1='sasuser.regstry'
```

指定レジストリキー以降のレジストリキーとの比較に制限します。 STARTAT=オプションでは、比較スコープが、CORE キーの下の EXPLORER サブキーに制限されます。デフォルトでは、両レジストリの内容全体が比較対象になります。

```
   startat='CORE\EXPLORER'
```

比較に使用する 2 番目のレジストリファイルを指定します。

```
   comparereg2='proclib.regstry';
run;
```

ログ

ログ 54.5 2 つのレジストリファイルを比較した結果

```
NOTE:Comparing registry SASUSER.REGSTRY to registry PROCLIB.REGSTRY NOTE:Diff in
Key (CORE\EXPLORER\MENUS\FILES\SAS) Item (1;&Open) SASUSER.REGSTRY Type:String
len  17 data PGM;INCLUDE '%s'; PROCLIB.REGSTRY Type:String len  15 data
WHOSTEDIT '%s'; NOTE:Diff in Key (CORE\EXPLORER\MENUS\FILES\SAS) Item
(3;&Submit) SASUSER.REGSTRY Type:String len  23 data PGM;INCLUDE '%s';SUBMIT
PROCLIB.REGSTRY Type:String len  21 data WHOSTEDIT '%s';SUBMIT NOTE:Diff in Key
(CORE\EXPLORER\MENUS\FILES\SAS) Item (4;&Remote Submit) SASUSER.REGSTRY
Type:String len  35 data SIGNCHECK;PGM;INCLUDE '%s';RSUBMIT; PROCLIB.REGSTRY
Type:String len  33 data SIGNCHECK;WHOSTEDIT '%s';RSUBMIT; NOTE:Diff in Key (CORE
\EXPLORER\MENUS\FILES\SAS) Item (@) SASUSER.REGSTRY Type:String len  17 data
PGM;INCLUDE '%s'; PROCLIB.REGSTRY Type:String len  15 data WHOSTEDIT '%s';
NOTE:Item (2;Open with &Program Editor) in key (CORE\EXPLORER\MENUS\FILES\TXT)
not found in registry PROCLIB.REGSTRY NOTE:Diff in Key (CORE\EXPLORER\MENUS\FILES
\TXT) Item (4;&Submit) SASUSER.REGSTRY Type:String len  24 data PGM;INCLUDE
'%s';SUBMIT; PROCLIB.REGSTRY Type:String len  22 data WHOSTEDIT '%s';SUBMIT;
NOTE:Diff in Key (CORE\EXPLORER\MENUS\FILES\TXT) Item (5;&Remote Submit)
SASUSER.REGSTRY Type:String len  35 data SIGNCHECK;PGM;INCLUDE '%s';RSUBMIT;
PROCLIB.REGSTRY Type:String len  33 data SIGNCHECK;WHOSTEDIT '%s';RSUBMIT;
```

例 5: STARTAT=オプションを使用し、キーシーケンス全体を指定する

要素: EXPORT オプション
STARTAT=オプション

詳細

次の例では、STARTAT=オプションの使用方法を示します。ルートキーの下の任意のサブキーでリスト表示を開始する場合、キーシーケンス全体を指定する必要があります。ルートキーはオプションです。

プログラム

```
proc registry export = my-fileref
startat='core\explorer\icons';
run;
```

例 6: フォントリストの表示

要素: LISTHELP オプション
STARTAT オプション

詳細

次の例では、ODS フォントのリストが SAS ログに書き込まれます。

プログラム

```
proc registry clearsasuser; run;
```

```
proc registry listhelp startat='ods\fonts'; run;
proc registry clearsasuser; run;
```

ログ

ログ 54.6 *SAS レジストリのフォントリストを表示した結果*

```
NOTE:Contents of SASHELP REGISTRY starting at subkey [ods\fonts] [  ods\fonts]
dings="Wingdings" monospace="Courier New" MTdings="Monotype Sorts"
MTmonospace="Cumberland AMT" MTsans-serif="Albany AMT" MTsans-serif-
unicode="Monotype Sans WT J" MTserif="Thorndale AMT" MTserif-unicode="Thorndale
Duospace WT J" MTsymbol="Symbol MT" sans-serif="Arial" serif="Times New Roman"
symbol="Symbol" [    ja_JP] dings="Wingdings" monospace="MS Gothic"
MTdings="Wingdings" MTmonospace="MS Gothic" MTsans-serif="MS PGothic" MTsans-
serif-unicode="MS PGothic" MTserif="MS PMincho" MTserif-unicode="MS PMincho"
MTsymbol="Symbol" sans-serif="MS PGothic" serif="MS PMincho" symbol="Symbol"
[    ko_KR] dings="Wingdings" monospace="GulimChe" MTdings="Wingdings"
MTmonospace="GulimChe" MTsans-serif="Batang" MTsans-serif-unicode="Batang"
MTserif="Gulim" MTserif-unicode="Gulim" MTsymbol="Symbol" sans-serif="Batang"
serif="Gulim" symbol="Symbol" [    th_TH] dings="Wingdings" monospace="Thorndale
Duospace WT J" MTdings="Monotype Sorts" MTmonospace="Cumberland AMT" MTsans-
serif="Monotype Sans WT J" MTsans-serif-unicode="Thorndale Duospace WT J"
MTserif="Thorndale Duospace WT J" MTserif-unicode="Monotype Sans WT J"
MTsymbol="Symbol MT" sans-serif="Angsana New" serif="Thorndale Duospace WT J"
symbol="Symbol"
```

55 章

REPORT プロシジャ

概要:REPORT プロシジャ

REPORT プロシジャの動作について

REPORT プロシジャは、PRINT プロシジャ、MEANS プロシジャ、TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートの作成が可能な 1 つのレポート作成ツールに組み合わせたものです。PROC REPORT は、次の 3 つの方法で使用できます。

- 非ウィンドウ環境での使用。この場合、その他の SAS プロシジャと同様に一連のステートメントを PROC REPORT ステートメントでサブミットします。これらのステートメントを PROC REPORT ステートメントでプログラムエディタ(デフォルトは NOWINDOWS)からサブミットするか、SAS をバッチ、非対話型または対話型のラインモードで実行できます(“Ways to Run Your SAS Session ” (*SAS Language Reference: Concepts*)に関する詳細を参照してください。)
- レポート作成時に入力を促すプロンプト機能を持つ対話型のレポートウィンドウ環境での使用。詳細については、56 章, “REPORT プロシジャウィンドウ” (1735 ページ)を参照してください。
- 入力を促すプロンプト機能を持たない対話型レポートウィンドウ環境での使用。詳細については、56 章, “REPORT プロシジャウィンドウ” (1735 ページ)を参照してください。

PROC REPORT で作成可能なレポートの種類について

詳細レポートには、レポートに選択された各オブザベーションにつき 1 行が含まれています。これらの行はそれぞれレポート行、**詳細レポート行**です。**要約レポート** では、1

行で複数のオブザベーションを表すようにデータがまとめられます。これらの行はそれぞれ詳細行、**要約レポート行**とも呼ばれます。

詳細レポートと要約レポートには、レポート行のほかに**要約レポート行**(ブレーク行)を含めることができます。要約行では、一連の詳細行またはすべての詳細行の数値データが要約されます。PROC REPORT は、デフォルト要約とカスタマイズされた要約を提供します(“ブレーク行の使用” (1593 ページ)を参照)。

この概要では、PROC REPORT で作成可能なレポートの種類を説明します。これらのレポートで使用されるデータセットと出力形式を作成するステートメントについては、“例 1: 変数の選択とレポートの要約行の作成” (1680 ページ)を参照してください。出力形式は、永久 SAS ライブラリに保存されます。追加レポートとそれを作成するステートメントについては、REPORT プロシジャの例を参照してください。

各種レポートの表示形式について

これらのレポートで使用されるデータセットには、食料品店チェーンの 8 店舗の 1 日の売上高が含まれます。

単一 PROC REPORT ステップにより、単一の PROC PRINT ステップにより作成されるレポートと似たレポートが作成されます。図 55.1 (1581 ページ)は、PROC REPORT で作成可能な最も単純な種類のレポートを示しています。レポートを作成するステートメントがその後に続きます。プログラムで使用するデータセットと出力形式は、“例 1: 変数の選択とレポートの要約行の作成” (1680 ページ)で作成されます。WHERE ステートメントと FORMAT ステートメントは重要ではありませんが、ここでは出力量を制限し、値をわかりやすくします。

```
libname proclib
'SAS-library';

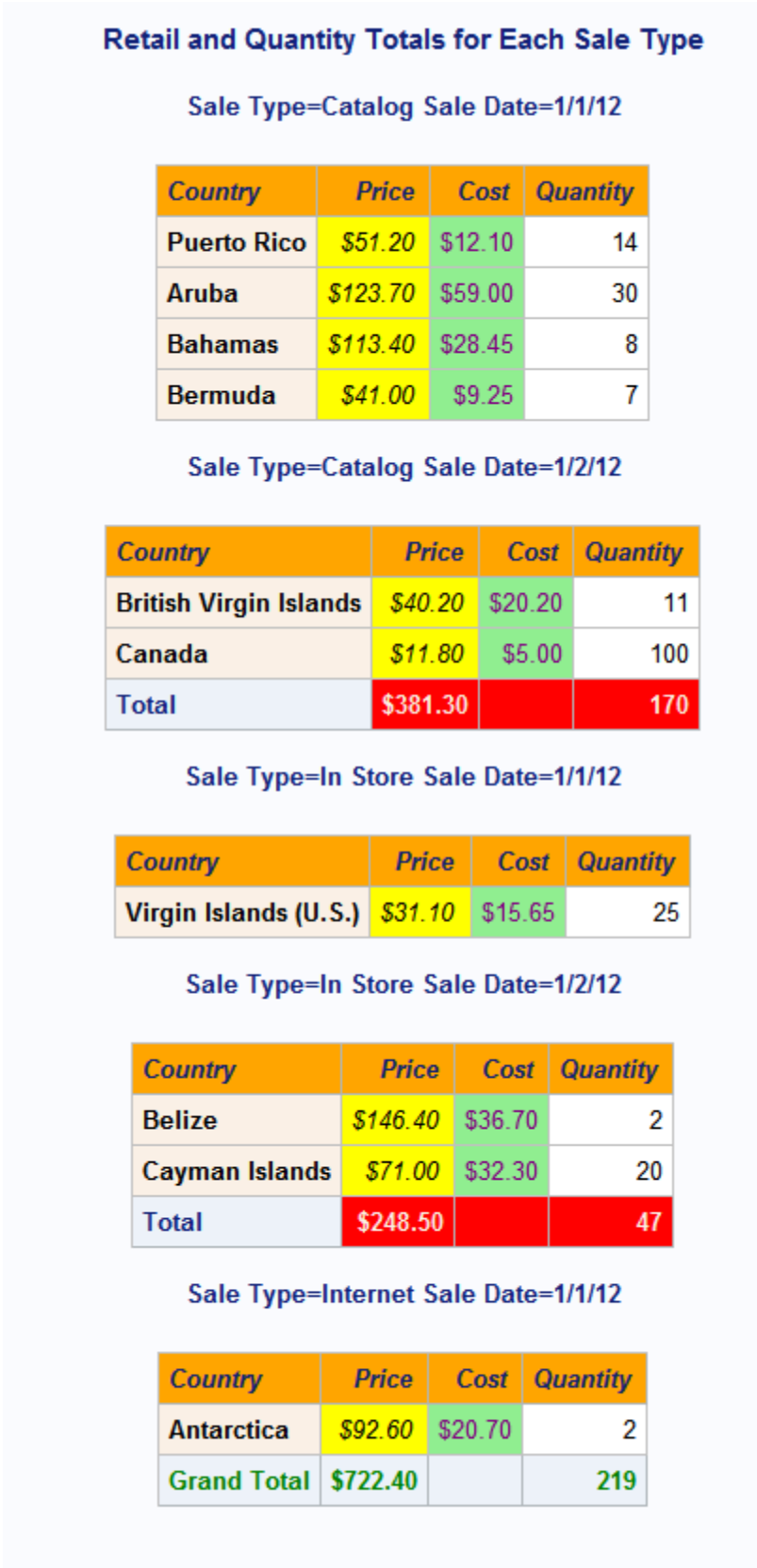
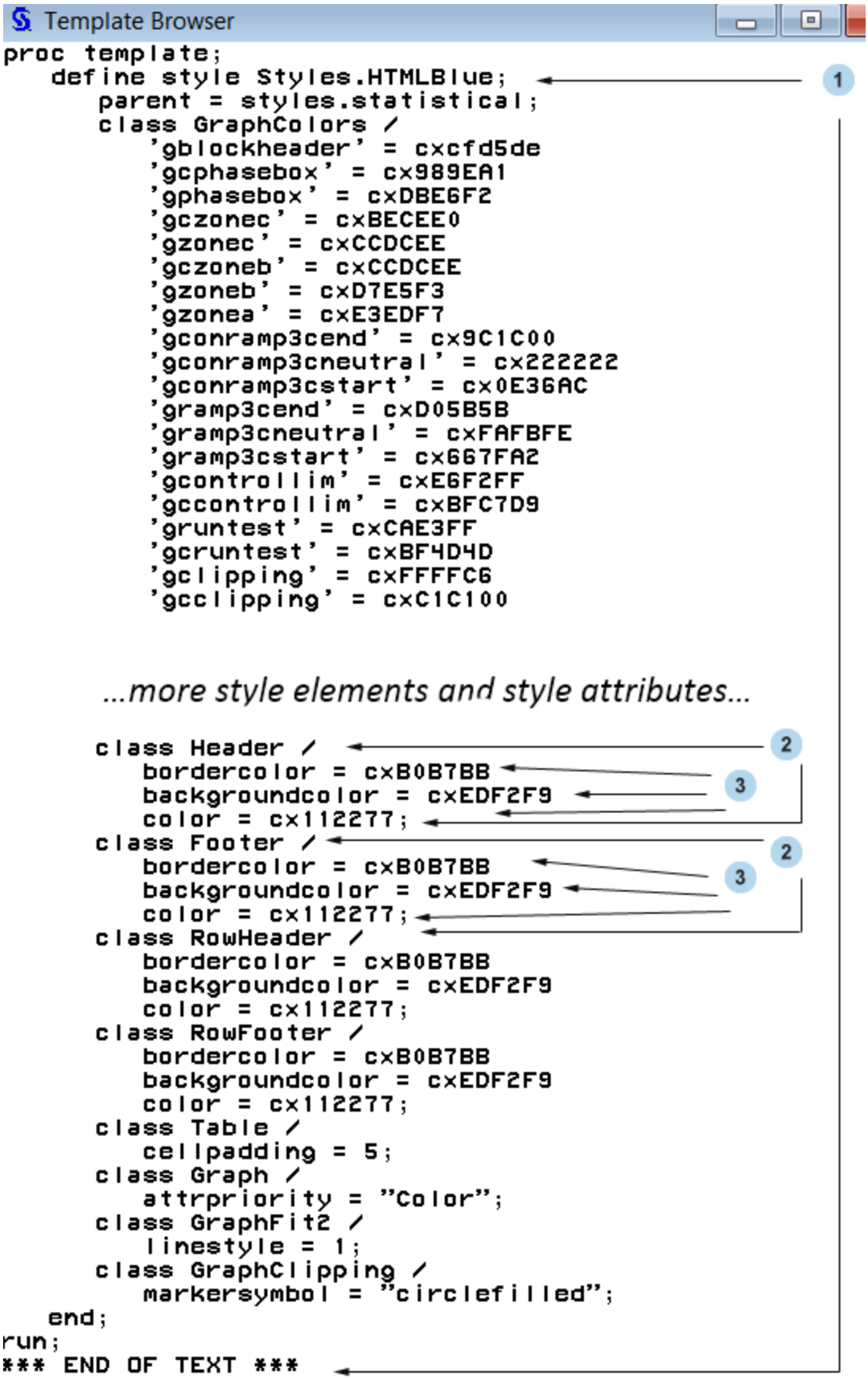
options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);

proc report data=grocery;
   where sector='se';
   format sector $sctrfmt. manager $mgrfmt.
          dept $deptfmt. sales dollar10.2;
run;
```

図 55.1 *各オブザベーションの詳細行を含む単純な詳細レポート*



次の図のレポートでは、上図と同じオブザベーションを使用しています。ただし、このレポートを作成するステートメントは、次を実行します。

- マネージャと部門の値によって行を並べ替える。
- マネージャの値ごとにデフォルト要約行を作成する。

- レポート全体のカスタマイズされた要約行を作成する。カスタマイズされた要約によって要約情報のコンテンツと表示形式を制御できますが、追加の PROC REPORT ステートメントを記述して作成する必要があります。

このレポートを作成するプログラムの説明については、“例 2: レポートでの行の並べ替え” (1684 ページ)を参照してください。

図 55.2 *デフォルト要約とカスタマイズされた要約を含む並べ替えられた詳細レポート*



次の図の要約レポートには、北セクタの店舗ごとに 1 行が含まれています。各詳細行は、入力データセットにおける 4 つのオブザベーション、部門ごとに 1 つのオブザベーションを表しています。個別の部門に関する情報は、このレポートには表示されません。代わりに、各詳細行の売上値は、すべての 4 つの部門の売上値の合計です。複数のオブザベーションをレポートの 1 行にまとめるだけでなく、このレポートを作成するステートメントは、次を実行します。

- 列ヘッダーのテキストをカスタマイズする
- 市の各セクタに対する売上を総計するデフォルト要約行を作成する
- 両方のセクタに対する売上を総計するカスタマイズされた要約行を作成する

このレポートを作成するプログラムの説明については、“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)を参照してください。

図 55.3 デフォルト要約とカスタマイズされた要約を含む要約レポート



次の図の要約レポートは、上図に似ています。大きな違いは、個別の部門に関する情報も含まれている点です。部門の選択値がそれぞれレポートの列となります。また、このレポートを作成するステートメントでは、入力データセットにない変数を計算し、表示します。

このレポートを作成するプログラムの説明については、“例 6: 変数の値ごとに列を作成する” (1695 ページ)を参照してください。

図 55.4 変数の各値の列を含む要約レポート



次の図のカスタマイズされたレポートには、各マネージャの店舗を個別のページに表示します。ここでは最初の 2 ページだけを表示しています。このレポートを作成するステートメントは、次を作成します。

- レポートのページごとにカスタマイズされたヘッダー
- 入力データセットにない計算済み変数(利益)

- マネージャの店舗の売上合計に依存するテキストを含むカスタマイズされた要約

このレポートを作成するプログラムの説明については、“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)を参照してください。

図 55.5 *カスタマイズされた要約レポート*



次の図のレポートでは、フォントの種類、フォントサイズ、位置合わせ、および表の枠の幅、セル間のスペースの幅などを制御するためのカスタマイズされたスタイル要素を使用しています。このレポートは、Output Delivery System (ODS)の HTML 出力先と STYLE=オプションをプロシジャの複数のステートメントで使用して作成されました。

このレポートを作成するプログラムの説明については、“例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1719 ページ)を参照してください。ODS の詳細については、“Output Delivery System (ODS)” (66 ページ)を参照してください。

図 55.6 HTML 出力



概念:REPORT プロシジャ

レポートのレイアウト

レイアウトの計画

レポート作成は、レポートの表示形式を明確に理解して取り組むと簡単です。決定すべき重要なことは、レポートのレイアウトです。レイアウトを設計するには、次の点について考えます。

- レポートの各列に表示する項目
- 各列を表示する順序
- 特定の変数の値ごとの列の表示の要否
- レポートで、オブザベーションごとに 1 行を使用する必要性、あるいは複数のオブザベーションの情報を 1 行にまとめる必要性の要否
- 各行を表示する順序

レポートのレイアウトを考えたら、PROC REPORT で COLUMN ステートメントと DEFINE ステートメントを使用して、レイアウトを構成します。

COLUMN ステートメントは、レポートの列に表示する項目をリストし、列の調整を記述し、複数の列にわたるヘッダーを定義します。レポート項目には、次が可能です。

- データセット変数
- プロシジャによって計算された統計量
- レポートのその他の項目から計算する変数

入力データセットのすべての変数をデータセットと同じ順序で含める場合、COLUMN ステートメントを省略します。

DEFINE ステートメントで、レポートの項目の特性を定義します。これらの特性には、レポートの項目、列ヘッダーのテキスト、値の表示に使用する出力形式を PROC REPORT がどのように使用するかが含まれます。

注: DEFINE ステートメントは、WINDOWS 環境を使用している場合の **DEFINITION** ウィンドウと同じです。

レポートでの変数の使い方

レポートのレイアウトの多くは、DEFINE ステートメントで変数に指定する使い方によって決定されます。データセット変数の場合、使い方は次のとおりです。

DISPLAY ORDER ACROSS GROUP ANALYSIS

レポートには、入力データセットにない変数を含めることができます。これらの変数の使い方が COMPUTED である必要があります。

表示変数

1 つ以上の表示変数を含むレポートには、入力データセットのオブザベーションごとに 1 行が含まれます。表示変数は、レポートの行の順序に影響しません。順序変数が表示変数の左側に表示されない場合、レポートの行の順序は、データセットのオブザベーションの順序を反映します。デフォルトで、PROC REPORT はすべての文字変数を表示変数として扱います。例については、“例 1: 変数の選択とレポートの要約行の作成” (1680 ページ)を参照してください。

順序変数

1 つ以上の順序変数を含むレポートには、入力データセットのオブザベーションごとに 1 行が含まれます。表示変数が順序変数の左側に表示されない場合、PROC REPORT は順序変数のフォーマットした値の昇順で詳細行を並べ替えます。DEFINE ステートメントで ORDER=および DESCENDING を使用して、デフォルトの順序を変更できます。

レポートに複数の順序変数が含まれている場合、PROC REPORT は、これらの変数をレポートの左から右へ並べ替えることにより詳細行の順序を作成します。PROC REPORT は、その左側の順序変数が値を変更しない限り、値が変わらない場合は、順序変数の値を行から行に繰り返しません。例については、“例 2: レポートでの行の並べ替え” (1684 ページ)を参照してください。

オブザベーションの順序は、DBMS テーブルに本来定義されていません。DBMS テーブルの入力データに対して ORDER=DATA オプションを指定する場合、PROC REPORT からデータベーステーブルに書き込まれる行の順序は保持されない可能性があります。

グループ変数

レポートに 1 つ以上のグループ変数が含まれている場合、PROC REPORT はすべてのグループ変数に対しフォーマットされた値の一意の組み合わせを持つデータセットからのすべてのオブザベーションを 1 行にまとめようとします。

PROC REPORT はグループを作成するとき、グループ変数のフォーマットされた値の昇順で詳細行を並べ替えます。デフォルトの順序は、DEFINE ステートメントの ORDER=と DESCENDING、または **DEFINITION** ウィンドウを使用して変更できます。

レポートに複数のグループ変数が含まれている場合、REPORT プロシジャは、これらの変数をレポートの左から右へ並べ替えることにより詳細行の順序を作成します。PROC REPORT は、その左側のグループ変数が値を変更しない限り、値が変わらない場合は、グループ変数の値を行から行に繰り返しません。

クラス変数を使用するプロシジャを熟知していると、グループ変数が PROC TABULATE の行ディメンションで使用されるクラス変数であることがわかります。

注: グループを常に作成できるわけではありません。レポートに 1 つ以上の統計量が関連付けられていない順序変数または表示変数が含まれている場合、PROC REPORT はオブザベーションをグループにまとめることはできません。(COLUMN ステートメント (1628 ページ)を参照)。対話型レポートウィンドウ環境で、PROC REPORT が直ちにグループを作成できない場合、プロシジャはすべての表示変数と順序変数をグループ変数に変更し、要求されたグループ変数を作成することができます。非ウィンドウ(デフォルト)環境では、グループを作成できなかった理由を説明するメッセージが SAS ログに返されます。代わりに、順序変数と同じようにグループ変数を表示する詳細レポートが作成されます。PROC REPORT が詳細レポートを作成する場合でも、グループ変数として定義する変数にはその使い方が定義に保持されます。

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)を参照してください。

分析変数

分析変数は、レポートのセルによって表されるすべてのオブザベーションに対する統計量の計算に使用される数値変数です。(列変数をグループ変数または順序変数と組み合わせ、セルが表すオブザベーションを決定します)。変数の定義または COLUMN ステートメントで統計量と分析変数を関連付けます。デフォルトで PROC REPORT は、数値変数を合計統計量の計算に使用される分析変数として使用します。

分析変数の値は、レポートで表示される場所によって異なります。

- 詳細レポートでは、詳細行の分析変数の値は、単一のオブザベーションに対して計算された変数と関連付けられた統計量の値です。単一のオブザベーションに対する統計量の計算は、実用的ではありません。ただし、変数を分析変数として使用することにより、オブザベーションセットまたはすべてのオブザベーションに対し要約行を作成できます。
- 要約レポートで、分析変数に対して表示されている値は、レポートのセルによって表されるオブザベーションセットに対して計算されるように指定する統計量の値です。
- レポートの要約行で、分析変数の値は、要約行のセルによって表されるすべてのオブザベーションに対して計算されるように指定する統計量の値です。

詳細については、“BREAK ステートメント” (1617 ページ)と “RBREAK ステートメント” (1649 ページ)を参照してください。

例については、次を参照してください。“例 2: レポートでの行の並べ替え” (1684 ページ)、“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)、“例

5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)、および “例 6: 変数の値ごとに列を作成する” (1695 ページ)

注: 要約行を含むレポートで SAS 日付を使用する際は注意が必要です。SAS 日付は数値変数です。日付は他種の変数(ORDER、GROUP、または DISPLAY)として明示的に定義しない限り、PROC REPORT によって要約されます。

列変数

PROC REPORT は、列変数の値ごとに列を作成します。PROC REPORT は、列を列変数のフォーマットした値の昇順で並べ替えます。DEFINE ステートメントで ORDER=および DESCENDING を使用して、デフォルトの順序を変更できます。その他の変数が列の定義に使用できない場合、PROC REPORT は N 統計量(レポートのセルに属している入力データセットのオブザベーション数)を表示します。COLUMN ステートメント (1628 ページ)を参照してください。

注: 表示変数と列変数が列を共有している場合、レポートには同じ列にはない別の変数も含まれている必要があります。列変数によって作成された列を参照する場合は、_cn_構文を使用する必要があります。

クラス変数を使用するプロシジャを熟知していると、列変数が PROC TABULATE で列ディメンションで使用されるクラス変数に似ていることがわかります。通常、列変数は、順序変数またはグループ変数を組み合わせて使用します。例については、“例 6: 変数の値ごとに列を作成する” (1695 ページ)を参照してください。

計算変数

計算変数は、レポートに対して定義する変数です。入力データセットにはなく、PROC REPORT によって入力データセットに追加されません。ただし、計算変数は作成されると出力データセットに含まれます。

計算変数は、次の方法で追加します。

- 計算変数を COLUMN ステートメントに含める
- 変数の使い方を DEFINE ステートメントで COMPUTED として定義する
- 変数と関連付けられている計算ブロックの変数値を計算する

例については、“例 6: 変数の値ごとに列を作成する” (1695 ページ)、“例 8: パーセントの計算” (1703 ページ)、および “例 10: 出力データセットの作成と計算変数の保存” (1710 ページ)を参照してください。

位置と使い方の相互作用

レポートの各変数の位置と使い方によって、レポートの構造とコンテンツが決まります。PROC REPORT は、順序変数とグループ変数のフォーマットされた値に従ってレポートの行を、COLUMN ステートメントで指定されているように左から右に並べ替えます。同様に、PROC REPORT は、列変数の列を変数の値に従って左から右に並べ替えます。

複数の項目は、レポートの列のコンテンツを一括で定義できます。たとえば次の図では、3 番目と 4 番目の列に表示される値は、売上(分析変数)および部門(列変数)によって一括で決定されます。この種類のレポートは、COLUMN ステートメントまたは対話型ウィンドウ環境でレポート項目を相互に置き換えることによって作成します。この調整は、レポートの積み重ね項目と呼ばれます。各項目がヘッダーを生成し、ヘッダーが相互に積み重ねられるためです。

```
options nodate pageno=1 fmtsearch=(proclib);
proc report data=grocery split='*';
   column sector manager department,sales perish;
   define sector / group format=$sctrfmt. 'Sector' '';
```

```
   define manager / group format=$mgrfmt. 'Manager* ';
   define department/ across format=$deptfmt. '_Department_';
   define sales / analysis sum format=dollar11.2 ' ';
   define perish / computed format=dollar11.2 'Perishable Total';
   break after manager / skip;
   compute perish;
      perish=_c3_+_c4_;
   endcomp;
   title "Sales Figures for Perishables in Northern Sectors";
   where sector contains 'n' and (department='p1' or department='p2');
run;
title;
```

図 55.7 *部門と売上の積み重ね*

Sales Figures for Perishables in Northern Sectors

Sector	Manager	_____Department_____ Meat/Dairy	Produce	Perishable Total
Northeast	Alomar	$190.00	$86.00	$276.00
	Andrews	$300.00	$125.00	$425.00
Northwest	Brown	$250.00	$73.00	$323.00
	Pelfrey	$205.00	$76.00	$281.00
	Reveiz	$600.00	$30.00	$630.00

複数の項目を使用して列のコンテンツを定義する場合、多くても次のうち 1 つの項目を 1 つの列に含めることができます。

- 表示変数(それ以上/以下の統計量を含む/含まない)
- 分析変数(それ以上/以下の統計量を含む/含まない)
- 順序変数
- グループ変数
- 計算変数

これらのうち複数の項目が 1 つの列にあると、表示する値に関して RROC REPORT で競合が生じます。

次の表に、列を共有できるレポート項目を示します。

注: 順序変数をその他のレポート項目と積み重ねることはできません。

表 55.1 *列を共有できるレポート項目*

	表示	分析	順序	グループ	計算	段組み	統計量
表示						X*	X
分析						X	X
順序							
グループ						X	

	表示	分析	順序	グループ	計算	段組み	統計量
計算変数						X	
段組み	X*	X			X	X	X
統計量	X	X				X	
*表示変数と列変数が列を共有する場合、レポートには同じ列にはない別の変数も含まれている必要があります。							

列が積み重ねレポート項目によって定義されている場合、PROC REPORT は ACROSS の使い方を持たない積み重ねの最下位レポート項目に指定される出力形式を使用することによって、列の値にフォーマットを適用します。

次の項目が列を専有できます。

- 表示変数
- 分析変数
- 順序変数
- グループ変数
- 計算変数
- 列変数
- N 統計量

注: 列変数によって専有される列の値は、度数カウントです。

計算ブロックの使用

計算ブロックについて

計算ブロックは、COMPUTE ステートメントと ENDCOMP ステートメントの間に表示される 1 つ以上のプログラミングステートメントです。PROC REPORT は、レポートの作成時にこれらのステートメントを実行します。計算ブロックはレポート項目(データセット変数、統計量、または計算変数)、または場所(レポートの上下、オブザベーションセットの前後)と関連付けることができます。計算ブロックは、COMPUTE ステートメントで作成できます。COMPUTE ステートメントの 1 つのフォームが計算ブロックをレポート項目と関連付けます。別のフォームは計算ブロックをレポートの場所と関連付けます(“ブレーク行の使用” (1593 ページ)を参照)。

注: COMPUTE ステートメントを使用する場合、対応する BREAK ステートメントまたは RBREAK ステートメントを使用する必要はありません(“例 2: レポートでの行の並べ替え” (1684 ページ)を参照。この例では、COMPUTE AFTER を使用しますが、RBREAK ステートメントは使用しません)。これらのステートメントは、1 つ以上の BREAK ステートメントまたは RBREAK ステートメントオプションを実装するときにのみ使用します。(COMPUTE AFTER MANAGER と BREAK AFTER MANAGER を使用する“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)を参照)。

計算ブロックの目的

レポート項目と関連付けられている計算ブロックは、次のタスクを実行できます。

- レポートの列に表示され、入力データセットには表示されない変数を定義する。

- レポート項目の表示属性を定義する(“CALL DEFINE ステートメント” (1624 ページ)を参照)。
- 要約行の“合計”の表示など、レポート項目の値を定義または変更する。

場所と関連付けられている計算ブロックは、カスタマイズされた要約を記述できます。

また、すべての計算ブロックは、ほとんどの SAS 言語要素を使用して計算を実行できます(“計算ブロックのコンテンツ” (1591 ページ)を参照)。PROC REPORT ステップには複数の計算ブロックを含めることができますが、それらをネストすることはできません。

計算ブロックのコンテンツ

計算ブロックは、COMPUTE ステートメントから始まり、ENDCOMP ステートメントで終わります。計算ブロック内では、次の SAS 言語要素を使用できます。

- %INCLUDE ステートメント
- DATA ステップステートメントは、次のとおりです。

  ARRAY
  配列参照
  割り当て
  CALL
  CONTINUE
  DO (すべてのフォーム) END
  END
  IF-THEN/ELSE
  LENGTH
  RETURN
  合計

- コメント
- Null ステートメント
- マクロ変数とマクロ呼び出し
- すべての DATA ステップ関数

また、計算ブロック内で、次のような PROC REPORT 機能を使用することもできます。

- カスタマイズされた要約の計算ブロックには、1 つ以上の LINE ステートメントを含めることができます。これによりカスタマイズされたテキストとフォーマットされた値が要約に挿入されます。(“LINE ステートメント” (1647 ページ)を参照)
- レポート項目の計算ブロックには、1 つ以上の CALL DEFINE ステートメントを含めることができます。これにより、項目の値がレポートに挿入されるたびに色や出力形式などの属性が設定されます。(“CALL DEFINE ステートメント” (1624 ページ)を参照)
- 計算ブロックは、自動変数_BREAK_を参照できます(“自動変数_BREAK_” (1594 ページ)を参照)。

計算ブロックのレポート項目を参照するための 4 つの方法

計算ブロックは、レポートの列を形成するレポート項目を参照できます(列が表示されるかどうか)。計算ブロックのレポート項目を次の 4 つの方法のうちいずれかで参照します。

- 名前。
- 変数と、変数を使用して計算する統計量の名前の両方を識別する複合名。複合名には次の形式があります。

  variable-name.statistic

- COLUMN ステートメントまたは **DEFINITION** ウィンドウを使用して作成する別名。
- 次の形式の列番号。

  '_C*n*_'
  ここで、*n* は、レポートの列の番号(左から右方向)です。

  注: 列番号が必要なのは、COMPUTED 変数が列を ACROSS 変数と共有している場合だけです。

  注: NOPRINT と NOZERO を使用して定義する列がレポートに表示されない場合でも、列を番号別に参照する場合はそれらの列をカウントする必要があります。"NOPRINT" (1641 ページ)と "NOZERO" (1642 ページ)の説明を参照してください。

注: 欠損値を含む変数を参照すると、欠損値が発生します。計算ブロックが欠損値を含む変数を参照する場合、PROC REPORT はその変数をブランク(文字変数の場合)またはピリオド(数値変数の場合)として表示します。

次の表に、計算ブロックにおける各種類の参照の使用方法を示します。

変数の種類	参照元	例
グループ	名前*	部門
順序	名前*	部門
計算	名前*	部門
表示	名前*	部門
統計量と列を共有している表示	複合名*	Sales.sum
分析	複合名*	Sales.mean
列変数と列を共有する種類	列番号**	'_c3_'

*変数に別名がある場合、その変数を別名で参照する必要があります。

**変数に別名がある場合でも、変数を列番号別に参照する必要があります。

"例 3: 別名を使用し、同一変数に複数の統計量を求める" (1687 ページ), (分析変数を別名別に参照)、"例 6: 変数の値ごとに列を作成する" (1695 ページ), (分析変数を列番号別に参照)、"例 8: パーセントの計算" (1703 ページ), (グループ変数と計算変数を名前別に参照)を参照してください。

計算ブロック処理

PROC REPORT は、計算ブロックを 2 つの異なる方法で処理します。

- 計算ブロックが場所と関連付けられている場合、PROC REPORT は計算ブロックをその場所でのみ実行します。PROC REPORT は実際にレポートの行を作成する前にグループの統計量を計算するため、レポート行セットの統計量は、これらの統計量に基づく変数の値のように、行の表示の前後に使用可能です。
- 計算ブロックがレポート項目と関連付けられている場合、PROC REPORT はその項目の列に関してレポートのすべての行の計算ブロックを実行します。レポート行

の計算変数の値は、計算ブロックでの DATA ステップステートメントの実行中に変数に割り当てられた最後の値です。PROC REPORT は値をレポート行の列に左から右へ割り当てます。その結果、レポートの右側に表示される変数に基づいて計算変数の計算を行うことはできません。

注: PROC REPORT は、計算変数をブレークで再計算します。計算ブロック処理の詳細については、"PROC REPORT でのレポート作成法 " (1670 ページ)を参照してください。

ブレーク行の使用

ブレーク行について

ブレーク行はテキストの行(ブランクを含む)で、レポートの**ブレーク**という特定の場所に表示されます。レポートには、複数のブレークを含めることができます。通常、ブレーク行はレポートの一部を視覚的に区切る、情報を要約する、またはその両方を行うために使用されます。ブレーク行は次の場所に示されます。

- レポートの始めと最後
- 各ページの上部または下部
- オブザベーションセット間(グループ変数または順序変数の値が変わるたび)

ブレーク行には次を含めることができます。

- テキスト
- 行セットまたはレポート全体に対して計算された値

ブレーク行の作成

ブレーク行の作成には、2 つの方法があります。最初の方法はより簡単です。デフォルトの要約を作成します。2 つ目の方法はより複雑です。カスタマイズされた要約を作成し、デフォルト要約を多少変更する方法を提供します。デフォルト要約とカスタマイズされた要約は、レポートの同じ場所で表示可能です。

デフォルト要約は BREAK ステートメント、RBREAK ステートメントを使用して作成します。デフォルト要約を使用して、レポートの一部を視覚的に区切る、数値変数に関する情報を要約する、またはその両方を行うことができます。オプションによりブレーク行の表示形式をある程度制御できますが、数値変数の要約を選択する場合、要約情報のコンテンツと配置を制御できません。(情報を要約するブレーク行は、要約行です)。

カスタマイズされた要約は、計算ブロックで作成されます。カスタマイズされた要約の表示形式とコンテンツを制御できますが、それを行うためのコードを記述する必要があります。

ブレーク行の順序

カスタマイズされた要約の行の順序を制御します。ただし、PROC REPORT はデフォルト要約の行の順序、デフォルト要約に関連したカスタマイズされた要約の配置を制御します。デフォルト要約に複数のブレーク行が含まれている場合、ブレーク行が表示される順序は次のとおりです。

1. 要約行
2. 改ページ

LISTING 出力の場合、ブレーク行が表示される順序は次のとおりです。

1. 上線または二重上線(LISTING 出力のみ)

2. 要約行
3. 下線または二重下線
4. ブランク行(LISTING 出力のみ)
5. 改ページ

同じ場所に対してカスタマイズされた要約を定義する場合、カスタマイズされたブレーク行が下線または二重下線の後に表示されます。これは、LISTING 出力のみで実行されます。

自動変数_BREAK_

PROC REPORT は、_BREAK_という変数を自動的に作成します。この変数には、次が含まれます。

- ブランク(現在行がブレークの一部でない場合)
- ブレーク変数の値 (現在行がオブザベーションセット間のブレークの一部である場合)
- 値 **_RBREAK_**(現在行がレポートの始めまたは最後のブレークの一部である場合)
- 値 **_PAGE_**(現在行がページの始めまたは最後のブレークの一部である場合)

複合名の使用

レポートの統計量を使用する場合、通常、Sales.sum などの複合名別に計算ブロックで参照します。ただし、レポートの部分によっては、同じ名前でもその意味が異なります。次の出力のレポートについて考えます。出力を作成するステートメントが後に続きます。使用されるユーザー定義の出力形式が PROC FORMAT ステップ (1682 ページ)によって作成されます。

```
libname proclib
'SAS-library';

options nodate pageno=1 fmtsearch=(proclib);
proc report data=grocery;
   column sector manager sales;
   define sector / group format=$sctrfmt.;
   define sales  / analysis sum
                   format=dollar9.2;
   define manager / group format=$mgrfmt.;
   break after sector / summarize;
   rbreak after / summarize;
   compute after;
      sector='Total:';
   endcomp;
run;
```

ログ 55.1 *Sales.sum の 3 つの異なる意味*

```
The SAS System 1 Sector     Manager      Sales Northeast  Alomar     $786.00
1 Andrews  $1,045.00 ---------            --------- Northeast
$1,831.00     2 Northwest  Brown      $598.00 Pelfrey    $746.00 Reveiz
$1,110.00 ---------           --------- Northwest           $2,454.00 Southeast
Jones      $630.00 Smith      $350.00 ---------           ---------
Southeast             $980.00 Southwest  Adams      $695.00 Taylor     $353.00
---------           --------- Southwest           $1,048.00 =========
========= Total:              $6,313.00     3 =========           =========
```

ここでは、Sales.sum には 3 つの異なる意味があります。

1 詳細行では、値は市のあるセクタの 1 人のマネージャの店舗の売上です。たとえば、レポートの最初の詳細行は、Alomar が経営する店舗の売上が$786.00 であったことを表しています。

2 グループ要約行では、値は 1 つのセクタのすべての店舗の売上です。たとえば、最初のグループ要約行は、北東セクタの売上が$1,831.00 であったことを表しています。

3 レポート要約行では、値$6,313.00 はその市のすべての店舗の売上です。

注: 計算ブロックの別名を持つ統計量を参照する場合、複合名は使用しないでください。通常、別名を使用する必要があります。ただし、統計量が列を列変数と共有する場合、列を列番号別に参照する必要があります(“計算ブロックのレポート項目を参照するための 4 つの方法” (1591 ページ)を参照)。

PROC REPORT によってサポートされている ODS 出力先

PROC REPORT ではすべての ODS 出力先をサポートしています。詳細については、“Understanding ODS Destinations” (*SAS Output Delivery System: User's Guide*)を参照してください。

PROC REPORT STYLE=オプションは、LISTING および OUTPUT を除くすべての ODS 出力先をサポートします。詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1655 ページ)を参照してください。

PROC REPORT は、ODS DOCUMENT プロシジャをサポートします。DOCUMENT 出力先により、分析を再実行したり、データベースクエリを繰り返すことなくデータを異なる方法で異なる出力先に再構築、ナビゲート、再生することができます。DOCUMENT 出力先により出力ストリーム全体が"生"形式で利用可能になり、アクセスしてカスタマイズできるようになります。出力は、データコンポーネントおよびテーブル定義として元の内部表現で保持されます。 詳細については、“The DOCUMENT Procedure” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。

PROC REPORT は、SAS 出力データセットの OUTPUT 出力先をサポートします。ODS ではデータの論理構造とネイティブフォームを認識しているため、プロシジャが内部的に使用した同じ結果のデータセットを表す SAS データセットを出力できます。詳細については、“ODS OUTPUT Statement ” (*SAS Output Delivery System: User's Guide*)を参照してください。

入力データセットのスレッド処理

THREADS オプションは、入力データセットの並列処理を有効化または無効化します。スレッド処理により、処理操作における一定の並列処理が達成されます。この並列処理の目的は、所定の操作を完了するための処理時間を削減し、それによって追加 CPU リソースのコストを抑えることです。詳細については、“Support for Parallel Processing” (*SAS Language Reference: Concepts*)を参照してください。

SAS システムオプション CPUCOUNT=の値はスレッド化された並べ替えのパフォーマンスに影響します。CPUCOUNT=は、スレッド化されたプロシジャで使用できるシステム CPU の数を示します。

詳細については、“THREADS System Option” (*SAS System Options: Reference*) および “CPUCOUNT= System Option” (*SAS System Options: Reference*)を参照してください。

構文: REPORT プロシジャ

ヒント: Output Delivery System をサポートします。詳細については、“Output Delivery System: Basic Concepts” (*SAS Output Delivery System: User's Guide*)を参照してください。

ステートメント ATTRIB、FORMAT、LABEL、WHERE を PROC SORT プロシジャと使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

発生するデータベース内処理に対し、データは SAS データベース内処理用に適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、“PROC REPORT のデータベース内処理” (1669 ページ)を参照してください。

```
PROC REPORT<option(s)>;
    BREAK location break-variable </ option(s)>
    BY variable-1
      <<DESCENDING> variable-2 …> <NOTSORTED>;
    COLUMNcolumn-specification(s);
        COMPUTE location <target>
          </ STYLE=<style-override(s)> >;
            LINE specification(s);
            ... select SAS language elements ...
            ENDCOMP;
        COMPUTE report-item </ type-specification>;
            CALL DEFINE (column-id ', < ' attribute-name', value>
                | _ROW_, < 'attribute-name', value>);
            ... select SAS language elements ...
            ENDCOMP;
    DEFINE report-item / <option(s)>;
    FREQ variable;
    RBREAK location </ option(s)>;
    WEIGHT variable;
```

ステートメント	タスク	例
“PROC REPORT ステートメント”	要約レポートまたは詳細レポートを作成します	Ex. 1, Ex. 2, Ex. 6, Ex. 4, Ex. 9, Ex. 10, Ex. 13
“BREAK ステートメント”	グループ変数または順序変数の値の変更時にデフォルト要約を作成します	Ex. 2, Ex. 5, Ex. 6, Ex. 7
“BY ステートメント”	各 BY グループの個別のレポートを作成します	
“CALL DEFINE ステートメント”	現在行の特定の列に属性値を設定します	Ex. 5, Ex. 13

ステートメント	タスク	例
“COLUMN ステートメント”	すべての列の調整と複数の列にわたるヘッダーの調整を記述します	Ex. 1, Ex. 3, Ex. 6, Ex. 4, Ex. 8, Ex. 9
“COMPUTE ステートメント”	PROC REPORT がレポートの作成時に実行する 1 つ以上のプログラミングステートメントを指定します	Ex. 2, Ex. 3, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 10, Ex. 13
“ENDCOMP ステートメント”	PROC REPORT がレポートの作成時に実行する 1 つ以上のプログラミングステートメントを指定します	Ex. 2
“DEFINE ステートメント”	レポート項目の使用方法と表示方法を記述します	Ex. 2, Ex. 3, Ex. 5, Ex. 6, Ex. 4, Ex. 7, Ex. 8, Ex. 10, Ex. 10, Ex. 11, Ex. 13
“FREQ ステートメント”	オブザベーションを入力データセットで複数回表示されるように扱います。	
“LINE ステートメント”	カスタマイズされた要約を記述するための PUT ステートメントの機能のサブセットを提供します	Ex. 2, Ex. 3, Ex. 5, Ex. 6, Ex. 7
“RBREAK ステートメント”	レポートの始めまたは最後、あるいは各 BY グループの始めと最後にデフォルト要約を作成します	Ex. 1, Ex. 8
“WEIGHT ステートメント”	統計量計算の分析変数の重みを指定します。	

PROC REPORT ステートメント

PRINT プロシジャ、MEANS プロシジャ、TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートの作成が可能な 1 つのレポート作成ツールに組み合わせます。

例:
- “例 1: 変数の選択とレポートの要約行の作成” (1680 ページ)
- “例 2: レポートでの行の並べ替え” (1684 ページ)
- “例 6: 変数の値ごとに列を作成する” (1695 ページ)
- “例 4: 1 つの変数に複数の統計量を表示する” (1690 ページ)
- “例 9: PROC REPORT での欠損値の処理法” (1706 ページ)
- “例 10: 出力データセットの作成と計算変数の保存” (1710 ページ)
- “例 10: 出力データセットの作成と計算変数の保存” (1710 ページ)
- “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1719 ページ)
- “例 14: CELLWIDTH=スタイル属性を PROC REPORT とともに使用する” (1726 ページ)

構文

PROC REPORT <*option(s)*>;

オプション引数の要約

DATA=*SAS-data-set*
入力データセットを指定します。

NOALIAS
計算ブロックに別名が必要になる以前に作成されたレポートを使用します。

NOCENTER
CENTER | NOCENTER を参照してください。

NOCOMPLETECOLS
COMPLETECOLS | NOCOMPLETECOLS を参照してください。

NOCOMPLETEROWS
COMPLETEROWS | NOCOMPLETEROWS を参照してください。

NOTHREADS
THREADS | NOTHREADS を参照してください。

NOWINDOWS
WINDOWS | NOWINDOWS を参照してください。

OUT=*SAS-data-set*
出力データセットを指定します。

PCTLDEF=
QNTLDEF=を参照してください。

THREADS | NOTHREADS
SAS システムオプション THREADS | NOTHREADS を無効にします。

WINDOWS | NOWINDOWS
対話型レポートウィンドウまたは非ウィンドウ環境を選択します。

Control classification levels

COMPLETECOLS | NOCOMPLETECOLS
列変数値のすべての可能な組み合わせを作成します。

COMPLETEROWS | NOCOMPLETEROWS
グループ変数値のすべての可能な組み合わせを作成します。

Control ODS output

CONTENTS='*link-text*'
出力に対し目次エントリのテキストを指定します。

SPANROWS
単一のセルが値が同じすべての行の列を占めるように指定します。

STYLE<(*location(s)*)>=<*style-override(s)*>
レポートの異なる部分に使用する 1 つ以上のスタイルの上書きを指定します。

Control the interactive report window environment

COMMAND
すべての REPORT ウィンドウのメニューバーでなくコマンド行を表示します。

HELP=*libref.catalog*
レポートのユーザー定義のヘルプを含むライブラリとカタログを識別します。

PROMPT
REPORT ウィンドウを開いて、PROMPT 機能を開始します。

Control the layout of the report

BOX
フォーマット文字を使用して、罫線文字をレポートに追加します。

BYPAGENO=*number*
BY グループ間のページ番号をリセットします。

CENTER | NOCENTER
レポートと要約テキストを中央揃えまたは左揃えにするかどうかを指定します。

COLWIDTH=*column-width*
計算変数または数値データセット変数を含む列に対して文字のデフォルト数を指定します。

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
レポートで罫線文字として使用する文字を定義します。

LS=*line-size*
レポートの行の長さを指定します。

MISSING
欠損値をグループ変数、順序変数、または列変数の有効値とみなします。

PANELS=*number-of-panels*
レポートの各ページのパネル数を指定します。

PS=*page-size*
レポートのページの行数を指定します。このオプションは、LISTING 出力にのみ影響します。

PSPACE=*space-between-panels*
パネル間のブランク文字の数を指定します。

SHOWALL
列の表示を非表示にする DEFINE ステートメントのオプションを無効にします。

SPACING=*space-between-columns*
列間のブランク文字の数を指定します。

WRAP
最初の列の別の値を表示する前に、必要に応じてレポートの列ごとに値を 1 つ連続行に表示します。

Control the statistical analysis

EXCLNPWGT
非正の重みがついたオブザベーションを分析から除外します。

QMARKERS=*number*
P^2 分位点推定方法に使用するサンプルサイズを指定します。

QMETHOD=OS|P2
分位点推定方法を指定します。

QNTLDEF=1 | 2 | 3 | 4 | 5
分位点を計算する算術定義を指定します。

VARDEF=*divisor*
分散の計算に使用する分母を指定します。

Customize column headings

NAMED

name=を、name=が値の列ヘッダーであるレポートの各値の前に書き込みます。

NOHEADER
列ヘッダーを非表示にします。

SPLIT='*character*'
区切り文字を指定します。

ODS Listing

HEADLINE
すべての列ヘッダーとそれらの間のスペースに下線を引きます。

HEADSKIP
すべての列ヘッダーの下にブランク行を書き込みます。

Store and retrieve report definitions, PROC REPORT statements, and your report profile

LIST
現在のレポートを作成する PROC REPORT コードを SAS ログに書き込みます。

NOEXEC
レポートの作成を行いません。

OUTREPT=*libref.catalog.entry*
サブミットする PROC REPORT ステップによって定義されるレポート定義を指定したカタログに保存します。

PROFILE=*libref.catalog*
使用するレポートプロファイルを識別します。

REPORT=*libref.catalog.entry*
使用するレポート定義を指定します。

オプション引数

BOX
フォーマット文字を使用して、罫線文字をレポートに追加します。これらの文字で、次を行います。

- レポートの各ページを囲む
- 列ヘッダーをレポートの本文から区切る
- 行と列を相互に区切る
- 要約行の値を同じ列のその他の値から区切る
- カスタマイズされた要約をレポートのその他の部分から区切る

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 BOX は、PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウで WRAP を使用する場合、または項目定義で FLOW を使用する場合、使用できません。

参照項目 説明 “FORMCHAR <(*position(s)*)>='*formatting-character(s)*'” (1603 ページ)

BYPAGENO=*number*
BY ステートメントが存在する場合、各 BY グループの開始時にページ番号を指定します。

範囲 0 より大きい正の整数。

制限事項 BY ステートメントが存在しない場合、このオプションは影響しません。

操作 BYPAGENO=オプションも ODS ESCAPECHAR THISPAGE 関数に影響します。

CENTER | NOCENTER

レポートと要約テキスト(カスタマイズされたブレーク行)を中央揃えまたは左揃えにするかどうかを指定します。

PROC REPORT は、検出したこれらの中央揃え指定のうち最初のものを有効化します。

- PROC REPORT ステートメントの CENTER オプションまたは NOCENTER オプション、または **ROPTIONS** ウィンドウの CENTER トグル
- PROC REPORT ステートメントの REPORT=を使用してロードされるレポート定義に保存される CENTER オプションまたは NOCENTER オプション
- SAS システムオプション CENTER または NOCENTER

操作 CENTER が有効な場合、PROC REPORT はレポートの最左の変数の前のスペースを無視します。

COLWIDTH=*column-width*

計算変数または数値データセット変数を含む列に対して文字のデフォルト数を指定します。

列幅の設定時に、PROC REPORT はまずその列に対する定義の WIDTH=を確認します。WIDTH=が存在しない場合、PROC REPORT は項目の出力形式の対応に十分な大きさの列幅を使用します出力形式が項目と関連付けられていない場合、列幅は次の表にある変数の種類に依存します。

表 55.2 *計算ブロックにおける参照の使用*

変数	結果の列幅
入力データセット内の文字変数	変数の長さ
入力データセットの数値変数	COLWIDTH=オプションの値
計算変数(数値または文字)	COLWIDTH=オプションの値

出力形式の詳細については、“FORMAT=*出力形式*” (1640 ページ)の説明を参照してください。

デフォルト 9

範囲 1 からラインサイズ

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。フォーマットされた ODS 出力先の場合、STYLE=オプションをスタイル属性 WIDTH=、CELLWIDTH=または OUTPUTWIDTH =と使用します。詳細については、“Style Attributes Tables ” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。スタイル属性 WIDTH=と CELLWIDTH=を PROC REPORT で使用する方法につ

いては、“例 14: CELLWIDTH=スタイル属性を PROC REPORT とともに使用する” (1726 ページ)を参照してください。

COMMAND
すべての REPORT ウィンドウのメニューバーでなくコマンド行を表示します。

対話型レポートウィンドウ環境で PROC REPORT を開始した後は、COMMAND コマンドを発行して、現在のウィンドウのメニューバーを表示できます。PMENU コマンドを発行して、すべての PROC REPORT ウィンドウのメニューバーを表示できます。PMENU コマンドは SAS セッションのすべてのウィンドウに影響します。これらのコマンドは、いずれもトグルです。

COMMAND の設定をレポートプロファイルに保存できます。PROC REPORT は、検出したこれらの設定のうち最初のものを有効化します。

- PROC REPORT ステートメントの COMMAND オプション
- レポートプロファイルの設定

制限事項 このオプションは、非ウィンドウ環境では影響しません。

COMPLETECOLS | NOCOMPLETECOLS
1 つ以上の組み合わせが入力データセット内で発生しない場合でも、列変数の値に可能なすべての組み合わせを作成します。結果として、列ヘッダーは単一 BY グループ内のレポートのすべての論理ページに対して同一です。

デフォルト COMPLETECOLS

操作 DEFINE ステートメントの PRELOADFMT オプションにより、度数がゼロの場合でも PROC REPORT が列変数の組み合わせに対しすべてのユーザー定義のフォーマット範囲を使用するようになります。

COMPLETEROWS | NOCOMPLETEROWS
1 つ以上の組み合わせが入力データセット内で発生しない場合でも、グループ変数の値のすべての可能な組み合わせを表示します。結果として、行ヘッダーは単一 BY グループ内のレポートのすべての論理ページに対して同一です。

デフォルト NOCOMPLETEROWS

操作 DEFINE ステートメントの PRELOADFMT オプションにより、度数がゼロの場合でも PROC REPORT がグループ変数の組み合わせに対しすべてのユーザー定義のフォーマット範囲を使用するようになります。

CONTENTS='*link-text*'
デフォルトで作成された、または STYLE=オプションをサポートする ODS 出力先のオプション設定で作成された目次のエントリのテキストを指定します。

ヒント OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

DATA=*SAS-data-set*
入力データセットを指定します。

参照項目 “入力データセット” (25 ページ)

EXCLNPWGT
非正(ゼロまたは負)の重みがついたオブザベーションを分析から除外します。デフォルトでは、PROC REPORT は負の重みのオブザベーションを重みが 0 のオブザベーションとして処理し、オブザベーション合計数に含めます。

別名 EXCLNPWGTS

要件 WEIGHT ステートメントを使用する必要があります。

参照項目 “WEIGHT ステートメント” (1653 ページ)

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
レポートで罫線文字として使用する文字を定義します。

position(s)
SAS フォーマット文字列における 1 つ以上の文字の位置を識別します。スペースまたはカンマで位置を区切ります。

デフォルト *position(s)*)を省略すると、すべての可能な 20 の SAS フォーマット文字を指定することになります

注 PROC REPORT は、SAS が提供する 20 のフォーマット文字のうち 12 を使用します。表 55.3 (1603 ページ)に、PROC REPORT が使用するフォーマット文字を示します。図 55.8 (1604 ページ)に、PROC REPORT からの出力で共通して使用される一部のフォーマット文字の使用を示します。

formatting-character(s)
指定位置に使用する文字をリストします。PROC REPORT は *formatting-character(s)*の文字をリストされている順序で *position(s)*に割り当てます。たとえば、次のオプションではアスタリスク(*)を 3 番目のフォーマット文字に、数字記号(#)を 7 番目の文字に割り当てます。その他の文字は変更されません。

```
formchar(3,7)='*#'
```

表 55.3 *PROC REPORT によって使用されるフォーマット文字*

Position	デフォルト	表示に使用
1	|	右枠、左枠、列間の区切り
2	-	上枠、下枠、および行間の水平区切り線。ブレーク行の下線と上線および HEADLINE オプションが描く下線
3	-	左枠の先頭文字
4	-	列を区切る文字の行の先頭文字
5	-	右枠の先頭文字
6	|	水平区切り線の行の最左文字

7	+	垂直文字の列と水平文字の行の交点
8	|	水平区切り線の行の最右文字
9	-	左枠の末尾文字
10	-	列を区切る文字の行の末尾文字
11	-	右枠の末尾文字
13	=	ブレーク行の二重上線と二重下線

図 55.8 *PROC REPORT 出力のフォーマット文字*



制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 SAS システムオプション FORMCHAR=では、デフォルトのフォーマット文字を指定します。システムオプションは、フォーマット文字の全体の文字列を定義します。プロシジャの FORMCHAR=オプションでは、選択した文字を再定義できます。

ヒント 16 進数文字を含む *formatting-characters* の文字を使用できます。16 進数文字を使用する場合、**x** を終了引用符の後に付ける必要があります。たとえば、次のオプションは 16 進数文字 2D を 3 番目のフォーマット文字に、16 進数文字 7C を 7 番目の文字に割り当てますが、その他の文字は変更しません。**formchar(3,7)='2D7C'x**

HEADLINE
レポートの各ページの上部のすべての列ヘッダーとそれらの間のスペースに下線を引きます。

HEADLINE オプションは、2 番目のフォーマット文字で下線を引きます(“FORMCHAR <(*position(s)*)>='*formatting-character(s)*'” (1603 ページ)の説明を参照)。

デフォルト　ハイフン(-)

制限事項　このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

ヒント　LISTING 出力では、HEADLINE を使用するかわりに各列ヘッダーの最終行として 2 つのハイフン ('--') を使用することにより、列ヘッダーに下線を引き、その間のスペースに下線を引かずにおくことができます。

HEADSKIP
レポートの各ページの上部のすべての列ヘッダーの下(または HEADLINE オプションで書き込む下線の下)にブランク行を書き込みます。

制限事項　このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

HELP=*libref.catalog*
レポートのユーザー定義のヘルプを含むライブラリとカタログを識別します。このヘルプは、CBT または HELP カタログエントリに置くことができます。SAS/AF ソフトウェアの BUILD プロシジャを使用して、レポートの各項目に対し CBT または HELP エントリを書き込むことができます。レポートに対するすべてのエントリを同じカタログに保存します。

特定のレポート項目に関するヘルプのエントリ名をそのレポート項目の **DEFINITION** ウィンドウまたは DEFINE ステートメントで指定します。

制限事項　このオプションは、レポートウィンドウでのみ機能します。

LIST
現在のレポートを作成する PROC REPORT コードを SAS ログに書き込みます。

このリストは、次の点でサブミットするステートメントと異なっていることがあります。

- 指定していない可能性のある一部のデフォルト値が表示される。
- REPORT プロシジャ固有でない一部のステートメントが、PROC REPORT ステップでサブミットするか、以前にサブミットしたかどうかに関係なく省略される。このようなステートメントには、次のようなものがあります。

BY　FOOTNOTE　FREQ　TITLE　WEIGHT　WHERE

- 次の PROC REPORT ステートメントオプションが省略される。

LIST
OUT=
OUTREPT=
PROFILE=
REPORT=
WINDOWS | NOWINDOWS

- 次の style(<*location*>)=オプションが含まれる。

CENTER　SPACING

HEADER USAGE
LEFT WIDTH
RIGHT

注: WIDTH および SPACING は、LISTING のみに適用されます。

- SAS システムオプションが省略される。
- 自動マクロ変数が解決される。

制限事項 DEFINE ステートメントで style(column)=オプションを指定すると、LIST オプションでは列別のスタイル設定がサポートされません。

LS=*line-size*
レポートの行の長さを指定します。

PROC REPORT は、検出したページサイズ指定を次の優先順位で有効化します。

- PROC REPORT ステートメントの LS=オプションまたは **ROPTIONS** ウィンドウの LINESIZE=
- PROC REPORT ステートメントの REPORT=を使用してロードされるレポート定義に保存される LS=設定
- SAS システムオプション LINESIZE=

注: PROC REPORT LS=オプションは、その他すべてのページサイズオプションよりも優先されます。

範囲 64-256 (整数)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

MISSING
欠損値をグループ変数、順序変数、または列変数の有効値とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ異なる値とみなされます。各欠損値のグループがレポートに表示されます。MISSING オプションを省略すると、PROC REPORT はグループ変数、順序変数、列変数に対する欠損値を含むオブザベーションをレポートに含めません。

参照項目 “Missing Values” (*SAS Language Reference: Concepts*)を参照

例 “例 9: PROC REPORT での欠損値の処理法” (1706 ページ)

NAMED
name=を、*name* が値の列ヘッダーであるレポートの各値の前に書き込みます。

操作 NAMED オプションの使用時、PROC REPORT は自動的に NOHEADER オプションを使用します。

ヒント NAMED を WRAP オプションと組み合わせて使用し、個別のページに広いレポートの列を置くのではなく、レポートの単一行に対しすべての列を連続行にラップするレポートを作成します。

NOALIAS
計算ブロックに別名が必要になる以前に作成されたレポートを使用できます。NOALIAS を使用すると、計算ブロックの別名を使用できません。

NOCENTER
“CENTER | NOCENTER ” (1601 ページ)を参照してください。

NOCOMPLETECOLS
“COMPLETECOLS | NOCOMPLETECOLS” (1602 ページ)を参照してください。

NOCOMPLETEROWS
“COMPLETEROWS | NOCOMPLETEROWS” (1602 ページ)を参照してください。

NOEXEC
レポートの作成を行いません。NOEXEC を OUTREPT=と使用して、レポート定義をカテゴリエントリに保存します。NOEXEC を LIST、REPORT=と使用して、指定したレポート定義のリストを表示します。

別名 NOEXECUTE

NOHEADER
複数の列にわたるヘッダーを含む列ヘッダーを非表示にします。

対話型レポートウィンドウ環境で列ヘッダーの表示を非表示にすると、レポート項目を選択できません。

NOTHREADS
“THREADS | NOTHREADS” (1615 ページ)を参照してください。

NOWINDOWS
“WINDOWS | NOWINDOWS ” (1617 ページ)を参照してください。

別名 NOWD

デフォルト デフォルトのモードは非ウィンドウ環境です。NOWINDOWS または NOWD を指定する必要はありません。

OUT=*SAS-data-set*
出力データセットを指定します。このデータセットが存在しない場合、PROC REPORT によって作成されます。データセットには、レポート行ごとに 1 つのオブザベーション、一意の要約行ごとに 1 つのオブザベーションが含まれます。カスタマイズされた要約とデフォルト要約をレポートの同じ場所で使用すると、出力データセットにはオブザベーションが 1 つだけ含まれます。これは、2 つの要約のデータ表示方法のみ異なるためです。カスタマイズに関する情報(下線、色、テキストなど)はデータではなく、出力データセットには保存されません。

出力データセットには、レポートの列ごとに変数が 1 つ含まれます。PROC REPORT は、出力データセットの対応する変数の名前としてレポート項目の名前を使用しようとします。ただし、データセット変数が列変数に満たない、または列変数を超える場合、またはデータセット変数が COLUMN ステートメントに別名なしで複数回表示される場合、この代替を実行することはできません。この場合、変数の名前は列番号(_C1_、_C2_など)に基づきます。

入力データセット変数から派生する出力データセット変数は、対応する出力形式を入力データセットに保持します。列を定義する唯一の項目が列変数でない限り、PROC REPORT はレポートの対応する列ヘッダーからこれらの変数に対しラベルを派生します。この場合、変数にラベルはありません。複数の項目が 1 つの列に積み重ねられると、対応する出力データセット変数のラベルが列の分析変数から取得されます。

出力データセットには、_BREAK_という文字変数も含まれます。出力データセットのオブザベーションがレポートの詳細行から派生する場合、_BREAK_の値は、欠損かブランクになります。オブザベーションが要約行から派生する場合、_BREAK_の値は要約行または**_RBREAK_**と関連付けられているブレーク変数の名前です。オブザベーションが COMPUTE BEFORE _PAGE_ステートメントまたは COMPUTE AFTER _PAGE_ステートメントから派生する場合、_BREAK_の値は**_PAGE_**となります。ただし、COMPUTE BEFORE _PAGE_と COMPUTE AFTER

_PAGE_の場合、**_PAGE_**値は出力データセットにのみ書き込まれます。この値はプロシジャの実行中は自動変数_BREAK_の値として利用できません。

ヒント 出力データセットは、ODS OUTPUT ステートメントを使用して作成できます。ODS OUTPUT によって作成されるデータセットは、OUT=オプションによって作成されるものと同じです。“ODS OUTPUT Statement ” (*SAS Output Delivery System: User's Guide*)を参照してください。

例 “例 10: 出力データセットの作成と計算変数の保存” (1710 ページ)

“例 10: 出力データセットの作成と計算変数の保存” (1710 ページ)

OUTREPT=*libref.catalog.entry*
サブミットする PROC REPORT ステップによって定義されるレポート定義を指定したカタログエントリに保存します。PROC REPORT はエントリに種類 REPT を割り当てます。

保存されたレポート定義は、サブミットするステートメントと次の点で異なります。

- REPORT プロシジャ固有でない一部のステートメントが、PROC REPORT ステップでサブミットするか、ステップのサブミット時にすでに有効かどうかに関係なく省略される。このようなステートメントには、次のようなものがあります。

BY	TITLE
FOOTNOTE	WEIGHT
FREQ	WHERE

- 次の PROC REPORT ステートメントオプションが含まれる。その他のオプションは省略される。

BOX	NOHEADER
CENTER	PAGESIZE
COLWIDTH	PANELS
COMPLETECOLS	PCTLDEF
COMPLETEROWS	PSPACE
FORMCHAR	QMARKERS
HEADLINE	QMETHOD
HEADSKIP	SHOWALL
HELP	SPACING
LINESIZE	SPLIT
MISSING	VARDEF
NAMED	WRAP

- SAS システムオプションが省略される。
- 自動マクロ変数が解決される。

PANELS=*number-of-panels*
レポートの各ページのパネル数を指定します。レポートの幅がページサイズの半分より小さい場合、データを複数の列セットに表示でき、他の場合は複数のページに表示される行が同じページに表示されます。各列セットが *panel* です。この種類のレポートのわかりやすい例は電話帳で、1 ページに複数のパネルの名前と電話番号が含まれます。

PROC REPORT がマルチパネルレポートを書き込む際は、1 つのパネルがいっぱいになってから次を開始します。

ページに合うパネル数は、次によって異なります。

- パネルの幅
- パネル間のスペース
- ページサイズ

デフォルト 1

注 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。ただし、ステートメント ODS PRINTER、ODS PDF、ODS RTF の COLUMNS=オプションの結果は似ています。詳細については、*SAS Output Delivery System: ユーザーガイド*のステートメントを参照してください。

ヒント *number-of-panels* がページに収まるパネル数より大きい場合、PROC REPORT はできるだけ多くのパネルを作成します。多くのパネル(たとえば 99)を指定して、PROC REPORT がページに収まる最大数のパネルにデータを入力するようにします。

参照項目 パネル間のスペースとページサイズの詳細については、PSPACE= (1610 ページ) の説明と LS= (1606 ページ)の説明を参照してください。

PCTLDEF=
QNTLDEF= (1611 ページ)を参照してください。

PROFILE=*libref.catalog*
使用するレポートプロファイルを識別します。レポートプロファイルは次を実行します。

- REPORT ウィンドウと COMPUTE ウィンドウに対し代替メニューバーとメニューを定義するメニューの場所を指定します
- WINDOWS、PROMPT、COMMAND のデフォルト値を設定します

PROC REPORT は、プロファイルとして指定するカタログのエントリ REPORT.PROFILE を使用します。そのようなエントリが存在しない場合、またはプロファイルを指定しない場合、PROC REPORT は SASUSER.PROFILE のエントリ REPORT.PROFILE を使用します。プロファイルがない場合、PROC REPORT はオプションのデフォルトメニューとデフォルト設定を使用します。

代替レポートウィンドウ環境での PROC REPORT の使用時に、**PROFILE** ウィンドウからプロファイルを作成します。プロファイルを作成するには、次の操作を実行します。

- PROC REPORT を WINDOWS オプションを使用して起動します。
- **ツール** ⇨ **レポートプロファイル**の順に選択します。
- ニーズに合うフィールドに入力します。
- **OK** を選択して、**PROFILE** ウィンドウを終了します。ウィンドウの終了時、PROC REPORT はプロファイルを SASUSER.PROFILE.REPORT.PROFILE に保存します。CATALOG プロシジャまたは **Explorer** ウィンドウを使用して、プロファイルを別の場所にコピーします。

  注: **PROFILE** ウィンドウを開いた後、プロファイルを作成しない場合、**CANCEL** を選択して、ウィンドウを閉じます。

PROMPT
REPORT ウィンドウを開いて、PROMPT 機能を開始します。この機能を使用して、新しいレポートの作成、データセット変数または統計量の既存レポートへの追加を行えます。

PROC REPORT をプロンプトで開始する場合、最初のウィンドウではプロンプト中に使用されるオブザベーション数を制限できます。対話モードの終了時、PROC REPORT は次の制限を削除します。

- PROC REPORT ステートメントの PROMPT オプション
- レポートプロファイルの設定

PROC REPORT ステートメントから PROMPT を省略すると、プロシジャはレポートプロファイルの設定を使用します(ある場合)。レポートプロファイルがない場合、PROC REPORT はプロンプト機能を使用しません。レポートプロファイルの詳細については、“プロファイル” (1749 ページ)を参照してください。

制限事項 PROMPT オプションを使用するときに、**REPORT** ウィンドウを開きます。**REPORT** ウィンドウが開いているときは、プロシジャ出力を ODS 出力先に送信できません。

ヒント PROMPT の設定をレポートプロファイルに保存できます。PROC REPORT は、検出したこれらの設定のうち最初のものを有効化します。

PS=*page-size*
レポートのページの行数を指定します。

PROC REPORT は、検出したこれらのページサイズ指定のうち最初のものを有効化します。

- PROC REPORT ステートメントの PS=オプション
- PROC REPORT ステートメントの REPORT=を使用して指定されるレポート定義の PS=設定
- SAS システムオプション PAGESIZE=

範囲 15-32、767 (整数)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

PSPACE=*space-between-panels*
パネル間のブランク文字の数を指定します。PROC REPORT は、レポートのすべてのパネルを同じ数のブランク文字によって区切ります。各パネルに対し、その幅とそれをパネルから左に区切るブランク文字の数の合計は、ページサイズを超えることはできません。

デフォルト 4

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

QMARKERS=*number*
P^2 推定方法に使用するマーカーのデフォルト数を指定します。マーカー数によって固定メモリ空間のサイズを制御します。

デフォルト デフォルト値は、要求する分位点によって異なります。中央値(P50)の場合、*number* は 7 です。分位点(P25 と P75)の場合、*number* は 25 です。分位点 P1、P5、P10、P90、P95 または P99 の場合、*number* は 105 です。複数の分位点を要求する場合、PROC REPORT は、*number* の最も大きいデフォルト値を使用します。

範囲 3 より大きい奇数の整数

ヒント　デフォルト設定を超えるマーカー数を増やし、推定の正確性を向上させます。マーカー数を減らしてコンピューティングリソースを節約することができます。

QMETHOD=OS|P2
PROC REPORT が分位点の計算時に入力データの処理に使用する方法を指定します。オブザベーション数が QMARKERS=オプションの値以下で、QNTLDEF=オプションの値が 5 の場合、いずれの方法でも結果は同じになります。

OS
順序統計量を使用します。PROC UNIVARIATE は、この方法を使用します。

注: この方法は、非常にメモリを消費する可能性があります。

P2
P^2 方法を使用して、分位点を概算します。

デフォルト　OS

制限事項　QMETHOD=P2 の場合、PROC REPORT は MODE と重み付き分位点を計算しません。

ヒント　QMETHOD=P2 の場合、一部の分位点(P1、P5、P95、P99)の信頼性のある推定は、裾の重い分布または歪度が高い分布を含むデータセットなど、一部のデータセットには可能でないことがあります。

QNTLDEF=1 | 2 | 3 | 4 | 5
QMETHOD=オプションの値が OS の場合にプロシジャが分位点の計算に使用する算術定義を指定します。QMETHOD=P2 の場合、QNTLDEF=5 を使用する必要があります。

別名　PCTLDEF=

デフォルト　5

参照項目　“分位数と関連統計量” (2083 ページ)

REPORT=*libref.catalog.entry*
使用するレポート定義を指定します。PROC REPORT は、すべてのレポート定義を種類 REPT のエントリとして SAS カタログに保存します。

操作　REPORT=を使用する場合、COLUMN ステートメントを使用できません。

参照項目　“OUTREPT=*libref.catalog.entry*” (1608 ページ)

SHOWALL
列の表示を非表示にする DEFINE ステートメントのオプションを無効にします。

参照項目　の NOPRINT と NOZERO “DEFINE ステートメント” (1633 ページ)

SPACING=*space-between-columns*
列間のブランク文字の数を指定します。各列について、その幅、その間のブランク文字、その左側の列の合計がページサイズを超えることはできません。

デフォルト　2

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 DEFINE ステートメントの SPACING=を使用して間隔を特定の項目の左に変更しない限り、PROC REPORT は、PROC REPORT ステートメントの SPACING=で指定するブランク文字の数によってレポートのすべての列を区切ります。

CENTER が有効な場合、PROC REPORT はレポートの最左の変数の前のスペースを無視します。

SPANROWS

GROUP 列または ORDER 列の値が複数の行で同じ場合、値が同じすべての行の列を占める単一のセルに値が表示されるように指定します。ボックスが列のその部分に対して作成され、そのボックスに行は表示されません。

SPANROWS オプションにより、値がページにわたって分割するときに GROUP 変数値と ORDER 変数値を繰り返すこともできます。PDF、PS、TAGSETS.RTF 出力先だけが、機能のこの部分をサポートします。

注 SPANROWS オプションは、レポートウィンドウ、データセット、LISTING、または OUTPUT 出力先に影響しません。

ページの下部に LINE ステートメントが表示される場合、値が RTF および TAGSETS.RTF ページにわたって分割するときに GROUP 変数値と ORDER 変数値は繰り返されません。

ヒント 要約行は、その他の場合は単一セルにまたがる一連の行の真ん中に表示される場合、列に独自のセルを挿入します。このアクションにより、要約行の後に来る GROUP 変数または ORDER 変数の値が変更されない場合でもまたがったセルが 2 つのセルに分割されます。

第 508 上に準拠した PROC REPORT 出力を含む HTML 出力を作成するため、PROC REPORT の SPANROWS オプションを指定し、HTML ステートメントの HEADER_DATA_ASSOCIATIONS=yes OPTIONS オプションを指定する必要があります。第 508 条は、リハビリテーション法第 508 条(1973 年制定)の改訂時に米国政府が採用した電子情報技術のユーザー補助標準です。サンプルコードは次のとおりです。

```
ods html file="sec508.html"
options(header_data_associations="yes");
proc report data=energy spanrows;
```

SPLIT='*character*'

区切り文字を指定します。PROC REPORT は、その文字に達すると列テキストを分割し、ヘッダーを次の行に続けます。区切り文字が発生するたびにラベルの最大 256 文字に考慮されますが、区切り文字自体は列ヘッダーまたはテキスト値の一部ではありません。

デフォルト slash (/)

制限事項 このオプションは、LISTING 出力以外の ODS 出力先の列ヘッダーでのみ機能します。

操作 DEFINE ステートメントの FLOW オプションは、区切り文字を有効化します。

STYLE<(*location(s)*)>=<*style-override(s)*>
レポートの異なる部分に使用する 1 つ以上のスタイルの上書きを指定します。

location(s)
レポートの中で STYLE=オプションの影響を受ける部分を識別します。次の表に、*location* の値によって影響を受けるレポート内の各部分を示します。

location の有効な値とデフォルト値は、STYLE=オプションが記述されるステートメントによって異なります。次の表では、*location* の有効な値とデフォルト値をステートメントごとに示します。同じ STYLE=オプションで複数の *location* を指定するには、各値をスペースで区切ります。

表 55.4 *PROC REPORT の各ステートメントの場所とデフォルトのスタイル要素*

ステートメント	有効な location 値	デフォルトの location 値	レポート内で影響を受ける部分	デフォルトのスタイル要素
PROC REPORT	COLUMN HEADER | HDR SUMMARY REPORT LINES CALLDEF	REPORT	レポート全体	Table
BREAK	SUMMARY LINES	SUMMARY	要約行	DataEmphasis
CALL DEFINE	CALLDEF	CALLDEF	CALL DEFINE ステートメントによって識別されるセル	Data
COMPUTE	LINES	LINES	LINE ステートメントによって生成される行	LineContent
DEFINE	COLUMN: HEADER| HDR	COLUMN HEADER	列のセル 列ヘッダー	COLUMN:Data HEADER:Header
RBREAK	SUMMARY LINES	SUMMARY	要約行 LINE ステートメントによって生成される行	DataEmphasis

次の表に表示されるすべての名前は、PROC ステートメントの *location(s)*の代わりに使用できます。DEFINE ステートメントには、column と header を受け入れます。BREAK および BRBREAK ステートメントは summary と lines を受け入れます。

図 55.9 PROC REPORT の Location および対応するステートメント



PROC REPORT location 以外の PROC REPORT ステートメントでの指定は、PROC REPORT ステートメントの同じ指定を無効にします。ただし、PROC REPORT ステートメントで指定し、別の PROC REPORT ステートメントで無効にしないスタイル属性は継承されます。たとえば、PROC REPORT ステートメントですべての列ヘッダーに対し青の背景と白い前景の指定し、PROCREPORT DEFINE ステートメントで変数の列ヘッダーに対し灰色の背景を指定すると、特定の列ヘッダーの背景は灰色になり、前景は(PROC REPORT ステートメントでの指定に従って)白になります。

style-override

レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。スタイルの無効化は次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

  注: これらのスタイルは、PROC ステートメントで指定されているスタイルよりも優先されます。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2 …*>]

注: 角かっこ([と])の代わりに中かっこ({と})を使用できます。

style-element-name

ODS スタイルテンプレートの各部分を表すスタイル要素の名前。SAS はいくつかの スタイルテンプレートを提供しています。ユーザーは TEMPLATE プロシジャを使用して、独自のスタイルテンプレートを作成できます。を参照してください。*SAS 9.4 Output Delivery System: Procedures Guide*

参照項目 PROC REPORT でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1655 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表は、“テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ)を参照してください。

style-attribute-name
変更する属性を指定します。PROC PRINT、PROC TABULATE および PROC REPORT の STYLE=オプションで設定できる一般的に使用されているスタイル属性のリストについては、表 55.9 (1663 ページ)を参照してください。

参照項目 PROC REPORT でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1655 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表は、“テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目 PROC REPORT でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1655 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用したスタイル属性値の割り当て” (1667 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表は、“テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ)を参照してください。

制限事項 OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目 “テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ) を参照。

例 “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1719 ページ)

THREADS | NOTHREADS
入力データセットの並列処理を有効化または無効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS |

NOTHREADS を無効にします。(制約を参照)。詳細については、“Support for Parallel Processing” (*SAS Language Reference: Concepts*) を参照してください。

デフォルト SAS システムオプション THREADS | NOTHREADS の値。

制限事項 サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

操作 PROC REPORT は、BY ステートメントが指定されるとき、または SAS システムオプション CPUCOUNT が 2 未満のとき以外は SAS システムオプション THREADS の値を使用します。PROC REPORT ステートメントで THREADS オプションを指定して、PROC REPORT がこうした状況で並列処理を使用するように強制することができます。

注 並列処理でもあるスレッド処理が有効な場合、オブザベーションは予想外の順序で返されることがあります。ただし、BY ステートメントが指定されている場合、オブザベーションは正しく並べ替えられます。

VARDEF=*divisor*

分散と標準偏差の計算に使用する分母を指定します。次の表に、*divisor* に可能な値と、関連する分母を示します。

表 55.5 *VARDEF=に可能な値*

値	分母	分母の式
DF	自由度	$n - 1$
N	オブザベーション数	n
WDF	重みの合計 - 1	$(\Sigma_i\, w_i) - 1$
WEIGHT | WGT	重みの合計	$\Sigma_i\, w_i$

プロシジャは分散を*CSS*/*divisor*として計算します。*CSS*は修正平方和で、$\Sigma\,(x_i - \bar{x})^2$と等しくなります。分析変数を重み付けする場合、*CSS*は$\Sigma\, w_i(x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト DF

要件 平均値の標準誤差とスチューデントの *t* 検定を比較するには、VARDEF= のデフォルト値を使用します。

ヒント WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。*i* 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$、およびw_iは *i*

番目のオブザベーションの重みです。これにより、オブザベーションの分散の推定が単位重みとともに生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな n に対し)$\sigma^2/\bar{w}$の推定になります。$\bar{w}$は、平均重みです。これにより、オブザベーションの分散の漸近的推定が平均重みとともに生成されます。

参照項目 "WEIGHT" (74 ページ)

WINDOWS | NOWINDOWS
対話型レポートウィンドウまたは非ウィンドウ環境を選択します。

WINDOWS を使用する場合、SAS は対話型レポートインターフェイス用の **REPORT** ウィンドウを開きます。これにより、レポートを繰り返し変更し、その変更をすぐに確認できます。NOWINDOWS を使用する場合、PROC REPORT は **REPORT** ウィンドウなしで実行し、その出力をオープンな出力先に送信します。

別名 WD | NOWD

デフォルト NOWD.非ウィンドウ環境で作業するために、NOWINDOWS または NOWD を指定する必要はありません。

制限事項 WINDOWS オプションを使用する場合、出力を SAS データセットまたは Printer 出力先にのみ送信できます。

参照項目 WINDOWS 環境を使用している場合は、PROFILE= (1609 ページ)のレポートプロファイルに関する情報を参照してください。

例 "例 1: 変数の選択とレポートの要約行の作成" (1680 ページ)

WRAP
最初の列の別の値を表示する前に、必要に応じてレポートの列ごとに値を 1 つ連続行に表示します。(デフォルトで、PROC REPORT は 1 ページに収まる列に対してのみ値を表示します。これらの列の値をページに入力してから、次のページで残りの列の値の表示を開始します。

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 WRAP が有効な場合、項目定義で PROC REPORT は PAGE を無視します。

ヒント 通常、列ヘッダーのラップを避けるために WRAP を NAMED オプションと組み合わせて使用します。

BREAK ステートメント

ブレーク(グループ変数または順序変数の値の変更)時にデフォルト要約を作成します。要約の情報は一連のオブザベーションに適用されます。オブザベーションは、レポートのブレーク変数、そのブレーク変数の左側のその他すべてのグループ変数または順序変数に対する値の一意の組み合わせを共有します。

例: "例 2: レポートでの行の並べ替え" (1684 ページ)
"例 5: 複数のオブザベーションをレポートの 1 行にまとめる" (1692 ページ)

“例 6: 変数の値ごとに列を作成する” (1695 ページ)
“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)

構文

BREAK *location break-variable* </ *option(s)*>;

オプション引数の要約

COLOR=*color*
: **REPORT** ウィンドウのブレーク行の色を指定します。

CONTENTS='*link-text*'
: 目次で使用されるリンクテキストを指定します。

DOL
: 各値に二重上線を引きます。

DUL
: 各値に二重下線を引きます。

OL
: 各値に上線を引きます。

PAGE
: 最終ブレーク行の後に新しいページを開始します。

SKIP
: 最終ブレーク行に対しブランク行を書き込みます。

STYLE<*location(s)*>=<*style-override*>
: デフォルト要約行、カスタマイズされた要約行、またはその両方に使用するスタイルの上書きを指定します。

SUMMARIZE
: ブレーク行の各グループの要約行を書き込みます。

SUPPRESS
: 要約行のブレーク変数の値の印刷、ブレーク変数を含む列のブレーク行の下線または上線の印刷を行いません。

UL
: 各値に下線を引きます。

必須引数

location
ブレーク行の配置を制御します。次の値のうちいずれかになります。

AFTER
: ブレーク変数の値が同じ各行セットの最終行の直後にブレーク行を置きます。

BEFORE
: ブレーク変数の値が同じ各行セットの開始行の直前にブレーク行を置きます。

break-variable
グループ変数または順序変数です。REPORT プロシジャはこの変数の値が変わるたびにブレーク行を書き込みます。

オプション引数

COLOR=*color*
REPORT ウィンドウのブレーク行の色を指定します。デフォルトの色は、**SASCOLOR** ウィンドウの**前景**の色です。次の色を使用できます。

表 55.6 *ブレーク行に使用できる色*

BLACK	MAGENTA
BLUE	ORANGE
BROWN	PINK
CYAN	RED
GRAY	WHITE
GREEN	YELLOW

デフォルト SASCOLOR ウィンドウの**前景**の色(詳細については、SASCOLOR ウィンドウのオンラインヘルプを参照してください)。

制限事項 このオプションは、対話型レポートウィンドウ環境の出力にのみ影響します。

注 すべての動作環境およびデバイスですべての色がサポートされているわけではありません。一部のオペレーティングシステムおよびデバイスでは、色の間でマップが行われることがあります。たとえば、**DEFINITION** ウィンドウに BROWN という言葉が黄色の文字で表示されている場合、BROWN を選択すると項目が黄色になります。

CONTENTS='*link-text*'
デフォルトで作成された、または STYLE=オプションをサポートする ODS 出力先のオプション設定で作成された目次のエントリのテキストを指定します。PAGE=オプションと *link-text* の CONTENTS=オプションが指定されると、PROC REPORT は *link-text* の値を目次で作成されたテーブルのリンクとして使用します。

デフォルト BREAK AFTER ステートメントで CONTENTS=オプションが指定されず、PAGE オプションが指定されている場合、目次のデフォルトのリンクテキストは“Table N”となります。N は整数です。

制限事項 CONTENTS=が指定され、PAGE オプションが指定されていない場合、PROC REPORT は SAS ログファイルに警告メッセージを生成します。

操作 DEFINE ステートメントにページオプションがあり、BREAK BEFORE ステートメントで PAGE オプションが指定され、指定されている CONTENTS=オプションの値が空の引用符以外である場合、PROC REPORT はディレクトリを目次に追加し、リンクをそのディレクトリのテーブルに追加します。この相互作用

の詳細については、DEFINE ステートメントの (1633 ページ)CONTENTS= オプションを参照してください。

BREAK BEFORE ステートメントが指定され CONTENTS=' 'オプションと PAGE=オプションが指定されている場合、PROC REPORT は目次でディレクトリを作成しません。かわりに、PROC REPORT は DEFINE ステートメントの CONTENTS=値を使用して、目次へのリンクを作成します。DEFINE ステートメントに CONTENTS=オプションがない場合、PROC REPORT は DEFINE ステートメントで記述されているデフォルトのテキストを使用してリンクを作成します。デフォルトのテキスト情報の説明については、DEFINE ステートメント (1633 ページ) CONTENTS=オプションを参照してください。

RTF 出力の場合、ODS RTF ステートメントで CONTENTS=YES オプションをオンにしない限り、CONTENTS=オプションは RTF Body ファイルに影響しません。その場合、**目次**ページは RTF 出力ファイルの前に挿入されます。次に、PROC REPORT の CONTENTS=オプションテキストがこの別の**目次**ページに表示されます。

注 BREAK BEFORE ステートメントが存在し、PAGE オプションが指定され、CONTENTS=オプションが指定されていない場合、デフォルトのリンクテキストは場所変数と場所変数の値です。場所変数は、BREAK 変数と関連付けられています。値は BREAK 変数値です。次のコードに示すように、値は rep で、場所は rep の前です。`break before rep / summarize page;`

ヒント 値が空の引用符である CONTENTS=オプションが指定されている場合、テーブルリンクは目次に作成されません。このコードの例は `CONTENTS=''` です。

複数の BREAK BEFORE ステートメントがある場合、リンクテキストはすべての CONTENTS=値またはすべてのデフォルト値の組み合わせです。

DOL
(二重上線の場合) 13 番目のフォーマット文字を使用して、各値に上線を引きます。

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト 等号(=)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 OL および DOL 両方のオプションを指定すると、PROC REPORT では OL のみ処理されます。

参照項目 FORMCHAR= (1603 ページ)の説明。

DUL
(二重下線の場合) 13 番目のフォーマット文字を使用して各値に下線を引きます。

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト 等号(=)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 UL オプションと DUL オプションを指定すると、PROC REPORT は UL だけを有効化します。

参照項目 FORMCHAR= (1603 ページ)の説明。

OL
(上線の場合) 2 番目のフォーマット文字を使用して各値に上線を引きます

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト ハイフン(-)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 OL および DOL 両方のオプションを指定すると、PROC REPORT では OL のみ処理されます。

参照項目 FORMCHAR= (1603 ページ)の説明。

PAGE
LISTING 出力で、新しいページを開始します。STYLE=オプションをサポートする ODS 出力先では、PAGE オプションは新しいテーブルを開始します。OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

制限事項 OUTPUT 出力先では、このオプションは影響しません。

操作 BREAK ステートメントで PAGE を使用し、レポートの最後にブレークを作成する場合、レポート全体の要約が別のページに表示されます。

例 “例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)

SKIP
最終ブレーク行に対しブランク行を書き込みます。

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

STYLE<*location(s)*>=<*style-override*>
BREAK ステートメントによって作成されるデフォルトの要約行に使用するスタイルの上書きを指定します。

スタイルの無効化は次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<style-attribute-name-2=style-attribute-value-2 …>]

制限事項 OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目 “テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ)

SUMMARIZE
ブレーク行の各グループの要約行を書き込みます。一連のオブザベーションに対する要約行には、次の値が含まれます。

- ブレーク変数(SUPPRESS オプションで非表示可能)
- ブレーク変数の左側のその他のグループ変数または順序変数
- 統計量
- 分析変数
- 計算変数

次の表に、BREAK ステートメントによって作成される要約行の各種レポート項目の値の PROC REPORT による計算方法を示します。

レポート項目	値
ブレーク変数	変数の現在の値(SUPPRESS を使用している場合は欠損値)
ブレーク変数の左側のグループ変数または順序変数	変数の現在値
ブレーク変数の右側のグループ変数または順序変数、またはレポートの任意の場所の表示変数	欠損*
統計量	セットのすべてのオブザベーションにわたる統計量の値。
分析変数	項目の定義の使い方のオプションとして指定される統計量の値。PROC REPORT は、セットのすべてのオブザベーションにわたる統計量の値を計算します。デフォルトの使用法は SUM です。
計算変数	対応する計算ブロックのコードに基づく計算結果 (“COMPUTE ステートメント” (1630 ページ)を参照)。

*カスタマイズされた要約行の欠損値を含む変数を参照する場合、PROC REPORT はその変数をブランク(文字変数の場合)またはピリオド(数値変数の場合)として表示します。

注: PROC REPORT は、順序変数または表示変数を含むレポートのグループを作成できません。

例 “例 2: レポートでの行の並べ替え” (1684 ページ)

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)

“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)

SUPPRESS
次の印刷を行いません。

- 要約行のブレーク変数の値
- ブレーク変数を含む列のブレーク行の下線および下線

操作 SUPPRESS を使用する場合、ブレーク変数の値は、ブレークと関連付けられている計算ブロックで値を割り当てない限り、カスタマイズされたブレーク行で使用できません(“COMPUTE ステートメント” (1630 ページ)を参照)。

例 “例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)

UL
(下線の場合) 2 番目のフォーマット文字を使用して各値に下線を引きます

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト ハイフン(-)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 UL オプションと DUL オプションを指定すると、PROC REPORT は UL だけを有効化します。

参照項目 FORMCHAR= (1603 ページ)の説明。

詳細

ブレーク行の順序

デフォルト要約にブレーク行が複数含まれている場合、次の順序でブレーク行が表示されます。

1. 上線または二重上線(OL または DOL)
2. 要約行(SUMMARIZE)
3. 下線または二重下線(UL または DUL)
4. スキップ行(SKIP)
5. 改ページ(PAGE)

注: ブレークに対しカスタマイズされた要約を定義する場合、カスタマイズされたブレーク行は下線または二重下線の後に表示されます。カスタマイズされたブレーク行の詳細については、“COMPUTE ステートメント” (1630 ページ) および “LINE ステートメント” (1647 ページ)を参照してください。

BY ステートメント

BY グループごとに個別のレポートを個別のページに作成します。

制限事項: BY ステートメントを使用する場合、非ウィンドウ環境で PROC REPORT ステートメント (NOWINDOWS または NOWD オプション)を使用する必要があります。

操作: BY 処理を使用するレポートで RBREAK ステートメントを使用する場合、PROC REPORT は BY グループごとにデフォルト要約を作成します。この場合、レポート全体に関する情報を要約できません。

ヒント: BY ステートメントを使用しても FIRST.変数と LAST.変数は計算ブロックで利用可能になりません。

参照項目: “BY” (68 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* …> <NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは指定するすべての変数によって並べ替えるか、適切にインデックスを付ける必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
データセットが BY ステートメントで文字 DESCENDING の直後に続く変数別に降順で並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。たとえば、データは時系列でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時には BY グループ処理に向けて保留にされます。実際、NOTSORTED を指定した場合は、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

CALL DEFINE ステートメント

現在行の特定の列に対して属性値を設定します。CALL DEFINE ステートメントは、その他のユーザーが対話型レポートウィンドウ環境で使用するレポート定義を記述するために頻繁に使用されます。属性 FORMAT、URL、URLBP、URLP だけが非ウィンドウ環境で影響します。実際に、URL、URLBP、URLP は、非ウィンドウ環境でのみ有効です。STYLE=および URL 属性は、ODS を使用して出力を作成する場合にのみ有効です。

制限事項: レポート項目に割り当てられている計算ブロックでのみ有効。

ヒント: OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

例:

```
compute sales;
      if sales.sum>100 and _break_=' ' then
          call define(_col_, "style",
                  "style=[backgroundcolor=yellow
                          fontfamily=helvetica
                          fontweight=bold]");
endcomp;

compute weight;
      if weight > 100.0 then
          call define(_row_, "style/merge", "style={font_weight=bold}");
endcomp;
```

例:

"例 5: 複数のオブザベーションをレポートの 1 行にまとめる" (1692 ページ)
"例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する" (1719 ページ)
"例 15: PROC REPORT CALL DEFINE ステートメントでの STYLE/MERGE の使用" (1728 ページ)
"例 16: PROC REPORT CALL DEFINE ステートメントでの STYLE/REPLACE の使用" (1731 ページ)

構文

CALL DEFINE (*column-id* | _ROW_, < ' *attribute-name*', *value*>

必須引数

column-id
列名または列番号(レポートの左端からの列の位置)を指定します。列 ID には、次のうちいずれかが可能です。

- 列名である文字リテラル(引用符で囲む)
- 列名を解決する文字式
- 列番号である数値リテラル
- 列番号を解決する数値式
- 形式'_C*n*_'の名前。*n* は列番号です。
- 自動変数_COL_.計算ブロックが割り当てられているレポート項目を含む列を識別します

ROW
現在の行全体を示す自動変数です。

attribute-name
定義する属性です。属性名については、表 55.7 (1626 ページ)を参照してください。

注: 属性 BLINK、HIGHLIGHT および RVSVIDEO は、すべてのデバイスで機能しません。

value
属性の値を設定します。各属性の値については、表 55.7 (1626 ページ)を参照してください。

表 55.7 属性の説明

属性	説明	値	影響
BLINK	現在値の点滅を制御します	1 は点滅をオンにします。0 は点滅をオフにします	対話型レポートウィンドウ環境
COLOR	**REPORT** ウィンドウの現在値の色を制御します	'青'、'赤'、'ピンク'、'緑'、'シアン'、'黄色'、'白'、'オレンジ'、'黒'、'マゼンダ'、'灰色'、'茶色'	対話型レポートウィンドウ環境
COMMAND	一連のコマンドが後に続くように指定します	コマンド行にサブミットする SAS コマンドの引用符付きの文字列	対話型レポートウィンドウ環境
FORMAT	列の出力形式を指定します	SAS 出力形式またはユーザー定義の出力形式	対話型のレポートウィンドウ環境および非ウィンドウ環境
HIGHLIGHT	現在値のハイライトを制御します	1 はハイライトをオンにします。0 はハイライトをオフにします	対話型レポートウィンドウ環境
RVSVIDEO	現在値の表示を制御します	1 は反転表示をオンにします。0 は反転表示をオフにします	対話型レポートウィンドウ環境
STYLE	列または行のスタイルの上書きを指定します	ODS スタイル属性	すべての ODS 出力先
STYLE/MERGE	同じ行または列で既存のスタイルを使用して指定されたスタイルをマージします	ODS スタイル属性	すべての ODS 出力先
STYLE/REPLACE	行または列の既存のスタイルを置き換えます	ODS スタイル属性	すべての ODS 出力先
URL	列の各セルのコンテンツを指定した Uniform Resource Locator (URL)へのリンクにします	引用符付きの URL (一重引用符または二重引用符が使用可能)	ODS HTML、HTML5、RTF、PDF、PowerPoint、EPUB 出力先

属性	説明	値	影響
URLBP	列の各セルのコンテンツをリンクにしますそのリンクのリンク先は、次の連結である Uniform Resource Locator (URL)です 1. ODS HTML ステートメントの BASE=オプションで指定される文字列 2. ODS HTML ステートメントの PATH=オプションで指定される文字列 3. URLBP 属性の値 *	引用符付きの URL (一重引用符または二重引用符が使用可能)	ODS HTML および HTML5 出力先
URLP	列の各セルのコンテンツをリンクにしますそのリンクのリンク先は、次の連結である Uniform Resource Locator (URL)です 1. ODS HTML ステートメントの PATH=オプションで指定される文字列 2. URLP 属性の値 *	引用符付きの URL (一重引用符または二重引用符が使用可能)	ODS HTML および HTML5 出力先

* BASE=オプションと PATH=オプションの詳細については、*SAS Output Delivery System: ユーザーガイド*の ODS HTML ステートメントのドキュメントを参照してください。

詳細

CALL DEFINE ステートメントによるスタイル属性の使用

STYLE 属性は、CALL DEFINE ステートメントによって影響を受けるセルで使用するスタイルの上書きを指定します。

STYLE=値は、PROC REPORT のその他のステートメントの STYLE=オプションのように機能します。ただし、ステートメントでオプションとして機能するかわりに、STYLE 属性の値になります。たとえば、次の CALL DEFINE ステートメントは、特定の列の背景色を黄色に、フォントサイズを 7 に設定します。

```
call define(_col_, "style",
            "style=[backgroundcolor=yellow fontsize=7]");
```

スタイル優先の詳細については、“データセルにスタイル属性を適用する際の優先順位” (1666 ページ)を参照してください。

制約:OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

相互作用:PROC REPORT ステートメントで CALLDEF 場所に対しスタイルの上書きを設定し、CALL DEFINE ステートメントでその設定したスタイルの上書きを使用する必要がある場合、次に示すように空の文字列を STYLE 属性の値として使用します。

```
call define (_col_, "STYLE", "" );
```

ヒント:文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照: “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1719 ページ)

STYLE/REPLACE および STYLE/MERGE スタイル属性の使用

STYLE/MERGE 属性および STYLE/REPLACE 属性は、CALL DEFINE ステートメントで指定したスタイルのみに対して機能します。STYLE(COLUMN)=で指定したスタイルをマージしたり、置き換えることはできません。STYLE/REPLACE および STYLE/MERGE は、1 つの CALL DEFINE ステートメントが含まれる複数の COMPUTE ブロックがあり、それらの CALL DEFINE ステートメントが同じセルを参照している場合に使用することをお勧めします。

STYLE=属性と STYLE/REPLACE 属性は ODS に使用されるスタイル要素を指定します。このセルまたは行に対しスタイルがすでに存在している場合、これらの STYLE 属性により CALL DEFINE が STYLE=オプションで指定されるスタイルを置き換えます。例については、“例 16: PROC REPORT CALL DEFINE ステートメントでの STYLE/REPLACE の使用” (1731 ページ)を参照してください。

STYLE/MERGE 属性により、CALL DEFINE は STYLE=値によって指定されるスタイルと、同じセルまたは行にある既存するスタイル属性をマージします。マージ対象となる既存する STYLE=値がない場合、STYLE/MERGE は STYLE 属性または STYLE/REPLACE 属性と同様に機能します。例については、“例 15: PROC REPORT CALL DEFINE ステートメントでの STYLE/MERGE の使用” (1728 ページ)を参照してください。

COLUMN ステートメント

すべての列の調整と複数の列にわたるヘッダーの調整を記述します。

制限事項: PROC REPORT ステートメントで REPORT=を使用する場合、COLUMN ステートメントは使用できません。

例:
“例 1: 変数の選択とレポートの要約行の作成” (1680 ページ)
“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)
“例 6: 変数の値ごとに列を作成する” (1695 ページ)
“例 4: 1 つの変数に複数の統計量を表示する” (1690 ページ)
“例 8: パーセントの計算” (1703 ページ)
“例 9: PROC REPORT での欠損値の処理法” (1706 ページ)

構文

COLUMN *column-specification(s)*;

必須引数

column-specification(s)
次のうちいずれかになります。

- *report-item(s)*
- *report-item-1*, *report-item-2 <...*, *report-item-n>*
- (*'header-1 ' < ...'header-n '> report-item(s)*)

- *report-item=name*

この場合、*report-item* は、データセット変数、計算変数または統計量の名前です。利用可能な統計量のリストについては、“PROC REPORT で使用できる統計量” (1654 ページ) を参照してください。

report-item(s)
レポートの列をそれぞれ形成する項目を識別します。

例 “例 1: 変数の選択とレポートの要約行の作成” (1680 ページ)

“例 9: PROC REPORT での欠損値の処理法” (1706 ページ)

***report-item-1*, *report-item-2* <..., *report-item-n*>**
列のコンテンツを一括して決定するレポート項目を識別します。これらの項目は、レポートで積み重ねられます。各項目がヘッダーを生成し、ヘッダーが相互に積み重ねられるためです。最左の項目のヘッダーが先頭です。項目のうち 1 つが分析変数、計算変数、グループ変数、または統計量である場合、その値はレポートのその部分のセルに入力されます。その他の場合、PROC REPORT はセルに度数カウントを入力します。

統計量と分析変数を積み重ねる場合、列ステートメントで指定する統計量が分析変数の定義の統計量よりも優先します。たとえば、次の PROC REPORT ステップでは、各セクタに対する売上の最小値を含むレポートを作成します。

```
proc report data=grocery;
column sector sales,min;
define sector/group;
define sales/analysis sum;
run;
```

列変数の下に表示変数を積み重ねる場合、その表示変数のすべての値がレポートに表示されます。

操作 積み重ねられた一連のレポート項目には、分析変数または統計量のいずれかを 1 つだけ含めることができます。複数の分析変数または統計量を含めると、PROC REPORT はエラーを返します。レポートのセルに挿入する値を決定できないためです。

ヒント かっこを使用して、相互に積み重ねるのではなく、ヘッダーを同じレベルで表示する必要があるレポート項目をグループ化することができます。

例 “例 6: 変数の値ごとに列を作成する” (1695 ページ)

“例 4: 1 つの変数に複数の統計量を表示する” (1690 ページ)

“例 8: パーセントの計算” (1703 ページ)

('*header-1* ' <... '*header-n* '> *report-item(s)*)
複数の列にわたる 1 つ以上のヘッダーを作成します。

header
レポートの 1 つ以上の列にわたる文字の文字列です。PROC REPORT は、各ヘッダーを個別の行に印刷します。ヘッダーで区切り文字を使用して、1 つのヘッダーを複数の行に分割することができます。SPLIT= (1612 ページ) の説明を参照してください。

LISTING 出力では、ヘッダーの最初と最後の文字が次のうちいずれかである場合、PROC REPORT はその文字を使用して、ヘッダーを拡張し、列にわたるスペースを埋めます。<>と><はペアである必要があります。- = . _ * + <> ><

同様に、ヘッダーの最初の文字が<で、最後の文字が>の場合、またはその逆の場合、PROC REPORT はヘッダーのテキストの前で最初の文字を、その後に最後の文字を繰り返すことによってヘッダーを拡張し、列にわたるスペースを埋めます。

注: 拡張文字の使用は、LISTING 出力先でのみサポートされています。したがって、PROC REPORT は、出力が他の出力先に送信されると、拡張文字を単純に削除します。詳細については、“Understanding ODS Destinations” (*SAS Output Delivery System: User's Guide*)を参照してください。

report-item(s)
またがる列を指定します。

例 “例 8: パーセントの計算” (1703 ページ)

report-item=name
レポート項目の別名を指定します。COLUMN ステートメントで同じレポート項目を複数回使用できます。ただし、指定名に使用できるのは、1 つの DEFINE ステートメントだけです。(DEFINE ステートメントは、出力形式、カスタマイズされた列ヘッダーなどの特性を指定します。項目に対して DEFINE ステートメントを省略すると、REPORT プロシジャはデフォルト値を使用します)。COLUMN ステートメントで別名を割り当てても、それだけでレポートは変わりません。ただし、変数または統計量が発生するたびに個別の DEFINE ステートメントを使用できるようになります。

注 常に別名を使用できるわけではありません。計算ブロックの別名を持つレポート項目を参照する場合、その別名を使用する必要があります。ただし、レポート項目が列を列変数と共有する場合、列を列番号別に参照する必要があります(“計算ブロックのレポート項目を参照するための 4 つの方法” (1591 ページ)を参照)。

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)

COMPUTE ステートメント

レポートの作成時に PROC REPORT が実行する 1 つ以上のプログラミングステートメントを含む計算ブロックを開始します。

制限事項: レポートを複数の ODS 出力先または後から再表示される ODS ドキュメントに送信している場合は、COMPUTE ブロックで非決定性関数(LAG、DIF、RANUNI、DATETIME など)を使用しないでください。レポートでこのような関数により作成されたデータを使用する必要がある場合は、DATA ステップで関数を呼び出し、その結果をデータセットに保存してから PROC REPORT を実行します。

操作: ENDCOMP ステートメントは計算ブロックのステートメントグループの終わりをマーク付けする必要があります。

注: 計算ブロックはレポート項目または場所と関連付けることができます(レポートの上下、ページの上下、オブザベーションセットの前後)。計算ブロックは、**COMPUTE** ウィンドウまたは COMPUTE ステートメントを使用して作成します。COMPUTE ステートメントの 1 つのフォームが計算ブロックをレポート項目と関連付けます。別のフォームが計算ブロックを場所と関連付けます。計算ブロックで使用できる SAS 言語要素のリストについては、“計算ブロックのコンテンツ” (1591 ページ)を参照してください。

例: “例 2: レポートでの行の並べ替え” (1684 ページ)
“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)

構文

COMPUTE *location* <*target*>
</ STYLE=<*style-override(s)* > >;
LINE *specification(s)*;
... select SAS language elements ...
ENDCOMP;
COMPUTE *report-item* </ *type-specification*>;
CALL DEFINE (*column-id*, '*attribute-name*', *value*);
... select SAS language elements ...
ENDCOMP;

必須引数

COMPUTE ステートメントで場所またはレポート項目のいずれかを指定する必要があります。

location
計算ブロックが *target* に関して実行する場所を決定します。

AFTER
次の場所のうちいずれかのブレークで計算ブロックを実行します。

- *target* として指定する変数に対し同じ値を持つ一連の行の最終行の直後、またはその変数にデフォルト要約がある場合は予備要約行の作成直後 (“PROC REPORT でのレポート作成法 ” (1670 ページ)を参照)。
- LISTING 出力では、_PAGE_を *target* として指定する場合、各ページの下近く、フットノートの直前。
- target を省略する場合はレポートの最後。

BEFORE
次の場所のうちいずれかのブレークで計算ブロックを実行します。

- *target* として指定する変数に対し同じ値を持つ行セットの最初の行の直前、またはその変数にデフォルト要約がある場合は予備要約行の作成直後 (“PROC REPORT でのレポート作成法 ” (1670 ページ)を参照)。
- LISTING 出力では、_PAGE_を *target* として指定する場合、各ページの上近く、タイトルと列ヘッダーの間。
- target を省略する場合、最初の詳細行の直前。

注 レポートに印刷ページに収まる以上の列が含まれている場合、PROC REPORT は残りの列を含めるための追加ページを生成します。この場合、_PAGE_を *target* として指定するとき、COMPUTE ブロックはこれらの各追加ページに対し再実行されません。COMPUTE ブロックが再実行されるのは、すべての行が印刷された後だけです。

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)

“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)

report-item
計算ブロックと関連付けるデータセット変数、計算変数または統計量を指定します。非ウィンドウ環境で作業をしている場合、レポート項目を COLUMN ステートメントに含める必要があります。項目が計算変数である場合、それに対し DEFINE ステートメントを含める必要があります。

注 計算変数の位置は重要です。PROC REPORT は値をレポート行の列に、左から右に割り当てます。その結果、レポートの右側に表示される変数に基づいて計算変数の計算を行うことはできません。

例 “例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)

“例 6: 変数の値ごとに列を作成する” (1695 ページ)

オプション引数

STYLE<(*location(s)*)>=<*style-override(s)*>
この計算ブロックで LINE ステートメントによって作成されるテキストに使用するスタイルを指定します。

スタイルの無効化は次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2* …>]

制限事項 OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目 “テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ)

例 “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1719 ページ)

target
いつ計算ブロックが実行するかを制御します。COMPUTE ステートメントに対し場所(BEFORE または AFTER)を指定する場合、*target* を指定することもできます。値は、次のうちいずれかになります。

break-variable
グループ変数または順序変数です。

ブレーク変数を指定する場合、PROC REPORT はブレーク変数の値が変わるたびに計算ブロックのステートメントを実行します。

PAGE </ justification>
LISTING 出力先で、タイトルの印刷直後またはフットノートの印刷直前に計算ブロックがページごとに一度実行されます。*justification* は、テキストと値の配置を制御します。次のうちいずれかになります。

CENTER
計算ブロックが書き込む各行を中央揃えにします。

LEFT
計算ブロックが書き込む各行を左寄せにします。

RIGHT
計算ブロックが書き込む各行を右寄せにします。

デフォルト CENTER

例 “例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)

type-specification
種類を指定します。(オプション)。*report-item* の長さも指定します。計算ブロックと関連付けられているレポート項目が計算変数である場合、種類の指定を使用してそれが文字変数になるように指定しない限り、PROC REPORT は数値変数であるとみなします。種類の指定には、次の形式があります。

CHARACTER <LENGTH=*length*>

この場合、

CHARACTER
計算変数が文字変数になるように指定します。長さを指定しない場合、変数の長さは 8 です。

別名 CHAR

例 “例 8: パーセントの計算” (1703 ページ)

LENGTH=*length*
計算文字変数の長さ(文字数)を指定します。

デフォルト 8

範囲 1 から 200

操作 長さを指定する場合、CHARACTER を使用して計算変数が文字変数であることを示す必要があります。

例 “例 8: パーセントの計算” (1703 ページ)

DEFINE ステートメント

レポート項目の使用方法と表示方法を説明します。

制限事項: 重みは、レポート項目にも適用しないと、*report-item* 別名に適用できません。WEIGHT= オプションは、DEFINE ステートメントの *report-item* に表示される必要があります。

ヒント: DEFINE ステートメントを使用しない場合、PROC REPORT はデフォルトの特性を使用します。

例: “例 2: レポートでの行の並べ替え” (1684 ページ)

“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)
“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)
“例 6: 変数の値ごとに列を作成する” (1695 ページ)
“例 4: 1 つの変数に複数の統計量を表示する” (1690 ページ)
“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)
“例 8: パーセントの計算” (1703 ページ)
“例 10: 出力データセットの作成と計算変数の保存” (1710 ページ)
“例 11: 出力形式を使用し、グループを作成する” (1713 ページ)
“例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1719 ページ)
“例 12: マルチラベル出力形式の使用” (1716 ページ)
“例 14: CELLWIDTH=スタイル属性を PROC REPORT とともに使用する” (1726 ページ)

構文

DEFINE *report-item* / <*option(s)*>;

オプション引数の要約

Control the placement of values and column headings

CENTER
列幅内のレポート項目のフォーマットされた値を中央揃えにし、値にわたる列ヘッダーを中央揃えにします。

COLOR=*color*
定義する列ヘッダーおよび項目の値の **REPORT** ウィンドウの色を指定します。

column-header
レポート項目に対し列ヘッダーを定義します。

LEFT
列幅内のレポート項目のフォーマットされた値を左寄せにし、値にわたる列ヘッダーを左寄せにします。

RIGHT
列幅内のレポート項目のフォーマットされた値を右寄せにし、値にわたる列ヘッダーを右寄せにします。

Customize the appearance of a report item

EXCLUSIVE
ユーザー定義の出力形式のすでに読み込まれた範囲に見つからない項目のすべての組み合わせを除外します。

FORMAT=*出力形式*
SAS またはユーザー定義の出力形式を項目に割り当てます。

MISSING
欠損値を項目の有効値とみなします。

MLF
PROC REPORT が出力形式ラベルを使用して、マルチラベル出力形式を含むサブグループの組み合わせを作成できるようにします。

ORDER=DATA|FORMATTED|FREQ|INTERNAL
グループ変数、順序変数、列変数の値を指定順序に従って並べ替えます。

PRELOADFMT

項目に対してすべての出力形式がすでに読み込まれているように指定します。

SPACING=*horizontal-positions*
LISTING 出力の場合、定義中の列とそのすぐ左の列の間を空けるブランク文字の数を定義します。

statistic
統計量と分析変数を関連付けます。

STYLE<(*location(s)*)>=<*style-overrides(s)*>
レポート項目に対してスタイル要素(Output Delivery System の)を指定します。

WEIGHT=*weight-variable*
分析変数の値に重みをつける数値変数を指定します。

WIDTH=*column-width*
PROC REPORT がレポート項目を表示する列の幅を定義します。

Specify how to use a report item

ACROSS
データセット変数である必要がある項目を列変数として定義します。

ANALYSIS
データセット変数である必要がある項目を分析変数として定義します。

COMPUTED
項目を計算変数として定義します。

DISPLAY
データセット変数である必要がある項目を表示変数として定義します。

GROUP
データセット変数である必要がある項目をグループ変数として定義します。

ORDER
データセット変数である必要がある項目を順序変数として定義します。

Specify options for a report item

CONTENTS='*link-text*'
目次でリンクを作成します。

DESCENDING
PROC REPORT がグループ変数、順序変数、列変数の行または値を表示する順序を逆順にします。

FLOW
その列の文字変数の値をラップします。

ID
定義している項目が ID 変数になるように指定します。

NOPRINT
レポート項目の表示を非表示にします。

NOZERO
レポート項目の表示を、その値がすべてゼロまたは欠損値の場合に非表示にします。

PAGE
レポート項目の値を含む最初の列を印刷する直前に改ページを挿入します。

必須引数

report-item
定義対象となるデータセット変数、計算変数または統計量の名前または別名(COLUMN ステートメントで作成)を指定して定義します。*report-item* に使用できる名前の種類を次に示します。

- SAS ID (VALIDVARNAME オプションで決定されます)
- 名前リテラル
- 番号範囲リスト
- 名前範囲リスト
- 特殊名リスト
- 名前接頭辞リスト
- 統計量

注 変数範囲リストの名前は、統計量名、または計算変数名ではなく、入力データセットの変数を表します。DEFINE ステートメントごとに 1 つの名前のみ使用します。ただし、1 つの名前で 1 つの範囲リストを表すことができます。変数範囲リストを使用した構文例は次のとおりです。`DEFINE Var1-Var3/ width=10 center "#Visit#Date";`

統計量の定義で使い方のオプションを指定しないでください。PROC REPORT は、統計量の名前でその使用方法を認識します。

参照項目 "Names in the SAS Language" (*SAS Language Reference: Concepts*)、"SAS Variable Lists" (*SAS Language Reference: Concepts*)、および "VALIDVARNAME= System Option" (*SAS System Options: Reference*).

オプション引数

ACROSS
データセット変数である必要がある *report-item* を列変数として定義します("列変数" (1588 ページ)を参照)。

例 "例 6: 変数の値ごとに列を作成する" (1695 ページ)

ANALYSIS
データセット変数である必要がある *report-item* を順序変数として定義します("分析変数" (1587 ページ)を参照)。

デフォルトで、PROC REPORT は分析変数に対する Sum 統計量を計算します。DEFINE ステートメントの *statistic* オプションで代替統計量を指定します。

注: DEFINE ステートメントでの統計量の指定は、ANALYSIS オプションを意味するため、ANALYSIS を指定する必要はありません。ただし、ANALYSIS を指定すると、コードがわかりやすくなります。

注: 特殊欠損値は、ANALYSIS 変数として定義されている場合、欠損値として表示されます。

例 "例 2: レポートでの行の並べ替え" (1684 ページ)

"例 3: 別名を使用し、同一変数に複数の統計量を求める" (1687 ページ)

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)

CENTER
列幅内のレポート項目のフォーマットされた値を中央揃えにし、値にわたる列ヘッダーを中央揃えにします。このオプションは、ページのレポートを中央揃えにする、PROC REPORT ステートメントの CENTER オプションに影響しません。

制限事項 このオプションは、LISTING 出力のヘッダーおよびデータに影響します。ODS 出力では、このオプションはデータのみに影響します。

COLOR=*color*
列ヘッダー、および定義している項目の値の **REPORT** ウィンドウの色を指定します。次の色を使用できます。

BLACK	MAGENTA
BLUE	ORANGE
BROWN	PINK
CYAN	RED
GRAY	WHITE
GREEN	YELLOW

注: すべての動作環境およびデバイスですべての色がサポートされているわけではありません。一部の動作環境およびデバイスでは、色の間でマップが行われることがあります。一部の動作環境およびデバイスでは、色の間でマップが行われることがあります。たとえば、**DEFINITION** ウィンドウに BROWN という言葉が黄色の文字で表示されている場合、BROWN を選択すると項目が黄色になります。

デフォルト **SASCOLOR** ウィンドウの**前景**の色(詳細については、**SASCOLOR** ウィンドウのオンラインヘルプを参照してください)。

制限事項 このオプションは、対話型レポートウィンドウ環境の出力にのみ影響します。

column-header
レポート項目に対し列ヘッダーを定義します。各ヘッダーを一重引用符または二重引用符で囲みます。複数の列ヘッダーを指定する場合、PROC REPORT は列ヘッダーごとに別の行を使用します。区切り文字も複数の行にわたる列ヘッダーを分割します。

LISTING 出力では、ヘッダーの最初と最後の文字が次のうちいずれかである場合、PROC REPORT はその文字を使用して、ヘッダーを拡張し、列にわたるスペースを埋めます。`:- = \_ .*+`

同様に、ヘッダーの最初の文字が<で、最後の文字が>の場合、またはその逆の場合、PROC REPORT はヘッダーのテキストの前で最初の文字を、その後に最後の文字を繰り返すことによってヘッダーを拡張し、列にわたるスペースを埋めます。

次の表に、デフォルトの変数と統計量を示します。

項目	ヘッダー
変数(ラベルなし)	変数名
変数(ラベルあり)	変数ラベル
統計量	統計量名

ヒント ラベルが存在するときに名前を使用する場合、PROC REPORT を起動する前に次の SAS ステートメントをサブミットします。`options nolabel;`

HEADLINE はすべての列ヘッダーとその間のスペースに下線を引きます。LISTING 出力では、HEADLINE を使用するかわりに各列ヘッダーの最終行として特殊文字 (`'---'`) を使用することにより、列ヘッダーに下線を引き、その間のスペースに下線を引かずにおくことができます(“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)を参照)。

参照項目 SPLIT= (1612 ページ)

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)

“例 6: 変数の値ごとに列を作成する” (1695 ページ)

COMPUTED

指定した項目を計算変数として定義します。計算変数は、レポートに対して定義する変数です。入力データセットにはなく、PROC REPORT によって入力データセットに追加されません。

対話型レポートウィンドウ環境では、計算変数を **COMPUTED VAR** ウィンドウからレポートに追加します。

非ウィンドウ環境では、計算変数を次を行うことによって追加します。

- 計算変数を COLUMN ステートメントに含める
- 変数の使い方を DEFINE ステートメントで COMPUTED として定義する
- 変数と関連付けられている計算ブロックの変数値を計算する

例 “例 6: 変数の値ごとに列を作成する” (1695 ページ)

“例 8: パーセントの計算” (1703 ページ)

CONTENTS='*link-text*'

デフォルトで作成された、または STYLE=オプションをサポートする ODS 出力先のオプション設定で作成された目次のエントリのテキストを指定します。DEFINE ステートメントで PAGE=オプションと CONTENTS=オプションが指定され、*link-text* 値が割り当てられている場合、PROC REPORT はディレクトリを目次に追加し、*link-text* の値を目次で作成されたテーブルのリンクとして使用します

デフ DEFINE ステートメントで PAGE オプションが指定され、CONTENTS=オプションが指定されていない場合、ディレクトリはディレクトリテキストとともに

デフォルト
COLA-COLB として作成されます。COLA は最左列の名前または別名で、COLB は最右列の名前または別名です。テーブルに 1 列のみ含まれている場合、ディレクトリテキストはその列の名前または別名です。

制限事項
CONTENTS=が指定され、PAGE オプションが指定されていない場合、PROC REPORT は SAS ログファイルに警告メッセージを生成します。

操作
DEFINE ステートメントにページオプションがあり、BREAK BEFORE ステートメントで PAGE オプションが指定され、指定されている CONTENTS=オプションの値が空の引用符以外である場合、PROC REPORT はディレクトリを目次に追加し、リンクをそのディレクトリのテーブルに追加します。

DEFINE ステートメントで PAGE=オプションが指定され、BREAK BEFORE ステートメントで PAGE=オプションが指定されていない場合、PROC REPORT は目次でディレクトリを作成しません。かわりに、PROC REPORT は DEFINE ステートメントの CONTENTS=値を使用して、目次へのリンクを作成します。DEFINE ステートメントに CONTENTS=オプションがない場合、PROC REPORT はデフォルトのテキスト COLA—COLB を使用してリンクを作成します。上記のデフォルトの説明を参照してください。

BREAK BEFORE ステートメントが指定され CONTENTS=' 'オプションと PAGE=オプションが指定されている場合、PROC REPORT は目次でディレクトリを作成しません。かわりに、PROC REPORT は DEFINE ステートメントの CONTENTS=値を使用して、目次へのリンクを作成します。DEFINE ステートメントに CONTENTS=オプションがない場合、PROC REPORT はデフォルトのテキスト COLA—COLB を使用してリンクを作成します。上記のデフォルトの説明を参照してください。

ヒント
DEFINE ステートメントで値が空の引用符である CONTENTS=オプションが指定されている場合、目次へのディレクトリは追加されません。このコードの例は、次のとおりです。`CONTENTS=''`

複数の BREAK BEFORE ステートメントがある場合、リンクテキストはすべての CONTENTS=値またはすべてのデフォルト値の組み合わせです。

OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

DESCENDING
PROC REPORT がグループ変数、順序変数、列変数の行または値を表示する順序を逆順にします。

ヒント
デフォルトで、PROC REPORT はグループ変数、順序変数、列変数をフォーマットされた値によって並び替えます。DEFINE ステートメントの ORDER=オプションを使用して、代替並べ替え順序を指定します。

DISPLAY
データセット変数である必要がある *report-item* を表示変数として定義します(“表示変数” (1586 ページ)を参照)。

EXCLUSIVE
ユーザー定義の出力形式のすでに読み込まれた範囲に見つからないグループ変数と列変数のすべての組み合わせをレポートと出力データセットから除外します。

要件 変数の出力形式を読み込むために、DEFINE ステートメントで PRELOADFMT オプションを指定する必要があります。

FLOW
その列の文字変数の値をラップします。FLOW オプションは区切り文字を有効化します。テキストに区切り文字が含まれていない場合、PROC REPORT はテキストをブランクで分割しようとします。

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

FORMAT=*出力形式*
SAS またはユーザー定義の出力形式を項目に割り当てます。この出力形式は、PROC REPORT で表示されるときに *report-item* に適用されます。出力形式は、データセット内の変数と関連付けられている出力形式を変えません。データセット変数の場合、PROC REPORT は検出されるこれらの出力形式のうち最初のものを有効化します。

- DEFINE ステートメントの FORMAT=と関連付けられている出力形式
- PROC REPORT の起動時に FORMAT ステートメントで割り当てられる出力形式
- データセット内の変数と関連付けられる出力形式

これらフォーマットが一切存在しない場合は、PROC REPORT により、数値変数には BEST*w.*が使用され、文字変数には$*w.*が使用されます。*w* の値は、デフォルトの列幅です。入力データセット内の文字変数の場合、デフォルトの列幅は変数の長さです。入力データセットの数値変数の場合と、計算変数(数値および文字の両方) の場合、デフォルトの列幅は PROC REPORT ステートメントの COLWIDTH= によって、または **ROPTIONS** ウィンドウで指定される値です。

対話型レポートウィンドウ環境で、使用する出力形式が不明な場合、**DEFINITION** ウィンドウの出力形式フィールドに疑問符(?)を入力して、**FORMATS** ウィンドウにアクセスします。

別名 F=

例 “例 2: レポートでの行の並べ替え” (1684 ページ)

“例 4: 1 つの変数に複数の統計量を表示する” (1690 ページ)

GROUP
データセット変数である必要がある *report-item* をグループ変数として定義します (“グループ変数” (1587 ページ)を参照)。

例 “例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)

“例 4: 1 つの変数に複数の統計量を表示する” (1690 ページ)

“例 11: 出力形式を使用し、グループを作成する” (1713 ページ)

ID
定義している項目が ID 変数になるように指定します。ID 変数とその左側のすべての列がレポートの各ページの左側に表示されます。ID によって、1 ページに収まる以上の列がレポートに含まれているときにレポートの各行を識別できるようになります。

LEFT
列幅内のレポート項目のフォーマットされた値を左寄せにし、値にわたる列ヘッダーを左寄せにします。出力形式の幅が列幅と同じ場合、LEFT オプションは値の配置に影響しません。

制限事項 このオプションは、LISTING 出力のヘッダーおよびデータに影響します。ODS 出力では、このオプションはデータのみに影響します。

MISSING
欠損値をレポート項目の有効値とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING オプションを省略すると、PROC REPORT は、グループ変数、順序変数または列変数が欠損値のすべてのオブザベーションをレポートおよび出力データセットから除外します。

MLF
PROC REPORT が指定の範囲や重複する範囲に出力形式ラベルを使用して、マルチラベル出力形式設定を使用するサブグループの組み合わせを作成できるようにします。これらのマルチラベル出力形式は、グループ変数と列変数とのみ使用されます。

MLF は、すべての ODS 出力先、LISTING 出力先、データセット、REPORT WINDOW でサポートされています。

注: PROC REPORT では、マルチラベル出力形式を持つ数値変数の文字変数への割り当てをサポートしています。

要件 VALUE ステートメントの PROC FORMAT と MULTILABEL オプションを使用して、マルチラベル出力形式を作成します。

ヒント 変数がマルチラベル出力形式と関連付けられていない限り、MLF オプションは影響しません。列と関連付けられている MULTILABEL 出力形式がない場合、既存する出力形式または入力形式を 1 つ以上の変数と関連付けるために追加の FORMAT ステートメントまたは DEFINE ステートメントの FORMAT=オプションが必要になります。MULTILABEL 出力形式がすでに列と関連付けられている場合(出力形式と変数を関連付けるための通常の方法を使用して)、追加の FORMAT ステートメントまたは FORMAT=オプションは不要です。

MLF オプションが省略されると、PROC REPORT は第 1 出力形式ラベルを使用して、サブグループの組み合わせを決定します。第 1 出力形式ラベルは、最初の外部出力形式値に対応しています。

参照項目 MULTILABEL オプション (870 ページ)(FORMAT プロシジャの VALUE ステートメント)。

例 “例 12: マルチラベル出力形式の使用” (1716 ページ)

NOPRINT
レポート項目の表示を非表示にします。このオプションは、次の場合に使用します。

- レポートに項目を表示せずに、その値を使用してレポートで使用するその他の値を計算する場合。

- レポートで行の順序を作成する場合。
- 項目を列として使用せずに、要約のその値にアクセスする場合(“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)を参照)。

操作 NOPRINT を使用して定義する列がレポートに表示されない場合でも、列を番号別に参照する場合はそれらの列をカウントする必要があります(“計算ブロックのレポート項目を参照するための 4 つの方法” (1591 ページ)を参照)。

PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウの SHOWALL は、NOPRINT のすべての発生を無効にします。

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)

“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)

NOZERO
レポート項目の表示を、その値がすべてゼロまたは欠損値の場合に非表示にします。

操作 NOZERO を使用して定義する列がレポートに表示されない場合でも、列を番号別に参照する場合はそれらの列をカウントする必要があります。(“計算ブロックのレポート項目を参照するための 4 つの方法” (1591 ページ)を参照)。

PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウの SHOWALL は、NOZERO の使用をすべて無効にします。

ORDER
データセット変数である必要がある *report-item* を順序変数として定義します(“順序変数” (1586 ページ)を参照)。

例 “例 2: レポートでの行の並べ替え” (1684 ページ)

ORDER=DATA|FORMATTED|FREQ|INTERNAL
グループ変数、順序変数、列変数の値を指定順序に従って並べ替えます。

DATA
入力データセットの順序に従って値を並べ替えます。

注 DBMS テーブルの入力データに対して ORDER=DATA オプションを指定する場合、PROC REPORT からデータベーステーブルに書き込まれる行の順序は保持されない可能性があります。

FORMATTED
フォーマットされた(外部)値によって値を並べ替えます。出力形式がクラス変数に割り当てられていない場合、デフォルトの出力形式 BEST12 が使用されます。

FREQ
昇順の度数カウントによって値を並べ替えます。

INTERNAL
フォーマットされていない値によって値を並べ替えます。これにより PROC SORT が作成するのと同じ順序を作成します。この順序は、動作環境によって異なります。この並べ替え順序は、日付を年代順に表示する場合に特に便利です。

デフォルト FORMATTED

操作 項目の定義の DESCENDING は、項目の並べ替え順序を逆順にします。デフォルトでは、順序は昇順です。

注 PROC REPORT の ORDER=オプションのデフォルト値は、その他の SAS プロシジャのデフォルト値とは異なります。その他の SAS プロシジャでは、デフォルトは ORDER=INTERNAL です。PROC REPORT のオプションのデフォルトは、その他のプロシジャと整合させるために、今後のリリースで変わることがあります。このため、フォーマットされた値によってレポート項目を並べ替えることが重要な本稼働ジョブでは、現在デフォルトである ORDER=FORMATTED を指定します。これにより、PROC REPORT はデフォルトが変わっても期待どおりのレポートの作成を続行します。

例 “例 2: レポートでの行の並べ替え” (1684 ページ)

PAGE

レポート項目の値を含む最初の列を印刷する直前に改ページを挿入します。

制限事項 このオプションは、OUTPUT 出力先では影響しません。

操作 WRAP を PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウで使用する場合、PAGE は無視されます。

ヒント リスト出力先では、DEFINE ステートメントの PAGE オプションにより PROC REPORT がこの列とすべての列を新しいページの右側に印刷します。ただし、非リスト出力先の場合は、レポート内のすべての行が印刷されるまでページブレークは発生しません。そのため、PROC REPORT はすべての列に対するすべての行を PAGE 列の左側に印刷してから、レポートの上部に戻って、PAGE 列とその列を右側に印刷します。

PRELOADFMT

変数に対して出力形式がすでに読み込まれているように指定します。

制限事項 PRELOADFMT は、グループ変数と列変数にのみ適用されます。

要件 PRELOADFMT は、EXCLUSIVE または ORDER=DATA を指定し、出力形式を変数に割り当てない限り、影響しません。

操作 レポートを入力データセット内にあるフォーマットされた変数値の組み合わせに制限するには、DEFINE ステートメントの EXCLUSIVE を使用します。

ユーザー定義の出力形式のすべての範囲と値を出力に含めるには、PROC REPORT ステートメントの COMPLETEROWS オプションを使用します。

注 列変数の定義時に NOCOMPLETECOLS を指定しない場合、レポートには、フォーマットされている値ごとに 1 列が含まれます。グループ変数の定義時に COMPLETEROWS を指定する場合、レポートには、フォーマットされた値ごとに 1 行が含まれます。レポートに列変数のフォーマットされた値ごとに 1 列、グループ変数のフォーマットされた値ごとに 1 行含まれている場合、行と列の組み合わせが意味をなさない場合があります。

例 “例 12: マルチラベル出力形式の使用” (1716 ページ)

RIGHT
列幅内の指定項目のフォーマットされた値を右寄せにし、値にわたる列ヘッダーを右寄せにします。出力形式の幅が列幅と同じ場合、RIGHT は値の配置に影響しません。

制限事項 このオプションは、LISTING 出力のヘッダーおよびデータに影響します。ODS 出力では、このオプションはデータのみに影響します。

SPACING=*horizontal-positions*
定義中の列とそのすぐ左の列の間を空けるブランク文字の数を定義します。各列について、その幅、その間のブランク文字、その左側の列の合計がページサイズを超えることはできません。

デフォルト 2

制限事項 このオプションは、LISTING 出力先以外の ODS 出力先に影響しません。

操作 PROC REPORT の CENTER オプションが有効な場合、PROC REPORT はレポートの最左の変数の前のスペースを無視します。

項目の定義の SPACING=は、PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウの SPACING=の値を無効にします。

statistic
統計量と分析変数を関連付けます。統計量とその定義のすべての分析変数を関連付ける必要があります。PROC REPORT は、レポートの各セルによって表されるオブザベーションに対する分析変数の値を計算するために指定する統計量を使用します。その他の種類の変数の定義で *statistic* を使用できません。

利用可能な統計量のリストについては、“PROC REPORT で使用できる統計量” (1654 ページ) を参照してください。

デフォルト SUM

注 PROC REPORT は、分析変数の名前を列のデフォルトヘッダーとして使用します。DEFINE ステートメントの *column-header* オプションを使用して、列ヘッダーをカスタマイズできます。

例 “例 2: レポートでの行の並べ替え” (1684 ページ)

“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)

STYLE<(*location(s)*)>=<*style-overrides(s)*>
このレポート項目に対する列ヘッダーおよびセル内のテキストに使用するスタイル要素を指定します。

スタイルの無効化は次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。

- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2 …*>]

制限事項 OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目 “テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ)

例 “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1719 ページ)

WEIGHT=*weight-variable*

DEFINE ステートメントで指定される分析変数の値に重みをつける数値変数を指定します。変数値は整数である必要はありません。次の表に、WEIGHT 変数のさまざまな値を PROC REPORT がどのように扱うかを示します。

重み値	PROC REPORT 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 未満	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負とゼロの重みを含むオブザベーションを分析から除外するには、PROC REPORT ステートメントの EXCLNPWGT オプションを使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

別名 WGT=

制限事項 重み付き分位点を比較するには、PROC REPORT ステートメントの QMETHOD=OS を使用します。

重みは、*report-item* にも適用しないと、*report-item* 別名に適用できません。WEIGHT=オプションは、DEFINE ステートメントの *report-item* に表示される必要があります。

ヒント WEIGHT=オプションを使用する場合、PROC REPORT ステートメントの VARDEF=オプションのどの値が適切かを考慮します。

変数に対し異なる重みを指定するため、別の変数定義の WEIGHT=オプションを使用します。

WIDTH=*column-width*
PROC REPORT が *report-item* を表示する列の幅を定義します。このオプションは、LISTING 出力にのみ影響します。

出力形式の詳細については、“FORMAT=*出力形式*” (1640 ページ)の説明を参照してください。

デフォルト 出力形式の処理に十分な大きさの列幅。出力形式がない場合、PROC REPORT は PROC REPORT ステートメントの COLWIDTH=オプションの値を使用します。

範囲 1 から SAS システムオプション LINESIZE=の値

制限事項 このオプションは、LISTING 出力以外の ODS 出力先に影響しません。ODS 出力先の場合、STYLE=オプションを WIDTH=スタイル属性または CELLWIDTH=スタイル属性と使用します。詳細については、“Style Attributes Tables ” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。スタイル属性 WIDTH=と CELLWIDTH=を PROC REPORT で使用する方法については、“例 14: CELLWIDTH=スタイル属性を PROC REPORT とともに使用する” (1726 ページ)を参照してください。

操作 項目定義の WIDTH=は、PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウの COLWIDTH=の値を無効にします。

ヒント 項目をレポートの同じ列に積み重ねる場合、一番下に積み重ねられる項目の幅によって列の幅が決定されます。

ENDCOMP ステートメント

PROC REPORT がレポートの作成時に実行する 1 つ以上のプログラミングステートメントの最後をマーク付けします。

制限事項: COMPUTE ステートメントは ENDCOMP ステートメントの前に置く必要があります。

参照項目: COMPUTE ステートメント

例: “例 2: レポートでの行の並べ替え” (1684 ページ)

構文

ENDCOMP;

FREQ ステートメント

オブザベーションを、入力データセットに複数回表示されたように処理します。

ヒント: FREQ ステートメントと WEIGHT ステートメントの影響は、自由度の計算時以外は似ています。

参照項目: FREQ ステートメントを使用する例については、“例” (73 ページ)を参照してください。

構文

FREQ *variable*;

必須引数

variable
値がオブザベーションの度数を表す数値変数を指定します。FREQ ステートメントを使用する場合、プロシジャは各オブザベーションが *n* オブザベーションを表すとみなします。*n* は *variable* の値です。*n* は整数でない場合、切り捨てられます。*n* が 1 未満または欠損値の場合、プロシジャはそのオブザベーションを統計量の計算に使用しません。

詳細

度数情報は保存されない
レポート定義を保存する場合、PROC REPORT は FREQ ステートメントを保存しません。

LINE ステートメント

カスタマイズされた要約を書き込むための PUT ステートメントの機能のサブセットを提供します。

制限事項: このステートメントは、レポートの場所と関連付けられている計算ブロックでのみ有効です。
LINE ステートメントを条件ステートメント(IF-THEN、IF-THEN/ELSE および SELECT)で使用できません。LINE ステートメントは、PROC REPORT が計算ブロックのその他すべてのステートメントを実行するまで実行されないためです。

例: “例 2: レポートでの行の並べ替え” (1684 ページ)
“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)
“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1692 ページ)
“例 6: 変数の値ごとに列を作成する” (1695 ページ)
“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)

構文

LINE *specification(s)*;

必須引数

specification(s)
次の形式のうちいずれかになります。異なる形式の指定を 1 つの LINE ステートメントに組み合わせることができます。

item item-format
表示する項目と表示に使用する出力形式を指定します。

item
レポートのデータセット変数、計算変数または統計量の名前です。レポート項目の参照の詳細については、“計算ブロックのレポート項目を参照するための 4 つの方法” (1591 ページ)を参照してください。

item-format
SAS 出力形式またはユーザー定義の出力形式です。項目ごとに出力形式を指定する必要があります。

例 “例 2: レポートでの行の並べ替え” (1684 ページ)

'character-string '
表示するテキストの文字列を指定します。文字列がブランクで、*specification(s)* にそれ以外何もない場合、PROC REPORT はブランク行を印刷します。

注: *character-string* 内で指定される 16 進値(`'DF'x` など)は引用符内で指定されるため、解決されません。16 進値を解決するには、`num` が 16 進値である`%sysfunc(byte(num))` 関数を使用します。マクロ関数が解決するように、*character-string* を二重引用符(" ")で囲みます。

例 “例 2: レポートでの行の並べ替え” (1684 ページ)

number-of-repetitions'character-string '*
文字列とそれを繰り返す回数を指定します。

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)

pointer-control
PROC REPORT が次の指定を表示する列を指定します。次の形式のうちいずれかをポインタコントロールに使用できます。

@column-number
指定リストで次の項目の表示を開始する列の数を指定します。

+column-increment
指定リストで次の項目の表示を開始する前にスキップする列の数を指定します。

column-number と *column-increment* は、変数値、リテラル値のいずれかが可能です。

制限事項 ポインタコントロールは、LISTING 出力向けに設計されています。これらのポインタコントロールは、他の ODS 出力先に影響しません。

詳細

LINE ステートメントと PUT ステートメントの違い

LINE ステートメントでは、PUT ステートメントの次の機能をサポートしていません。

- 等号記号(=)によって示される自動ラベル付け。名前付き出力ともいいます
- 引数_ALL_、_INFILE_、_PAGE_と OVERPRINT オプション
- 1 つの出力形式を項目リストに適用する項目と出力形式のグループ化
- 式を使用したポインタコントロール
- 行ポインタコントロール(#と/)
- 符号(@と@@)で終了
- 出力形式修飾子
- 配列要素

RBREAK ステートメント

レポートの始めまたは最後、あるいは各 BY グループの始めと最後にデフォルト要約を作成します。

例: “例 1: 変数の選択とレポートの要約行の作成” (1680 ページ)
“例 8: パーセントの計算” (1703 ページ)

構文

RBREAK *location* </ *option(s)*>;

オプション引数の要約

COLOR=*color*
: **REPORT** ウィンドウのブレーク行の色を指定します。

CONTENTS='*link-text*'
: 目次で使用されるリンクテキストを指定します。

DOL
: 各値に二重上線を引きます。

DUL
: 各値に二重下線を引きます。

OL
: 各値に上線を引きます。

PAGE
: レポートの始めのブレークの最終ブレーク行の後に新しいページを開始します。

SKIP
: レポートの始めのブレークの最終ブレーク行に対しブレーク行を書き込みます。

STYLE<(*location(s)*)>=<*style-overrides(s)*>
: デフォルト要約行、カスタマイズされた要約行またはその両方に対し、スタイル要素(Output Delivery System の)を指定します。

SUMMARIZE
: 要約行をブレーク行の 1 つとして含めます。

UL
: 各値に下線を引きます。

必須引数

location
ブレーク行の配置を制御します。次のうちいずれかになります。

AFTER
: レポートの最後にブレーク行を置きます。

BEFORE
: レポートの始めにブレーク行を置きます。

オプション引数

COLOR=*color*
REPORT ウィンドウのブレーク行の色を指定します。次の色を使用できます。

BLACK	MAGENTA
BLUE	ORANGE
BROWN	PINK
CYAN	RED
GRAY	WHITE
GREEN	YELLOW

デフォルト **SASCOLOR** ウィンドウの**前景**の色(詳細については、SASCOLOR ウィンドウのオンラインヘルプを参照してください)。

制限事項 このオプションは、対話型レポートウィンドウ環境の出力にのみ影響します。

注 すべての動作環境およびデバイスですべての色がサポートされているわけではありません。一部の動作環境およびデバイスでは、色の間でマップが行われることがあります。一部の動作環境およびデバイスでは、色の間でマップが行われることがあります。たとえば、**DEFINITION** ウィンドウに BROWN という言葉が黄色の文字で表示されている場合、BROWN を選択すると項目が黄色になります。

CONTENTS='*link-text*'
デフォルトで作成された、または STYLE=オプションをサポートする ODS 出力先のオプション設定で作成された目次のエントリのテキストを指定します。PAGE オプションと SUMMARIZE オプションが指定されている RBREAK BEFORE ステートメントだけが目次内にテーブルを作成します。CONTENTS=オプションと、PAGE オプション、SUMMARIZE オプションが指定されると、PROC REPORT は *link-text* の値を使用して、そのテキストを作成したテーブルの目次におきます。CONTENTS=の値が空の引用符の場合、リンクは目次で作成されません。

デフォルト RBREAK BEFORE ステートメントが存在し、PAGE オプションと SUMMARIZE オプションが指定され、CONTENTS=オプションが指定されていない場合、目次のデフォルトのリンクテキストには**要約**が表示されます。

制限事項 CONTENTS=が指定され、PAGE オプションが指定されていない場合、PROC REPORT は SAS ログファイルに警告メッセージを生成します。SUMMARIZE オプションと PAGE オプションが指定されている RBREAK BEFORE/だけが、実際に目次にテーブルを作成できます。

ヒント HTML 出力には現在、追加のアンカータグを含めることができます。

OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

DOL
(二重上線の場合) 13 番目のフォーマット文字を使用して、各値に上線を引きます。

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト 等号(=)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 OL および DOL 両方のオプションを指定すると、PROC REPORT では OL のみ処理されます。

参照項目 FORMCHAR= (1603 ページ)の説明。

DUL
(二重下線の場合) 13 番目のフォーマット文字を使用して各値に下線を引きます。

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト 等号(=)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 UL オプションと DUL オプションを指定すると、PROC REPORT は UL だけを有効化します。

参照項目 FORMCHAR= (1603 ページ)の説明

OL
(上線の場合) 2 番目のフォーマット文字を使用して各値に上線を引きます

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト ハイフン(-)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 OL および DOL 両方のオプションを指定すると、PROC REPORT では OL のみ処理されます。

参照項目 FORMCHAR= (1603 ページ)の説明。

PAGE
レポートの始めのブレークの最終ブレーク行の後に新しいページを開始します。RBREAK BEFORE で、PAGE オプションは新しいテーブルを開始します。

制限事項 このオプションは、OUTPUT 出力先では影響しません。

SKIP
レポートの始めのブレークの最終ブレーク行の後にブレーク行を書き込みます。

制限事項 このオプションは、LISTING 出力先以外の ODS 出力先に影響しません。

STYLE<(*location(s)*)>=<*style-overrides(s)*>
RBREAK ステートメントを使用して作成されるデフォルトの要約行に使用するスタイル要素を指定します。

制限事項 OUTPUT および LISTING を除くすべての ODS 出力先が、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目 “テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ)

SUMMARIZE
要約行をブレーク行の 1 つとして含めます。レポートの始めと最後の要約行には、次の値が含まれます。

- 統計量
- 分析変数
- 計算変数

次の表に、RBREAK ステートメントによって作成される要約行の各種レポート項目の値の PROC REPORT による計算方法を示します。

レポート項目	結果値
統計量	セットのすべてのオブザベーションにわたる統計量の値。
分析変数	DEFINE ステートメントの使い方のオプションとして指定される統計量の値。PROC REPORT は、セットのすべてのオブザベーションにわたる統計量の値を計算します。デフォルトの使用法は SUM です。
計算変数	対応する計算ブロックのコードに基づく計算結果(“COMPUTE ステートメント” (1630 ページ)を参照)。

例 “例 1: 変数の選択とレポートの要約行の作成” (1680 ページ)

“例 8: パーセントの計算” (1703 ページ)

UL
(下線の場合) 2 番目のフォーマット文字を使用して各値に下線を引きます

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます

デフォルト ハイフン(-)

制限事項 このオプションは、LISTING 出力にのみ影響します。他の ODS 出力には影響しません。

操作 UL オプションと DUL オプションを指定すると、PROC REPORT は UL だけを有効化します。

参照項目 FORMCHAR= (1603 ページ)の説明。

詳細

ブレーク行の順序

デフォルト要約に複数のブレーク行が含まれている場合、ブレーク行が表示される順序は次のとおりです。

1. 上線または二重上線(OL または DOL、LISTING 出力のみ)
2. 要約行(SUMMARIZE)
3. 下線または二重下線(UL または DUL、LISTING 出力のみ)
4. スキップ行(SKIP、LISTING 出力のみ)
5. 改ページ(PAGE)

注: ブレークに対しカスタマイズされた要約を定義する場合、カスタマイズされたブレーク行は下線または二重下線の後に表示されます。カスタマイズされたブレーク行の詳細については、COMPUTE ステートメント (1630 ページ) と LINE ステートメント“LINE ステートメント” (1647 ページ)を参照してください。

WEIGHT ステートメント

統計量計算の分析変数に対し重みを指定します。

参照項目: 重み付き統計量の計算の詳細については、“重み付き統計量の計算” (75 ページ)を参照してください。WEIGHT ステートメントを使用する例については、“重み付き統計量の例” (76 ページ)を参照してください。

構文

WEIGHT *variable*;

必須引数

variable
分析変数の値に重みをつける数値変数を指定します。変数の値は整数である必要はありません。

重み値	PROC REPORT 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 未満	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。

重み値	PROC REPORT 応答
欠損値	オブザベーションを除外します

制限事項 重み値がアクティブの場合、PROC REPORT は MODE を計算しません。代わりに、MODE の計算が必要で、重み変数がアクティブの場合は、PROC UNIVARIATE を使用しようとします。

ヒント WEIGHT ステートメントを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。詳細については、VARDEF= (1616 ページ)および “キーワードと式” (2078 ページ)の重み付き統計量の計算を参照してください。

詳細

重み情報は保存されない

レポート定義を保存する場合、PROC REPORT は WEIGHT ステートメントを保存しません。

PROC REPORT で使用できる統計量

次のキーワードを使用して、TABLE ステートメントで統計量を要求するか、KEYWORD ステートメントまたは KEYLABEL ステートメントで統計量キーワードを指定します。

注: 変数名(分類または分析)と統計量名が同じ場合、統計量名を一重引用符で囲みます。(例:`'MAX'`)

表 55.8 *PROC REPORT で使用できる統計量*

記述統計量キーワード	
CSS	PCTSUM
CV	RANGE
MAX	STD
MEAN	STDERR
MIN	SUM
MODE	SUMWGT
N	USS
NMISS	VAR

PCTN	
四分位範囲統計量キーワード	
MEDIAN | P50	Q3 | P75
P1	P60
P5	P70
P10	P80
P20	P90
P30	P95
P40	P99
Q1 | P25	QRANGE
仮説検定キーワード	
PRT | PROBT	T

これらの統計量、その計算に使用される式、そのデータ要件については、“キーワードと式” (2078 ページ)を参照してください。

標準誤差とスチューデントの **t** 検定を計算するには、VARDEF=のデフォルト値 DF を使用する必要があります。

N 以外のすべての統計量が変数と関連付けられている必要があります。数値表示変数以上または以下の統計量を置き換える、または分析変数に対し DEFINE ステートメントまたは **DEFINITION** ウィンドウで使い方のオプションとして統計量を指定することにより、統計量と変数を関連付けます。

N はどこにでも置くことができます。これは、N がレポートのセルの値に使用する入力データセット内のオブザベーション数であるためです。N の値は、特定の変数に依存していません。

注: PROC REPORT ステートメントで MISSING オプションを使用する場合、N には欠損しているグループ変数、順序変数、列変数を含むオブザベーションが含まれています。

基本的なレポート記述プロシジャを使用した ODS スタイルの使用

概要

ODS をサポートする Base SAS プロシジャの多くは、1 つ以上のテーブルテンプレートを使用して出力オブジェクトを生成します。これらのテーブルテンプレートには、テーブ

ル要素(列、ヘッダー、フッター)に対するテンプレートが含まれています。各テーブル要素で、出力のさまざまな部分に 1 つ以上のスタイル要素を使用することを指定できます。これらのスタイル要素はプロシジャの構文内で指定することはできませんが、使用する ODS 出力先に合わせてカスタマイズされたスタイルを使用できます。テーブルとスタイルのカスタマイズの詳細については、“TEMPLATE Procedure: Creating a Style Template” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。

Base SAS レポートプロシジャの PROC PRINT、PROC REPORT および PROC TABULATE を使用してデータをすばやく分析し、読みやすいテーブルに整理することができます。これらのプロシジャステートメントで STYLE=オプションを使用してレポートの表示を変更できます。STYLE=オプションを使用すると、すべての出力のデフォルトスタイルを変更せずに、出力の各セションに変更を加えることができます。プロシジャ内の特定のステートメントで STYLE=オプションを指定して、プロシジャ出力の特定のセクションをカスタマイズすることができます。

次のプログラムでは、STYLE=オプションを使用して以下の PROC REPORT 出力の背景色を作成します。

```
title "Height and Weight by Gender and Age";
proc report nowd data=sashelp.class
   style(header)=[background=white];
   col age (('Gender' sex),(weight height));
   define age / style(header)=[background=lightgreen];
   define sex / across style(header)=[background=yellow] ' ';
   define weight / style(header)=[background=orange];
   define height / style(header)=[background=tan];
run;
```

図 55.10 *PROC REPORT 出力の拡張*



次のプログラムでは、STYLE=オプションを使用して以下の PROC TABULATE 出力の色を作成します。

```
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

proc tabulate data=prdsale;
   class region division prodtype / style=[background=lightgreen];
   classlev region division prodtype  / style=[background=yellow];
   var actual  / style=[background=tan];
   keyword all sum / style=[background=linen color=blue];
```

```
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         box=[label='Region by Division and Type' style=[backgroundcolor=orange]];

title 'Actual Product Sales';
title2 '(millions of dollars)';
run;
```

図 55.11 PROC TABULATE 出力の拡張

Actual Product Sales
(millions of dollars)

Region by Division and Type		Product type		Total
		FURNITURE	OFFICE	
		Actual Sales	Actual Sales	Actual Sales
		Sum	Sum	Sum
Region	Division			
EAST	CONSUMER	$72,570	$108,686	$181,256
	EDUCATION	$73,901	$115,104	$189,005
	Total	$146,471	$223,790	$370,261
WEST	Division			
	CONSUMER	$76,209	$105,020	$181,229
	EDUCATION	$67,945	$110,902	$178,847
	Total	$144,154	$215,922	$360,076
Total	Division			
	CONSUMER	$148,779	$213,706	$362,485
	EDUCATION	$141,846	$226,006	$367,852
	Total	$290,625	$439,712	$730,337

次のプログラムでは、STYLE=オプションを使用して以下の PROC PRINT 出力の色を作成します。

```
proc print data=exprev noobs sumlabel='Total' GRANDTOTAL_LABEL="Grand Total"
      style(table)=[frame=box rules=groups]
      style(bysumline)=[background=red foreground=linen]
      style(grandtotal)=[foreground=green]
      style(header)=[font_style=italic background=orange];
   by sale_type order_date;
   sum price quantity;
   sumby sale_type;
   label sale_type='Sale Type' order_date='Sale Date';
format price dollar10.2 cost dollar10.2;
   var Country / style(data)=[font_face=arial font_weight=bold background=linen];
   var Price / style(data)=[font_style=italic background=yellow];
```

```
   var Cost / style(data)=[foreground=hgt. background=lightgreen];
title 'Retail and Quantity Totals for Each Sale Type';
run;
```

完全な入力データセットについては、“EXPREV” (2154 ページ)を参照してください。

図 55.12 PROC PRINT 出力の拡張



Retail and Quantity Totals for Each Sale Type

Sale Type=Catalog Sale Date=1/1/12

Country	Price	Cost	Quantity
Puerto Rico	$51.20	$12.10	14
Aruba	$123.70	$59.00	30
Bahamas	$113.40	$28.45	8
Bermuda	$41.00	$9.25	7

Sale Type=Catalog Sale Date=1/2/12

Country	Price	Cost	Quantity
British Virgin Islands	$40.20	$20.20	11
Canada	$11.80	$5.00	100
Total	$381.30		170

Sale Type=In Store Sale Date=1/1/12

Country	Price	Cost	Quantity
Virgin Islands (U.S.)	$31.10	$15.65	25

Sale Type=In Store Sale Date=1/2/12

Country	Price	Cost	Quantity
Belize	$146.40	$36.70	2
Cayman Islands	$71.00	$32.30	20
Total	$248.50		47

Sale Type=Internet Sale Date=1/1/12

Country	Price	Cost	Quantity
Antarctica	$92.60	$20.70	2
Grand Total	$722.40		219

スタイル、スタイル要素およびスタイル属性

SAS 出力の表示はスタイルテンプレート(スタイル)で制御されます。スタイルは、SAS 出力の視覚側面(色、フォント、罫線、マーカーなど)を定義する ODS テンプレートの一種です。スタイルにより、そのスタイルを使用するドキュメントの全体的な表示が決まります。スタイルテンプレートは、スタイル要素とスタイル属性で構成されています。

- スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性の名前付きコレクションです。ODS 出力の各領域には、その領域に関連付けられているスタイル要素名があります。スタイル要素名で、スタイル属性が適用される場所を指定します。たとえば、スタイル要素に、列ヘッダーの表示またはセル内のデータの表示に関する指示を含めることができます。スタイル要素で、スタイルを使用する出力のデフォルトの色およびフォントも指定することもできます。
- スタイル属性は、色、フォントのプロパティ、罫線の特性などの視覚プロパティで、予約された名前と値を使用して ODS で定義されます。スタイル属性は、スタイルテンプレート内のスタイル要素によりまとめて参照されます。各*スタイル属性*は、表示の 1 つの側面の値を指定します。たとえば、BACKGROUNDCOLOR=属性では、HTML テーブルまたは印刷出力の色付きテーブルの背景色を指定します。FONTSTYLE=属性では、Roman フォントとイタリックフォントのどちらを使用するか指定します。

注: スタイルはデータの表示を制御するので、LISTING、DOCUMENT または OUTPUT の出力先に移動する出力オブジェクトには影響しません。

使用可能なスタイルは、SASHELP.TMPLMST アイテムストアに含まれています。SAS Enterprise Guide では、スタイルシートのリストがスタイルウィザードで表示されます。バッチモードまたは SAS Studio では、次のコードをサブミットして使用可能なスタイルテンプレートのリストを表示できます。

```
proc template;
list styles / store=sashelp.tmplmst;
run;
```

ODS スタイルの表示に関する詳細は、“Viewing ODS Styles Supplied by SAS” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

デフォルトでは、HTML 出力は HTMLBlue スタイルテンプレートを使用します。スタイル、スタイル要素およびスタイル属性を熟知するために、これらの関係を確認します。以下の図は、スタイル、スタイル要素およびスタイル属性の関係を示しています。次の図は、スタイルの構造を示す例です。

図 55.13 HtmlBlue スタイルの図



以下のリストは、上記の図の番号の付いた項目に対応しています。

1 Styles.HtmlBlue は*スタイル*です。スタイルは、SAS 出力の表示側面(色、フォント、フォントのサイズなど)の表示方法を記述します。スタイルにより、そのスタイルを使用する ODS ドキュメントの全体的な表示が決まります。HTML 出力のデフォルトのスタイルは HtmlBlue です。各スタイルはスタイル要素で構成されています。各出力先には、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルが 1 つあります。

- HTML 出力のデフォルトのスタイルは HTMLBlue です。
- PRINTER の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

を使用して新しいスタイルを作成できます。“DEFINE STYLE Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*).新しいスタイルは既存のスタイルとは別に作成することも、既存のスタイルに基づいて作成することもできます。既存のスタイルから新しいスタイルを作成するには、“PARENT= Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*) を使用できます。ODS スタイルに関する詳細は、“Style Templates” (*SAS Output Delivery System: User's Guide*)を参照してください。

2 *スタイル要素*の例として、ヘッダーとフッターがあります。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。たとえば、スタイル要素に、列ヘッダーの表示またはテーブルセル内のデータの表示に関する指示を含めることができます。スタイル要素で、スタイルを使用する出力のデフォルトの色およびフォントも指定することもできます。スタイル要素はスタイル内に存在し、1 つ以上のスタイル属性で構成されています。スタイル要素はユーザーが定義することも、SAS から提供されるものを使用することもできます。ユーザー定義のスタイル要素は “STYLE Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*)で作成できます。

   注: HTML およびマークアップ言語とその継承に使用されるデフォルトのスタイル要素のリストについては、“Style Elements” (*SAS Output Delivery System: User's Guide*)を参照してください。

3 *スタイル属性*の例として、BORDERCOLOR=、BACKGROUNDCOLOR=、COLOR=があります。スタイル属性では、スタイル要素が適用される出力の領域の 1 つの側面の値を指定します。たとえば、COLOR=属性では、フォントの色に **`cx112277`** の値を指定します。SAS で提供されるスタイル属性のリストについては、“Style Attributes” (*SAS Output Delivery System: User's Guide*)を参照してください。

   スタイル属性はスタイル参照を使用して参照できます。スタイル参照の詳細については、“style-reference” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

次の表に、PROC PRINT、PROC TABULATE および PROC REPORT.ステートメントの STYLE=オプションで設定できる一般的に使用されているスタイル属性を示します。これらの属性のうちほとんどは、セル以外のテーブルの各部分(テーブルの罫線、列と行の間の罫線など)に適用されます。すべての属性がすべての出力先で有効であるわけではありません。これらのスタイル属性、有効な値および適用可能な出力先に関する詳細は、“Style Attributes Tables ” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。

表 55.9 PROC REPORT、PROC TABULATE および PROC PRINT のスタイル属性

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
ASIS=	X	X		X		X
BACKGOUNDCOLOR=	X	X	X	X	X	X
BACKGOUNDIMAGE=	X	X	X	X	X	X
BORDERBOTTOMCOLOR=	X	X		X		
BORDERBOTTOMSTYLE=	X	X	X	X		
BORDERBOTTOMWIDTH=	X	X	X	X		
BORDERLEFTCOLOR=	X	X		X		
BORDERLEFTSTYLE=	X	X	X	X		
BORDERLEFTWIDTH=	X	X	X	X		
BORDERCOLOR=	X	X		X	X	X
BORDERCOLORDARK=	X	X	X	X	X	X
BORDERCOLORLIGHT=	X	X	X	X	X	X
BORDERRIGHTCOLOR=	X	X		X		
BORDERRIGHTSTYLE=	X	X	X	X		
BORDERRIGHTWIDTH=	X	X	X	X		
BORDERTOPCOLOR=	X	X		X		
BORDERTOPSTYLE=	X	X	X	X		
BORDERTOPWIDTH=	X	X	X	X		
BORDERWIDTH=	X	X	X	X	X	X

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
CELLPADDING=	X		X		X	
CELLSPACING=	X		X		X	
CELLWIDTH=	X	X	X	X		X
CLASS=	X	X	X	X	X	X
COLOR=	X	X	X			
FLYOVER=	X	X		X		X
FONT=	X	X	X	X	X	X
FONTFAMILY=	X	X	X	X	X	X
FONTSIZE=	X	X	X	X	X	X
FONTSTYLE=	X	X	X	X	X	X
FONTWEIGHT=	X	X	X	X	X	X
FONTWIDTH=	X	X	X	X		X
FRAME=	X		X		X	
HEIGHT=	X	X		X	X	X
HREFTARGET=		X		X		X
HTMLSTYLE=	X	X	X	X	X	
NOBREAKSPACE=	X	X		X		X
OUTPUTWIDTH=	X	X	X	X	X	
POSTHTML=*	X	X	X	X	X	X
POSTIMAGE=	X	X	X	X	X	X
POSTTEXT=*	X	X	X	X	X	X
PREHTML=*	X	X	X	X	X	X
PREIMAGE=	X	X	X	X	X	X
PRETEXT=*	X	X	X	X	X	X

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
PROTECTSPECIALCHARS=		X		X	X	X
RULES=	X		X		X	
TAGATTR=	X	X	X	X	X	X
TEXTALIGN=	X	X	X	X	X	X
URL=		X		X		X
VERTICALALIGN=		X		X		X
WIDTH=	X	X	X	X	X	

* これらの属性をこの場所で使用する場合には、属性 PRETEXT=、POSTTEXT=、PREHTML=、POSTHTML=で指定されるテキストにのみ影響します。表に表示されるテキストの前景色またはフォントを変更するには、表ではなくセルに影響する場所で対応する属性を設定する必要があります。スタイル属性とその値の詳細な説明については、“Style Attributes” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

テーブル領域のスタイル要素とスタイル属性

次の表に、PROC REPORT 出力の各領域のデフォルトのスタイル要素およびスタイル属性を示します。この表の各場所は、(1613 ページ)の各場所に対応しています。この表には、最もよく使用される ODS 出力先である HTML、PDF および RTF のデフォルトがリストされています。それぞれの出力先には、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルテンプレートが含まれます。

- HTML 出力のデフォルトスタイルは HTMLBlue です。
- PRINTER の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

ODS 出力先とそのデフォルトのスタイルに関する詳細なドキュメントは、“Style Templates” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

表 55.10 テーブルの各場所のデフォルトのスタイル要素とスタイル属性

場所	スタイル要素	HTML スタイル属性	PDF スタイル属性	RTF スタイル属性
ヘッダー	Header	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb
Column	Data	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 8pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000 BORDERWIDTH = NaN COLOR = cx000000	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000
要約	DataEmphasis	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 8pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000 BORDERWIDTH = NaN	FONTFAMILY =""Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = italic COLOR = cx000000

データセルにスタイル属性を適用する際の優先順位

PROC REPORT は、スタイルのデフォルトの優先順位に基づいて特定のセルに使用するスタイル属性を決定します。要約行と要約行以外の行のスタイルの優先順位を次に示します。

要約行以外の行のスタイル優先順位

1. PROC REPORT は、デフォルトのスタイル要素(データ)のデフォルトのスタイル属性を使用します。
2. PROC REPORT ステートメントの STYLE (COLUMN)=オプションによってデフォルトのスタイル属性は上書きされます。
3. DEFINE ステートメントの STYLE(COLUMN)=オプションは、列のすべてのセルに適用されます。

4. CALL DEFINE ステートメントの_ROW_引数によって指定される行のスタイルは、行のすべてのセルに適用されます。
5. CALL DEFINE ステートメントの *column-id* _COL_によって指定されるスタイルは、列のすべてのセルに適用されます。

要約行のスタイル優先順位

1. PROC REPORT は、要約行のデフォルトのスタイル要素(DataEmphasis)のデフォルトのスタイル属性を使用します。
2. PROC REPORT ステートメントの STYLE (SUMMARY)=オプションによって、要約のデフォルトのスタイル属性は上書きされます。
3. BREAK ステートメントの STYLE=オプションによって指定されるスタイルは、要約セルに適用されます。
4. 行のスタイルは、CALL DEFINE ステートメントで、_ROW_引数を使用して指定されます。
5. CALL DEFINE ステートメントで、column-id _COL_、列名または列番号を使用して指定されたスタイルは、列のすべての要約セルに適用されます。

この優先順位を変更するには、TABLE ステートメント (1904 ページ)で STYLE_PRECEDENCE=オプションを使用します。たとえば、STYLE_PRECEDENCE=ROW を指定し、行次元で STYLE=オプションを指定すると、これらのスタイル属性値はテーブルセルに指定されているその他すべてに優先します。

出力形式を使用したスタイル属性値の割り当て

出力形式を使用して、スタイル属性値を割り当てることができます。たとえば、次のコードを使用して、値が負になっている Difference 列に赤い背景色を割り当てます。

```
proc format;
value proffmt low-<0='red'
0-high='green';
run;

proc report data=sashelp.prdsale;
column country predict actual diff;
define country /group;
define diff /'Difference' computed format=dollar12.2
   style(column)=[backgroundcolor=proffmt.];

compute diff;
diff = predict.sum - actual.sum;
endcomp;
run;
```

Country	Predicted Sales	Actual Sales	Difference
CANADA	$233,019.00	$246,990.00	$-13,971.00
GERMANY	$231,554.00	$245,998.00	$-14,444.00
U.S.A.	$241,722.00	$237,349.00	$4,373.00

行間隔の制御

ユーザーは、ページのより多くの行に合うようにレポートを"縮小"する必要があることがよくあります。レポートの縮小により、フォントサイズや行間隔が変わります。プログラマーに最大の柔軟性を提供するため、ODS では行間隔の計算に REPORT の場所に指定されているフォントサイズを使用します。そのため、表を縮小するには、REPORT の場所と COLUMN の場所のフォントサイズを変更します。次に例を示します。

```
proc report
data=libref.data-set-name
            style(report)=[fontsize=8pt]
            style(column)=[font=(Arial, 8pt)];
```

レポートの印刷

ODS による印刷

Output Delivery System を使用したレポートの印刷は、ここで文書化されている以前の印刷方法よりも非常に簡単で、魅力的な出力を提供します。最適な結果を得るには、ODS プリンタグループまたは RTF の出力先を使用します。これらの出力先および ODS ステートメントの使用の詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

非対話型モードまたはバッチモードからの印刷

非対話型モードまたはバッチモードを使用する場合、出力が表示ファイルまたは外部ファイルのいずれかに書き込まれます。これは、動作環境と使用する SAS オプションによって異なります。これらのファイルの指定方法および保存場所に関する情報については、動作環境の SAS ドキュメントを参照してください。

出力ファイルを直接印刷することも、PROC PRINTTO を使用して出力を別のファイルにリダイレクトすることもできます。いずれの場合でもフォームは使用されませんが、出力先が印刷ファイルの場合はキャリッジコントロール文字が書き込まれます。

動作環境コマンドを使用して、ファイルをプリンタに送信します。

PROC PRINTTO の使用

PROC PRINTTO により、SAS 出力と SAS ログの出力先が定義されます。(46 章, “PRINTTO プロシジャ” (1427 ページ)を参照してください。)

PROC PRINTTO はフォームを使用しませんが、印刷ファイルに書き込んでいる場合はキャリッジコントロール文字を書き込みます。

注: 2 つの PROC PRINTTO ステップが必要です。最初の PROC PRINTTO ステップは、PROC REPORT ステップの前になります。出力をファイルにリダイレクトします。2 つ目の PROC PRINTTO ステップは、PROC REPORT ステップの後に続きます。デフォルトの出力先が再構築され、出力ファイルが解放されます。ファイルは PROC PRINTTO によって解放されるまで印刷できません。

レポート定義の格納と再利用

PROC REPORT ステートメントの OUTREPT=オプションは、レポート定義を指定したカタログエントリに保存します。

注: レポート定義がレポートを作成する SAS プログラムと異なっている場合があります。OUTREPT= (1608 ページ)の説明を参照してください。

レポート定義を使用して、レポート定義で使用されるものと同じ名前の変数を含む SAS データセットに対して同一に構築されたレポートを作成できます。PROC REPORT ステートメントの REPORT=オプションを使用して、PROC REPORT の開始時にレポート定義をロードします。詳細については、“REPORT=*libref.catalog.entry*” (1611 ページ)を参照してください。

PROC REPORT のデータベース内処理

データベース内処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。セキュリティの強化は、機密データをデータベース管理システム(DBMS)から抽出する必要がないため可能です。データが、比較的低速なネットワーク接続を介して移送されるかわりに、高速の二次記憶装置を使用して、DBMS 上でローカル操作されます。DBMS が使用されるのは、DBMS に自由に使えるより多くの処理リソースがあり、高度に並列かつスケーラブルな方法で実行するクエリを最適化できるためです。

DATA=入力データセットが DBMS でテーブルまたはビューとして保存される場合、PROC REPORT プロシジャはデータベース内処理を使用して、データベース内の作業のほとんどを実行できます。データベース内処理は、より迅速な処理と、データベースと SAS ソフトウェア間のデータ転送の減少という利点を提供できます。

PROC REPORT のデータベース内処理では、次のデータベース管理システムがサポートされています。

- Aster
- DB2
- Greenplum
- HADOOP
- HAWQ
- Impala
- Oracle
- SAP HANA
- Teradata
- Netezza

PROC REPORT は、SQL 暗黙のパススルーを使用してデータベース内処理を実行します。プロシジャはステートメントに基づく SQL クエリと、プロシジャで指定される出力統計量と同様に使用される PROC REPORT オプションを生成します。データベースは

これらの SQL クエリを実行し、クエリの結果は PROC REPORT に送信されます。生成された SQL を検証するには、SASTRACE=オプションを設定します。

SAS 出力形式定義がデータベースに配置されていない場合、データベース内集計が未加工値で発生し、結果のセットが PROC REPORT 内部構造に結合されるときに関連出力形式が SAS によって適用されます。詳細については、*SAS/ACCESS for Relational Databases: Reference* のセクション“Deploying and Using SAS Formats”を参照してください。

使い方の種類が DISPLAY または ORDER の変数が PROC REPORT ステップに含まれている場合、データベース内処理は発生しません。

データベース内処理に対し、次の統計量がサポートされています。N、NMISS、MIN、MAX、MEAN、 RANGE、SUM、SUMWGT、CSS、USS、VAR、STD、STDERR、CV.

データベース内処理の重みづけは、N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN に対してのみサポートされています。

SQLGENERATION システムオプションまたは LIBNAME ステートメントオプションは、データベース内プロシジャがデータベース内で実行されるかどうか、およびその実行方法を制御します。デフォルトで、データベース内プロシジャは可能な場合はデータベース内で実行されます。データベース内処理を実行しない、多くのデータセットオプションがあります。完全なリストについては、*SAS/ACCESS for Relational Databases: Reference* の“In-Database Procedures”を参照してください。

データベース内処理の詳細については、*SAS/ACCESS for Relational Databases: Reference* を参照してください。

PROC REPORT でのレポート作成法

イベントの順序

このセクションでは、レポート作成の一般的なプロセスについて説明します。このプロセスの例については、“レポートの作成例” (1672 ページ)を参照してください。プログラミングステートメントまたは対話型レポートウィンドウ環境のいずれを使用しても、イベントの順序は同じです。

レポート作成のプロセスを理解するには、レポート変数と一時変数の違いを理解する必要があります。**レポート変数**は、COLUMN ステートメントで指定される変数です。入力データセットから取得することも、計算する(変数の DEFINE ステートメントが COMPUTED オプションを指定)こともできます。計算ブロックに表示されることも、表示されないこともあります。1 つ以上の計算ブロックだけに表示される変数は、**一時変数**です。一時変数はレポートに表示されず、出力データセットに書き込まれません(要求される場合)。

PROC REPORT は、レポートの各行の始めのレポート変数を欠損値に初期化します。**一時変数**の値は、PROC REPORT がレポートの行の構成を開始する前に欠損値に初期化され、値を割り当てるまで欠損値のままです。PROC REPORT は、計算ブロック間の実行からの**一時変数**の値を保持します。

PROC REPORT はレポートを次のように構成します。

1. グループ変数、順序変数、列変数ごとにデータをまとめます。レポートのすべての統計量、詳細行の統計量、およびブレークの要約行の統計量を計算します。統計量には、分析変数に対して計算される統計量が含まれます。PROC REPORT は要約行の統計量を、レポートに表示されるかどうかに関係なく計算します。

2. すべての一時変数を欠損値に初期化します。
3. レポートの行の構成を開始します。
   a. 各行の始めに、すべてのレポート変数を欠損値に初期化します。
   b. レポート変数の値を左から右に入力します。
      - 計算変数の値は、対応する計算ブロックのステートメント実行によって取得されます。
      - その他すべての変数の値は、データセット、またはレポート作成プロセスの開始時に計算された要約統計量から取得されます。
   c. ブレークに関しては、PROC REPORT はまず BREAK ステートメントまたは RBREAK ステートメント、または **BREAK** ウィンドウのオプションで作成されるブレーク行を構成します。
   d. 計算ブロックがブレークに割り当てられている場合、PROC REPORT は計算ブロックのステートメントを実行します。詳細については、“要約行の構造” (1671 ページ) を参照してください。

   注: PROC REPORT がレポートを作成する方法によって、次を行うことができます。
      - グループ変数の前のブレークに対する計算ブロックのグループ統計量を使用する
      - レポートの始めの計算ブロックの全体レポートに対する統計量を使用する

      このドキュメントは、適切な複合名の統計量を参照します。レポート項目の参照の詳細については、“計算ブロックのレポート項目を参照するための 4 つの方法” (1591 ページ)を参照してください。

   注: LINE ステートメントを条件ステートメント(IF-THEN、IF-THEN/ELSE および SELECT)で使用できません。LINE ステートメントは、PROC REPORT が計算ブロックのその他すべてのステートメントを実行するまで実行されないためです。
4. 各レポート行が完了すると、PROC REPORT は行を現在オープンなすべての ODS 出力先に送信します。

要約行の構造

次の条件のいずれかが真の場合、PROC REPORT はブレークの要約行を構成します。

- ブレークの数値変数を要約します。
- ブレークの計算ブロックを使用します。(BREAK ステートメントまたは RBREAK ステートメントを使用せずに、または **BREAK** ウィンドウでオプションを選択せずに計算ブロックをブレークに添付できます)。

  計算ブロックの使用の詳細については、“計算ブロックの使用” (1590 ページ) と “COMPUTE ステートメント” (1630 ページ)を参照してください。

PROC REPORT がこの時点で構成する要約行は、予備です。計算ブロックがブレークに割り当てられていない場合、予備要約行は最終要約行となります。ただし、計算ブロックがブレークに割り当てられている場合、計算ブロックのステートメントは予備要約行の値を変える可能性があります。

PROC REPORT が要約行を印刷するのは、ブレークの数値変数を要約する場合だけです。

レポートの作成例

グループを使用したレポートとレポート要約の作成

ログ 55.2 (1673 ページ) のレポートには、5 つの列が含まれています。

- Sector と Department はグループ変数です。
- Sales は、Sum 統計量の計算に使用される分析変数です。
- Profit は、値が Department の値に基づく計算変数です。
- N 統計量は、各行が表すオブザベーション数を示します。

レポートの最後で、ブレークはレポートの統計量と計算変数を要約して、Sector に **TOTALS:** の値を割り当てます。

次のステートメントは ログ 55.2 (1673 ページ)を作成します。使用されるユーザー定義の出力形式が PROC FORMAT ステップ (1681 ページ)によって作成されます。

```
libname proclib
'SAS-library';

options nodate pageno=1 linesize=64
        pagesize=60 fmtsearch=(proclib);

ods html close;
ods listing;
proc report data=grocery headline headskip;
    column sector department sales Profit N ;
    define sector / group format=$sctrfmt.;
    define department   / group format=$deptfmt.;
    define sales  / analysis sum
                    format=dollar9.2;
    define profit / computed format=dollar9.2;

    compute before;
    totprof = 0;
    endcomp;

    compute profit;
    if sector ne ' ' or department ne ' ' then do;
       if department='np1' or department='np2'
          then profit=0.4*sales.sum;
       else profit=0.25*sales.sum;
       totprof = totprof + profit;
       end;
    else
       profit = totprof;
    endcomp;

    rbreak after / dol dul summarize;
    compute after;
       sector='TOTALS:';
    endcomp;

    where sector contains 'n';
    title 'Report for Northeast and Northwest Sectors';
```

```
  run;
ods listing close;
```

ログ 55.2 グループを含むレポートとレポート要約

```
Report for Northeast and Northwest Sectors          1 Sector     Department
Sales     Profit          N
---------------------------------------------------- Northeast  Canned
$840.00    $336.00          2 Meat/Dairy    $490.00    $122.50          2
Paper         $290.00     $116.00         2 Produce       $211.00
$52.75          2 Northwest  Canned      $1,070.00    $428.00          3 Meat/
Dairy  $1,055.00    $263.75          3 Paper         $150.00     $60.00
3 Produce       $179.00     $44.75          3 =========              =========
=========  ========= TOTALS:                $4,285.00  $1,423.75         20
=========              =========  =========  =========
```

PROC REPORT によるこのレポートの作成方法についての説明は、次のとおりです。

1. PROC REPORT は、データ(Sector と Department はグループ変数です)をまとめ、レポートの最後に各詳細行およびブレークに対し統計量(Sales.sum と N)を計算することにより、レポートの作成を開始します。
2. これで、PROC REPORT はレポートの最初の行の作成を開始できます。このレポートにはレポートの始めのブレークまたはグループの前のブレークが含まれていないため、レポートの開始行は詳細行になります。次の図にあるように、プロシジャはすべてのレポート変数を欠損値に初期化します。文字変数の欠損値はブランクで表され、数値変数の欠損値はピリオドで表されます。

   図 55.14 値が初期化された最初の詳細行

Sector	Department	Sales	Profit	N
		.	.	.

3. 次の図に、行の最初の 3 列の構造を示します。PROC REPORT は、行の値を左から右に入力します。値は、レポート作成プロセスの開始時に計算された統計量から取得されます。

   図 55.15 値が左から右に入力された最初の詳細行

Sector	Department	Sales	Profit	N
Northeast		.	.	.

Sector	Department	Sales	Profit	N
Northeast	Canned	.	.	.

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	.	.

4. レポートの次の列には、計算変数 Profit が含まれます。この列に達すると、PROC REPORT は Profit に割り当てられている計算ブロックのステートメントを実行します。非生鮮品目(値が **np1** または **np2**)は利益の 40%を占め、生鮮品目(値が **p1** または **p2**)は利益の 25%を占めています。

```
if department='np1' or department='np2'
   then profit=0.4*sales.sum;
else profit=0.25*sales.sum;
```

行は次の図のようになります。

注: 計算変数の位置は重要です。PROC REPORT は値をレポート行の列に、左から右に割り当てます。その結果、レポートの右側に表示される変数に基づいて計算変数の計算を行うことはできません。

図 55.16 *最初の詳細行に追加された計算変数*

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	$336.00	.

5. 次に、PROC REPORT は N 統計量に対する値を入力します。値は、レポート作成プロセスの開始時に作成された統計量から取得されます。次の図に、完了した行を示します。

図 55.17 *最初の完了詳細行*

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	$336.00	2

6. プロシジャは完了した行をレポートに書き込みます。
7. PROC REPORT は、レポートの詳細行ごとにステップ 2、3、4、5、6 を繰り返します。
8. レポートの最後のブレークで、PROC REPORT は RBREAK ステートメントによって記述されるブレーク行を構成します。これらの行には、二重下線、二重上線、要約行の予備バージョンが含まれます。要約行の統計量は前に計算されています。(ステップ 1 を参照)。計算変数の値は、詳細行と同様に、PROC REPORT が適切な列に達すると計算されます。PROC REPORT はこれらの値を使用して、要約行の予備バージョンを作成します。(次の図を参照)。

図 55.18 *予備要約行*

Sector	Department	Sales	Profit	N
		$4,285.00	$1,071.25	20

9. 計算ブロックがブレークに割り当てられていない場合、要約行の予備バージョンは最終バージョンと同じです。ただしこの例では、計算ブロックはブレークに割り当てられています。そのため、PROC REPORT は計算ブロックのステートメントを今実行します。この場合、計算ブロックにはステートメントが 1 つ含まれています。

```
sector='TOTALS:';
```

このステートメントは、デフォルトで要約行にはない Sector の値を **TOTALS:**という言葉と置き換えます。PROC REPORT はステートメントを実行した後、要約行を変更して、この変更を Sector の値に反映します。要約行の最終バージョンが次の図に表示されます。

図 55.19 *最終要約行*

Sector	Department	Sales	Profit	N
TOTALS:		$4,285.00	$1,071.25	20

10. 最後に、PROC REPORT はすべてのブレーク行を下線、上線、最終要約行とともにレポートに書き込みます。

一時変数を使用するレポートの作成

PROC REPORT は、レポートの各行の始めのレポート変数を欠損値に初期化します。一時変数の値は、PROC REPORT がレポートの行の構成を開始する前に欠損値に初期化され、値を割り当てるまで欠損値のままです。PROC REPORT は、計算ブロック間の実行からの一時変数の値を保持します。

すべての計算ブロックですべての変数の現在の値を共有しているため、レポートの始めのブレークまたはブレーク変数の前のブレークで一時変数を初期化できます。このレポートは、Sector の前のブレークで一時変数 Sctrtot を初期化します。

注: PROC REPORT は、対応する計算ブロックを実行する前に、ブレークの予備要約行を作成します。要約行に計算変数が含まれている場合、計算は予備要約行の要因となる変数の値に基づいています。計算ブロックで設定した値に基づいて計算変数を再計算する場合は、計算ブロックで明示的に行う必要があります。このレポートでは、この方法について示します。計算ブロックがブレークに割り当てられていない場合、予備要約行は最終要約行となります。

ログ 55.3 (1677 ページ) のレポートには、5 つの列が含まれています。

- Sector と Department はグループ変数です。
- Sales はこのレポートで二度使用される分析変数です。Sum 統計量の計算と Pctsum 統計量の計算にそれぞれ一度ずつ使用されます。
- Sctrpct は、値が Sales、およびセクタの売上合計である一時変数 Sctrtot の値に基づく計算変数です。

レポートの始めに、カスタマイズされたレポート要約によりすべての店舗の売上が次のようになるように指定されます。部門の各オブザベーショングループの前のブレークで、デフォルト要約はそのセクタのデータを要約します。各グループの最後にブレークはブレーク行を挿入します。

次のステートメントは、ログ 55.3 (1677 ページ)を作成します。使用されるユーザー定義の出力形式が PROC FORMAT ステップ (1681 ページ)によって作成されます。

注: パーセントの計算は、その結果に 100 を掛けません。PROC REPORT はその結果を PERCENT.出力形式で印刷するためです。

```
libname proclib
'SAS-library';

options nodate pageno=1 linesize=64
        pagesize=60 fmtsearch=(proclib);
```

```
ods html close;
ods listing;
proc report data=grocery noheader;
  column sector department sales
         Sctrpct sales=Salespct;

  define sector     / 'Sector' group
                      format=$sctrfmt.;
  define department / group format=$deptfmt.;
  define sales      / analysis sum
                      format=dollar9.2 ;
  define sctrpct    / computed
                      format=percent9.2 ;
  define salespct   / pctsum format=percent9.2;

  compute before;
     line ' ';
     line @16 'Total for all stores is '
          sales.sum dollar9.2;
     line ' ';
     line @29 'Sum of' @40 'Percent'
          @51 'Percent of';
     line @6 'Sector' @17 'Department'
          @29 'Sales'
          @40 'of Sector' @51 'All Stores';
     line @6 55*'=';
     line ' ';
  endcomp;

  break before sector / summarize ul;
  compute before sector;
     sctrtot=sales.sum;
     sctrpct=sales.sum/sctrtot;
  endcomp;

  compute sctrpct;
     sctrpct=sales.sum/sctrtot;
  endcomp;

  break after sector/skip;
  where sector contains 'n';
  title 'Report for Northeast and Northwest Sectors';
run;
ods listing close;
```

ログ 55.3 *一時変数を含むレポート*

```
Report for Northeast and Northwest Sectors          1 Total for all stores is
$4,285.00 Sum of     Percent    Percent of Sector     Department  Sales      of
Sector  All Stores ==========================================================
Northeast              $1,831.00   100.00%     42.73% ---------
---------  ---------  --------- Northeast  Canned        $840.00    45.88%
19.60% Meat/Dairy    $490.00    26.76%     11.44% Paper         $290.00
15.84%      6.77% Produce       $211.00    11.52%      4.92%
Northwest              $2,454.00   100.00%     57.27% ---------
---------  ---------  --------- Northwest  Canned      $1,070.00    43.60%
24.97% Meat/Dairy  $1,055.00    42.99%     24.62% Paper         $150.00
6.11%      3.50% Produce       $179.00     7.29%      4.18%
```

PROC REPORT によるこのレポートの作成方法についての説明は、次のとおりです。

1. PROC REPORT は、データ(Sector と Department はグループ変数です)をまとめ、各詳細行、レポートの始めのブレーク、各グループの前のブレーク、および各グループの後のブレークに対し統計量(Sales.sum と Sales.pctsum)を計算することにより、レポートの作成を開始します。
2. PROC REPORT は、一時変数 Sctrtot を欠損値に初期化します。(次の図を参照)。

   図 55.20 *初期化された一時変数*

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
		.	.	.	.

3. この PROC REPORT ステップには COMPUTE BEFORE ステートメントが含まれているため、プロシジャはレポートの始めのブレークに対し予備要約行を構成します。この予備要約行には、統計量(Sales.sum と Sales.pctsum)と計算変数(Sctrpct)に対する値が含まれます。

   このブレークで、Sales.sum は全店舗の売上で、Sales.pctsum はこれらの売上が全店舗に対して表すパーセント(100%)です。PROC REPORT は、これらの統計量の値をレポート作成プロセスの開始時に計算された統計量から取得します。

   Sctrpct の値は、対応する計算ブロックのステートメントの実行から取得されます。Sctrtot の値が欠損しているため、PROC REPORT は Sctrpct の値を計算できません。そのため、(この場合は印刷されない)予備要約行のこの変数にも欠損値が含まれています(次の図を参照)。(次の図を参照)。

   COMPUTE BEFORE ブロックのステートメントは、変数を変えることはありません。そのため、最終要約行は、予備要約行と同じです。

   注: COMPUTE BEFORE ステートメントは、レポートの始めにブレークを作成します。RBREAK ステートメントを使用する必要はありません。

   図 55.21 *レポートの始めのブレークの予備/最終要約行*

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
		$4,285.00	.	100.00%	.

4. SUMMARIZE オプションを指定した RBREAK ステートメントがプログラムに含まれていないため、PROC REPORT は最終要約行をレポートに書き込みません。かわりに LINE ステートメントを使用して、Sales.sum の値を文に組み込むカスタマイズされた要約を書き込み、カスタマイズされた列ヘッダーを書き込みます。(PROC

REPORT ステートメントの NOHEADER オプションは、カスタマイズされた要約の前に表示されるデフォルト列ヘッダーを非表示にします)。

5. 次に、PROC REPORT はオブザベーションの最初のグループの前のブレークに対して予備要約行を構成します。(このブレークは BREAK ステートメントの SUMMARIZE オプションを使用し、計算ブロックが割り当てられています。これらの条件のうちいずれかが要約行を生成します)。予備要約行には、ブレーク変数(Sector)、統計量(Sales.sum と Sales.pctsum)、計算変数(Sctrpct)に対する値が含まれています。このブレークでは、Sales.sum は 1 つのセクタ(北東セクタ)の売上です。PROC REPORT は Sector、Sales.sum、Sales.pctsum の値をレポート作成プロセスの開始時に計算された統計量から取得します。

   Sctrpct の値は、対応する計算ブロックのステートメントを実行することにより取得されます。Sctrtot の値がまだ欠損しているため、PROC REPORT は Sctrpct の値を計算できません。そのため、予備要約行では、Sctrpct に欠損値が含まれています。(次の図を参照)。

   図 55.22 *オブザベーションの最初のグループの前のブレークに対する予備要約行*

	Report Variables				Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast		$1,831.00	.	42.73%	.

6. PROC REPORT は、COMPUTE BEFORE SECTOR 計算ブロックのステートメントを実行することにより、要約行の最終バージョンを作成します。これらのステートメントは、Sector の値が変わるたびに一度実行します。

   - 最初のステートメントは、レポートのその部分で 1 つの Sector の合計売上を表す Sales.sum の値を変数 Sctrtot に割り当てます。
   - 2 番目のステートメントは、Sctrtot の新しい値から Sctrpct を再計算することによって要約行を完了します。次の表に、最終要約行を示します。

   注: この例では、最終要約行の Sctrpct の値を再計算する必要があります。再計算しない場合、Sctrpct の値は欠損値になります。COMPUTE Sctrpct ブロックの実行時点で Sctrtot の値が欠損値であるためです。

   図 55.23 *オブザベーションの最初のグループの前のブレークに対する最終要約行*

	Report Variables				Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast		$1,831.00	100.00%	42.73%	$1,831.00

7. プログラムには SUMMARIZE オプションが指定されている BREAK BEFORE ステートメントが含まれているため、PROC REPORT は最終要約行をレポートに書き込みます。BREAK ステートメントの UL オプションは、要約行に下線を引きます。

8. これで、PROC REPORT は最初のレポート行の作成を開始できます。すべてのレポート変数は欠損値に初期化されます。一時変数の値は変わりません。次の表に、この時点での最初の詳細行を示します。

   図 55.24 *値が初期化された最初の詳細行*

	Report Variables				Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
		.	.	.	$1,831.00

9. 次の図に、行の最初の 3 列の構造を示します。PROC REPORT は、行の値を左から右に入力します。値は、レポート作成プロセスの開始時に計算された統計量から取得されます。

図 55.25 *左から右への値の入力*

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast		.	.	.	$1,831.00

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast	Canned	.	.	.	$1,831.00

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast	Canned	$840.00	.	.	$1,831.00

10. レポートの次の列には、計算変数 Sctrpct が含まれます。この列に達すると、PROC REPORT は Sctrpct に割り当てられている計算ブロックのステートメントを実行します。このステートメントは、この部門が占めるセクタの売上合計のパーセントを計算します。

```
sctrpct=sales.sum/sctrtot;
```

行は次の図のようになります。

図 55.26 *最初の計算変数が追加された最初の詳細行*

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast	Canned	$840.00	45.88%	.	$1,831.00

11. レポートの次の列には、統計量 Sales.pctsum が含まれます。PROC REPORT はこの値を、レポート作成プロセスの開始時に作成された統計量から取得します。最初の詳細行はこれで完了です。(次の図を参照)。

図 55.27 *最初の完了詳細行*

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast	Canned	$840.00	45.88%	19.60%	$1,831.00

12. PROC REPORT は詳細行をレポートに書き込みます。グループの詳細行ごとにステップ 8、9、10、11、12 を繰り返します。

13. グループの最後の詳細行をレポートに書き込んだ後、PROC REPORT はデフォルトグループ要約を構成します。計算ブロックがこのブレークに割り当てられておらず、BREAK AFTER ステートメントに SUMMARIZE オプションが含まれていないため、PROC REPORT は要約行を構成しません。このブレークでの唯一のアクションとして、BREAK AFTER ステートメントの SKIP オプションがグループの最後の詳細行の後にブランク行を書き込みます。

14. これで、ブレーク変数の値が `Northeast` から `Northwest` に変わります。PROC REPORT は、このオブザベーショングループの前のブレークに対し予備要約行を構成します。行の始めに、ROC REPORT はすべてのレポート変数を欠損値に初

期化しますが、一時変数の値は保持します。次に、ブレーク変数(Sector)、統計量(Sales.sum と Sales.pctsum)、計算変数(Sctrpct)に対する適切な値を含む予備要約行を入力します。このブレークで、Sales.sum は北西セクタの売上です。COMPUTE BEFORE Sector ブロックが未実行のため、Sctrtot の値は北東セクタの値である $1,831.00 のままです。そのため、PROC REPORT がこの予備要約行の Sctrpct に対して計算する値は、正しくありません。(次の図を参照)。このブレークの計算ブロックのステートメントは正しい値を計算します(次のステップを参照)。

図 55.28 *オブザベーションの 2 番目のグループの前のブレークに対する予備要約行*

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northwest		$2,454.00	134.00%	57.27%	$1,831.00

注意:

結果が正しくなるように、ブレーク行の計算変数の値を同期します。 PROC REPORT ステップがそのブレークに割り当てられている計算ブロックの Sctrpct を再計算しない場合、最終要約行の値は要約行のその他の値と同期されず、レポートは正しくなくなります。

15. PROC REPORT は、COMPUTE BEFORE Sector 計算ブロックのステートメントを実行することによって、要約行の最終バージョンを作成します。これらのステートメントは、Sector の値が変わるたびに一度実行します。

    - 最初のステートメントは、レポートのその部分で 1 つの北西セクタの売上を表す Sales.sum の値を変数 Sctrtot に割り当てます。
    - 2 番目のステートメントは、Sctrtot の新しい適切な値から Sctrpct を再計算することによって要約行を完了します。次の表に、最終要約行を示します。

図 55.29 *オブザベーションの 2 番目のグループの前のブレークに対する最終要約行*

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northwest		$2,454.00	100.00%	57.27%	$2,454.00

プログラムには SUMMARIZE オプションが指定されている BREAK BEFORE ステートメントが含まれているため、PROC REPORT は最終要約行をレポートに書き込みます。BREAK ステートメントの UL オプションは、要約行に下線を引きます。

16. これで、PROC REPORT はこのオブザベーショングループの最初の行の作成を開始できます。入力データセットのすべてのオブザベーションが処理されるまで、ステップ 8 から 16 まで(最後のオブザベーショングループについてはステップ 14 まで)が繰り返されます。

例: REPORT プロシジャ

例 1: 変数の選択とレポートの要約行の作成

要素: PROC REPORT ステートメントオプション
COLUMN ステートメント

RBREAK ステートメントオプション
AFTER
STYLE=
SUMMARIZE

他の要素: LIBNAME statement
PROC FORMAT ステートメント
DATA ステップ
OPTIONS ステートメント
FORMAT statement
WHERE statement
TITLE statement

詳細

この例では、永久データセットと永久出力形式を使用して、次を含むレポートを作成します。

- オブザベーションごとに 1 行
- レポート全体のデフォルト要約

プログラム

```
libname proclib 'SAS-library';

data grocery;
   input Sector $ Manager $ Department $ Sales @@;
   datalines;
se 1 np1 50    se 1 p1 100   se 1 np2 120   se 1 p2 80
se 2 np1 40    se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60    nw 3 p1 600   nw 3 np2 420   nw 3 p2 30
nw 4 np1 45    nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45    nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53    sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
sw 6 np1 40    sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90    ne 7 p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;

proc format library=proclib;
   value $sctrfmt 'se' = 'Southeast'
                  'ne' = 'Northeast'
                  'nw' = 'Northwest'
                  'sw' = 'Southwest';

   value $mgrfmt '1' = 'Smith'   '2' = 'Jones'
                 '3' = 'Reveiz'  '4' = 'Brown'
                 '5' = 'Taylor'  '6' = 'Adams'
                 '7' = 'Alomar'  '8' = 'Andrews'
                 '9' = 'Pelfrey';

   value $deptfmt 'np1' = 'Paper'
                  'np2' = 'Canned'
                  'p1'  = 'Meat/Dairy'
                  'p2'  = 'Produce';
```

```
run;

options fmtsearch=(proclib);

proc report data=grocery;

   column manager department sales;

   rbreak after / summarize style=[font_weight=bold];

   where sector='se';

   format manager $mgrfmt. department $deptfmt.
          sales dollar11.2;

   title 'Sales for the Southeast Sector';
   title2 "for &sysdate";
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

GROCERY データセットを作成します。 GROCERY には、Grocery Mart チェーンの 8 つの店舗の 1 日の売上が含まれています。オブザベーションにはそれぞれ 1 つの店舗の 1 つの部門に対する 1 日の売上データが含まれています。

```
data grocery;
   input Sector $ Manager $ Department $ Sales @@;
   datalines;
se 1 np1 50    se 1 p1 100   se 1 np2 120   se 1 p2 80
se 2 np1 40    se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60    nw 3 p1 600   nw 3 np2 420   nw 3 p2 30
nw 4 np1 45    nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45    nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53    sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
sw 6 np1 40    sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90    ne 7 p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;
```

出力形式$SCTRFMT.、$MGRFMT.、$DEPTFMT.を作成します。 PROC FORMAT は、Sector、Manage、Department に対し永久出力形式を作成します。LIBRARY=オプションは、出力形式が後続の SAS セッションで使用できるように永久保管場所を指定します。これらの出力形式は、このセッション全体にわたって例に使用されます。

```
proc format library=proclib;
   value $sctrfmt 'se' = 'Southeast'
                  'ne' = 'Northeast'
                  'nw' = 'Northwest'
                  'sw' = 'Southwest';

   value $mgrfmt '1' = 'Smith'   '2' = 'Jones'
                 '3' = 'Reveiz'  '4' = 'Brown'
                 '5' = 'Taylor'  '6' = 'Adams'
                 '7' = 'Alomar'  '8' = 'Andrews'
                 '9' = 'Pelfrey';
```

```
    value $deptfmt 'np1' = 'Paper'
                   'np2' = 'Canned'
                   'p1'  = 'Meat/Dairy'
                   'p2'  = 'Produce';
run;
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションを指定します。 デフォルトでは、REPORT ウィンドウなしで REPORT プロシジャが実行され、その出力が空いている出力先に送信されます。

```
proc report data=grocery;
```

レポート列を指定します。 レポートには、Manager、Department および Sales に対する列が含まれています。これらの変数に対し DEFINE ステートメントがないため、PROC REPORT は表示変数として文字変数(Manager と Department)を、合計統計量の計算に使用される分析変数として数値変数(Sales)を使用します。

```
    column manager department sales;
```

レポートの要約を作成します。 RBREAK ステートメントは、レポートの最後にデフォルト要約を作成します。SUMMARIZE はレポートのすべてのオブザベーションに対する Sales の値を合計します。STYLE=は、要約値を太字にするために使用します。

```
    rbreak after / summarize style=[font_weight=bold];
```

処理するオブザベーションを選択します。 WHERE ステートメントは、南東セクタの店舗のオブザベーションのみレポートに選択します。

```
    where sector='se';
```

レポート列をフォーマットします。 FORMAT ステートメントはレポートで使用する出力形式を割り当てます。FORMAT ステートメントは、データセット変数とのみ使用できます。

```
    format manager $mgrfmt. department $deptfmt.
           sales dollar11.2;
```

タイトルを指定します。 SYSDATE は、SAS ジョブまたは SAS セッション開始時に日付を返す自動マクロ変数です。TITLE2 ステートメントは、マクロ変数が解決するように一重引用符ではなく二重引用符を使用します。

```
    title 'Sales for the Southeast Sector';
    title2 "for &sysdate";
run;
```

出力:出力:HTML

アウトプット 55.1 *変数の選択とレポートの要約行の作成*

Sales for the Southeast Sector for 03MAY14

Manager	Department	Sales
Smith	Paper	$50.00
Smith	Meat/Dairy	$100.00
Smith	Canned	$120.00
Smith	Produce	$80.00
Jones	Paper	$40.00
Jones	Meat/Dairy	$300.00
Jones	Canned	$220.00
Jones	Produce	$70.00
		$980.00

例 2: レポートでの行の並べ替え

要素: PROC REPORT ステートメントオプション
COLUMN ステートメント
DEFINE ステートメントオプション
ANALYSIS
FORMAT=
ORDER
ORDER=
SUM
BREAK ステートメントオプション
AFTER
SUMMARIZE
STYLE=

他の要素: LIBNAME statement
OPTIONS ステートメント
WHERE statement
TITLE statement

データセット: GROCERY

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例では、次を行います。

- Manager のフォーマットされた値と Department の内部値によって行をアルファベット順に調整する(非生鮮商品を販売する 2 つの部門の売上が生鮮商品を販売する 2 つの部門の売上の前に来るように)
- デフォルトの列幅および列間のスペースを制御する
- マネージャごとに売上のデフォルト要約を作成する
- レポート全体の売上のカスタマイズされた要約を作成する

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc report data=grocery;

   column manager department sales;

   define manager / order order=formatted format=$mgrfmt.;
   define department / order order=internal format=$deptfmt.;

   define sales / analysis sum format=dollar7.2;

   break after manager / summarize
                         style=[font_style=italic];

   where sector='se';

   title 'Sales for the Southeast Sector';
 run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT は REPORT ウィンドウを使用せずに実行され、出力は開いている出力先に送信されます。

```
proc report data=grocery;
```

レポート列を指定します。 レポートには、Manager、Department および Sales に対する列が含まれています。

```
   column manager department sales;
```

並べ替え順序変数を定義します。 DEFINE ステートメントで ORDER オプションが指定されているすべての変数の値によって、レポートの行の順序が決定されます。このレポ

ートで PROC REPORT は最初に Manager の値(これが COLUMN ステートメントの最初の変数のため)、次に Department の値によって行を調整します。ORDER=は変数の並べ替え順序を指定します。このレポートは Manager の出力適用値と Department の内部値(np1、np2、p1、p2)に従って行を調整します。FORMAT=で、レポートに使用する出力形式を指定します。

```
define manager / order order=formatted format=$mgrfmt.;
define department / order order=internal format=$deptfmt.;
```

分析変数を定義します。 Sum は、現在の行によって表されるすべてのオブザベーションに対する合計統計量を計算します。このレポートで、行はそれぞれ 1 つのオブザベーションだけを表します。そのため、Sum 統計量は、入力データセットのそのオブザベーションの Sales の値と同じになります。このレポートで Sales を分析変数として使用すると、各グループおよびレポートの最後で値を要約できます。

```
define sales / analysis sum format=dollar7.2;
```

レポートの要約を作成します。 この BREAK ステートメントは、各マネージャの最終行の後にデフォルト要約を作成します。PROC REPORT は、Sales が Sum 統計量の計算に使用される分析変数であるため、各マネージャに対する Sales の値を合計します。SKIP は要約行の後にブランク行を書き込みます。

```
break after manager / summarize
                      style=[font_style=italic];
```

処理するオブザベーションを選択します。 WHERE ステートメントは、南東セクタの店舗のオブザベーションのみレポートに選択します。

```
where sector='se';
```

タイトルを指定します。

```
  title 'Sales for the Southeast Sector';
run;
```

出力:出力:HTML

アウトプット 55.2 *レポートでの行の並べ替え*

Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	$40.00
	Canned	$220.00
	Meat/Dairy	$300.00
	Produce	$70.00
Jones		*$630.00*
Smith	Paper	$50.00
	Canned	$120.00
	Meat/Dairy	$100.00
	Produce	$80.00
Smith		*$350.00*

例 3: 別名を使用し、同一変数に複数の統計量を求める

要素: PROC REPORT ステートメントオプション

COLUMN ステートメント
- 別名

COMPUTE ステートメント引数
- AFTER

DEFINE ステートメントオプション
- ORDER=
- ANALYSIS
- SUM
- FORMAT=
- MAX
- MIN
- NOPRINT
- 列ヘッダーのカスタマイズ

LINE ステートメント
- 引用符付きテキスト
- 変数値と出力形式
- ブランク行の書き込み

他の要素: LIBNAME statement

OPTIONS ステートメント

WHERE statement
TITLE statement
自動マクロ変数
SYSDATE

データセット: GROCERY
出力形式: $MGRFMT
出力形式: $DEPTFMT

詳細

このレポートの最後のカスタマイズされた要約には、南東セクタの店舗の全部門に関して売上の最小値と最大値が表示されます。これらの値を決定するため、PROC REPORT はレポートの各行の売上の MIN および MAX 統計量が必要になります。ただし、レポートを簡潔にするため、これらの統計量の表示は非表示になります。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc report data=grocery;

   column manager department sales
          sales=salesmin
          sales=salesmax;

   define manager / order
                    order=formatted
                    format=$mgrfmt.
                    'Manager';
   define department    / order
                    order=internal
                    format=$deptfmt.
                    'Department';

   define sales / analysis sum format=dollar7.2 'Sales';

   define salesmin / analysis min noprint;
   define salesmax / analysis max noprint;

compute after;
      line 'Departmental sales ranged from'
           salesmin dollar7.2  " " 'to' " " salesmax dollar7.2;
   endcomp;

   where sector='se';

   title 'Sales for the Southeast Sector';
   title2 "for &sysdate";
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT は REPORT ウィンドウを使用せずに実行され、出力は開いている出力先に送信されます。

```
proc report data=grocery;
```

レポート列を指定します。 レポートには Manager と Department の列が含まれています。Sales の 3 列も含まれています。列指定 SALES=SALESMIN と SALES=SALESMAX は、Sales の別名を作成します。これらの別名によって 3 つの列のそれぞれに対し別の Sales の定義を使用できます。

```
column manager department sales
       sales=salesmin
       sales=salesmax;
```

並べ替え順序変数を定義します。 DEFINE ステートメントで ORDER オプションが指定されているすべての変数の値によって、レポートの行の順序が決定されます。このレポートで PROC REPORT は最初に Manager の値(これが COLUMN ステートメントの最初の変数のため)、次に Department の値によって行を調整します。ORDER=オプションは変数の並べ替え順序を指定します。このレポートは、Manager の値をフォーマットされた値によって調整し、Department の値を内部値(np1、np2、p1、p2)によって調整します。FORMAT=で、レポートに使用する出力形式を指定します。引用符内のテキストは列ヘッダーを指定します。

```
define manager / order
                 order=formatted
                 format=$mgrfmt.
                 'Manager';
define department    / order
                 order=internal
                 format=$deptfmt.
                 'Department';
```

分析変数を定義します。 レポートの行の分析変数の値は、関連付けられている統計量の値(この場合は Sum)で、行によって表されるすべてのオブザベーションに対して計算されます。詳細レポートの各行は 1 つのオブザベーションだけを表します。そのため、Sum 統計量は、入力データセットのそのオブザベーションの Sales の値と同じになります。

```
define sales / analysis sum format=dollar7.2 'Sales';
```

要約で使用する追加の分析変数を定義します。 これらの DEFINE ステートメントでは、分析変数 Sales の MIN 統計量と MAX 統計量に対し別の列を作成するため COLUMN からの別名を使用します。NOPRINT はこれらの統計量の印刷を行いません。PROC REPORT は列のこれらの値を印刷しませんが、これらの値にアクセスして要約に印刷できます。

```
define salesmin / analysis min noprint;
define salesmax / analysis max noprint;
```

カスタマイズした要約を作成します。 この COMPUTE ステートメントは、レポートの最後に実行する計算ブロックを開始します。LINE ステートメントは引用符で囲まれたテキスト、Salesmin の値(DOLLAR7.2 出力形式)、引用符で囲まれたスペース、引用符で囲

まれたテキスト"to"、スペース、Salesmax(DOLLAR7.2 出力形式)を書き込みます .(プログラムは、変数をその別名によって参照する必要があります)ENDCOMP ステートメントは計算ブロックを終了します。

```
compute after;
      line 'Departmental sales ranged from'
            salesmin dollar7.2  " " 'to' " " salesmax dollar7.2;
   endcomp;
```

処理するオブザベーションを選択します。 WHERE ステートメントは、南東セクタの店舗のオブザベーションのみレポートに選択します。

```
   where sector='se';
```

タイトルを指定します。 SYSDATE は、SAS ジョブまたは SAS セッション開始時に日付を返す自動マクロ変数です。TITLE2 ステートメントは、マクロ変数が解決するように一重引用符ではなく二重引用符を使用します。

```
   title 'Sales for the Southeast Sector';
   title2 "for &sysdate";
run;
```

出力:出力:HTML

アウトプット 55.3 *別名を使用し、同一変数に複数の統計量を求める*

Sales for the Southeast Sector
for 27MAY14

Manager	Department	Sales
Jones	Paper	$40.00
	Canned	$220.00
	Meat/Dairy	$300.00
	Produce	$70.00
Smith	Paper	$50.00
	Canned	$120.00
	Meat/Dairy	$100.00
	Produce	$80.00
Departmental sales ranged from $40.00 to $300.00		

例 4: 1 つの変数に複数の統計量を表示する

要素: PROC REPORT ステートメントオプション
COLUMN ステートメント

積み上げ変数の統計量の指定

DEFINE ステートメントオプション
FORMAT=
GROUP

他の要素: LIBNAME statement
OPTIONS ステートメント
TITLE statement

データセット: GROCERY

出力形式: $MGRFMT

詳細

この例のレポートでは、各マネージャの店舗の売上に対する 6 つの統計量を表示します。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc report data=grocery;

column sector manager (Sum Min Max Range Mean Std),sales;

   define manager / group format=$mgrfmt.;
   define sector / group format=$sctrfmt.;
   define sales / format=dollar11.2 ;

   title 'Sales Statistics for All Sectors';
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションを指定します。

```
proc report data=grocery;
```

レポート列を指定します。 この COLUMN ステートメントは、Sector、Manager、および Sales と関連付けられている 6 つの統計量のそれぞれに対する列を作成します。

```
column sector manager (Sum Min Max Range Mean Std),sales;
```

グループ変数と分析変数を定義します。 このレポートでは、Sector と Manager がグループ変数です。レポートの各詳細行に、グループ変数の値が同じすべてのオブザベーショ

ンに関する情報がまとめられます。FORMAT=で、レポートに使用する出力形式を指定します。

```
define manager / group format=$mgrfmt.;
define sector / group format=$sctrfmt.;
define sales / format=dollar11.2 ;
```

タイトルを指定します。

```
title 'Sales Statistics for All Sectors';
run;
```

出力:出力:HTML

アウトプット 55.4 *1 つの変数に複数の統計量を表示する*

Sales Statistics for All Sectors

		Sum	Min	Max	Range	Mean	Std
Sector	Manager	Sales	Sales	Sales	Sales	Sales	Sales
Northeast	Alomar	$786.00	$86.00	$420.00	$334.00	$196.50	$156.57
	Andrews	$1,045.00	$125.00	$420.00	$295.00	$261.25	$127.83
Northwest	Brown	$598.00	$45.00	$250.00	$205.00	$149.50	$105.44
	Pelfrey	$746.00	$45.00	$420.00	$375.00	$186.50	$170.39
	Reveiz	$1,110.00	$30.00	$600.00	$570.00	$277.50	$278.61
Southeast	Jones	$630.00	$40.00	$300.00	$260.00	$157.50	$123.39
	Smith	$350.00	$50.00	$120.00	$70.00	$87.50	$29.86
Southwest	Adams	$695.00	$40.00	$350.00	$310.00	$173.75	$141.86
	Taylor	$353.00	$50.00	$130.00	$80.00	$88.25	$42.65

例 5: 複数のオブザベーションをレポートの 1 行にまとめる

要素: PROC REPORT ステートメントオプション

COLUMN ステートメント

DEFINE ステートメントオプション
- ANALYSIS
- GROUP
- SUM
- FROMAT=

BREAK ステートメントオプション
- AFTER
- SUMMARIZE
- SUPPRESS

CALL DEFINE ステートメント
COMPUTE ステートメント引数
AFTER
CALL DEFINE ステートメント
LINE ステートメント
引用符付きテキスト
変数値

他の要素: LIBNAME statement
OPTIONS ステートメント
TITLE statement
WHERE statement

データセット: GROCERY

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例では、次を行う要約レポートを作成します。

- Sector と Manager の各組み合わせに関する情報をレポートの 1 行にまとめる
- セクタごとの売上のデフォルト要約を含む
- すべてのセクタの売上のカスタマイズされた要約を含む
- 詳細行と要約行で異なる売上の出力形式を使用する
- カスタマイズされた列ヘッダーを使用する

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc report data=grocery;

   column sector manager sales;

   define sector / group format=$sctrfmt.'Sector';
   define manager / group format=$mgrfmt.'Manager';
   define sales / analysis sum format=comma10.2 'Sales';

   break after sector / summarize
                        style=[font_style=italic]
                        suppress;

   compute after;
      line 'Combined sales for the northern sectors were '
           sales.sum dollar9.2 '.';
   endcomp;

   compute sales;
      if _break_ ne ' ' then
      call define(_col_,"format","dollar11.2");
   endcomp;

   where sector contains 'n';
```

```
   title 'Sales Figures for Northern Sectors';
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT は REPORT ウィンドウを使用せずに実行され、出力は開いている出力先に送信されます。

```
proc report data=grocery;
```

レポート列を指定します。 レポートには、Sector、Manager、Sales の列が含まれます。

```
   column sector manager sales;
```

グループ変数と分析変数を定義します。 このレポートでは、Sector と Manager がグループ変数です。Sales は、Sum 統計量の計算に使用される分析変数です。詳細行はそれぞれ、すべてのグループ変数に対するフォーマットされた値の一意の組み合わせを持つ一連のオブザベーションを表します。各詳細行の Sales の値は、グループのすべてのオブザベーションに対する Sales の合計です。FORMAT=はレポートで使用する出力形式を指定します。DEFINE ステートメントの引用符内のテキストは、列ヘッダーを指定します。

```
   define sector / group format=$sctrfmt.'Sector';
   define manager / group format=$mgrfmt.'Manager';
   define sales / analysis sum format=comma10.2 'Sales';
```

レポートの要約を作成します。 この BREAK ステートメントは、各セクタの最終行の後にデフォルト要約を作成します。SUMMARIZE は Sales の値を要約行に書き込みます。要約値は、セクタごとの値です。セクタの売上では、すべてのマネージャの売上を合計します。SUPPRESS は PROC REPORT が要約行に Sector 値を表示するのを防ぎます。要約行は、イタリック体になります。

```
   break after sector / summarize
                        style=[font_style=italic]
                        suppress;
```

カスタマイズした要約を作成します。 この計算ブロックは、レポートの最後にカスタマイズされた要約を作成します。LINE ステートメントは引用符付きテキストと Sales.sum の値 (DOLLAR9.2 出力形式)を要約に書き込みます。ENDCOMP ステートメントは計算ブロックを終了する必要があります。

```
   compute after;
      line 'Combined sales for the northern sectors were '
           sales.sum dollar9.2 '.';
   endcomp;
```

要約行の出力形式を指定します。 詳細行で、PROC REPORT は定義で指定された出力形式(COMMA10.2)を持つ Sales の値を表示します。計算ブロックは、要約行の現在

の列で使用する代替出力形式を指定します。要約行は、_BREAK_に対するブランク以外の値として識別されます。

```
compute sales;
   if _break_ ne ' ' then
   call define(_col_,"format","dollar11.2");
endcomp;
```

処理するオブザベーションを選択します。 WHERE ステートメントは、北東セクタと北西セクタの店舗のオブザベーションだけをレポートに選択します。TITLE ステートメントはタイトルを指定します。

```
where sector contains 'n';
```

タイトルを指定します。

```
   title 'Sales Figures for Northern Sectors';
run;
```

出力:出力:HTML

アウトプット 55.5 *複数のオブザベーションをレポートの 1 行にまとめる*

Sales Figures for Northern Sectors

Sector	Manager	Sales
Northeast	Alomar	786.00
	Andrews	1,045.00
		$1,831.00
Northwest	Brown	598.00
	Pelfrey	746.00
	Reveiz	1,110.00
		$2,454.00

Combined sales for the northern sectors were $4,285.00

例 6: 変数の値ごとに列を作成する

要素: PROC REPORT ステートメントオプション
SPLIT=
COLUMN ステートメント
積み上げ変数
DEFINE ステートメントオプション
GROUP
ACROSS

ANALYSIS
COMPUTED
SUM
FORMAT=

COMPUTE ステートメント引数
計算変数(*report-item*)
AFTER

LINE ステートメント

ENDCOMP ステートメント

他の要素: LIBNAME statement
OPTIONS ステートメント
TITLE statement
WHERE statement

データセット: GROCERY

出力形式: $SCTRFMT

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例のレポートでは、次を行います。

- 複数のオブザベーションを 1 行にまとめる
- レポートに選択される部門(生鮮品目を販売する部門)の値ごとの列を含む
- 入力データセットにない変数を含む
- 一部ブランクを含むカスタマイズされた列ヘッダーを使用する
- カスタマイズした要約で変数値を使用する

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc report data=grocery split='*';

   column sector manager department,sales perish;

   define sector / group format=$sctrfmt. 'Sector' '';
   define manager / group format=$mgrfmt. 'Manager* ';

   define department / across format=$deptfmt. 'Department';

   define sales / analysis sum format=dollar11.2 ' ';

   define perish / computed format=dollar11.2
               'Perishable*Total';

   compute perish;
      perish=sum(_c3_, _c4_);
   endcomp;

   compute after;
            line  'Combined sales for meat and dairy : '
```

```
            _c3_ dollar11.2 '';
      line  'Combined sales for produce : '
            _c4_ dollar11.2 '';
      line  'Combined sales for all perishables: '
            _c5_ dollar11.2 '';
         endcomp;

   where sector contains 'n'
         and (department='p1' or department='p2');

   title 'Sales Figures for Perishables in Northern Sectors';
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT は REPORT ウィンドウを使用せずに実行され、出力は開いている出力先に送信されます。デフォルトの区切り文字(/)は部門名の一部であるため、SPLIT=は区切り文字をアスタリスク(*)として定義します。

```
proc report data=grocery split='*';
```

レポート列を指定します。 Department と Sales は定義する列のコンテンツを一括で決定できるように、COLUMN ステートメントでカンマで区切られます。項目はそれぞれヘッダーを生成しますが、Sales のヘッダーは定義でブランクに設定されます。Sales が分析変数であるため、その値はこれらの 2 つの変数によって作成されたセルに入力されます。

```
   column sector manager department,sales perish;

   define sector / group format=$sctrfmt. 'Sector' '';
   define manager / group format=$mgrfmt. 'Manager* ';
```

列変数を定義します。 PROC REPORT は、列変数 Department の各フォーマットされた値に対し列と列ヘッダーを作成します。PROC REPORT はこれらの値によって列を並べ替えます。PROC REPORT もこれらのすべての列にわたる列ヘッダーを生成します。Department に対する DEFINE ステートメントの引用符付きテキストは、このヘッダーをカスタマイズします。

```
   define department / across format=$deptfmt. 'Department';
```

分析変数を定義します。 Sales は、Sum 統計量の計算に使用される分析変数です。いずれの場合も、Sales の値は 1 つのグループの 1 つの部門のすべてのオブザベーションに対する Sales の合計です(この場合、値は 1 つのオブザベーションを表します)。

```
   define sales / analysis sum format=dollar11.2 ' ';
```

計算変数を定義します。 COMPUTED オプションは、PROC REPORT が Perish の値を計算する必要があることを示します。Perish と関連付けられている計算ブロックの変数の値を計算します。

```
define perish / computed format=dollar11.2
            'Perishable*Total';
```

計算変数の値を計算します。 この計算ブロックは、肉/乳製品部門と農産物部門の値からの Perish の値を計算します。変数 Sales と Department がこれらの列を一括して定義するため、PROC REPORT への値を名前によって識別することはできません。そのため、割当ステートメントでは列番号を使用して、使用する値を明確に指定します。PROC REPORT は Perish の値が必要となるたび、レポートのその行の 3 番目と 4 番目の列の値を合計します。

```
compute perish;
   perish=sum(_c3_, _c4_);
endcomp;
```

カスタマイズした要約を作成します。 この計算ブロックは、レポートの最後にカスタマイズされた要約を作成します。LINE ステートメントは、指定した列に引用符付きのテキストと変数_C3_、_C4_および_C5_の値(DOLLAR11.2 形式)を書き込みます。

```
compute after;
         line  'Combined sales for meat and dairy : '
         _c3_ dollar11.2 '';
   line  'Combined sales for produce : '
         _c4_ dollar11.2 '';
   line  'Combined sales for all perishables: '
         _c5_ dollar11.2 '';
      endcomp;

where sector contains 'n'
      and (department='p1' or department='p2');
```

タイトルを指定します。

```
   title 'Sales Figures for Perishables in Northern Sectors';
run;
```

出力:出力:HTML

アウトプット 55.6 *変数の値ごとに列を作成する*

Sales Figures for Perishables in Northern Sectors

		Department		
		Meat/Dairy	Produce	
Sector	Manager			Perishable Total
Northeast	Alomar	$190.00	$86.00	$276.00
	Andrews	$300.00	$125.00	$425.00
Northwest	Brown	$250.00	$73.00	$323.00
	Pelfrey	$205.00	$76.00	$281.00
	Reveiz	$600.00	$30.00	$630.00

Combined sales for meat and dairy: $1,545.00
Combined sales for produce: $390.00
Combined sales for all perishables: $1,935.00

例 7: ページごとにカスタマイズされた要約を書き込む

要素: PROC REPORT ステートメントオプション

COLUMN ステートメント

DEFINE ステートメントオプション
- NOPRINT
- GROUP
- COMPUTED
- ANALYSIS

COMPUTE ステートメント引数
- BEFORE
- AFTER
- BEFORE _PAGE_
- STYLE=
- TEXT=

BREAK ステートメントオプション
- PAGE
- SUMMARIZE
- AFTER
- STYLE=

LINE ステートメント

他の要素: LIBNAME statement

OPTIONS ステートメント
TITLE statement

データセット: GROCERY
出力形式: $SCTRFMT
出力形式: $MGRFMT
出力形式: $DEPTFMT

詳細

この例のレポートでは、各店舗に対する 1 日の売上の記録を表示します。行は、1 つの店舗に関するすべての情報がまとめられ、店舗ごとの情報が新しいページで開始されるように調整されます。一部の変数が列に表示されます。その他の変数は、セクタと店舗のマネージャを識別するページヘッダーにのみ表示されます。

各ページの上部に表示されるヘッダーは、COMPUTE ステートメントの_PAGE_引数によって作成されます。

Profit は、Sales と Department の値に基づく計算変数です。

ページの下部に表示されるテキストは、店舗の Sales の合計によって異なります。ここでは、レポートの最初の 2 ページだけを表示します。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc report data=grocery;

   title 'Sales for Individual Stores';

   column sector manager department sales Profit;

   define sector / group noprint;
   define manager / group noprint;
   define profit / computed format=dollar11.2;
   define sales / analysis sum format=dollar11.2;
   define department / group format=$deptfmt.;

   compute profit;
      if department='np1' or department='np2'
         then profit=0.4*sales.sum;
      else profit=0.25*sales.sum;
   endcomp;

    compute before _page_ / style={just=left};
      line sector $sctrfmt. ' Sector';
      line 'Store managed by ' manager $mgrfmt.;
         endcomp;

   break after manager / summarize style=[font_style=italic] page;

   compute after manager;

      length text $ 35;

      if sales.sum lt 500 then
         text='Sales are below the target region.';
      else if sales.sum ge 500 and sales.sum lt 1000 then
```

```
      text='Sales are in the target region.';
   else if sales.sum ge 1000 then
      text='Sales exceeded goal!';
   line text $35.;
endcomp;
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT は REPORT ウィンドウを使用せずに実行され、出力は開いている出力先に送信されます。

```
proc report data=grocery;
```

タイトルを指定します。

```
title 'Sales for Individual Stores';
```

レポート列を指定します。 レポートには、Sector、Manager、Department、Sales、Profit の列が含まれていますが、NOPRINT オプションは、Sector と Manager の列の印刷を行いません。ページヘッダー(後でプログラムで作成)には、それらの値が含まれています。これらの変数値をページヘッダーに入力するには、Sector と Manager が COLUMN ステートメントにある必要があります。

```
column sector manager department sales Profit;
```

グループ変数、計算変数、分析変数を定義します。 このレポートでは、Sector、Manager、Department はグループ変数です。レポートの各詳細行に、グループ変数の値が同じすべてのオブザベーションに関する情報がまとめられます。Profit は値がプログラムの次のセクションで計算される計算変数です。FORMAT=で、レポートに使用する出力形式を指定します。

```
define sector / group noprint;
define manager / group noprint;
define profit / computed format=dollar11.2;
define sales / analysis sum format=dollar11.2;
define department / group format=$deptfmt.;
```

計算変数を計算します。 利益は Sales のパーセントとして計算されます。非生鮮品目の場合、利益は販売価格の 40%です。生鮮品目の場合、利益は 25%です。計算ブロックでは、計算する変数と統計量の両方を識別する複合名(Sales.sum)を持つ変数 Sales を参照する必要があります。

```
compute profit;
   if department='np1' or department='np2'
      then profit=0.4*sales.sum;
   else profit=0.25*sales.sum;
endcomp;
```

カスタマイズされたページヘッダーを作成します。 この計算ブロックは、PROC REPORT がタイトルを書き込んだ後に各ページの上部で実行します。現在のマネージャの店舗に対するページヘッダーを書き込みます。LEFT オプションは LINE ステートメントのテキストを左寄せにします。LINE ステートメントは、変数名の直後に指定される出力形式を持つ変数値を書き込みます。

```
compute before _page_ / style={just=left};
  line sector $sctrfmt. ' Sector';
  line 'Store managed by ' manager $mgrfmt.;
     endcomp;
```

レポートの要約を作成します。 この BREAK ステートメントは、各マネージャの最終行の後にデフォルト要約を作成します。SUMMARIZE は Sales の値(分析変数または計算変数のみ)を要約行に書き込みます。PAGE オプションは、前の計算ブロックで作成されるページヘッダーが常に適切なマネージャに関連するように、各デフォルト要約の後に新しいページを開始します。STYLE=オプションで、要約情報をイタリック体に設定します。

```
break after manager / summarize style=[font_style=italic] page;
```

カスタマイズした要約を作成します。 この計算ブロックは、各マネージャに対する最後の詳細行の後に表示される条件テキストをカスタマイズされた要約に置きます。

```
compute after manager;
```

カスタマイズされた要約テキストの長さを指定します。 LENGTH ステートメントは長さ 35 を一時変数 TEXT に割り当てます。こうした特殊な場合、最長バージョンが最初の IF/THEN ステートメントで表示されるため、LENGTH ステートメントは不要です。ただし LENGTH ステートメントにより、条件ステートメントの順序が変わっても、TEXT は最長バージョンを含めるのに十分な長さになります。

```
   length text $ 35;
```

カスタマイズされた要約テキストに対し条件付きロジックを指定します。 LINE ステートメントは条件ステートメント(IF-THEN、IF-THEN/ELSE、SELECT)で使用できません。LINE ステートメントは、PROC REPORT が計算ブロックのその他すべてのステートメントを実行するまで実行されないためです。これらの IF-THEN/ELSE ステートメントは、要約行の Sales.sum の値に基づいて値を TEXT に割り当てます。LINE は値が発生する変数を書き込みます

```
   if sales.sum lt 500 then
      text='Sales are below the target region.';
   else if sales.sum ge 500 and sales.sum lt 1000 then
      text='Sales are in the target region.';
   else if sales.sum ge 1000 then
      text='Sales exceeded goal!';
   line text $35.;
endcomp;
run;
```

出力:出力:HTML

アウトプット 55.7 *ページごとにカスタマイズされた要約を書き込む*

Sales for Individual Stores

Northeast Sector
Store managed by Alomar

Department	Sales	Profit
Canned	$420.00	$168.00
Meat/Dairy	$190.00	$47.50
Paper	$90.00	$36.00
Produce	$86.00	$21.50
	$786.00	*$196.50*

Sales are in the target region.

Sales for Individual Stores

Northeast Sector
Store managed by Andrews

Department	Sales	Profit
Canned	$420.00	$168.00
Meat/Dairy	$300.00	$75.00
Paper	$200.00	$80.00
Produce	$125.00	$31.25
	$1,045.00	*$261.25*

Sales exceeded goal!

例 8: パーセントの計算

要素: PROC REPORT ステートメントオプション

COLUMN ステートメント引数
- PCTSUM
- SUM

DEFINE ステートメントオプション
COMPUTED
STYLE(COLUMN)=
GROUP

COMPUTE ステートメントオプション
CHAR
LENGTH=

RBREAK ステートメントオプション
SUMMARIZE
STYLE=

他の要素: LIBNAME statement
OPTIONS ステートメント
TITLE statement

データセット: GROCERY

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例の要約レポートでは、店舗ごとの合計売上と、これらの売上の全店舗の売上に対するパーセントを表示します。これらの列にはそれぞれ独自のヘッダーがあります。また、1 つのヘッダーはすべての列にわたります。

レポートには、計算文字変数 COMMENT が含まれます。これは、売上率が非常に高い店舗のフラグをオンにします。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc report data=grocery;
   title;

   column ('Individual Store Sales as a Percent of All Sales'
            sector manager sales,(sum pctsum) comment);

   define manager / group
                    format=$mgrfmt.;
   define sector / group
                   format=$sctrfmt.;
   define sales / format=dollar11.2
                  '';
   define sum / format=dollar9.2
                'Total Sales';

   define pctsum / 'Percent of Sales' format=percent6.;
   define comment / computed style(column)=[cellwidth=2.5in];

   compute comment / char length=40;

      if sales.pctsum gt .15 and _break_ = ' '
      then comment='Sales substantially above expectations.';
      else comment=' ';
   endcomp;
```

```
   rbreak after / summarize style=[font_style=italic];
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションを指定します。 データセットとその他のオプションを指定します。

```
proc report data=grocery;
   title;
```

レポート列を指定します。 COLUMN ステートメントは、引用符内のテキストをヘッダーとして使用します。ヘッダーは、ヘッダーを含む一対のかっこにすべて含まれているため、レポートの全列にわたります。COLUMN ステートメントは、2 つの統計量 Sum と Pctsum を売上と関連付けます。Sum 統計量は、レポートの行に含まれるすべてのオブザベーションに対する Sales の値を合計します。Pctsum 統計量は、集計する Sales のレポートのすべてのオブザベーションに対するパーセントを示します。

```
column ('Individual Store Sales as a Percent of All Sales'
         sector manager sales,(sum pctsum) comment);
```

グループ列と分析列を定義します。 このレポートでは、Sector と Manager がグループ変数です。詳細行はそれぞれ、すべてのグループ変数に対するフォーマットされた値の一意の組み合わせを持つ一連のオブザベーションを表します。Sales はデフォルトで、Sum 統計量の計算に使用される分析変数です。ただし、統計量は COLUMN ステートメントで Sales と関連付けられているため、これらの統計量はデフォルトより優先されます。FORMAT=で、レポートに使用する出力形式を指定します。引用符の間のテキストは列ヘッダーを指定します。

```
define manager / group
                 format=$mgrfmt.;
define sector / group
                format=$sctrfmt.;
define sales / format=dollar11.2
               '';
define sum / format=dollar9.2
             'Total Sales';
```

パーセント列と計算列を定義します。 Pctsum の DEFINE ステートメントで、列のヘッダーと出力形式を指定します。PERCENT.出力形式は、Pctsum の値を小数ではなく、パーセントで表示します。COMMENT の DEFINE ステートメントは、計算変数を定義してそれを列に割り当てます。

```
define pctsum / 'Percent of Sales' format=percent6.;
define comment / computed style(column)=[cellwidth=2.5in];
```

計算変数を計算します。 COMPUTE ステートメントのオプションは、COMMENT を長さ 40 の文字変数として定義します。

```
    compute comment / char length=40;
```

計算変数に対し条件付きロジックを指定します。 売上が全店舗の売上の 15%を超えたすべての店舗について、この計算ブロックは **Sales substantially above expectations** というコメントを作成します。レポートの要約行では、Pctsum の値は 100 となります。ただし、この行を例外的な売上があるとしてフラグ設定することは適切ではありません。自動変数_BREAK_によって詳細行と要約行が区別されます。詳細行では、_BREAK_の値はブランクです。THEN ステートメントは Pctsum の値が 0.15 を超える詳細行でのみ実行します。

```
      if sales.pctsum gt .15 and _break_ = ' '
      then comment='Sales substantially above expectations.';
      else comment=' ';
  endcomp;
```

レポート要約を作成します。 この RBREAK ステートメントは、レポートの最後にデフォルト要約を作成します。SUMMARIZE は要約行に Sales.sum と Sales.pctsum の値を書き込みます。STYLE=で、要約行をイタリック体に設定します。

```
    rbreak after / summarize style=[font_style=italic];
run;
```

出力:出力:HTML

アウトプット 55.8 *パーセントの計算*

Individual Store Sales as a Percent of All Sales

Sector	Manager	Total Sales	Percent of Sales	comment
Northeast	Alomar	$786.00	12%	
	Andrews	$1,045.00	17%	Sales substantially above expectations.
Northwest	Brown	$598.00	9%	
	Pelfrey	$746.00	12%	
	Reveiz	$1,110.00	18%	Sales substantially above expectations.
Southeast	Jones	$630.00	10%	
	Smith	$350.00	6%	
Southwest	Adams	$695.00	11%	
	Taylor	$353.00	6%	
		$6,313.00	*100%*	

例 9: PROC REPORT での欠損値の処理法

要素: PROC REPORT ステートメントオプション
MISSING
COLUMN ステートメント
N 統計量
DEFINE ステートメントオプション
STYLE(COLUMN)=
GROUP
RBREAK ステートメントオプション

AFTER
SUMMARIZE
STYLE=

他の要素: LIBNAME statement
OPTIONS ステートメント
TITLE statement

出力形式: $MGRFMT

詳細

この例では、MISSING オプションを使用した場合と使用しない場合の、PROC REPORT でのグループ(または順序、段組み)変数の欠損値の処理方法を示します。行ごとに N の値を比較し、レポートの最後のデフォルト要約の合計を比較すると、レポートの違いは明らかです。

プログラム – 欠損値のないデータセット

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

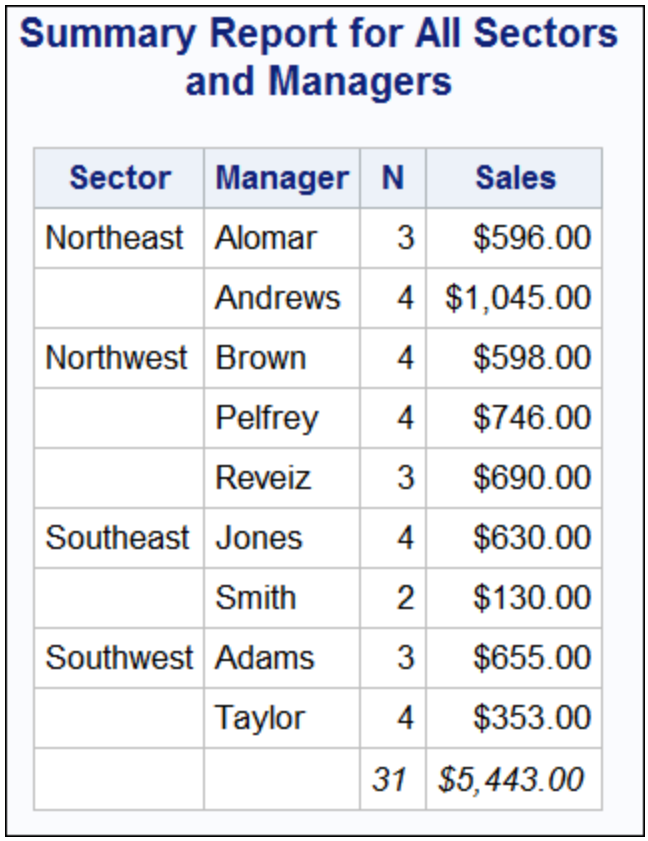
data grocmiss;
   input Sector $ Manager $ Department $ Sales @@;
datalines;
se 1 np1 50    .  1 p1 100   se . np2 120   se 1 p2 80
se 2 np1 40    se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60    nw 3 p1 600   .  3 np2 420   nw 3 p2 30
nw 4 np1 45    nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45    nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53    sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
.  . np1 40    sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90    ne . p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;

proc report data=grocmiss;

   column sector manager N sales;

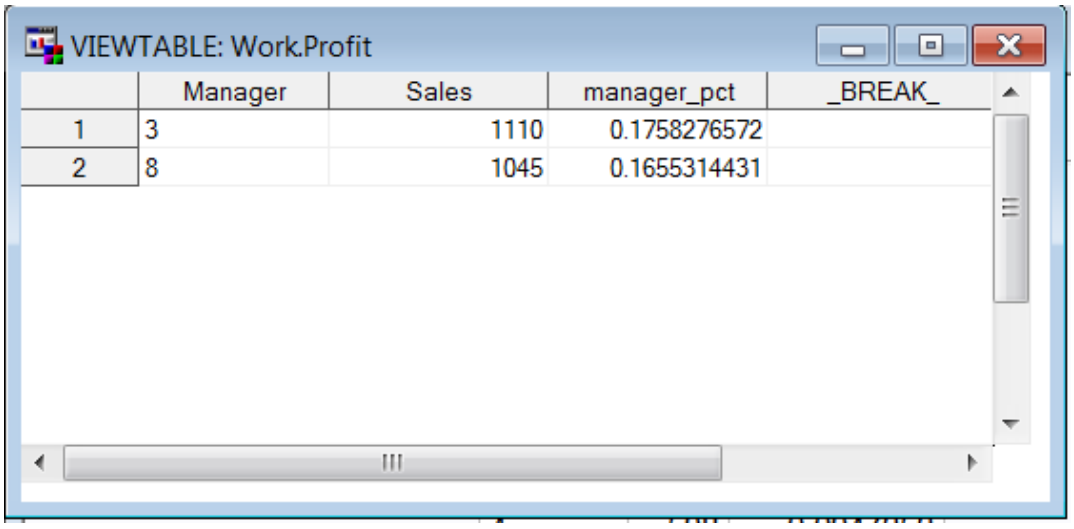
   define sector / group format=$sctrfmt.;
   define manager / group format=$mgrfmt.;
   define sales / format=dollar9.2;

   rbreak after / summarize style=[fontstyle=italic];

   title 'Summary Report for All Sectors and Managers';
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

GROCMISS データセットを作成します。 GROCMISS は、Sector、Manager、またはその両方の欠損値を含む一部のオブザベーションを含むという以外は、GROCERY と同じです。

```
data grocmiss;
   input Sector $ Manager $ Department $ Sales @@;
datalines;
se 1 np1 50    .  1 p1 100   se . np2 120   se 1 p2 80
se 2 np1 40    se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60    nw 3 p1 600   .  3 np2 420   nw 3 p2 30
nw 4 np1 45    nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45    nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53    sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
.  . np1 40    sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90    ne . p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT は REPORT ウィンドウを使用せずに実行され、出力は開いている出力先に送信されます。

```
proc report data=grocmiss;
```

レポート列を指定します。 レポートには、Sector、Manager、N 統計量、Sales の列が含まれています。

```
   column sector manager N sales;
```

グループ変数と分析変数を定義します。 このレポートでは、Sector と Manager がグループ変数です。Sales はデフォルトで、Sum 統計量の計算に使用される分析変数です。詳細行はそれぞれ、すべてのグループ変数に対するフォーマットされた値の一意の組み合わせを持つ一連のオブザベーションを表します。各詳細行の Sales の値は、グループのすべてのオブザベーションに対する Sales の合計です。この PROC REPORT ステップで、プロシジャにはグループ変数の値が欠損値であるオブザベーションは含まれません。FORMAT=は、レポートで使用する出力形式を指定します。

```
   define sector / group format=$sctrfmt.;
   define manager / group format=$mgrfmt.;
   define sales / format=dollar9.2;
```

レポートの要約を作成します。 この RBREAK ステートメントは、レポートの最後にデフォルト要約を作成します。SUMMARIZE は要約行に N と Sales.sum の値を書き込みます。STYLE=で、要約行をイタリック体に設定します。

```
   rbreak after / summarize style=[fontstyle=italic];
```

タイトルを指定します。

```
   title 'Summary Report for All Sectors and Managers';
run;
```

出力:出力:HTML

アウトプット 55.9 *欠損値を含まない出力*



Summary Report for All Sectors and Managers

Sector	Manager	N	Sales
Northeast	Alomar	3	$596.00
	Andrews	4	$1,045.00
Northwest	Brown	4	$598.00
	Pelfrey	4	$746.00
	Reveiz	3	$690.00
Southeast	Jones	4	$630.00
	Smith	2	$130.00
Southwest	Adams	3	$655.00
	Taylor	4	$353.00
		31	*$5,443.00*

プログラム – 欠損値があるデータセット

```
proc report data=grocmiss missing;
   column sector manager N sales;
   define sector / group format=$sctrfmt.;
   define manager / group format=$mgrfmt.;
   define sales / format=dollar9.2;
   rbreak after / summarize style=[fontstyle=italic];
run;
```

プログラムの説明

欠損値を含めます。 2 番目の PROC REPORT ステップの MISSING オプションには、グループ変数の値が欠損値のオブザベーションが含まれます。

```
proc report data=grocmiss missing;
   column sector manager N sales;
   define sector / group format=$sctrfmt.;
   define manager / group format=$mgrfmt.;
   define sales / format=dollar9.2;
   rbreak after / summarize style=[fontstyle=italic];
run;
```

出力:出力:HTML

アウトプット 55.10 *欠損値を含む出力*

Summary Report for All Sectors and Managers

Sector	Manager	N	Sales
		1	$40.00
	Reveiz	1	$420.00
	Smith	1	$100.00
Northeast		1	$190.00
	Alomar	3	$596.00
	Andrews	4	$1,045.00
Northwest	Brown	4	$598.00
	Pelfrey	4	$746.00
	Reveiz	3	$690.00
Southeast		1	$120.00
	Jones	4	$630.00
	Smith	2	$130.00
Southwest	Adams	3	$655.00
	Taylor	4	$353.00
		36	*$6,313.00*

例 10: 出力データセットの作成と計算変数の保存

要素: PROC REPORT ステートメントオプション
- OUT=

COLUMN

DEFINE ステートメントオプション
- GROUP
- COMPUTED
- ANALYSIS
- SUM

COMPUTE

他の要素: データセットオプション
- WHERE=

LIBNAME statement
OPTIONS ステートメント
TITLE

データセット: GROCERY

出力形式: $MGRFMT

詳細

この例では、出力データセットの作成時に WHERE 処理を使用します。この方法を使用すると、複数のオブザベーションを 1 つの行にまとめた後に WHERE 処理を実行できます。

注: この方法が必要である理由は、分析変数の結果に関しては部分集合を抽出できないためです。PROC REPORT によって計算された値に関しては部分集合を抽出できません。

最初の PROC REPORT ステップでは、各行が 1 人のマネージャの入力データセットの全オブザベーションを表すレポートを作成します。2 番目の PROC REPORT ステップでは、出力データセットからのレポートを作成します。

出力データセットを作成するためのプログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc report data=grocery
            out=profit( where=(sales gt 1000 and _break_='') );
   column manager sales manager_pct;

   define manager / group;
   define manager_pct / computed;
   compute before;
      total_sales = sales.sum;
   endcomp;
   compute manager_pct;
      manager_pct = sales.sum /total_sales;
   endcomp;
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

レポートオプションと列を指定します。 デフォルトでは、PROC REPORT は REPORT ウィンドウを使用せずに実行され、出力は開いている出力先に送信されます。OUT=は出力データセット PROFIT を作成します。出力データセットには、レポートの各列(Manager、Sales および計算列 manager_pct)の変数のほか、この例では使用されていない変数

_BREAK_の変数が含まれています。データセット内の各オブザベーションは、レポートの行を表します。Manager がグループ変数、Sales が Sum 統計量の計算に使用される分析変数であるため、レポートの各行(および出力データセット内の各オブザベーション)は、入力データセットからの複数のオブザベーションを表します。特に、出力データセットの Sales の各値は、マネージャに対する Sales のすべての値の合計です。OUT=オプションの WHERE=データセットオプションは、PROC REPORT が出力データセットを作成するときにこれらの行をフィルタします。売上が$1,000 を超えるオブザベーションだけが出力データセットのオブザベーションになります。

```
proc report data=grocery
            out=profit( where=(sales gt 1000 and _break_='') );
   column manager sales manager_pct;
```

グループ変数および分析変数を定義して、売上のパーセント値を計算します。 全体の合計額が一時変数、total_sales に配置され、売上のパーセント値が計算されます。

```
   define manager / group;
   define manager_pct / computed;
   compute before;
      total_sales = sales.sum;
   endcomp;
   compute manager_pct;
      manager_pct = sales.sum /total_sales;
   endcomp;
run;
```

出力:出力:HTML

PROC REPORT によって作成されるデータセットを次に示します。このデータセットは、2 番目の PROC REPORT ステップで入力セットとして使用されます。

アウトプット 55.11 *出力データセット PROFIT*



	Manager	Sales	manager_pct	_BREAK_
1	3	1110	0.1758276572	
2	8	1045	0.1655314431	

出力データセットを使用するプログラム

```
proc report data=profit;
   column manager sales manager_pct;
   define manager / group format=$mgrfmt.;
   define sales / analysis sum format=dollar11.2;
   define manager_pct /'Percent of Total Sales' format=percent8.2;
```

```
   title 'Managers with Daily Sales';
   title2 'of over';
   title3 'One Thousand Dollars';
run;
```

プログラムの説明

レポートオプションと列を指定し、グループ列と分析列を定義し、タイトルを指定します。
DATA=は最初の PROC REPORT ステップからの出力データセットをこのレポートの入力データセットとして指定します。TITLE ステートメントはレポートのタイトルを指定します。

```
proc report data=profit;
   column manager sales manager_pct;
   define manager / group format=$mgrfmt.;
   define sales / analysis sum format=dollar11.2;
   define manager_pct /'Percent of Total Sales' format=percent8.2;
   title 'Managers with Daily Sales';
   title2 'of over';
   title3 'One Thousand Dollars';
run;
```

出力:出力:HTML

アウトプット 55.12 *出力データセットに基づくレポート*

Managers with Daily Sales
of over
One Thousand Dollars

Manager	Sales	Percent of Total Sales
Andrews	$1,045.00	16.55%
Reveiz	$1,110.00	17.58%

例 11: 出力形式を使用し、グループを作成する

要素: PROC REPORT ステートメント

DEFINE ステートメントオプション
- GROUP
- ORDER
- ACROSS
- ANALYSIS
- SUM
- STYLE=

COMPUTE ステートメントオプション
- AFTER
- STYLE=

LINE ステートメント

他の要素: FORMAT プロシジャ
LIBNAME statement
OPTIONS ステートメント
TITLE

データセット: GROCERY

出力形式: $MGRFMT

詳細

この例では、出力形式を使用して PROC REPORT が作成するグループ数を制御する方法を示します。プログラムでは、4 つの部門を 2 つの種類 生鮮と非生鮮のいずれかに分類する Department の出力形式を作成します。結果として、Department が列変数の場合、PROC REPORT は 4 つの列ではなく 2 つの列だけを作成します。列ヘッダーは、変数のフォーマットされた値です。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc format;
   value $perish 'p1','p2'='Perishable'
                 'np1','np2'='Nonperishable';
run;

proc report data=grocery;


   column manager department,sales sales;

   define manager / group order=formatted
                    format=$mgrfmt.;
   define department / across order=formatted
                 format=$perish. '';

   define sales / analysis sum
                  format=dollar9.2 style=[cellwidth=13];

   compute after / style=[font_style=italic];
      line 'Total sales for these stores were: '
           sales.sum dollar9.2;
   endcomp;

title 'Sales Summary for All Stores';
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

$PERISH.出力形式を作成します。 PROC FORMAT は Department の出力形式を作成します。この変数にはデータセットに 4 つの異なる値がありますが、出力形式には 2 つの値だけがあります。

```
proc format;
   value $perish 'p1','p2'='Perishable'
                 'np1','np2'='Nonperishable';
run;
```

レポートオプションを指定します。 デフォルトでは、REPORT ウィンドウなしで REPORT プロシジャが実行され、その出力が空いている出力先に送信されます。

```
proc report data=grocery;
```

レポート列を指定します。 Department と Sales は定義する列のコンテンツを一括で決定できるように、COLUMN ステートメントでカンマで区切られます。Sales が分析変数であるため、その値はこれらの 2 つの変数によって作成されたセルに入力されます。レポートには、単独の Manager の列と Sales の列(すべての部門の売上)も含まれます。

```
      column manager department,sales sales;
```

グループ変数と列変数を定義します。 Manager はグループ変数です。レポートの各詳細行には、Manager の値が同じすべてのオブザベーションに関する情報がまとめられます。Department は列変数です。PROC REPORT は Department のフォーマットされた値に対してそれぞれ列と列ヘッダーを作成します。ORDER=FORMATTED は Manager と Department の値をそのフォーマットされた値に従ってアルファベット順に調整します。FORMAT=は使用する出力形式を指定します。Department の定義のブランクの引用符は、ブランクの列ヘッダーを指定します。そのため、ヘッダーはすべての部門にわたりません。ただし、PROC REPORT は Department の出力形式が定義された値を使用して、個別の部門に対しそれぞれ列ヘッダーを作成します。

```
      define manager / group order=formatted
                       format=$mgrfmt.;
      define department / across order=formatted
                   format=$perish. '';
```

分析変数を定義します。 Sales は、Sum 統計量の計算に使用される分析変数です。Sales は COLUMN ステートメントで二度記述され、同じ定義が二度とも適用されます。FORMAT=はレポートで使用する出力形式を指定します。STYLE=は列幅を指定します。Department と Sales が作成する列の列ヘッダーは、Department のヘッダーと Sales の(デフォルト)ヘッダーの組み合わせです。

```
      define sales / analysis sum
                     format=dollar9.2 style=[cellwidth=13];
```

カスタマイズした要約を作成します。 この COMPUTE ステートメントは、レポートの最後にカスタマイズされた要約を作成する計算ブロックを開始します。LINE ステートメントは、引用符付きテキストと Sales.sum の値(DOLLAR9.2 出力形式)を要約に置きます。STYLE=オプションで、要約情報をイタリック体に設定します。ENDCOMP ステートメントは計算ブロックを終了する必要があります。

```
      compute after / style=[font_style=italic];
```

```
        line 'Total sales for these stores were: '
             sales.sum dollar9.2;
    endcomp;
```

タイトルを指定します。

```
title 'Sales Summary for All Stores';
run;
```

出力:出力:HTML

アウトプット 55.13 *出力形式を使用し、グループを作成する*

Sales Summary for All Stores

	Nonperishable	Perishable	
Manager	**Sales**	**Sales**	**Sales**
Adams	$265.00	$430.00	$695.00
Alomar	$510.00	$276.00	$786.00
Andrews	$620.00	$425.00	$1,045.00
Brown	$275.00	$323.00	$598.00
Jones	$260.00	$370.00	$630.00
Pelfrey	$465.00	$281.00	$746.00
Reveiz	$480.00	$630.00	$1,110.00
Smith	$170.00	$180.00	$350.00
Taylor	$173.00	$180.00	$353.00
Total sales for these stores were: $6,313.00			

例 12: マルチラベル出力形式の使用

要素: PROC REPORT ステートメント
COLUMN ステートメント
DEFINE ステートメントオプション
- FORMAT=
- GROUP
- MLF
- ORDER=
- PRELOADFMT
- MEAN

他の要素: FORMAT プロシジャオプション

MULTILABEL
NOTSORTED

LIBNAME statement

OPTIONS ステートメント

TITLE statement

WHERE statement

詳細

この例では、マルチラベル出力形式を使用して、次を行うレポートを作成します。

- PROC FORMAT の VALUE ステートメントでマルチラベル出力形式を指定する方法を示す
- DEFINE ステートメントで MLF オプションを使用して、マルチラベル出力形式処理をアクティブにする方法を示す
- NOTSORTED および PRELOADFMT で、PROC FORMAT に示されている並べ替え順序を使用する方法を示す

プログラム

```
proc format;
  value agelfmt (multilabel notsorted)
     11='11'
     12='12'
     13='13'
     14='14'
     15='15'
     16='16'
     11-12='11 or 12'
     13-14='13 or 14'
     15-16='15 or 16'
     low-13='13 and below'
     14-high='14 and above' ;
run;

ods html file="example.html";

title "GROUP Variable with MLF Option";

proc report data=sashelp.class;

   col age ('Mean' height weight);

   define age / group mlf format=agelfmt. 'Age Group' order=data preloadfmt;
   define height / mean format=6.2 'Height (in.)';
   define weight / mean format=6.2 'Weight (lbs.)';
run;
```

プログラムの説明

AGE1FMT.出力形式を作成します。 FORMAT プロシジャは、MULTILABEL オプションを使用して年齢に対するマルチラベル出力形式を作成します。マルチラベル出力形式は、複数のラベルを同一の値に割り当てることができる出力形式です。各値は発生する各範囲のテーブルに表されます。NOTSORTED を使用して、その並べ替え順序が維持されるようにします。

```
proc format;
  value agelfmt (multilabel notsorted)
     11='11'
     12='12'
     13='13'
     14='14'
     15='15'
     16='16'
     11-12='11 or 12'
     13-14='13 or 14'
     15-16='15 or 16'
     low-13='13 and below'
     14-high='14 and above' ;
run;
```

ODS HTML 出力ファイル名を指定します。

```
ods html file="example.html";
```

タイトルを指定します。

```
title "GROUP Variable with MLF Option";
```

レポートオプションを指定します。 デフォルトでは、REPORT ウィンドウなしで REPORT プロシジャが実行され、その出力が空いている出力先に送信されます。

```
proc report data=sashelp.class;
```

レポート列を指定します。 レポートには、Age、Height、Weight の列が含まれています。Height と Weight の Mean が計算されます。

```
   col age ('Mean' height weight);
```

変数を定義します。 AGE はグループ変数です。AGE 変数は MLF オプションを使用してマルチラベル出力形式処理をアクティブにします。MLF は、グループ変数および列変数のみとともに使用する必要があります。FORMAT=はマルチラベル出力形式 AGE1FMT.が使用されるように指定します。PROC FORMAT で、ORDER=DATA オプションを NOTSORTED オプションとともに使用することにより、望ましい並べ替え順序を維持します。レポートの各詳細行に、グループ変数の値が同じすべてのオブザベーションに関する情報がまとめられます。平均値が HEIGHT と WEIGHT の列値に対して計算されます。PRELOADFMT オプションで、変数に対して出力形式があらかじめ読み込まれるように指定します。

```
   define age / group mlf format=agelfmt. 'Age Group' order=data preloadfmt;
   define height / mean format=6.2 'Height (in.)';
   define weight / mean format=6.2 'Weight (lbs.)';
run;
```

出力:出力:HTML

アウトプット 55.14 *マルチラベル出力形式の使用*

GROUP Variable with MLF Option

	Mean	
Age Group	Height (in.)	Weight (lbs.)
11	54.40	67.75
12	59.44	94.40
13	61.43	88.67
14	64.90	101.88
15	65.63	117.38
16	72.00	150.00
11 or 12	58.00	86.79
13 or 14	63.41	96.21
15 or 16	66.90	123.90
13 and below	59.03	87.35
14 and above	66.01	114.11

例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する

要素: PROC REPORT ステートメントオプション STYLE=オプション
STYLE=
DEFINE ステートメントオプション
ORDER
STYLE=
BREAK ステートメントオプション
AFTER
SUMMARIZE
COMPUTE ステートメントオプション
AFTER
STYLE=
CALL DEFINE
STYLE=
LINE ステートメント

他の要素: FORMAT プロシジャ
LIBNAME statement
OPTIONS ステートメント

TITLE statement
WHERE statement
ODS PDF ステートメント
ODS RTF ステートメント

データセット: GROCERY

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例では、HTML、PDF および RTF ファイルを作成し、PROC REPORT ステートメントでレポートの各場所に対しスタイル要素を設定します。次に、その他のステートメントでスタイル要素を指定することにより、これらの設定のうち一部を無効にします。詳細については、“テーブル領域のスタイル要素とスタイル属性 ” (1665 ページ)を参照してください。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods pdf file='external-PDF-file';
ods rtf file='external-RTF-file';

proc report data=grocery

style(report)=[cellspacing=5 borderwidth=10 bordercolor=blue]

style(header)=[color=yellow
               fontstyle=italic fontsize=6]

style(column)=[color=moderate brown
               fontfamily=helvetica fontsize=4]

style(lines)=[color=white backgroundcolor=black
              fontstyle=italic fontweight=bold fontsize=5]

style(summary)=[color=cx3e3d73 backgroundcolor=cxaeadd9
                fontfamily=helvetica fontsize=3 textalign=r];

   column manager department sales;

   define manager / order
                    order=formatted
                    format=$mgrfmt.
                    'Manager'

                    style(header)=[color=white
                                   backgroundcolor=black];

   define department / order
                       order=internal
                       format=$deptfmt.
                       'Department'

                       style(column)=[fontstyle=italic];

   break after manager / summarize;
```

```
   compute after manager
           / style=[fontstyle=roman fontsize=3 fontweight=bold
             backgroundcolor=white color=black];

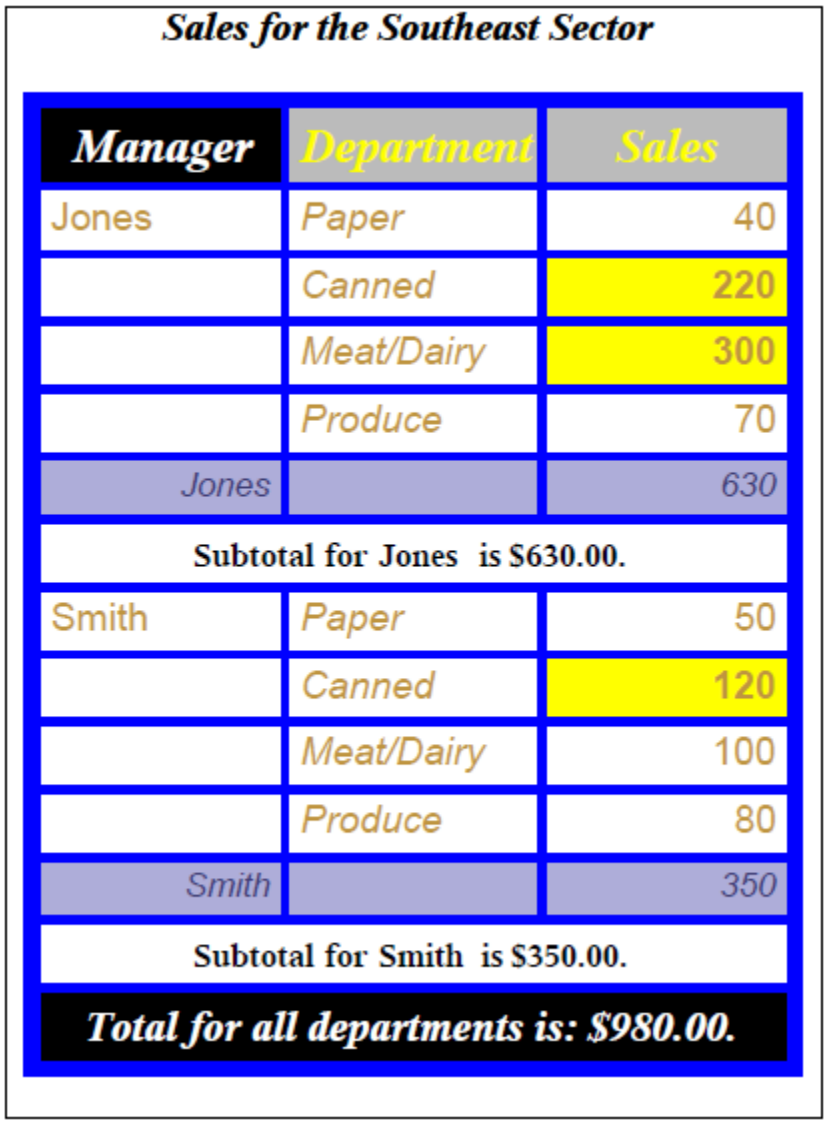
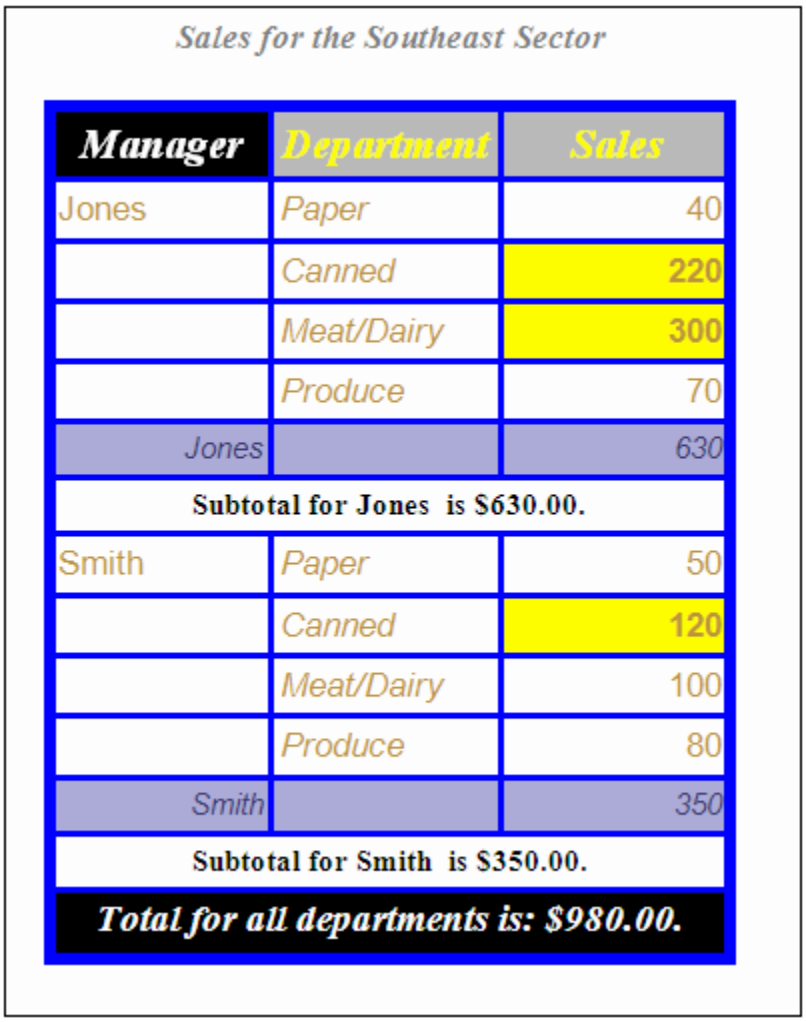
      line 'Subtotal for ' manager $mgrfmt. 'is '
           sales.sum dollar7.2 '.';
   endcomp;

   compute sales;
      if sales.sum>100 and _break_=' ' then
      call define(_col_, "style",
                  "style=[backgroundcolor=yellow
                          fontfamily=helvetica
                          fontweight=bold]");
   endcomp;

   compute after;
       line 'Total for all departments is: '
            sales.sum dollar7.2 '.';
   endcomp;

   where sector='se';

   title 'Sales for the Southeast Sector';
run;

ods pdf close;
ods rtf close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力ファイル名を指定します。 複数の ODS 出力先を開くことによって、一度の実行で複数の出力ファイルを作成できます。HTML 出力はデフォルトで作成されます。ODS PDF ステートメントは出力を Portable Document Format (PDF)で作成します。ODS RTF ステートメントは出力を Rich Text Format (RTF)で作成します。PROC REPORT からの出力は、これらのファイルにそれぞれ行われます。

```
ods pdf file='external-PDF-file';
ods rtf file='external-RTF-file';
```

レポートオプションを指定します。 デフォルトでは、REPORT ウィンドウなしで PROC REPORT が実行されます。この場合、SAS は出力を従来のプロシジャ出力、HTML Body ファイル、RTF ファイル、PDF ファイルに書き込みます。

```
proc report data=grocery
```

レポートに対しスタイル属性を指定します。 この STYLE=オプションは、レポートの構造部分に対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(report)=[cellspacing=5 borderwidth=10 bordercolor=blue]
```

列ヘッダーに対しスタイル属性を指定します。この STYLE=オプションは、すべての列ヘッダーに対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(header)=[color=yellow
               fontstyle=italic fontsize=6]
```

レポート列に対しスタイル属性を指定します。この STYLE=オプションは、すべての列のすべてのセルに対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(column)=[color=moderate brown
               fontfamily=helvetica fontsize=4]
```

計算ブロック行に対しスタイル属性を指定します。この STYLE=オプションは、すべての計算ブロックのすべての LINE ステートメントに対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(lines)=[color=white backgroundcolor=black
              fontstyle=italic fontweight=bold fontsize=5]
```

レポート要約に対しスタイル属性を指定します。この STYLE=オプションは、すべてのデフォルト要約行に対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(summary)=[color=cx3e3d73 backgroundcolor=cxaeadd9
                fontfamily=helvetica fontsize=3 textalign=r];
```

レポート列を指定します。レポートには、Manager、Department、Sales の列が含まれています。

```
column manager department sales;
```

最初の並べ替え順序変数を定義します。このレポートで、Manager は順序変数です。PROC REPORT はまず Manager の値(COLUMN ステートメントの最初の変数であるため)別に行を調整します。ORDER=は Manager の値がそのフォーマットされた値に従って調整されるように指定します。FORMAT=は、この変数に使用する出力形式を指定します。引用符内のテキストは、列ヘッダーを指定します。

```
define manager / order
                 order=formatted
                 format=$mgrfmt.
                 'Manager'
```

最初の並べ替え順序変数列ヘッダーに対しスタイル属性を指定します。STYLE=オプションは、Manager の列ヘッダーの前景と背景の色を設定します。

```
                 style(header)=[color=white
                                backgroundcolor=black];
```

2 番目の並べ替え順序変数を定義します。このレポートで、Department は順序変数です。PROC REPORT はまず Manager の値(COLUMN ステートメントの最初の変数であるため)別に、次に Department の値別に行を調整します。ORDER=は Department の値

がその内部値に従って調整されるように指定します。FORMAT=は、この変数に使用する出力形式を指定します。引用符内のテキストは、列ヘッダーを指定します。

```
define department / order
                    order=internal
                    format=$deptfmt.
                    'Department'
```

---

2 番目の並べ替え順序変数列に対しスタイル属性を指定します。 STYLE=オプションは、列 Department のセルのフォントをイタリックに設定します。セルのスタイル属性は、PROC REPORT ステートメントで COLUMN 場所に対して作成されたものと一致します。

```
                    style(column)=[fontstyle=italic];
```

レポートの要約を作成します。 BREAK ステートメントは各マネージャの最終行の後にデフォルト要約を作成します。SUMMARIZE は Sales の値(レポートの分析変数または計算変数のみ)を要約行に書き込みます。PROC REPORT は、Sales が Sum 統計量の計算に使用される分析変数であるため、各マネージャに対する Sales の値を合計します。

```
break after manager / summarize;
```

カスタマイズした要約を作成します。 COMPUTE ステートメントは、レポートの最後にカスタマイズされた要約を作成する計算ブロックを開始します。この STYLE=オプションは、この計算ブロックの LINE ステートメントによって作成されるテキストに使用するスタイル要素を指定します。このスタイル要素は、PROC REPORT ステートメントで LINES 場所に対して指定された前景と背景の色を切り替えます。フォントスタイル、フォントの太さ、フォントサイズも変更します。

```
compute after manager
        / style=[fontstyle=roman fontsize=3 fontweight=bold
          backgroundcolor=white color=black];
```

カスタマイズされた要約に対しテキストを指定します。 LINE ステートメントは引用符付きテキストと Manager と Sales.sum の値(出力形式$MGRFMT.と DOLLAR7.2)を要約に置きます。ENDCOMP ステートメントは計算ブロックを終了する必要があります。

```
   line 'Subtotal for ' manager $mgrfmt. 'is '
        sales.sum dollar7.2 '.';
endcomp;
```

分析列に対するカスタマイズされた背景を作成します。 この計算ブロックは、100 以上の値を含む、要約行にはない Sales 列のすべてのセルに対し、背景色と太字フォントを指定します。

```
compute sales;
   if sales.sum>100 and _break_=' ' then
   call define(_col_, "style",
               "style=[backgroundcolor=yellow
                       fontfamily=helvetica
                       fontweight=bold]");
endcomp;
```

カスタマイズされたレポートの最後の要約を作成します。 この COMPUTE ステートメントは、レポートの最後に実行する計算ブロックを開始します。LINE ステートメントは引用符付きテキストと Sales.sum の値(出力形式 DOLLAR7.2)を書き込みます。ENDCOMP ステートメントは計算ブロックを終了する必要があります。

```
compute after;
```

```
        line 'Total for all departments is: '
             sales.sum dollar7.2 '.';
    endcomp;
```

処理するオブザベーションを選択します。 WHERE ステートメントは、南東セクタの店舗のオブザベーションのみレポートに選択します。

```
    where sector='se';
```

タイトルを指定します。

```
    title 'Sales for the Southeast Sector';
run;
```

ODS 出力先をクローズします。

```
ods pdf close;
ods rtf close;
```

出力:出力:HTML

アウトプット 55.15 *複数のステートメントの ODS 出力のスタイル要素*

Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		630
Subtotal for Jones is $630.00.		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		350
Subtotal for Smith is $350.00.		
Total for all departments is: $980.00.		

出力:PDF

アウトプット 55.16 *複数のステートメントの ODS 出力のスタイル要素*



Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		630
Subtotal for Jones is $630.00.		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		350
Subtotal for Smith is $350.00.		
Total for all departments is: $980.00.		

出力:RTF

アウトプット 55.17 *複数のステートメントの ODS 出力のスタイル要素*



Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		630
Subtotal for Jones is $630.00.		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		350
Subtotal for Smith is $350.00.		
Total for all departments is: $980.00.		

例 14: CELLWIDTH=スタイル属性を PROC REPORT とともに使用する

要素: PROC REPORT ステートメントオプション
STYLE=
COLUMN ステートメント
DEFINE ステートメントオプション
STYLE(COLUMN)=

他の要素: TITLE statement

詳細

この例では、PROC REPORT を使用して、テーブルを作成し、ODS スタイル属性を使用します。この例では、次の操作を実行します。

- レポート全体のセルの幅を設定する

- ODS 出力の列のセル幅を定義する

  注: CELLWIDTH=および WIDTH=は、別名です。

詳細については、“Style Attributes Tables ” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。

注: DEFINE ステートメントの WIDTH=オプションは、LISTING 出力のテーブル出力の幅だけを変更します。

プログラム

```
proc report data=sashelp.class;

   col name age sex;

   define name / style(column)=[cellwidth=1in];
define age / style(column)=[cellwidth=.5in];
define sex / style(column)=[cellwidth=.5in];

title "Using the CELLWIDTH= Style with PROC REPORT";
run;
```

プログラムの説明

ODS 出力のすべての列に対してセルの幅を指定します。 デフォルトでは、REPORT ウィンドウなしで REPORT プロシジャが実行され、その出力が空いている出力先に送信されます。ODS HTML はこの例でデフォルトとして使用される出力先です。

```
proc report data=sashelp.class;
```

使用する列を指定します。

```
   col name age sex;
```

CELLWIDTH=スタイル属性を使用して列の幅を定義します。 STYLE=オプションを使用して NAME 列、AGE 列および SEX 列のディメンションを定義します。

```
   define name / style(column)=[cellwidth=1in];
define age / style(column)=[cellwidth=.5in];
define sex / style(column)=[cellwidth=.5in];
```

テーブルタイトルを指定します。 テーブル名を入力して、SAS プログラムを実行します。

```
title "Using the CELLWIDTH= Style with PROC REPORT";
run;
```

出力:出力:HTML

アウトプット 55.18 *CELLWIDTH=スタイル属性を PROC REPORT とともに使用する*

Using CELLWIDTH= Style with PROC REPORT

Name	Age	Sex
Alfred	14	M
Alice	13	F
Barbara	13	F
Carol	14	F
Henry	14	M
James	12	M
Jane	12	F
Janet	15	F
Jeffrey	13	M
John	12	M
Joyce	11	F
Judy	14	F
Louise	12	F
Mary	15	F
Philip	16	M
Robert	12	M
Ronald	15	M
Thomas	11	M
William	15	M

例 15: PROC REPORT CALL DEFINE ステートメントでの STYLE/MERGE の使用

要素: PROC REPORT ステートメント
COLUMN ステートメント
COMPUTE ステートメント

CALL DEFINE ステートメントオプション
STYLE
STYLE/MERGE

他の要素: TITLE statement

詳細

この例では、PROC REPORT を使用してテーブルを作成し、CALL DEFINE ステートメントの STYLE/MERGE オプションを使用します。

プログラム

```
proc report data=sashelp.class;

col name sex age height weight;

define name--weight / display;

   compute sex;
    if sex = 'M' then
        call define('name', "style", "style=[background=cyan]");
    endcomp;

   compute age;
    if age > 13 then
        call define('name', "style/merge", "style=[color=red]");
    endcomp;

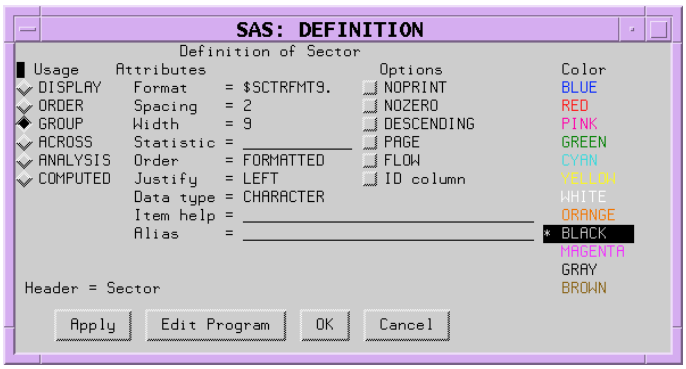
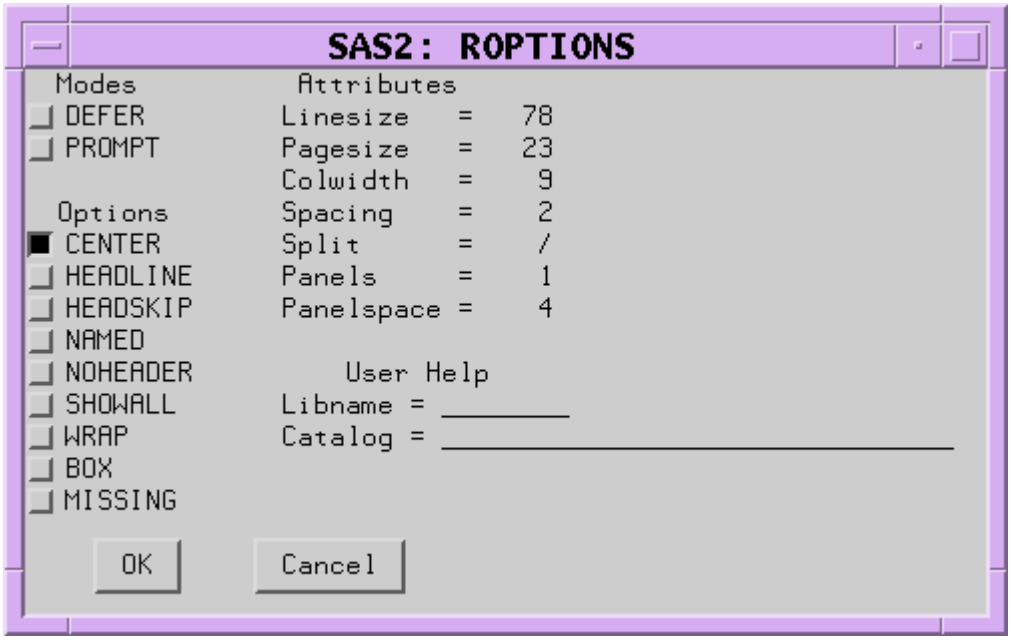
title "Using STYLE/MERGE Style with PROC REPORT";
run;
```

プログラムの説明

ODS 出力のすべての列に対してセルの幅を指定します。 デフォルトでは、REPORT ウィンドウなしで REPORT プロシジャが実行され、その出力が空いている出力先に送信されます。ODS HTML はこの例でデフォルトとして使用される出力先です。

```
proc report data=sashelp.class;
```

使用する列を指定します。

```
col name sex age height weight;
```

すべての列を表示します。

```
define name--weight / display;
```

スタイル属性を適用します。 男性の名前の背景にシアン色を適用します。

```
   compute sex;
    if sex = 'M' then
        call define('name', "style", "style=[background=cyan]");
    endcomp;
```

スタイル属性をマージします。 13 歳を超えている被験者の名前に赤色を適用します。この名前の色を背景色がシアン色のセルにマージします。

```
   compute age;
```

```
if age > 13 then
    call define('name', "style/merge", "style=[color=red]");
endcomp;
```

テーブルタイトルを指定します。テーブル名を入力して、SAS プログラムを実行します。

```
title "Using STYLE/MERGE Style with PROC REPORT";
run;
```

出力:出力:HTML

アウトプット 55.19 *STYLE/MERGE の使用*

Using STYLE/MERGE

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

例 16: PROC REPORT CALL DEFINE ステートメントでの STYLE/REPLACE の使用

要素: PROC REPORT ステートメント
COLUMN ステートメント
COMPUTE ステートメント
CALL DEFINE ステートメントオプション
STYLE
STYLE/REPLACE

他の要素: TITLE statement

詳細

この例では、PROC REPORT を使用してテーブルを作成し、CALL DEFINE ステートメントの STYLE/REPLACE オプションを使用します。

プログラム

```
proc report data=sashelp.class;

col name sex age height weight;

define name--weight / display;

   compute sex;
    if sex = 'M' then
        call define('name', "style", "style=[background=cyan]");
    endcomp;

   compute age;
    if age > 13 then
        call define('name', "style/replace", "style=[color=red]");
    endcomp;

title "Using STYLE/REPLACE";
run;
```

プログラムの説明

ODS 出力のすべての列に対してセルの幅を指定します。 デフォルトでは、REPORT ウィンドウなしで REPORT プロシジャが実行され、その出力が空いている出力先に送信されます。ODS HTML はこの例でデフォルトとして使用される出力先です。

```
proc report data=sashelp.class;
```

使用する列を指定します。

```
col name sex age height weight;
```

すべての列を表示します。

```
define name--weight / display;
```

スタイル属性を適用します。 男性の名前の背景にシアン色を適用します。

```
compute sex;
 if sex = 'M' then
     call define('name', "style", "style=[background=cyan]");
 endcomp;
```

スタイル属性を置き換えます。 13 歳を超えている被験者の名前に赤色を適用します。13 歳を超えている被験者の名前で、赤い名前の色を背景色シアンに置き換えます。

```
compute age;
 if age > 13 then
     call define('name', "style/replace", "style=[color=red]");
 endcomp;
```

テーブルタイトルを指定します。 テーブル名を入力して、SAS プログラムを実行します。

```
title "Using STYLE/REPLACE";
run;
```

出力:出力:HTML

アウトプット 55.20 *STYLE/REPLACE の使用*

Using STYLE/REPLACE

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

56 章

REPORT プロシジャウィンドウ

REPORT プロシジャウィンドウの概要

PROC REPORT の対話型レポートウィンドウ環境では、基本的にステートメントと同じ機能が提供されます。唯一の重要な例外は、対話型レポートウィンドウ環境からはアウトプットデリバリシステムを使用できないということです。

ディクショナリ

BREAK

グループ変数や順序変数の値を変更する場合、またはレポートの上部や下部での PROC REPORT のアクションを制御します。

詳細

パス

編集 ⇨ 要約

要約を選択すると、PROC REPORT では、ブレーク場所として 4 つの選択肢が提供されます。

- **Before Item**
- **After Item**
- **At the top**
- **At the bottom**

場所を選択すると、**要約**ウィンドウが表示されます。

注: 詳細行の前や後(グループ変数や順序変数の値変更時)にブレークを作成するには、**要約**ウィンドウを開く前に変数を選択しておく必要があります。

説明

注: このウィンドウの直線描画オプションで使用されるフォーマット文字の変更の詳細については、の説明を参照してください。"FORMCHAR <(*position(s)*)>='*formatting-character(s)*'" (1603 ページ)

オプション

上に線を追加する
2 番目のフォーマット文字を使用して各値に上線を引きます。

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト ハイフン(-)

操作 上線と二重上線のオプションを指定すると、PROC REPORT では上線が引かれます。

上に二重線を追加する
13 番目のフォーマット文字を使用して各値に上線を引きます。

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト 等号(=)

操作 上線と二重上線のオプションを指定すると、PROC REPORT では上線が引かれます。

下線を追加する
2 番目のフォーマット文字を使用して各値に下線を引きます。

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト ハイフン(-)

操作 下線と二重下線のオプションを指定すると、PROC REPORT では下線が引かれます。

二重下線を追加する
13 番目のフォーマット文字を使用して各値に下線を引きます。

- 要約行に表示されます。
- SUMMARIZE オプションを指定した場合、要約行に表示されます。

デフォルト 等号(=)

操作 下線と二重下線のオプションを指定すると、PROC REPORT では下線が引かれます。

ブレーク後の行をスキップする
最終ブレーク行に対しブランク行を書き込みます。

このオプションは、レポートの最後のブレークで使用した場合は、無効になります。

改ページ
最終ブレーク行の後に新しいページを開始します。このオプションは、レポートの最後のブレークでは無効になります。

操作 変数のブレークでこのオプションを使用し、レポートの最後にブレークを作成する場合、レポート全体の要約が別のページに表示されます。

分析変数を要約する
ブレーク行の各グループの要約行を書き込みます。要約行には次の値が含まれます。

- 統計量
- 分析変数
- 計算変数

一連のオブザベーションの要約行には、次も含まれます。

- ブレーク変数(**ラベルを抑制する**で非表示可能)
- ブレーク変数の左側のその他のグループ変数または順序変数

次の表に、**要約**ウィンドウによって作成される要約行の各種レポート項目の値の PROC REPORT による計算方法を示します。

レポート項目	結果値
ブレーク変数	変数の現在の値(**ラベルを抑制する**を選択している場合は欠損値)。
ブレーク変数の左側のグループ変数または順序変数	変数の現在値。
ブレーク変数の右側のグループ変数または順序変数、またはレポートの任意の場所の表示変数	欠損*
統計量	セットのすべてのオブザベーションにわたる統計量の値。
分析変数	項目の定義の使い方のオプションとして指定される統計量の値。PROC REPORT は、セットのすべてのオブザベーションにわたる統計量の値を計算します。デフォルトの使用法は SUM です。
計算変数	対応する計算ブロックのコードに基づく計算結果(“COMPUTE ステートメント” (1630 ページ)ステートメントを参照してください)。

*カスタマイズされた要約行の欠損値を含む変数を参照する場合、PROC REPORT はその変数をブランク(文字変数の場合)またはピリオド(数値変数の場合)として表示します。

ラベルを抑制する
次の印刷を行いません。

- 要約行のブレーク変数の値
- ブレーク変数を含む列のブレーク行の下線および下線

ラベルを抑制するを選択する場合、ブレーク変数の値は、ブレークと関連付けられている計算ブロックで値を割り当てない限り、カスタマイズされたブレーク行で使用できません。

色
列ヘッダー、および定義している項目の値の **REPORT** ウィンドウで使用する色を、色リストから選択します。デフォルトは、**SASCOLOR** ウィンドウの**前景**の色です(詳細については、**SASCOLOR** ウィンドウのオンラインヘルプを参照してください)。

注: すべての動作環境およびデバイスですべての色がサポートされているわけではありません。一部の動作環境およびデバイスでは、色の間でマップが行われることがあります。たとえば、**DEFINITION** ウィンドウに BROWN という言葉が黄色の文字で表示されている場合、BROWN を選択すると項目が黄色になります。

ボタン
プログラム編集
プログラム編集ウィンドウを開き、計算ブロックとレポート内の場所を関連付けられます。

OK
要約ウィンドウの情報をレポートに適用し、ウィンドウを閉じます。

キャンセル
情報をレポートに適用せずに**要約**ウィンドウを閉じます。

COMPUTE

計算ブロックをレポート項目かレポートの場所に割り当てます。SAS テキストエディタのコマンドを使用して、このウィンドウでテキストを操作します。

詳細

パス
計算列、**定義**または**要約**ウィンドウの**プログラム編集**。

説明
プログラム編集ウィンドウで使用できる SAS 言語機能の詳細については、“計算ブロックのコンテンツ” (1591 ページ)を参照してください。

計算列

入力データセットにない変数をレポートに追加します。

詳細

パス
列を選択します。次に、**編集** ⇨ **追加** ⇨ **計算列**を選択します。

計算列を選択すると、PROC REPORT で、選択した列に関連する計算列の場所の入力が求められます。場所を選択すると、**計算列**ウィンドウが表示されます。

説明

プロンプトで変数名を入力します。文字変数の場合、**文字データ**チェックボックスを選択し、必要に応じて、**幅**フィールドに値を入力します。長さには 1 から 200 までの任意の整数を指定できます。フィールドをブランクのままにしておくと、PROC REPORT では、長さ 8 が変数に割り当てられます。

変数名の入力後に、**プログラム編集**を選択して**プログラム編集**ウィンドウを開きます。**プログラム編集**ウィンドウでプログラミングステートメントを使用して、計算変数を定義します。**プログラム編集**および**計算列**ウィンドウを閉じた後で、**定義**ウィンドウを開き、計算変数の表示方法を記述します。

注: 計算変数の位置は重要です。PROC REPORT は値をレポート行の列に、左から右に割り当てます。その結果、レポートの右側に表示される変数に基づいて計算変数の計算を行うことはできません。

変数

入力データセットの変数をすべてリスト表示して、レポートにデータセット変数を 1 つ以上追加できるようにします。

詳細

パス

レポート項目を選択します。次に、**編集** ⇨ **追加** ⇨ **変数**を選択します。

変数を選択すると、PROC REPORT で、選択した列に関連する計算列の場所の入力を求められます。場所を選択すると、**変数**ウィンドウが表示されます。

説明

レポートに追加する変数を 1 つ以上選択します。最初に選択した変数は、ウィンドウのリストの上部に移動します。複数の変数を選択した場合、後で選択した変数は選択変数リストの下部に移動します。アスタリスク(*)によって選択変数が識別されます。選択変数の上から下への順序によって、レポートでは左から右へ順序が決定されます。

データセットを開く

現在のレポート定義にデータセットをロードします。

詳細

パス

ファイル ⇨ **データセットを開く**

説明

データセットを開くウィンドウの最初のリストボックスには、SAS セッションに対して定義したライブラリ参照名がすべてリスト表示されます。2 番目には、選択したライブラリ内の SAS データセットがすべてリスト表示されます。

注: 現在のレポート定義との互換性があるデータを使用する必要があります。ロードするデータセットには、現在のレポート定義の変数名と同じ名前の変数が含まれている必要があります。

ボタン

OK
選択したデータセットを現在のレポート定義にロードします。

キャンセル
新しいデータをロードせずに**データセットを開く**ウィンドウを閉じます。

定義

レポートの項目に関連付けられた特性を表示し、変更できるようにします。

詳細

パス

レポート項目を選択します。次に、**編集** ⇨ **定義**を選択します。

注: または、選択した項目をダブルクリックします(すべての動作環境で、この**定義**ウィンドウの開き方がサポートされているわけではありません)。

説明



使い方

各タイプの使い方の説明については、“レポートのレイアウト ” (1585 ページ) を参照してください。

表示
選択した項目を表示変数として定義します。表示変数が文字変数のデフォルトです。

ORDER
選択した項目を順序変数として定義します。

GROUP
選択した項目をグループ変数として定義します。

ACROSS
選択した項目を列変数として定義します。

ANALYSIS
選択した項目を分析変数として定義します。分析変数の統計量(統計量属性の説明を参照 (1743 ページ))を指定する必要があります。分析変数は数値変数のデフォルトです。

COMPUTED
選択した項目を計算変数として定義します。計算変数は、レポートに対して定義する変数です。入力データセットにはなく、PROC REPORT によって入力データセットに追加されません。ただし、計算変数は作成されると出力データセットに含まれます。

対話型レポートウィンドウ環境では、計算変数を **COMPUTED VAR** ウィンドウからレポートに追加します。

属性

出力形式
SAS またはユーザー定義の出力形式を項目に割り当てます。この出力形式は、PROC REPORT で表示されるときに選択項目に適用されます。出力形式によって、データセット内の変数と関連付けられている出力形式が変更されることはありません。データセット変数の場合、PROC REPORT は検出されるこれらの出力形式のうち最初のものを有効化します。

- **定義**ウィンドウで出力形式と関連付けられている出力形式
- PROC REPORT の開始時に FORMAT ステートメントで割り当てられる出力形式
- データセット内の変数と関連付けられる出力形式

これらフォーマットが一切存在しない場合は、PROC REPORT により、数値変数には BEST*w*.が使用され、文字変数には$*w*.が使用されます。*w* の値は、デフォルトの列幅です。入力データセット内の文字変数の場合、デフォルトの列幅は変数の長さです。入力データセットの数値変数の場合と、計算変数(数値および文字の両方)の場合、デフォルトの列幅は**オプション**ウィンドウの列の幅属性の値です。

使用する出力形式が不明な場合、**定義**ウィンドウの出力形式フィールドに疑問符(?)を入力して、**出力形式**ウィンドウにアクセスします。

スペース
定義中の列とそのすぐ左の列の間を空けるブランク文字の数を定義します。各列について、その幅、その間のブランク文字、その左側の列の合計がページサイズを超えることはできません。

デフォルト 2

操作 PROC REPORT の CENTER オプションが有効な場合、PROC REPORT はレポートの最左の変数の前のスペースを無視します。

項目の定義のスペースは、PROC REPORT ステートメントまたは**オプション**ウィンドウのスペースの値より優先されます。

表示の幅
PROC REPORT が選択項目を表示する列の幅を定義します。

デフォルト 出力形式の処理に十分な大きさの列幅。出力形式がない場合、PROC REPORT は COLWIDTH=の値を使用します。

範囲 1 から SAS システムオプション LINESIZE=の値まで

注 項目をレポートの同じ列に積み重ねる場合、一番下に積み重ねられる項目の幅によって列の幅が決定されます。

統計量

統計量と分析変数を関連付けます。統計量とその定義のすべての分析変数を関連付ける必要があります。PROC REPORT は、レポートの各セルによって表されるオブザベーションの分析変数の値を計算するために指定する統計量を使用します。その他の種類の変数の定義で *statistic* を使用することはできません。

注: PROC REPORT は、分析変数の名前を列のデフォルトヘッダーとして使用します。**定義**ウィンドウの**ラベル**フィールドを使用して、列ヘッダーをカスタマイズできます。

次の値を*統計量*として使用できます。

記述統計量キーワード	
CSS	PCTSUM
CV	RANGE
MAX	STD
MEAN	STDERR
MIN	SUM
N	SUMWGT
NMISS	USS
PCTN	VAR
四分位範囲統計量キーワード	
MEDIAN | P50	Q3 | P75
P1	P90
P5	P95
P10	P99
Q1 | P25	QRANGE
仮説検定キーワード	
PRT|PROBT	T

キーワード、その計算に使用される式、およびデータ要件の説明については、付録 1, “基本的な SAS 統計プロシジャ” (2077 ページ)を参照してください。

デフォルト SUM

要件 標準誤差とスチューデントの *t* 検定を計算するには、VARDEF=のデフォルト値 DF を使用する必要があります。

参照項目 これらの統計量の定義については、“キーワードと式” (2078 ページ)を参照してください。

順序

グループ変数、順序変数、列変数の値を指定順序に従って並べ替えます。

DATA
入力データセットの順序に従って値を並べ替えます。

FORMATTED
フォーマットされた(外部)値によって値を並べ替えます。デフォルトでは、順序は昇順です。

FREQ
昇順の度数カウントによって値を並べ替えます。

INTERNAL
フォーマットされていない値によって値を並べ替えます。これにより PROC SORT が作成するのと同じ順序が作成されます。この順序は、動作環境によって異なります。この並べ替え順序は、日付を年代順に表示する場合に特に便利です。

デフォルト FORMATTED

操作 項目の定義の DESCENDING は、項目の並べ替え順序を逆順にします。

注 PROC REPORT の ORDER=オプションのデフォルト値は、その他の SAS プロシジャのデフォルト値とは異なります。その他の SAS プロシジャでは、デフォルトは ORDER=INTERNAL です。PROC REPORT のオプションのデフォルトは、その他のプロシジャと整合させるために、今後のリリースで変わることがあります。このため、フォーマットされた値によってレポート項目を並べ替えることが重要な本稼働ジョブでは、現在デフォルトである ORDER=FORMATTED を指定します。これにより、PROC REPORT はデフォルトが変わっても期待どおりにレポートの作成が続行されます。

位置の調整

次の 3 つの方法のいずれかで、列ヘッダー、および列内に定義している項目の値の配置を揃えられます。

LEFT
列幅内に定義している項目のフォーマットされた値を左寄せにし、値にわたる列ヘッダーを左寄せにします。出力形式の幅が列幅と同じ場合、LEFT は値の配置に影響を及ぼしません。

RIGHT
列幅内に定義している項目のフォーマットされた値を右寄せにし、値にわたる列ヘッダーを右寄せにします。出力形式の幅が列幅と同じ場合、RIGHT は値の配置に影響を及ぼしません。

CENTER
列幅内に定義している項目のフォーマットされた値を中央揃えにし、値にわたる列ヘッダーを中央揃えにします。このオプションは、SAS システムオプション CENTER の設定には影響を及ぼしません。

値を揃える際、PROC REPORT では、列内の項目の出力形式によって定義されたフィールド幅が揃えられます。このため、数字は常に揃えられます。

項目の属性
レポート項目が数値か文字かを示します。このフィールドは変更できません。

項目のヘルプ
選択した項目に関するヘルプ情報を含む HELP エントリまたは CBT エントリを参照します。SAS/AF ソフトウェアの PROC BUILD を使用して、レポート項目に対し HELP エントリまたは CBT エントリを作成します。レポートの All HELP エントリと CBT エントリは、同じカタログ内にある必要があり、カタログを PROC REPORT ステートメントの HELP=オプションで、または **ROPTIONS** ウィンドウの**ユーザーヘルプ**フィールドから指定する必要があります。

レポートからヘルプエントリにアクセスするには、項目を選択して、HELP コマンドを発行します。PROC REPORT はまず *entry-name*.CBT という名前のエントリを検索し、表示します。そのようなエントリが存在しない場合、PROC REPORT は *entry-name*.HELP を検索します。選択した項目に対し CBT エントリも HELP エントリも存在しない場合、PROC REPORT のヘルプの開始フレームが表示されます。

別名
別名フィールドに名前を入力して、定義しているレポート項目の別名を作成します。別名を使用すると、使用方法が異なる同じレポート項目同士を見分けられます。計算ブロックの別名を持つレポート項目を参照する場合、その別名を使用する必要があります(“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1687 ページ)を参照)。

オプション

NOPRINT
定義している項目の表示を非表示にします。このオプションは、次の場合に使用します。

- レポートに項目を表示せずに、その値を使用してレポートで使用するその他の値を計算する場合。
- レポートで行の順序を作成する場合。
- 項目を列として使用せずに、要約のその値にアクセスする場合(“例 7: ページごとにカスタマイズされた要約を書き込む” (1699 ページ)を参照)。

操作 NOPRINT を使用して定義する列がレポートに表示されない場合でも、列を番号別に参照する場合はそれらの列をカウントする必要があります(“計算ブロックのレポート項目を参照するための 4 つの方法” (1591 ページ) を参照)。

PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウの SHOWALL は、NOPRINT のすべての発生を無効にします。

NOZERO
定義している項目の表示を、その値がすべてゼロまたは欠損値の場合に非表示にします。

操作 NOZERO を使用して定義する列がレポートに表示されない場合でも、列を番号別に参照する場合はそれらの列をカウントする必要があります(“計算ブロックのレポート項目を参照するための 4 つの方法” (1591 ページ)を参照)。

PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウの SHOWALL は、NOZERO のすべての発生を無効にします。

DESCENDING
PROC REPORT がグループ変数、順序変数、列変数の行または値を表示する順序を逆順にします。

PAGE
選択した項目の値を含む最初の列を印刷する直前に改ページを挿入します。

操作 WRAP を PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウで使用する場合、PAGE は無視されます。

FLOW
その列の文字変数の値をラップします。FLOW オプションは区切り文字を有効化します。テキストに区切り文字が含まれていない場合、PROC REPORT はテキストをブランクで分割しようとします。

ID 列
定義している項目が ID 変数になるように指定します。ID 変数とその左側のすべての列がレポートの各ページの左側に表示されます。ID によって、1 ページに収まる以上の列がレポートに含まれているときにレポートの各行を識別できるようになります。

色

列ヘッダー、および定義している項目の値の **REPORT** ウィンドウで使用する色を、色リストから選択します。デフォルトは、**SASCOLOR** ウィンドウの**前景**の色です(詳細については、**SASCOLOR** ウィンドウのオンラインヘルプを参照してください)。

注: すべての動作環境およびデバイスですべての色がサポートされているわけではありません。一部の動作環境およびデバイスでは、色の間でマップが行われることがあります。たとえば、**DEFINITION** ウィンドウに BROWN という言葉が黄色の文字で表示されている場合、BROWN を選択すると項目が黄色になります。

ボタン

適用
開いているウィンドウの情報をレポートに適用し、ウィンドウを開いたままにしておきます。

プログラム編集
プログラム編集ウィンドウを開き、計算ブロックと定義している変数を関連付けられます。

OK
定義ウィンドウの情報をレポートに適用し、ウィンドウを閉じます。

キャンセル
適用で行われた変更を適用せずに、**定義**ウィンドウを閉じます。

ページ番号

レポートの特定ページを表示します。

詳細

パス

表示 ⇨ ページ番号

説明

ページ番号の最大値を入力すると、レポートの最終ページにアクセスできます。レポートの最終ページを表示すると、PROC REPORT で、**REPORT** ウィンドウのメッセージ行に注記が送信されます。

エクスプローラ

データを試せます。

制限事項: レポートにグループ変数か順序変数が少なくとも 1 つ含まれていなければ、エクスプローラウィンドウは開きません。

詳細

パス

編集 ⇨ **エクスプローラデータ**

説明

エクスプローラウィンドウでは、次を実行できます。

- リストボックスによるデータのサブセット化
- **表示しない**チェックボックスによる列の非表示
- **列の入れ替え**による列の順序変更

注: **エクスプローラ**ウィンドウでの操作の結果は、**REPORT** ウィンドウには表示されますが、レポート定義には保存されません。

ウィンドウ機能

リストボックス

エクスプローラウィンドウには、3 つのリストボックスがあります。これらの 3 つのボックスには、値**すべての水準**に加えて、レポートの最初の 3 つのグループ変数または順序変数の実際の値が含まれます。値には、有効な WHERE 句処理がすべて反映されます。たとえば、WHERE 句を使用してデータをサブセット化し、北東セクタと北西セクタのみを含めるようにした場合、セクタのリストボックスに表示される値は、すべての**水準**、**北東**および**北西**のみになります。この場合に**すべての水準**を選択すると、北東セクタと北西セクタのレポート行のみが表示されます。すべてのセクタのデータを参照するには、**エクスプローラ**ウィンドウを開く前に WHERE 句をクリアする必要があります。

リストボックスで値を選択すると、**REPORT** ウィンドウの表示が、選択した値に制限されます。互換性のない値を選択すると、PROC REPORT では、エラーが返されます。

表示しない

エクスプローラウィンドウの各リストボックスの上に、**表示しない**というラベルの付いたチェックボックスがあります。このチェックボックスを選択し、変更を適用すると、**REPORT** ウィンドウから列が削除されます。チェックボックスをクリアし、その変更を適用すると、簡単に列を復元できます。

ボタン

OK

エクスプローラウィンドウの情報をレポートに適用し、ウィンドウを閉じます。

適用
エクスプローラウィンドウの情報をレポートに適用し、ウィンドウを開いたままにしておきます。

列の入れ替え
リストボックスでの変数の表示順序を変更します。左に 1 列移動可能な各変数が移動し、最左の変数は 3 列目に移動します。

キャンセル
適用で行われた変更を適用せずに、**エクスプローラ**ウィンドウを閉じます。

出力形式

出力形式のリストを表示し、それぞれのサンプルを提供します。

詳細

パス

定義ウィンドウから**出力形式**フィールドに疑問符(?)を入力し、**キャンセル**以外のいずれかのボタンを選択するか、Enter キーを押します。

説明

出力形式ウィンドウで出力形式を選択すると、**例:**フィールドにその出力形式のサンプルが表示されます。定義している変数に対して使用する出力形式を選択します。

ボタン

OK
選択した出力形式を**定義**ウィンドウの**出力形式**フィールドに書き込み、**出力形式**ウィンドウを閉じます。レポートの出力形式を参照するには、**定義**ウィンドウで**適用**を選択します。

キャンセル
出力形式フィールドに出力形式を書き込まずに、**出力形式**ウィンドウを閉じます。

レポートを開く

保存されたレポート定義をロードします。

詳細

パス

ファイル ⇨ レポートを開く

説明

レポートを開くウィンドウの最初のリストボックスには、SAS セッションに対して定義したライブラリ参照名がすべてリスト表示されます。2 番目のリストボックスには、選択したライブラリ内のカタログがすべてリスト表示されます。3 番目のリストボックスには、選択したカタログ内に保存されたすべてのレポート定義(エントリの種類 REPT)の説明が

リスト表示されます。エントリに説明がなければ、リストボックスにはエントリの名前が含まれます。

ボタン

OK
選択したレポート定義に現在のデータをロードします。

キャンセル
新しいレポート定義をロードせずに**レポートを開く**ウィンドウを閉じます。

注: **REPORT** ウィンドウで END コマンドを発行すると、前のレポート定義(現在のデータを表示)に戻ります。

メッセージ

自動的に開き、PROC REPORT によって返された注、警告およびエラーが表示されます。

詳細

PROC REPORT の使用を続行する前に、**OK** を選択して**メッセージ**ウィンドウを閉じる必要があります。

プロファイル

レポートプロファイルを作成して、PROC REPORT 環境の一部機能をカスタマイズします。

詳細

パス

ツール ⇨ **レポートプロファイル**

説明

プロファイルウィンドウでは、次のようなレポートプロファイルが作成されます。

- **REPORT** および**プログラム編集**ウィンドウで使用するための代替メニューを定義する SAS ライブラリ、カタログおよびエントリを指定します。PROC PMENU を使用すると、これらのメニューを定義する種類 PMENU のカタログエントリを作成できます。両方のウィンドウの PMENU エントリが同じカタログに存在する必要があります。
- WINDOWS、PROMPT、COMMAND のデフォルト値を設定します。PROC REPORT では、PROC REPORT ステートメントのオプションを指定して無効にしない限り、プロシジャを開始するたびにデフォルトオプションが使用されます。

PROC REPORT ステートメントの PROFILE=オプションで使用するプロファイルを含めるカタログを指定します (“PROFILE=*libref.catalog*” (1609 ページ)の説明を参照)。

ボタン

OK
SASUSER.PROFILE.REPORT.PROFILE というファイルにプロファイルを保存します。

注: PROC CATALOG または**エクスプローラ**ウィンドウを使用して、プロファイルを別の場所にコピーします。

キャンセル
プロファイルを保存せずにウィンドウを閉じます。

対話モード

レポートへの項目の追加時に、情報の入力を求めます。

詳細

パス

PROC REPORT の開始時に PROMPT オプションを指定するか、**オプション**ウィンドウから対話モードで使用するを選択します。次にレポートに項目を追加したときに、**対話モード**ウィンドウが表示されます。

説明

この対話モードでは、ウィンドウでレポートの作成に最もよく使用される部分がガイドされます。**対話モード**ウィンドウのコンテンツが変更されると、ウィンドウのタイトルが、対話モードを使用しない場合にタスクの実行に使用するウィンドウの名前に変更されます。タイトル変更は、ウィンドウと機能を関連付けたり、後で変更を加えることを決定した場合にどのウィンドウを使用すべきかを確認したりするのに役立ちます。

PROC REPORT をプロンプトで開始する場合、最初のウィンドウではプロンプト中に使用されるオブザベーション数を制限できます。対話モードを終了すると、PROC REPORT によってその制限が削除されます。

ボタン

OK
開いているウィンドウの情報をレポートに適用し、プロンプトプロセスを続行します。

注: 最後のプロンプトウィンドウで **OK** を選択すると、PROC REPORT では、処理対象のオブザベーション数の制限がすべて削除されます。

適用
開いているウィンドウの情報をレポートに適用し、ウィンドウを開いたままにしておきます。

戻る
前の**対話モード**ウィンドウに戻ります。

閉じる
レポートにこれ以上の変更は適用せずに**対話モード**ウィンドウを閉じます。プロンプト中に使用するオブザベーション数を制限していた場合、PROC REPORT によってその制限が削除されます。

REPORT

レポート表示の基になる外観です。

詳細

パス

PROC REPORT ステートメントで WINDOWS または PROMPT を使用します。

説明

REPORT ウィンドウでは、列ヘッダー以外はどの部分にも直接書き込むことはできません。レポートのその他の外観を変更するには、次コマンドのターゲットとしてレポート項目(列ヘッダーなど)を選択し、コマンドを発行します。項目を選択するには、マウスまたはカーソルキーを使用してカーソルを合わせます。次に、マウスボタンをクリックするか、Enter を押します。コマンドを実行するには、**REPORT** ウィンドウ上部のメニューバーから選択します。更新時の指定オプションをオンにしない限り、PROC REPORT では、コマンドの影響が即座に表示されます。

注: **REPORT** ウィンドウで END コマンドを発行すると、現在のデータによる前のレポート定義に戻ります。前のレポート定義が存在しなければ、END では、**REPORT** ウィンドウが閉じられます。

注: **REPORT** ウィンドウのファイルメニューには、レポートをファイルに保存するための、名前を付けて保存オプションがありません。かわりに次の操作を実行します。

1. **REPORT** ウィンドウからデータセットの保存を選択します。ダイアログボックスで、そのデータセットの保存先の SAS ライブラリとファイル名を入力します。
2. プログラムエディタウィンドウから PROC PRINT を実行します。
3. ファイルメニューで、名前を付けて保存を選択し、生成された出力をファイルに保存します。

オプション

レイアウト制御やレポート全体表示の選択肢を表示し、レポート項目の CBT または HELP エントリを含む SAS ライブラリとカタログを識別します。

詳細

パス

ツール ⇨ **オプション** ⇨ **レポート**

説明



モード

更新時の指定

更新時の指定モードをオフにした場合や**表示** ⇨ **最新の情報に更新**を選択した場合、即時に変更情報が保存され、適用されます。

更新時の指定は、レポートに複数の変更を加える必要があると判明しているが、中間レポートの確認は不要な場合に特に便利です。

デフォルトでは、項目の追加や削除、**定義**ウィンドウでの情報変更、または**要約**ウィンドウでの情報変更によってレポートを再定義するたびに、PROC REPORT で **REPORT** ウィンドウにレポートが再表示されます。

対話モードで使用する

次にレポートに項目を追加したときに、**対話モード**ウィンドウが開きます。

オプション

CENTER

レポートと要約テキスト(カスタマイズされたブレーク行)を中央揃えにします。中央揃えを選択しなければ、レポートは左寄せになります。

PROC REPORT は、検出したこれらの中央揃え指定のうち最初のものを有効化します。

- PROC REPORT ステートメントの CENTER オプションまたは NOCENTER オプション、または**オプション**ウィンドウの中央揃えの切り替え
- PROC REPORT ステートメントの REPORT=を使用して開かれたレポート定義に保存された CENTER オプションまたは NOCENTER オプション
- SAS システムオプション CENTER または NOCENTER

PROC REPORT の CENTER オプションが有効な場合、PROC REPORT はレポートの最左の変数の前のスペースを無視します。

HEADLINE

レポートの各ページの上部のすべての列ヘッダーとそれらの間のスペースに下線を引きます。

ヘッダーラインは、2 番目のフォーマット文字で下線を引きます("FORMCHAR <(position(s))>='formatting-character(s)'" (1603 ページ)の説明を参照)。

デフォルト ハイフン(-)

ヒント 従来の(monospace) SAS 出力では、ヘッダーラインを使用するかわりに各列ヘッダーの最終行として'---' を使用することにより、列ヘッダーに下線を引き、その間のスペースに下線を引かなくて済みます。

HEADSKIP
レポートの各ページの上部のすべての列ヘッダーの下(または HEADLINE オプションで書き込む下線の下)にブランク行を書き込みます。

NAMED
name=を、*name* が値の列ヘッダーであるレポートの各値の前に書き込みます。

操作 列ラベルの追加の使用時、PROC REPORT は自動的にヘッダーの抑制を使用します。

ヒント 列ラベルの追加を 1 ページに表示と組み合わせて使用し、個別のページに広いレポートの列を置くのではなく、レポートの単一行に対しすべての列を連続行にラップするレポートを作成します。

NOHEADER
複数の列にわたるヘッダーを含む列ヘッダーを非表示にします。

対話型レポートウィンドウ環境で列ヘッダーの表示を非表示にすると、レポート項目を選択できません。

SHOWALL
列の表示を非表示にする定義部分(表示しないおよび欠損値の抑制)を無効にします。DEFINE ステートメントを使用するか、**定義**ウィンドウで、レポート項目を定義します。

WRAP
最初の列の別の値を表示する前に、必要に応じてレポートの列ごとに値を 1 つ連続行に表示します。デフォルトで、PROC REPORT は 1 ページに入る分の列に対する値を表示します。これらの列の値をページに入力してから、次のページで残りの列の値の表示を開始します。

操作 WRAP が有効な場合、項目定義で PROC REPORT は PAGE を無視します。

ヒント 通常、列ヘッダーのラップを避けるために 1 ページに表示を列ラベルの追加と組み合わせて使用します。

BOX
フォーマット文字を使用して、罫線文字をレポートに追加します。これらの文字で、次を行います。

- レポートの各ページを囲む
- 列ヘッダーをレポートの本文から区切る
- 行と列を相互に区切る

操作 罫線は、PROC REPORT ステートメントまたは **ROPTIONS** ウィンドウで 1 ページに表示を使用する場合、または項目定義で折り返しを使用する場合、使用できません。

参照項目 フォーマット文字の詳細については、“FORMCHAR <(position(s))>='formatting-character(s)'” (1603 ページ)の説明を参照してください。

MISSING
欠損値をグループ変数、順序変数、または列変数の有効値とみなします。数値を表すために使用される特殊欠損値(文字 A から Z、アンダースコア(_)文字)は、それぞれ異なる値とみなされます。各欠損値のグループがレポートに表示されます。欠損値オプションを省略すると、PROC REPORT は 1 つ以上のグループ変数、順序変数、列変数に対する欠損値を含むオブザベーションをレポートに含めません。

属性

LINESIZE=
レポートのラインサイズを指定します。PROC REPORT は、検出したこれらラインサイズ指定のうち最初のものを有効化します。

- PROC REPORT ステートメントの LS=または **ROPTIONS** ウィンドウのラインサイズ
- PROC REPORT ステートメントの REPORT=を使用してロードされるレポート定義に保存される LS=設定
- SAS システムオプション LINESIZE=

範囲 64-256 (整数)

ヒント ラインサイズが **REPORT** ウィンドウの幅より大きい場合は、SAS 対話型レポートウィンドウ環境コマンド RIGHT および LEFT を使用して、レポートの現在表示されていない部分を表示します。

PAGESIZE=
レポートのページサイズを指定します。PROC REPORT は、検出したこれらページサイズ指定のうち最初のものを有効化します。

- PROC REPORT ステートメントの PS=または **ROPTIONS** ウィンドウのページサイズ
- PROC REPORT ステートメントの REPORT=を使用して開かれたレポート定義に保存された PS=設定
- SAS システムオプション PAGESIZE=

範囲 15-32,767 (整数)

列の幅
計算変数または数値データセット変数を含む列に対して文字のデフォルト数を指定します。

列幅の設定時に、PROC REPORT はまずその列に対する定義の WIDTH=を確認します。WIDTH=が存在しない場合、PROC REPORT は項目の出力形式の対応に十分な大きさの列幅を使用します(出力形式の詳細については、“出力形式” (1742 ページ)の説明を参照してください)。出力形式が項目と関連付けられていない場合、列幅は変数の種類によって異なります。

変数	結果の列幅
入力データセット内の文字変数	変数の長さ

変数	結果の列幅
入力データセットの数値変数	COLWIDTH=オプションの値
計算変数(数値または文字)	COLWIDTH=オプションの値

デフォルト 9

範囲 1 からラインサイズ

SPACING=*space-between-columns*
列間のブランク文字の数を指定します。各列について、その幅、その間のブランク文字、その左側の列の合計がページサイズを超えることはできません。

デフォルト 2

操作 特定の項目の定義のスペースを使用して間隔をその項目の左に変更しない限り、PROC REPORT は、PROC REPORT ステートメントまたは**オプション**ウィンドウのスペースで指定するブランク文字の数によってレポートのすべての列を区切ります。

CENTER が有効な場合、PROC REPORT はレポートの最左の変数の前のスペースを無視します。

SPLIT='*character*'
区切り文字を指定します。PROC REPORT は、その文字に達すると列ヘッダーを分割し、ヘッダーを次の行に続けます。区切り文字が発生するたびにラベルの最大 40 文字に考慮されますが、区切り文字自体は列ヘッダーの一部ではありません。

デフォルト slash (/)

操作 DEFINE ステートメントの FLOW オプションは、区切り文字を有効化します。

ヒント ヘッダーを(**対話**モードまたは**定義**ウィンドウから入力するかわりに)上書き入力した場合は、画面を更新しなければ区切り文字の効果を確認できません。画面を更新するには、項目を追加または削除するか、定義または要約ウィンドウの内容を変更するか、**表示** ⇨ **最新の情報に更新**を選択します。

PANELS=*number-of-panels*
レポートの各ページのパネル数を指定します。レポートの幅がページサイズの半分より小さい場合、データを複数の列セットに表示でき、他の場合は複数のページに表示される行が同じページに表示されます。各列セットが *panel* です。この種類のレポートのわかりやすい例は電話帳で、1 ページに複数のパネルの名前と電話番号が含まれます。

PROC REPORT がマルチパネルレポートを書き込む際は、1 つのパネルがいっぱいになってから次を開始します。

ページに合うパネル数は、次によって異なります。

- パネルの幅

- パネル間のスペース
- ページサイズ

デフォルト 1

ヒント *number-of-panels* がページに収まるパネル数より大きい場合、PROC REPORT はできるだけ多くのパネルを作成します。多くのパネル(たとえば 99)を指定して、PROC REPORT がページに収まる最大数のパネルにデータを入力するようにします。

参照項目 パネル間のスペース指定の詳細については、パネル間の余白の説明を参照してください。ラインサイズ設定の詳細については、"LINESIZE=" (1754 ページ)の説明を参照してください。

PSPACE=*space-between-panels*
パネル間のブランク文字の数を指定します。PROC REPORT は、レポートのすべてのパネルを同じ数のブランク文字によって区切ります。各パネルに対し、その幅とそれをパネルから左に区切るブランク文字の数の合計は、ページサイズを超えることはできません。

デフォルト 4

ユーザーヘルプ
レポートのユーザー定義のヘルプを含むライブラリとカタログを識別します。このヘルプは、CBT または HELP カタログエントリにおくことができます。SAS/AF ソフトウェアの BUILD プロシジャを使用して、レポートの各項目に対し CBT または HELP エントリを書き込むことができます。レポートに対するすべてのエントリを同じカタログに保存する必要があります。

特定のレポート項目に関するヘルプのエントリ名をそのレポート項目の**定義**ウィンドウまたは DEFINE ステートメントで指定します。

データセットの保存

現在のレポートのデータを保存する出力データセットを指定できます。

詳細

パス
ファイル ⇨ データセットの保存

説明
出力データセットを指定するには、**データセットの保存**ウィンドウで作成する SAS ライブラリの名前とデータセットの名前(ウィンドウでは `member`)を入力します。

ボタン
OK
出力データセットを作成し、**データセットの保存**ウィンドウを閉じます。

キャンセル
出力データセットを作成せずに**データセットの保存**ウィンドウを閉じます。

保存

レポート定義を保存し、同一データセットや類似データセットで次回に実行できます。

詳細

パス
ファイル ⇨ **レポートの保存**

説明
保存ウィンドウでは、現在のレポートの定義を保存するカタログエントリの完全名、およびオプションのレポート説明の入力を求められます。この説明は**レポートを開く**ウィンドウに表示され、適切なレポートの選択に役立ちます。

SAS では、レポート定義が種類 REPT のカタログエントリとして保存されます。レポート定義を使用して、レポート定義で使用される変数と同じ名前の変数を含む SAS データセットに対して同一に構築されたレポートを作成できます。

ボタン
OK
レポート定義を作成し、**保存**ウィンドウを閉じます。

キャンセル
レポート定義を作成せずに**保存**ウィンドウを閉じます。

ソース

現在のレポートを作成する PROC REPORT ステートメントをリストします。

詳細

パス
ツール ⇨ **ソース**

統計量

PROC REPORT で使用できる統計量を表示します。

詳細

パス
編集 ⇨ **追加** ⇨ **統計量**

統計量を選択すると、PROC REPORT で、選択した列に関連する統計量の場所の入力を求められます。場所を選択すると、**統計量**ウィンドウが表示されます。

説明

レポートに含める統計量を選択し、ウィンドウを閉じます。最初に選択した統計量は、ウィンドウのリストの上部に移動します。複数の統計量を選択すると、後で選択した統計量は選択統計量リストの下部に移動します。アスタリスク(*)は、各選択統計量を示します。選択統計量の上から下への順序によって、レポートでは左から右へ順序が決定されます。

注: 統計量をダブルクリックすると、PROC REPORT で即座に統計量がレポートに追加されます。**統計量**ウィンドウは開いたままです。

標準誤差とスチューデントの *t* 検定を計算するには、VARDEF=のデフォルト値 DF を使用する必要があります。

選択した統計量をすべてレポートに追加するには、**ファイル** ⇨ **選択の確定**を選択します。**ファイル** ⇨ **閉じる**を選択すると、**統計量**ウィンドウは、選択した統計量をレポートに追加せずに閉じられます。

WHERE

指定した条件に合うオブザベーションをデータセットから選択します。

詳細

パス

サブセット ⇨ **WHERE 条件の定義**

説明

WHERE 条件の定義フィールドに *where-expression* を入力します。*where-expression* は、通常は一連のオペランドと演算子から成る演算式または論理式です。*where-expression* の構成の詳細については、“WHERE Statement” (*SAS Statements: Reference*)のドキュメントを参照してください。

注: *where-expressions* をすべてクリアするには、**WHERE 条件の定義**フィールドは空のままで、**OK** を選択します。

ボタン

OK
where-expression をレポートに適用し、**WHERE 条件の定義**ウィンドウを閉じます。

キャンセル
レポートを変更せずに **WHERE 条件の定義**ウィンドウを閉じます。

WHERE 条件の追加

指定した条件、およびすでに有効なその他の条件すべてに合うオブザベーションをデータセットから選択します。

詳細

パス

サブセット ⇨ **WHERE 条件の追加**

説明

WHERE 条件の追加フィールドに *where-expression* を入力します。*where-expression* は、通常は一連のオペランドと演算子から成る演算式または論理式です。*where-expression* の構成の詳細については、“WHERE Statement” (*SAS Statements: Reference*)のドキュメントを参照してください。

ボタン

OK

where-expression を、すでに有効なその他の *where-expressions* に追加し、レポートにすべて適用します。さらに、**WHERE 条件の追加**ウィンドウを閉じます。

キャンセル

レポートを変更せずに **WHERE 条件の追加**ウィンドウを閉じます。

57 章

SCAPROC プロシジャ

概要: SCAPROC プロシジャ

SCAPROC プロシジャは SAS コードアナライザを実装します。実行中に SAS ジョブからの入力、出力、マクロシンボルの使用に関する情報を取得します。SAS コードアナライザは、この情報と元の SAS ファイルにある情報を、ユーザー指定のファイルへ書き込みます。SCAPROC プロシジャは、個別のジョブを同時に実行可能なグリッド対応ジョブを生成することもできます。オペレーティングシステムのコマンドライン、または **SAS エディタ**ウィンドウの SAS コードで SCAPROC プロシジャを発行できます。

次のコマンドはオペレーティングシステムのコマンドラインから SAS ジョブと SAS コードアナライザを実行します。

```
sas yourjob.sas -initstmt "proc scaproc;  record 'yourjob.txt' ; run;"
```

sas
SAS を起動するためにサイトで使用されるコマンドです。

yourjob.sas
分析する SAS ジョブの名前です。

yourjob.txt
SAS コードのコピーが含まれるファイルの名前です。ファイルには、入力と出力情報、マクロシンボルの使用、およびジョブの他の情報を示すために挿入されるコメントも含まれます。SAS コードでの PROC SCAPROC の発行の詳細については、例を参照してください。

グリッド対応ジョブの一部のタスクは、前のタスクとの依存関係があります。PROC SCAPROC は、前のタスクとの依存関係に基づいて、これらのタスクを組み合わせて

並べ替えます。タスクを組み合わせて同一の作業単位でサブミットすることにより、タスクのより迅速な処理が可能です。GRID オプションの NOOPTIMIZE 引数は、グリッド対応ジョブのタスクの組み合わせと並べ替えを無効にします。

注: GRID ステートメントを実行するには、SAS Grid Manager または SAS/CONNECT のライセンスが必要です。SAS Grid Manager では、分散マシンのグリッドで生成グリッドジョブを実行できます。SAS/CONNECT では、1 つの対称型マルチプロセッシング(SMP)マシンの並列 SAS セッションで生成グリッドジョブを実行できます。

概念:SCAPROC プロシジャ

グロバールステートメントの処理

生成されたグリッドジョブの各リモートセッションにサブミットする一連のステートメントをマークできます。一連のステートメントの前に**/* SCAPROC GLOBAL BEGIN */**コメントを、後ろに**/* SCAPROC GLOBAL END */**コメントを加えます。PROC SCAPROC により、ジョブのステップのいずれかがサブミットされる前に、マークしたセッション内のステートメントが各リモートセッションにサブミットされます。

In the following example, PROC SCAPROC submits the statements between the **/* SCAPROC GLOBAL BEGIN */** and **/* SCAPROC GLOBAL END */** comments with each of the PROC SUMMARY statements.

```
/* SCAPROC GLOBAL BEGIN */;
    libname one "SAS-library-1";
    libname two "SAS-library-1";
   %let year=2013;
/* SCAPROC GLOBAL END */;

Proc summary data=one.sales;
   where year=&year
   class month;
   var sales;
   output out=sasuser.sales;
run;
Proc summary data=two.expenses;
   where year=&year
   class month;
   var expenses;
   output out=sasuser.expenses;
run;
```

生成されたグリッドジョブと併用できるあらゆる SAS ステートメントを付加できます。あらゆる大文字または小文字のテキスト、または大文字と小文字が混在するテキストを使用できます。

構文: SCAPROC プロシジャ

PROC SCAPROC;
RECORD *filespec* <ATTR> <OPENTIMES> <INTCON> <EXPANDMACROS>
<GRID *filespec* <RESOURCE "*resource name*"> <INHERITLIB> <NOOPTIMIZE> >;
WRITE;

ステートメント	タスク	例
"PROC SCAPROC ステートメント"	SAS コードアナライザを実装します。	Ex. 1, Ex. 2
"RECORD ステートメント"	SAS コードアナライザの出力情報を含めるファイル名またはファイル参照名を指定します。	Ex. 1, Ex. 2
"WRITE ステートメント"	記録ファイルへ情報を出力します。	Ex. 1

PROC SCAPROC ステートメント

SAS が SAS ジョブと SAS コードアナライザを実行するように指定します。

例: "例 1: 記録ファイルの指定" (1768 ページ)
"例 2: グリッドジョブジェネレータの指定" (1769 ページ)

構文

PROC SCAPROC;

RECORD ステートメント

SAS コードアナライザの出力情報を含めるファイル名またはファイル参照名を指定します。

例: "例 1: 記録ファイルの指定" (1768 ページ)
"例 2: グリッドジョブジェネレータの指定" (1769 ページ)

構文

RECORD *filespec* <ATTR> <OPENTIMES> <INTCON> <EXPANDMACROS>
<GRID *filespec* <RESOURCE "*resource name*"> <INHERITLIB> <NOOPTIMIZE> >;

必須引数

filespec
SAS コードアナライザの出力情報を含めるファイルを指定する、物理ファイル名(引用符で囲む)またはファイル参照名を示します。出力情報は、元の SAS ソースとジョブに関する情報を含むコメントです。出力情報の詳細については、“結果” (1765 ページ)を参照してください。

オプション引数

ATTR
入力データセットと入力ビューにある変数に関する追加情報を出力します。

OPENTIMES
入力データセットが開かれた時間、サイズ、物理ファイル名を出力します。

INTCON
テーブルー貫性制約に関する出力情報です。

EXPANDMACROS
マクロ呼び出しを個別のタスクに展開します。

GRID

filespec
グリッドジョブジェネレータの出力を含めるファイルを指す物理ファイル名(引用符で囲む)またはファイル参照名を示します。

RESOURCE “*resource name*”
`grdsvs_enable` 関数呼び出しで使用するリソースを示します。デフォルトは SASApp です。

INHERITLIB
グリッド対応ジョブで SIGNON ステートメントに INHERITLIB=(USER)を追加します。デフォルトでは、各リモートセッションに USER ライブラリ(指定されている場合)が割り当てられます。

注: INHERITLIB は、すべてのリモートセッションで USER ライブラリがグローバルにアクセス可能でない場合にのみ使用します。

NOOPTIMIZE
グリッド対応ジョブのタスクの組み合わせと並べ替えを無効にします。

WRITE ステートメント

記録ファイルへの出力情報を示します。

例: “例 1: 記録ファイルの指定” (1768 ページ)

構文

WRITE;

引数なし

WRITE ステートメントは、記録ファイルが RECORD ステートメントにより指定されている場合、SAS コードアナライザによる記録ファイルへの情報の出力を指定します。グリッドジョブジェネレータが指定されている場合、グリッドジョブジェネレータもこの時点で

実行されます。SAS を終了する場合も、SAS コードアナライザによって指定の記録ファイルに情報が出力されます。

結果

次のリストに、SAS コードアナライザにより、PROC SCAPROC で指定される記録ファイルへ書き込まれるコメントの説明を示します。出力コメントは、記録ファイル内で/*と*/コメントタグによって区切られています。記録ファイル参照時にわかりやすいように、次にその形式を示します。

/* JOBSPLIT:DATASET INPUT|OUTPUT|UPDATE SEQ|MULTI *name* */
データセットが読み込み、書き込み、更新のために開かれたことを示します。

INPUT
データセットが読み込まれたことを示します。

OUTPUT
データセットが書き込まれたことを示します。

UPDATE
データセットが更新されたことを示します。

SEQ
データセットが順次アクセスのために開かれたことを示します。

MULTI
データセットがマルチパスアクセスのために開かれたことを示します。

name
データセットの名前を示します。

/* JOBSPLIT:CATALOG INPUT|OUTPUT|UPDATE *name* */
カタログが読み込み、書き込み、更新のために開かれたことを示します。

INPUT
カタログが読み込まれたことを示します。

OUTPUT
カタログが書き込まれたことを示します。

UPDATE
カタログが更新されたことを示します。

name
カタログ名を指定します。

/* JOBSPLIT:FILE INPUT|OUTPUT|UPDATE *name* */
外部ファイルが読み込み、書き込み、更新のために開かれたことを示します。

INPUT
ファイルが読み込まれたことを示します。

OUTPUT
ファイルが書き込まれたことを示します。

UPDATE
ファイルが更新されたことを示します。

name
ファイルの名前を示します。

/* JOBSPLIT:ITEMSTOR INPUT|OUTPUT|UPDATE *name* */
ITEMSTOR が読み込み、書き込み、更新のために開かれたことを示します。

INPUT
ITEMSTOR が読み込まれたことを示します。

OUTPUT
ITEMSTOR が書き込まれたことを示します。

UPDATE
ITEMSTOR が更新されたことを示します。

name
ITEMSTOR の名前を示します。

/* JOBSPLIT:OPENTIME *name* DATE:*date* PHYS:*phys* SIZE:*size* */
データセットが入力のために開かれたことを示します。ファイルの OPENTIME と SIZE が出力されます。

name
データセットの名前を示します。

DATE
データセットが開かれた日付と時刻を示します。DATE に返される値はファイルの作成日時ではありません。

PHYS
開かれたデータセットの完全な物理名を示します。

SIZE
データセットのサイズをバイト単位で示します。

/* JOBSPLIT:ATTR *name* INPUT|OUTPUT VARIABLE:*variable name*
TYPE:CHARACTER|NUMERIC LENGTH:*length* LABEL:*label* FORMAT:*format*
INFORMAT:*informat* */
データセットのクローズ日時を示します。再度開かれて、各変数の属性が出力されます。各変数に対して、ATTR 行が 1 つずつ生成されます。

name
データセットの名前を示します。

INPUT
データセットが読み込まれたことを示します。

OUTPUT
データセットが書き込まれたことを示します。

VARIABLE
現在の変数の名前を示します。

TYPE
変数が文字または数値であるか示します。

LENGTH
変数の長さをバイト単位で示します。

LABEL
変数ラベルを示します(ある場合)。

FORMAT
変数の出力形式を示します(ある場合)。

INFORMAT
変数の入力形式を示します(ある場合)。

/* JOBSPLIT:SYMBOL SET|GET *name* */
マクロシンボルがアクセスされたことを示します。

SET
シンボルが設定されたことを示します。例:シンボル **sym1** を次のコードで設定。**%let sym1=sym2**

GET
シンボルが取得されたことを示します。例:シンボル **sym** を次のコードで取得。**a="&sym"**

name
シンボルの名前を示します。

/* JOBSPLIT:ELAPSED *number* */
相対的なタスクの実行回数を決定するために使用する数値を示します。

number
相対的なタスクの実行回数を決定するために使用する数値を示します。

/* JOBSPLIT:USER *useroption* */
SAS によりグリッドジョブコードと USER オプションを併用して、単一レベルのデータセット名が Work ライブラリに存在できるようにすることを示します。

useroption
コード実行中に使用される値を示します。

/* JOBSPLIT:_DATA_ */
予約データセット名_DATA_を使用することを示します。

/* JOBSPLIT:_LAST_ */
予約データセット名_LAST_を使用することを示します。

/* JOBSPLIT:PROCNAME procname|DATASTEP */
このステップの SAS プロシジャの名前または DATA ステップを示します。

/* JOBSPLIT:LIBNAME <libname options> */
LIBNAME ステートメントに付与された、または内部設定された LIBNAME オプションを示します。

/* JOBSPLIT:SYSSCP <sysscp> */
SAS ジョブが実行されたときの SYSSC 自動マクロ変数の値を示します。

/* JOBSPLIT:JOBSTARTTIME <datetime> */
ジョブ開始時の日時を記録します。

/* JOBSPLIT:JOBENDTIME <datetime> */
ジョブ終了時の日時を記録します。

/* JOBSPLIT:TASKSTARTTIME <datetime> */
タスク開始時の日時を記録します。

/* JOBSPLIT:INTCON <table name> <INPUT | OUTPUT> NAME:<name>
TYPE:<UNIQUE|CHECK|NOTNULL|PRIMARY|FOREIGN|REFERENTIAL>
VARIABLES:<names> WHERE:<where clause>
REFERENCE:ONDELETE:ONUPDATE:MESSAGE:<msg> MESSAGETYPE:<USER>
***/**

table name
テーブルの名前を示します。

INPUT
テーブルが入力のために開かれたことを示します。

OUTPUT
テーブルが出力のために開かれたことを示します。

NAME
一貫性制約の名前を示します。

TYPE
一貫性制約のタイプを示します。

UNIQUE	unique
CHECK	check
NOTNULL	not null
PRIMARY	primary key
FOREIGN	foreign key
REFERENTIAL	a FOREIGN integrity constraint in another table that references this table

VARIABLES
一貫性制約に関連する変数を示します。その他の場合は空欄になります。

WHERE
一貫性制約に対し WHERE 表現を示します。その他の場合は空欄になります。

REFERENCE
次の一貫性制約の中の 1 つを示します。

- FOREIGN 一貫性制約については、外部キーが参照するテーブルを示します。
- REFERENTIAL 一貫性制約については、外部キーを含んでいるテーブルを示します。
- その他の場合は空欄になります。

ONDELETE
FOREIGN および REFERENTIAL 一貫性制約に対する ON DELETE 参照アクションを示します。その他の場合は空欄になります。

ONUPDATE
FOREIGN および REFERENTIAL 一貫性制約に対する ON UPDATE 参照アクションを示します。その他の場合は空欄になります。

MESSAGE
エラーメッセージのテキストがあれば、それを示します。

MESSAGETYPE
MESSAGETYPE=USER が指定された場合、USER を示します。その他の場合は空欄になります。

例: SCAPROC プロシジャ

例 1: 記録ファイルの指定

要素: RECORD ステートメント
WRITE ステートメント

この例では、記録ファイル'record.txt'を指定し、SAS コードアナライザから記録ファイルに情報を書き込みます。

プログラム

```
proc scaproc;
   record 'record.txt';
run;

data a;
   do i = 1 to 100000;
      j = cos(i);
      output;
   end;
run;

proc print data=a(obs=25);
run;

proc means data=a;
run;

proc scaproc;
   write;
run;
```

出力

アウトプット 57.1 *record.txt ファイルの内容*

```
/* JOBSPLIT:DATASET OUTPUT SEQ WORK.A.DATA */ /* JOBSPLIT:LIBNAME WORK ENGINE
V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS Temporary Files\_TD1252 */ /*
JOBSPLIT:ELAPSED 3984  */ /* JOBSPLIT:PROCNAME DATASTEP */ /* JOBSPLIT:STEP
SOURCE FOLLOWS */ data a; do i = 1 to 1000000; j = cos(i); output; end; run; /*
JOBSPLIT:ITEMSTOR INPUT SASUSER.TEMPLAT */ /* JOBSPLIT:ITEMSTOR INPUT
SASHELP.TMPLMST */ /* JOBSPLIT:DATASET INPUT SEQ WORK.A.DATA */ /*
JOBSPLIT:LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS
Temporary Files\_TD1252 */ /* JOBSPLIT:ELAPSED 5187  */ /* JOBSPLIT:PROCNAME
PRINT */ /* JOBSPLIT:STEP SOURCE FOLLOWS */ proc print data=a(obs=25); run; /*
JOBSPLIT:DATASET INPUT SEQ WORK.A.DATA */ /* JOBSPLIT:LIBNAME WORK ENGINE V9
PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS Temporary Files\_TD1252 */ /*
JOBSPLIT:FILE OUTPUT C:\winnt\profiles\userid\record.txt */ /* JOBSPLIT:SYMBOL
GET SYSSUMTRACE */ /* JOBSPLIT:ELAPSED 2750  */ /* JOBSPLIT:PROCNAME MEANS
*/ /* JOBSPLIT:STEP SOURCE FOLLOWS */ proc means data=a; run; /* JOBSPLIT:END */
```

例 2: グリッドジョブジェネレータの指定

要素: RECORD ステートメント
GRID ステートメント

詳細

この例では、SAS コードアナライザから **1.txt** という名前のファイルに情報を書き込みます。サンプルコードでは、グリッドジョブジェネレータも実行し、その情報を **1.grid** という名前のファイルに書き込みます。このサンプルコードには、次のコードを含む終了ステートメントはありません。

```
proc scaproc;
   write;
run;
```

SAS の終了時に、PROC SCAPROC により自動的に保留中の RECORD や GRID ステートメントが実行されます。

GRID ステートメントを実行するには、SAS Grid Manager または SAS/CONNECT のライセンスが必要です。SAS Grid Manager では、分散マシンのグリッドで生成グリッドジョブを実行できます。SAS/CONNECT では、1 つの対称型マルチプロセッシング(SMP)マシンの並列 SAS セッションで生成グリッドジョブを実行できます。

プログラム

```
proc scaproc;
   record '1.txt' grid '1.grid';
run;

data a;
   do i = 1 to 100000;
      j = cos(i);
      output;
   end;
run;

proc print data=a(obs=25);
run;

proc means data=a;
run;
```

出力

アウトプット 57.2 *1.txt ファイルの内容*

```
/* JOBSPLIT:DATASET OUTPUT SEQ WORK.A.DATA */ /* JOBSPLIT:LIBNAME WORK ENGINE
V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS Temporary Files\_TD1252 */ /*
JOBSPLIT:ELAPSED 375  */ /* JOBSPLIT:PROCNAME DATASTEP */ /* JOBSPLIT:STEP
SOURCE FOLLOWS */ data a; do i = 1 to 1000000; j = cos(i); output; end; run; /*
JOBSPLIT:DATASET INPUT SEQ WORK.A.DATA */ /* JOBSPLIT:LIBNAME WORK ENGINE V9
PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS Temporary Files\_TD1252 */ /*
JOBSPLIT:ELAPSED 46  */ /* JOBSPLIT:PROCNAME PRINT */ /* JOBSPLIT:STEP SOURCE
FOLLOWS */ proc print data=a(obs=25); run; /* JOBSPLIT:DATASET INPUT SEQ
WORK.A.DATA */ /* JOBSPLIT:LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid
\LOCALS~1\Temp\SAS Temporary Files\_TD1252 */ /* JOBSPLIT:FILE OUTPUT C:\WINNT
\Profiles\userid\1.txt */ /* JOBSPLIT:SYMBOL GET SYSSUMTRACE */ /*
JOBSPLIT:ELAPSED 81453  */ /* JOBSPLIT:PROCNAME MEANS */ /* JOBSPLIT:STEP
SOURCE FOLLOWS */ proc means data=a; run; /* JOBSPLIT:END */
```

58 章

SOAP プロシジャ

概要: SOAP プロシジャ

SOAP(Simple Object Access Protocol)プロシジャは、ファイル参照名を持つファイルから XML 入力を読み込み、別のファイル参照名を持つファイルへ XML 出力を書き込みます。メッセージコンポーネントである、サービス要求に対応する XML ドキュメントは、ファイル参照名のコンテンツの一部です。PROC SOAP の IN オプションで定義されます。入力 XML は、SOAPEnvelope 要素、またはウェブサービスの呼び出しに必要な SOAPEnvelope 内部の要素です。

概念: SOAP プロシジャ

PROC SOAP を使用して、XML ファイルに追加の SOAPEnvelope 要素を含めることができます。これは、SOAPHeader 要素にカスタム情報を含める場合に実行します。

SOAP エンベロープは、アプリケーション固有のメッセージ言語を持つメッセージをラップします。SOAPHeader コンテンツが実際の SAS 登録ウェブサービス要求に追加されます。この追加は、PROC SOAP に渡された XML ファイルには含まれていない、WS-Addressing または WS-Security 対応の要素を含む、追加の SOAPHeader 要素があるかもしれないため、実行されます。この場合、送信される XML コードと提供されるコードが完全には一致しない可能性があります。

要求は、SOAP エンベロープに含まれている必要はありません。エンベロープを指定しない場合、エンベロープが追加されます。エンベロープを指定する場合、送信されるエンベロープに組み込まれます。応答は、ENVELOPE プロパティが設定されている場合のみ、エンベロープ内に戻されます。デフォルトの動作では、エンベロープのコンテンツのみ返されます。

CONFIGFILE オプションを使用して、ウェブサービスからの応答を待機する時間を設定できます。デフォルトの待機時間は、60 秒です。

エンベロープが含まれている場合でも、要求にエンコーディング宣言を含めることはできません。要求をファイルから読み込む場合、セッションエンコーディングと同一のエンコーディングでファイルをエンコードする必要があります。要求はウェブサービスへ送信される前に、UTF-8 としてエンコードされます。

z/OS 固有

SAS 登録サービスの呼び出しは、z/OS 動作環境では使用できません。SAS 登録ウェブサービスには、WS-Security と Password Digest が必須であり、z/OS では Password Digest はサポートされていません。

構文: SOAP プロシジャ

制限事項: SAS がロック状態のときは、SOAP プロシジャは使用できません。サーバー管理者は、ロック状態でもこのプロシジャにアクセスできるように再有効化できます。FILENAME の場合、URL アクセスメソッドは LOCKDOWN ENABLE_AMS =ステートメントを使用して再有効化され、SOAP プロシジャは自動的に再有効化されます。詳細については、“SAS Processing Restrictions for Servers in a Locked-Down State” (*SAS Language Reference: Concepts*)を参照してください。

参照項目: の「LOCKDOWN ステートメント」 *SAS Intelligence Platform: Application Server Administration Guide*

PROC SOAP *option(s)* <*properties*>;

ステートメント	タスク	例
“PROC SOAP ステートメント”	SOAPEnvelope 要素でウェブサービスを呼び出します。	Ex. 1
	SOAPEnvelope 要素を使用せずに、ウェブサービスを呼び出します。	Ex. 2
	プロキシを使用してウェブサービスを呼び出します。	Ex. 3
	サービスレジストリサービス を使用して SAS 登録ウェブサービスを呼び出します。	Ex. 4
	SAS 環境ファイルを使用して SAS 登録ウェブサービスを呼び出します。	Ex. 5

ステートメント	タスク	例
	ウェブサービスの呼び出しのデフォルトタイムアウトを変更します。	Ex. 6

PROC SOAP ステートメント

Java ネイティブインターフェース(JNI)を介してウェブサービスを呼び出します。

構文

PROC SOAP *option(s)* <*properties*>;

オプション引数の要約

CONFIGFILE
ウェブサービス呼び出しのタイムアウト制限を設定します。

DEBUG
出力ログファイルを指定します。

ENVFILE
SAS 環境ファイルの場所を指定します。

ENVIRONMENT
SAS 環境ファイルで定義されている環境を使用するように指定します。

MUSTUNDERSTAND
mustUnderstand 属性の設定を指定します。

OUT=*ファイル参照名* '*your-output-file*'
応答出力のファイル参照名を指定します。

PROXYDOMAIN
HTTP プロキシサーバードメインを指定します。

PROXYHOST
HTTP プロキシサーバーのホスト名を指定します。

PROXYPASSWORD
HTTP プロキシサーバーのパスワードを指定します。

PROXYPORT
HTTP プロキシサーバーのポートを指定します。

PROXYUSERNAME
HTTP プロキシサーバーのユーザー名を指定します。

SOAPACTION
SOAPAction 要素を指定します。

SRSURL
System Registry Service の URL を指定します。

WEBAUTHDOMAIN
メタデータからユーザー名とパスワードの取得を行うように指定します。

WEBDOMAIN
ユーザー名とパスワードのドメインまたはレルムを指定します。

WEBPASSWORD
ウェブサービス認証のパスワードを指定します。

WEBUSERNAME
ウェブサービス認証のユーザー名を指定します。

WSSAUTHDOMAIN
SAS メタデータサーバー r へのアクティブな接続が認証情報の取得に使用されるように指定します。

WSSPASSWORD
WS-Security パスワードを指定します。

WSSUSERNAME
WS-Security ユーザー名を指定します。

必須引数

IN=*ファイル参照名* '*your-input-file*'
PROC SOAP 要求を含む XML データの入力に使用されるファイル参照名を指定します。

ファイル参照名には、コンテンツの一部として SOAPEnvelope 要素と SOAPHeader 要素が含まれる場合がありますが、指定する固有のヘッダー情報がないかぎり必須ではありません。

SERVICE
呼び出す SAS 登録ウェブサービスを指定します。

ヒント SERVICE オプションを使用する場合、URL オプションは使用しません。

URL
ウェブサービスエンドポイントの URL を指定します。

ヒント URL オプションを使用する場合、SERVICE オプションは使用しません。

オプション引数

CONFIGFILE
ウェブサービス呼び出しのタイムアウト制限を設定できます。デフォルトのタイムアウトは 60 秒です。

DEBUG
出力ログファイルを指定します。デバッグオプションは、httpclient wire ロギングを有効にし、出力を指定ファイルへ書き出します。このオプションの値は指定出力先へのパスまたはファイル名です。

ENVFILE
SAS 環境ファイルの場所を指定します。

ENVIRONMENT
SAS 環境ファイルで定義されている環境を使用するように指定します。

MUSTUNDERSTAND
PROC SOAP ヘッダーの mustUnderstand 属性の設定を指定します。

OUT=*ファイル参照名* '*your-output-file*'
PROC SOAP XML 応答出力が書き込まれるファイル参照名を指定します。

PROXYDOMAIN
HTTP プロキシサーバードメインを指定します。

ヒント このオプションは、お使いのプロキシサーバーで、ドメイン修飾またはレルム修飾の認証情報が必要な場合にのみ必須です。

PROXYHOST
HTTP プロキシサーバーのホスト名を指定します。

PROXYPASSWORD
HTTP プロキシサーバーのパスワードを指定します。PROC PWENCODE によって生成されるエンコーディングがサポートされています。

ヒント このオプションは、プロキシサーバーで認証情報が必要な場合のみ要求されます。

PROXYPORT
HTTP プロキシサーバーのポートを指定します。

PROXYUSERNAME
HTTP プロキシサーバーのユーザー名を指定します。

ヒント このオプションは、プロキシサーバーで認証情報が必要な場合のみ要求されます。

SOAPACTION
ウェブサービスを呼び出すための SOAPAction 要素を指定します。

SRSURL
System Registry Service の URL を指定します。

WEBAUTHDOMAIN
ユーザー名とパスワードを指定された認証ドメインのメタデータから取得するように指定します。

WEBDOMAIN
ユーザー名とパスワードのドメインまたはレルムを指定します。

WEBPASSWORD
基本ウェブサービス認証のパスワードを指定します。PROC PWENCODE によって生成されるエンコーディングがサポートされています。

WEBUSERNAME
基本ウェブサービス認証のユーザー名を指定します。

WSSAUTHDOMAIN
指定される認証ドメインで、SAS メタデータサーバーへのアクティブな接続を使用して認証情報を取得するように指定します。

認証情報が検出されると、WS-Security UsernameToken の認証情報として使用されます。

WSSUSERNAME
WS-Security ユーザー名を指定します。値が設定されると WS-Security が使用され、UsernameToken がウェブサービス要求と一緒に送信され、ユーザーの認証、セキュリティ、暗号化に使用されます。

WSSPASSWORD
WSSUSERNAME のパスワードである、WS-Security パスワードを指定します。PROC PWENCODE によって生成されるエンコーディングがサポートされています。

プロパティ

ENVELOPE
応答に SOAP エンベロープを含めるように指定します。

トランスポートレイヤーセキュリティ(TLS)を用いた PROC SOAP の使用

TLS とデータ暗号化

トランスポートレイヤーセキュリティ(TLS)とそれに先行するセキュアソケットレイヤー(SSL)は、データの暗号化により、保護された接続を経由してウェブブラウザとウェブサーバーが通信できるようにします。ブラウザとサーバーでは、データが送信前に暗号化されます。受信するブラウザまたはサーバーでは、データが処理前に解読されます。

注: TLS に関するすべての説明は、先行するプロトコルであるセキュアソケットレイヤー(SSL)にも当てはまります。

注: TLS には現在、認められていないテキストが暗号化されたデータストリームの最初または最後に追加されないようにする再ネゴシエーション機能があります。この機能は特に、証明書に基づいたクライアント認証により使用されます。デフォルトでは、この機能により、Java セキュアソケット拡張(JSSE)での TLS 再ネゴシエーションは無効になります。その結果、証明書に基づいたクライアント認証を必要とするウェブリソースに、透過プロキシを通じてアクセスしようとすると、次の Java TLS エラーメッセージが生成されます。

```
(javax.net.ssl.SSLException): HelloRequest followed by an
                                        unexpected handshake message
```

ただし、JSSE ライブラリが初期化される前に新しいシステムプロパティ

```
sun.security.ssl.allowUnsafeRenegotiation
```

を true に設定することで、Java での TLS の再ネゴシエーションを再有効化できます。

HTTPS プロトコルを使用して PROC SOAP 呼び出しを作成する

HTTPS プロトコルを使用して PROC SOAP 呼び出しを行うため、信頼されるサービスの認証を含む信頼ソースを設定する必要があります。jreoptions を使用して Java システムオプションを設定し、このトラストストアとそのパスワードを SAS セッションへ提供する必要があります。この情報は SAS コマンドラインまたは SAS 構成ファイルで指定できます。次の構文を使用します。次のエントリを 1 行で入力します。

```
-jreoptions (-Djavax.net.ssl.trustStore=full-path-to-the-trust-store
-Djavax.net.ssl.trustStorePassword=trustStorePassword)
```

次の例に、SAS コマンドラインのエントリの使用方法を示します。例では、Windows 動作環境を使用します。次のエントリを 1 行で入力します。

```
"C:\Program Files\SAS\SASFoundation\9.3\sas.exe" -CONFIG "C:\Program
Files\SAS\SASFoundation\9.3\nls\en\SASV9.CFG" -jreoptions
(- Djavax.net.ssl.trustStore=C:\Documents and Settings\mydir\.keystore
-Djavax.net.ssl.trustStorePassword=trustpassword)
```

SAS (R) ウェブサービスの呼び出し法

2 つの方法を使用して、SAS 登録ウェブサービスを呼び出すことができます。1 つめの方法では、サービスレジストリサービスの URL と呼び出すサービスのエンドポイントの URL を知っている必要があります。SRSURL オプションに Service Registry Service の URL を設定する必要があります。URL オプションは、呼び出すサービスのエンドポイントを示します。例については、“例 4: Service Registry Service を使用した SAS (R) ウェブサービスの呼び出し” (1781 ページ)を参照してください。

SAS 登録 Web サービスの呼び出しに使用される 2 つめの方法では、SAS 環境ファイルを使用して、呼び出すサービスのエンドポイントを指定します。この方法を使用する場合、次の 2 つの方法のうちのいずれかで SAS 環境ファイルの場所を指定できます。

- PROC SOAP の ENVFILE オプションを使用
- SAS コマンドラインまたは SAS 構成ファイルで、JREOPTIONS の Java プロパティ **env.definition.location** を定義

次の JREOPTIONS 構文を使用します。

```
-jreoptions (-Denv.definition.location=http://your-SAS-environment.xml)
```

ENVIRONMENT オプションを使用してそのファイル内に目的の環境を指定し、SERVICE オプションを使用して呼び出すサービスの名前を指定する必要もあります。例については、“例 5: SAS (R) 環境ファイルを使用した SAS ウェブサービスの呼び出し” (1781 ページ)を参照してください。

どちらの場合も、WSUSERNAME オプションと WSPASSWORD オプションは、Security Token Service と接続するために必要なユーザー名とパスワードに設定されます。

認証情報を提供しない SAS 保護サービスの呼び出し

SRSURL オプションまたは ENVFILE オプションまたは ENVIRONMENT オプションを使用すれば、一連の認証情報を提供しなくても SAS 保護サービスを呼び出せます。この場合、WSSUSERNAME オプションや WSSPASSWORD オプションや WSSAUTHDOMAIN オプションを使用する必要はありません。この機能により、ユーザー認証情報をメタデータに保存する必要がなくなり、ネットワーク上を流れる認証情報の量が減少します。メタデータサーバーへの接続が必要です。メタデータサーバーに接続されているユーザーに対して、有効使用回数が 1 回である認証情報が生成されます。

出力ログファイルの指定

ログファイルには、PROC SOAP での HTTP 要求の送信時にサーバー間で送受信された、HTTP ヘッダーとデータが含まれます。DEBUG オプションで作成されたログファイルを使用して、デバッグ出力を記録できます。

記録を有効にして、全 SAS セッションの SOAP 要求と応答を参照するには、–jreoptions コマンドラインオプションを使用して SAS を再起動する必要があります。

PROC SOAP では、要求と応答を log4j を使用して記録するため、トレースが可能です。発行される要求と受信される応答を含むログファイルを作成するには、次のコンテンツを含むファイルを作成します。

```
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=wire.log
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern =%d %5p [%c] %m%n

log4j.logger.httpclient.wire=DEBUG, FILE
```

SAS コマンドラインまたは SAS 構成ファイルで、–jreoptions を使用して Java システムオプションを設定し、記録を有効にします。次の構文に、システムオプションを設定する方法を示します。

```
-jreoptions (-Dlog4j.configuration=path-to-log4j-config-file)
```

次の例に、SAS コマンドラインのエントリの使用方法を示します。例では、Windows 動作環境を使用します。1 行でエントリを入力します。

```
"C:\Program Files\SAS\SASFoundation\9.2\sas.exe" -CONFIG "C:\Program
Files\SAS\SASFoundation\9.2\nls\en\SASV9.CFG" -jreoptions
(- Dlog4j.configuration=file:/c:/public/log4j.properties)
```

前述の構成ファイルと jreoptions メソッドを使用して、全 SAS セッションの記録が有効になります。個別の PROC SOAP 呼び出しに httpclient.wire を有効にするには、DEBUG オプションを使用します。

DEBUG オプションを使用して、PROC SOAP 呼び出し時の Wire ロギングを有効にできます。DEBUG オプションの値は出力ファイルへのパスまたはファイル名です。

例: SOAP プロシジャ

例 1: SOAPEnvelope 要素を使用した SOAP プロシジャ

詳細

この例は、WS-Security を必要とするサービスを呼び出します。エンベロープが提供されます。

プログラム

```
filename request 'c:\temp\simpleTest_REQUEST.xml';
filename response 'c:\temp\simpleTest_RESPONSE.xml';

data _null_;
   file request;
   input;
   put _infile_;
   datalines4;
```

```
<soapenv:Envelope xmlns:add="http://tempuri.org/addintegersWS"
                  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header/>
   <soapenv:Body>
      <add:addintegers>
         <add:parameters>
            <add:int1>20</add:int1>
            <add:int2>30</add:int2>
         </add:parameters>
      </add:addintegers>
   </soapenv:Body>
   </soapenv:Envelope>
;;;;
run;

%let response=response;
proc soap in=request
          out=&response
          url="http://localhost:8080/SASBIWS/services/addintegersWS"
          wssusername="user-name"
          wsspassword="password";
run;
```

例 2: SOAPEnvelope 要素を使用しない SOAP プロシジャ

詳細

この例では、“例 1: SOAPEnvelope 要素を使用した SOAP プロシジャ” (1778 ページ)で説明されているプロセスで呼び出されるサービスと同じサービスを呼び出しています。ここではこのサービスはエンベロープなしで呼び出されます。

プログラム

```
filename request 'c:\temp\simpleTest_REQUEST.xml';
filename response 'c:\temp\simpleTest_RESPONSE.xml';

data _null_;
   file request;
   input;
   put _infile_;
   datalines4;

<add:addintegers xmlns:add="http://tempuri.org/addintegersWS">
   <add:parameters>
      <add:int1>20</add:int1>
         <add:int2>30</add:int2>
   </add:parameters>
</add:addintegers
;;;;
run;

%let response=response;
proc soap in=request
          out=&response
```

```
                                    url="http://localhost:8080/SASBIWS/services/addintegersWS"
                                    wssusername="user-name"
                                    wsspassword="password";
                          run;
```

例 3: プロキシを使用したウェブサービスの呼出し

詳細

この例では外部ウェブサービスを呼び出しているため、プロキシを使用しています。

プログラム

```
filename request temp;
filename response "c:\temp\Output.xml";

data _null_;
   file request;
   input;
   put _infile_;
   datalines4;

<soapenv:Envelope

                xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                xmlns:ndf="http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl">

   <soapenv:Header/>
     <soapenv:Body>
          <ndf:NDFDgenByDay soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <latitude xsi:type="xsd:decimal">35.79</latitude>
                      <longitude xsi:type="xsd:decimal">-78.82</longitude>
          <startDate xsi:type="xsd:date">2006-10-03</startDate>
              <numDays xsi:type="xsd:integer">3</numDays>
          <format xsi:type="dwml:formatType"
                  xmlns:dwml="http://www.weather.gov/forecasts/xml/SWMLgen/schema/DWML.xsd">24 hourly</format>
        </ndf:NDFDgenByDay>
   </soapenv:Body>
</soapenv:Envelope>
;;;;

proc soap in=request
          out=response
          url="http://www.weather.gov/forecasts/xml/SOAP_server/ndfdXMLserver.php"
          soapaction="http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl#NDFDgenByDay"
          proxyhost="proxygw.abc.sas.com"
          proxyport=80;
run;
```

例 4: Service Registry Service を使用した SAS (R) ウェブサービスの呼び出し

詳細

この例は、サービスの URL とサービスレジストリサービスを使用して SAS 登録ウェブサービスを呼び出します。

プログラム

```
filename request temp;
filename response "c:\temp\output.xml";

data _null_;
   file request;
   input;
   put _infile_;
   datalines4;

<soapenv:Envelope xmlns:rep="http://www.sas.com/xml/schema/sas-svcs/reportrepository-9.2"
                  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
      <Action
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">
       http://www.sas.com/xml/schema/sas-svcs/reportrepository-9.2/DirectoryServiceInterface/isDirectory</Action>
  </soapenv:Header>
  <soapenv:Body>
     <rep:isDirectoryDirectoryServiceInterfaceRequest>
        <rep:dirPathUrl>SBIP:path-name</rep:dirPathUrl>
     </rep:isDirectoryDirectoryServiceInterfaceRequest>
  </soapenv:Body>
</soapenv:Envelope>
;;;;
run;

proc soap in=request out=response
   url="http://machine-name:port-number/SASWIPSoapServices/services/ReportRepositoryService"
   soapaction="http://www.sas.com/xml/schema/sas-svcs/reportrepository-9.2/DirectoryServiceInterface/isDirectory"
   srsurl="http://machine-name:port-number/SASWIPSoapServices/services/ServiceRegistry"
   wssusername="user-name"
   wsspassword="password";
run;
```

例 5: SAS (R) 環境ファイルを使用した SAS ウェブサービスの呼び出し

詳細

この例は、SAS 環境ファイルとテスト環境を使用して SAS 登録ウェブサービスを呼び出します。

プログラム

```
filename request temp;
filename response "c:\temp\output.xml";

data _null_;
   file request;
   input;
   put _infile_;
   datalines4;

<soapenv:Envelope xmlns:rep="http://www.sas.com/xml/schema/sas-svcs/reportrepository-9.2"
                  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
      <Action
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">
       http://www.sas.com/xml/schema/sas-svcs/reportrepository-9.2/DirectoryServiceInterface/isDirectory</Action>
  </soapenv:Header>
  <soapenv:Body>
     <rep:isDirectoryDirectoryServiceInterfaceRequest>
        <rep:dirPathUrl>SBIP:path-name</rep:dirPathUrl>
     </rep:isDirectoryDirectoryServiceInterfaceRequest>
  </soapenv:Body>
</soapenv:Envelope>
;;;;
run;

proc soap in=request out=response service="ReportRepositoryService"
soapaction="http://machine-name:port-number/SASWIPSoapServices/services/ReportRepositoryService"
                  envfile="http://file-server-name.abc.xyz.com/sas-environment.xml"
                  environment="test";
                  wssusername="user-name"
                  wsspassword="password";
run;
```

例 6: ウェブサービスの呼び出しに対するデフォルトタイムアウトの変更

詳細

この例では、CONFIGFILE オプションを使用して、SAS 登録ウェブサービスの応答を待機する時間をミリ秒単位で設定します。この例では、soTimeout 値は 20000 ミリ秒です。次のコンテンツを使用して構成ファイルを作成し、値を変更して異なるタイムアウトを設定できます。次の例は、C:Public ディレクトリに保存されている soap-client-config.xml ファイルを使用します。

プログラム

```
      /* This section of code is not part of the example. The code        */
      /* is the content of the file that is pointed to by the CONFIGFILE  */
      /* option in the PROC SOAP command. The soTimeout property that is  */
      /* defined in this file is what changes the timeout.                */

   <?xml version="1.0" encoding="UTF-8" ?>
```

```
-<beans xmlns="http://www.springframework.org/schema/beans"

   xmlns:aop="http://www.springframework.org/schema/aop"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
   http://www.springframework.org/schema/aop
          http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">
-<bean id="httpClientParams"
       class="org.apache.commons.httpclient.params.HttpClientParams">
       <property name="soTimeout" value="20000" />
  </bean>
</beans>



   /* This is the beginning of the example. */

filename request "c:\temp\AddInts_request.xml" ;
filename response "c:\temp\AddInts_response.xml" ;

data _null_;
   file request;
   input;
   put _infile_;
   datalines4;
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:add="http://tempuri.org/AddInts">
   <soapenv:Header/>
   <soapenv:Body>
      <add:addInts>
         <add:parameters>
            <add:int1>2</add:int1>
            <add:int2>3</add:int2>
         </add:parameters>
      </add:addInts>
   </soapenv:Body>
</soapenv:Envelope>
;;;;

proc soap;
   in=request;
   out=response;
   url="http://somehost.abc.xyz.com:8080/SASBIWS/services/AddInts"
   soapaction="http://tempuri.org/AddInts/addInts"
   configfile="c:\public\soap-client-config.xml";
run;
```

59 章

SORT プロシジャ

概要: SORT プロシジャ

SORT プロシジャの動作について

SORT プロシジャは、SAS データセットオブザベーションを 1 つ以上の文字変数または数値変数の値を基準にして並べ替えます。SORT プロシジャは、元のデータセットを置

き換えるか、新しいデータセットを作成します。PROC SORT は、出力データセットのみを作成します。詳細については、“プロシジャの出力” (1812 ページ)を参照してください。

注: 入力データセットに拡張属性が定義されている場合、PROC SORT は拡張属性を出力データセットに伝播します。拡張属性の詳細については、“拡張属性” (445 ページ)を参照してください。

動作環境の情報
本章で説明する並べ替え機能は、すべての動作環境で使用できます。また、SAS システムオプション SORTPGM=の HOST 値を使用すると、この他にも使用する動作環境で使用可能な並べ替えオプションを使用できる場合があります。他の並べ替え機能の詳細については、動作環境に関する SAS ドキュメントを参照してください。

SAS データセットの並べ替え

次の例では、元のデータセットは姓のアルファベット順に並べられていました。PROC SORT を実行すると、元のデータセットを従業員 ID 番号順に並べ替えられたデータセットに置き換えることができます。次のログは、この PROC SORT ステップの実行結果を示しています。アウトプット 59.1 (1786 ページ)アウトプット 55.1(1716 ページ)は、PROC PRINT ステップの結果を示しています。出力を生成するステートメントは次のとおりです。

```
proc sort data=employee;
   by idnumber;
run;

proc print data=employee;
run;
```

ログ 59.1 *PROC SORT により生成される SAS ログ*

```
NOTE:There were six observations read from the data set WORK.EMPLOYEE.NOTE:The data set WORK.EMPLOYEE
has six observations and three variables.NOTE:PROCEDURE SORT used: real time           0.01 seconds
cpu time            0.01 seconds
```

アウトプット 59.1 *1 つの変数の値によるオブザベーションの並べ替え*

```
The SAS System                          1 Obs    Name          IDnumber 1
Belloit         1988 2     Wesley          2092 3     Lemeux          4210
4     Arnsbarger      5466 5     Pierce          5779 6     Capshaw         7338
```

次の出力には、3 つの変数を基準にしたより複雑な並べ替えの結果が表示されます。この例の企業は、町、債務(金額の高い順)、アカウント番号の順に並べ替えられます。この出力を生成するプログラムの説明については、“例 2: 降順の並べ替え” (1815 ページ)を参照してください。

アウトプット 59.2 3 つの変数値によるオブザベーションの並べ替え

```
Customers with Past-Due Accounts                        1 Listed by Town, Amount, Account Number
Account Obs    Company                  Town              Debt      Number 1    Paul's
Pizza              Apex              83.00      1019 2    Peter's Auto Parts
Apex              65.79      7288 3    Watson Tabor Travel       Apex              37.95      3131
4    Tina's Pet Shop           Apex              37.95      5108 5    Apex Catering
Apex              37.95      9923 6    Deluxe Hardware           Garner           467.12      8941
7    Boyd & Sons Accounting    Garner           312.49      4762 8    World Wide Electronics
Garner           119.95      1122 9    Elway Piano and Organ     Garner            65.79      5217
10    Ice Cream Delight         Holly Springs    299.98      2310 11    Tim's Burger Stand
Holly Springs    119.95      6335 12    Strickland Industries     Morrisville      657.22      1675
13    Pauline's Antiques        Morrisville      302.05      9112 14    Bob's Beds
Morrisville      119.95      4998
```

概念: SORT プロシジャ

スレッド化された並べ替え

THREADS システムオプションを使用すると、スレッド化された並べ替えを有効にできます。スレッド化された並べ替えにより、処理操作における一定の並列処理が達成されます。この並列処理の目的は、所定の操作を完了するための処理時間を削減し、それによって追加 CPU リソースのコストを抑えることです。詳細については、“Support for Parallel Processing” (*SAS Language Reference: Concepts*)を参照してください。

PROC SORT ステートメントで THREADS オプションを指定している場合、マルチスレッド化された SAS 並べ替えも呼び出すことができます。マルチスレッド化された並べ替えでは、UTILLOC=システムオプションで指定された場所の 1 つにある単一のユーティリティファイルにすべての一時データが格納されます。このユーティリティファイルのサイズは、入力データセットから読み込まれるデータの量に比例します。入力データセットから読み込まれるデータ量が大きい場合、または SORT プロシジャで使用できるメモリの量が小さい場合は、同じサイズの 2 つ目のユーティリティファイルを別の場所に作成することができます。詳細については、“UTILLOC= System Option” (*SAS System Options: Reference*)を参照してください。

注: TAGSORT オプション (1806 ページ)は、スレッド化された並べ替えをサポートしていません。

マルチスレッド化された SAS 並べ替えは、THREAD システムオプションが指定され、CPUCOUNT=システムオプションの値が 1 より大きい場合に呼び出すことができます。SAS システムオプション CPUCOUNT=の値はスレッド化された並べ替えのパフォーマンスに影響します。CPUCOUNT=は、スレッド化されたプロシジャで使用できるシステム CPU の数を示します。

詳細については、“THREADS System Option” (*SAS System Options: Reference*) および “CPUCOUNT= System Option” (*SAS System Options: Reference*)を参照してください。

数値変数の並べ替え順序

数値変数について、昇順の比較シーケンスを次に示します。

1. SAS 欠損値(ピリオドや特殊欠損値で表示)
2. 負数値

3. ゼロ
4. 正数値

文字変数の並べ替え順序

デフォルトの照合シーケンス

英数字の並べ替え順序は、照合シーケンスと呼ばれます。この並べ替え順序は、セッションエンコーディングによって決定されます。

デフォルトで PROC SORT は、文字値を比較する際、プロシジャの実行環境に応じて EBCDIC 照合シーケンスと ASCII 照合シーケンスのいずれかを使用します。

さまざまな照合シーケンスとそれらの使用タイミングについては、“Collating Sequence” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

注: ASCII と EBCDIC は、セッションエンコーディングのファミリ名を表します。並べ替え順序は、エンコーディングを参照して決定できます。

EBCDIC 順序

z/OS 動作環境では、EBCDIC 照合シーケンスが使用されます。

英語の EBCDIC シーケンスの並べ替え順序は、次の並べ替え順序の例と一致しています。

表 59.1 *EBCDIC 並べ替え順序の例*

blank< (+ | & !$ *) ; ¬ - / , % _ > ?:## @ ' = "
a b c d e f g h i j k l m n o p q r ~ s t u v w x y z
{ A B C D E F G H I } J K L M N O P Q R \S T
U V W X Y Z
0 1 2 3 4 5 6 7 8 9

EBCDIC シーケンスの主な特徴は、小文字の後に大文字が並べ替えられ、その後に数字が並べ替えられることです。また、アルファベット順序の途中に特殊文字がいくつか割り込んでいるので注意してください。表示可能な最小文字はブランクです。

ASCII 順序

ASCII 照合シーケンスを使用する動作環境には、次のものがあります。

- UNIX とその派生 OS
- Windows
- OpenVMS

英語の ASCII シーケンスは、表示可能な文字を昇順に並べると、次の表に示す順序と一致します。

表 59.2 ASCII 並べ替え順序の例

blank !"# $ % & ' ( ) * + , - ./0 1 2 3 4 5 6 7 8 9:; < = > ?@
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z[\] ◦_
a b c d e f g h i j k l m n o p q r s t u v w x y z { } ~

ASCII シーケンスの主な特徴は、数字の後に大文字が並べ替えられ、その後に小文字が並べ替えられることです。表示可能な最小文字はブランクです。

文字変数の並べ替え順序の指定

オプション EBCDIC、ASCII、NATIONAL、DANISH、SWEDISH、REVERSE を使用すると、HOST カタログに格納されている照合シーケンスをを指定できます。

独自の照合シーケンスの作成や提供されている照合シーケンスの変更を行う必要がある場合は、TRANTAB プロシジャを使用して変換テーブルを作成または変更します。独自の変換テーブルを作成すると、その変換テーブルは PROFILE カタログに格納され、HOST カタログの同名の変換テーブルよりも優先されます。詳細については、"TRANTAB" (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

言語照合では、言語のルールに従ってデータが並べ替えられます。言語照合の詳細については、"Collating Sequence" (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

注: システム管理者は、新規作成したテーブルを PROFILE カタログから HOST カタログにコピーして、HOST カタログを変更できます。その場合、すべてのユーザーが新規または修正された変換テーブルにアクセスできます。

格納された並べ替え情報

PROC SORT は、データセットの並べ替えに使用する BY 変数、照合シーケンスおよび文字セットを記録します。この情報は、不要な並べ替えを回避するためにデータセットとともに格納されます。

PROC SORT は、データセットを並べ替える前に、格納されている並べ替え情報を確認します。現在の並べ替え方法でデータセットを並べ替えようとすると、PROC SORT は並べ替えを実行せず、その旨を示すメッセージをログに書き込みます。この動作を無効にするには、FORCE オプションを使用します。現在の並べ替えと同じ方法でデータセットを並べ替える場合に OUT=データセットを指定しても、PROC SORT は DATA=データセットのコピーを作成するだけです。

PROC SORT が格納した並べ替え情報を上書きするには、_NULL_値と SORTEDBY=データセットオプションを使用します。参照してください"SORTEDBY= Data Set Option" (*SAS Data Set Options: Reference*).

既存データセットの並べ替え情報を変更する場合は、DATASETS プロシジャの MODIFY ステートメントで SORTEDBY=データセットオプションを使用します。詳細については、"MODIFY ステートメント" (508 ページ)を参照してください。

データセットと格納されている並べ替え情報にアクセスするには、PROC DATASETS で CONTENTS ステートメントを使用します。詳細については、"CONTENTS ステートメント" (472 ページ)を参照してください。

PROC SORT でデータセットを並べ替えるときの基準となる変数の数は、利用可能なメモリでのみ制限されます。PROC SQL を使用して結果セットの行を並べ替えるときの

基準となる列の数も利用可能なメモリでのみ制限されます。ソートインジケータは、基本データセットのメタデータに格納されるか、メモリで表されるかにかかわらず、127 個の変数に制限されます。このため、ソートインジケータに格納できるまたは SORTEDBY=データセットオプションでリストできる変数は最大 127 個です。127 を超える数の変数で並べ替えすると、最初の 127 個だけがソートインジケータに記録されます。データセットを BY 変数のリスト全体でもう一度並べ替えると、(127 を超える)追加の変数はソートインジケータ内に見つからないため、データセットは並べ替えられているものと認識されません。詳細については、“What Is a Sort Indicator?” (*SAS Language Reference: Concepts*)を参照してください。

事前に並べ替えられた入力データセット

“PRESORTED” (1804 ページ) オプションを指定すると、すでに並べ替えられたデータセットは並べ替えられません。並べ替えの前に、入力データセット内のオブザベーションの順序がチェックされ、オブザベーションが順番どおりかどうかが決定されます。データセットがすでに BY ステートメントで指定したキー変数に従った順序であることがわかっている、またはそう思われる場合に、PRESORTED オプションを使用します。データセット内のオブザベーションの順序をチェックするには、データセットを読み取り、読み取った各オブザベーションの BY 変数と、その前のオブザベーションの BY 変数を比較します。この処理は、すべてのデータセットが読み取られるか、順序の外れたオブザベーションが検出されるまで続行されます。

データセットをすべて読み取っても順序を外れたオブザベーションが見つからなかった場合は、2 つのアクションのどちらかが実行されます。出力データセットを指定していない場合は、入力データセットの並べ替え順序メタデータが更新され、順序が検証済みであることが示されます。この検証には、データセットが指定された BY 変数に従って正当に並べ替えられていることが示されます。これとは違って、オブザベーション順序が検証済みで出力データセットを指定している場合は、入力データセットのオブザベーションが出力データセットにコピーされます。出力データセットのメタデータに、データが BY 変数に従って正当に並べ替えられていることが示されます。

データセット内のオブザベーションが順序どおりでない場合、データセットが並べ替えられます。

“NODUPKEY” (1803 ページ) オプションを指定した場合は、順序チェックによって、データセットにキーが重複するオブザベーションが存在するかどうかが判定されます。指定しないと、NODUPKEY オプションを指定して、キーの重複するオブザベーションが検出された場合、入力データセットは並べ替えの対象外と判断されます。

入力データセットのメタデータで、データがすでに BY ステートメントに表示されるキー変数に従って並べ替え済みで、その入力データセットが検証済みであることが示されている場合は、順序チェックも並べ替えも実行されません。

“Sorted Data Sets” (*SAS Language Reference: Concepts*) および“SORTVALIDATE System Option” (*SAS System Options: Reference*)との相互作用を参照してください。

データセットの言語ソートと ICU

言語照合では、言語とロケールに関連付けられたルールに従って、文化的に配慮した方法で文字が並べ替えられます。ルールとデフォルトの照合シーケンスは、現在のロケール設定で指定された言語に基づきます。この実装は、International Components for Unicode (ICU)ライブラリにより提供されます。その結果、Unicode 照合アルゴリズム(UCA)との互換性が高くなります。

SAS は、Base SAS プロシジャの PROC SORT で言語オプション (SORTSEQ=LINGUISTIC)が指定されている場合に ICU 照合を提供します。SAS 9.4 のメンテナンスリリース 3 より、SQL プロシジャで SORTSEQ=オプションを使用し、

SORTSEQ=LINGUISTIC システムオプションを指定して言語照合を指定できるようになりました。

注: SORTSEQ=LINGUISTIC システムオプションを指定した場合、PROC SORT および PROC SQL のみ影響を受けます。

SORTSEQ=LINGUISTIC オプションを指定すると、SAS は Unicode 照合アルゴリズム (UCA)の参照実装として、またデファクトスタンダードとして ICU ライブラリに依存します。UCA アルゴリズムまたは International Components for Unicode (ICU)ライブラリ実装の詳細については、Download the ICU 4.8 Release および CLDR 2.0 Release Note を参照してください。

SAS 9.4 では、SAS で組み込まれ、PROC SORT で使用されている ICU ライブラリがバージョン 4.8.1 にアップグレードされます。この新しい ICU バージョンでは、バージョン 2.0 の Unicode Common Locale Data Repository (CLDR)のロケールデータを使用します。

言語照合のために PROC SORT で使用されている ICU のバージョンを変更すると、別のバージョンの SAS で並べ替えられたデータセットの解釈に影響する可能性があります。2 つの SAS バージョンが異なるバージョンの ICU を使用しているときに、一方のバージョンの SAS でデータセットが 1 つ以上の文字変数により言語ソートされている場合、そのたデータセットはもう一方のバージョンの SAS でアクセスされたときに並べ替えられているものと認識されます。照合ルールは ICU バージョン間で変更されることがあるため、ルールの変更によって、PROC SORT で作成されたオブザベーションの順序が異なるものになる場合があります。順序の違いが無視されると、処理中に予期しない結果が生じる可能性があります。

言語ソート時に、SAS で使用されている ICU バージョンは、データセットヘッダーに格納されているソートインジケータに記録されます。データセットが並べ替え済みと見なすかどうかを判定するときに、ICU バージョンが確認されます。使用中の ICU バージョンと、データセットのソートインジケータに記録された ICU バージョンに違いがあると、SAS システムは示された並べ替え順序を無視し、データセットが並べ替えられていないものと見なします。

注: PROC CONTENTS 出力には、使用中の ICU バージョンが示されます。

永久データセットのソートインジケータを無視する場合は、処理を容易にするため、順序を再度指定し、データセットでソートインジケータを再確立した方が望ましい場合があります。これを行うには、PRESORTED オプションを指定した PROC SORT を使用します。多くの場合、データセット内のオブザベーションの順序は乱れておらず、正しい可能性があるため、SORT プロシジャは完全な並べ替えを実行しなくても、データセットを順番に読み込むだけでインジケータを再確立できます。オブザベーションの順序が正しくない場合、SORT プロシジャが必要に応じてオブザベーションを並べ替えます。

COPY プロシジャと MIGRATE プロシジャの両方で、入力データセットで記録された ICU バージョンが SAS システムで使用中のバージョンと異なる場合、入力データセットのソートインジケータは無視され、出力データセットは並べ替え済みとしてマークされず、メッセージが SAS ログに書き込まれます。ただし、どちらのプロシジャも、入力から読み込んだときと同じ順序でオブザベーションを出力データセットに書き込みます。物理的な順序が OUT=出力先ライブラリに使用されているエンジンでサポートされている場合、この順序は保持されます。このため、新しいリリースの SAS に移行するときは、PRESORTED オプションを指定した PROC SORT を使用して永久データセットの並べ替え順序を再確立することを検討してください。

SAS での言語照合の使用方法の詳細については、次のドキュメントおよび PROC SORT SORTSEQ=LINGUISTIC システムオプションで説明されています。

- “SORTSEQ=sort-table” (*SAS SQL Procedure User's Guide*)を参照してください。
- PROC SORT オプション“ LINGUISTIC<*(collating-options)*>” (1796 ページ)を参照してください。

- “Specifying Linguistic Collation” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。
- 13 章, “CONTENTS プロシジャ,” (401 ページ)を参照してください。
- 14 章, “COPY プロシジャ,” (409 ページ)を参照してください。
- 関連項目 38 章, “MIGRATE プロシジャ,” (1176 ページ)
- 付録 4, “ICU ライセンス” (2187 ページ)を参照してください。

次の SAS の資料には、言語照合に関する詳細な情報が記載されています。

- Creating Order out of Character Chaos:Collation Capabilities of the SAS System
- A Sampler of What's New in Base SAS 9.2
- Linguistic Collation:Everyone Can Get What They Expect
- Processing Multilingual Data with the SAS 9.2 Unicode Server
- New Language Features in SAS 9.2 for the Global Enterprise

次は、言語照合の詳細について参照する必要のあるサードパーティの資料のリストです。

- Unicode Collation Algorithm (UCA) Specification を参照してください。
- ICU User Guide の Collation セクションを参照してください。
- ICU Locale Explorer に表示される照合ルールの詳細については、ICU Locale Explorer を参照してください。Locale Explorer では、各種照合オプションを使用して単語のリストを並べ替えることができるデモを実行できます。詳細については、Collation Rules for English (United States)を参照してください。

構文: SORT プロシジャ

要件 BY ステートメント

ヒント: ステートメント ATTRIB、FORMAT、LABEL、WHERE を PROC SORT プロシジャと使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

発生するデータベース内処理に対し、データは SAS データベース内処理用に適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、“データベース内処理:PROC SORT” (1810 ページ)を参照してください。

参照項目: “SORT Procedure: Windows” (*SAS Companion for Windows*), “SORT Procedure: z/OS” (*SAS Companion for z/OS*), “SORT Procedure: UNIX” (*SAS Companion for UNIX Environments*).

PROC SORT <*collating-sequence-option*> <*other option(s)*>;
BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …>;
KEY *variable(s)* </ *option*>;

ステートメント	タスク	例
“PROC SORT ステートメント”	SAS データセットオブザベーションを 1 つ以上の文字変数または数値変数の値を基準にして並べ替えます	Ex. 1, Ex. 3, Ex. 4

ステートメント	タスク	例
“BY ステートメント”	並べ替え変数を指定します。	Ex. 1, Ex. 2, Ex. 4
“KEY ステートメント”	並べ替えキーと変数を指定します。	

PROC SORT ステートメント

SAS データセットオブザベーションを 1 つ以上の文字変数または数値変数の値を基準にして並べ替えます。

例: “例 1: 複数の変数の値によって並べ替える” (1813 ページ)
“例 3: BY グループ内でオブザベーションの相対順序を維持する” (1817 ページ)
“例 4: 各 BY グループの最初のオブザベーションを維持する” (1820 ページ)

構文

PROC SORT <*collating-sequence-option*> <*other option(s)*>;

オプション引数の要約

ASCII
ASCII を指定します。

DATA=*SAS-data-set*
入力データセットを指定します。

DATECOPY
作成日と変更日を変更せずに SAS データセットを並べ替えます。

FORCE
強制的に冗長な並べ替えを行います。

OVERWRITE
置換出力データセットが作成される前に、入力データセットを削除します。

PRESORTED
データセットがすでに並べ替えられた可能性があるかどうかを指定します。

REVERSE
文字変数の照合シーケンスを逆順にします。

SORTSIZE=*memory-specification*
利用可能なメモリを指定します。

TAGSORT
一時的なディスク使用量を減らします。

Create output data sets

DUPOUT=*SAS-data-set*
重複オブザベーションの書き込み先となる出力データセットを指定します。

OUT= *SAS-data-set*
出力データセットを指定します。

UNIQUEOUT= *SAS-data-set*
除外されたオブザベーションの出力データセットを指定します。

Eliminate duplicate observations

NODUPKEY
: BY 値が重複するオブザベーションを削除します。

Eliminate unique observations

NOUNIQUEKEY
: 一意のソートキーがある出力データセットからオブザベーションを除外します。

Override SAS system option THREADS

NOTHREADS
: スレッド化された並べ替えは実行されません。

THREADS | NOTHREADS
: スレッド化された並べ替えの有効化を可能にするか行わないようにします。

Specify the collating sequence

DANISH
: デンマーク語を明記

EBCDIC
: EBCDIC を指定します。

FINNISH
: フィンランド語を指定します。

NATIONAL
: カスタマイズされた順序を指定します。

NORWEGIAN
: ノルウェー語を指定します。

POLISH
: ポーランド語を指定します。

SORTSEQ= *collating-sequence*
: 照合シーケンスを指定します。

SWEDISH
: スウェーデン語を指定します。

Specify the output order

EQUALS | NOEQUALS
: BY グループ内の相対順序を指定します。

NOEQUALS
: BY グループ内の相対順序を保持しません。

Collating-Sequence-Options

動作環境の情報

DANISH、FINNISH、NORWEGIAN、SWEDISH*collating-sequence-option* の動作環境固有の動作については、動作環境に関する SAS ドキュメントを参照してください。PROC SORT ステップでは、

collating-sequence-option は 1 つだけ、*other options* は複数指定できます。この 2 種類のオプションの順序は問われません。

DANISH

デンマーク語とノルウェー語の規則に従って文字を並べ替えます。

デンマーク語とノルウェー語の照合シーケンスを 図 59.1 (1796 ページ)に示します。

ASCII
ASCII 照合シーケンスを使用して文字変数を並べ替えます。EBCDIC がネイティブ照合シーケンスであるシステムで ASCII 順序に並べ替える場合のみ、このオプションが必要になります。

参照項目 “文字変数の並べ替え順序 ” (1788 ページ)

EBCDIC
EBCDIC 照合シーケンスを使用して文字変数を並べ替えます。ASCII がネイティブ照合シーケンスであるシステムで EBCDIC 順序に並べ替える場合のみ、このオプションが必要になります。

参照項目 “文字変数の並べ替え順序 ” (1788 ページ)

POLISH
ポーランド語の規則に従って文字を並べ替えます。

FINNISH
フィンランド語とスウェーデン語の規則に従って文字を並べ替えます。フィンランド語とスウェーデン語の照合シーケンスを 図 59.1 (1796 ページ)に示します。

NATIONAL
インストール時に設定した、各国の国別規則を反映した代替照合シーケンスを使用して文字変数を並べ替えます。このオプションを使用するには、カスタマイズされた各国並べ替え順序をサイトに定義しておく必要があります。サイトの SAS 導入担当者に連絡して、カスタマイズされた各国並べ替え順序が使用可能かどうかを確認します。

NORWEGIAN
“DANISH” (1794 ページ)を参照してください。

SWEDISH
次を参照してください。“FINNISH” (1795 ページ)

SORTSEQ= *collating-sequence*
表示されている照合シーケンス(ASCII、EBCDIC、DANISH、FINNISH、ITALIAN、NORWEGIAN、POLISH、SPANISH、SWEDISH、NATIONAL)、その他のシステム提供変換テーブル名(POLISH、SPANISH)、ユーザー作成変換テーブル名のいずれかを指定します。エンコーディングを指定できます。また、ロケールに適した照合シーケンスを実行するために、キーワード LINGUISTIC または UCA を指定できます。*collating-sequence* には collating-sequence-option、変換テーブル、エンコーディング、キーワード LINGUISTIC を指定できます。指定できる照合シーケンスは、1 つだけです。詳細については、“Collating Sequence” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

次に、照合シーケンスについて説明します。

collating-sequence-option* | *translation_table
変換テーブル(SAS の提供する変換テーブルかユーザー定義変換テーブルが可能)、または PROC SORT ステートメント *collating-sequence-options* のいずれかを指定します。PROC SORT ステートメント“Collating-Sequence-Options” (1794 ページ)を参照してください。PROC TRANTAB および PROC SORT の SORTSEQ=との使用例については、“Using Different Translation Tables for Sorting” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

使用可能な変換テーブルは次のとおりです。

- ASCII
- DANISH

- EBCDIC
- FINNISH
- ITALIAN
- NORWEGIAN
- POLISH
- REVERSE
- SPANISH
- SWEDISH

次の図に、各言語の英数字の並べ替え方法を示します。

図 59.1 *英数字の各国照合シーケンス*

```
Danish:     0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅabcdefghijklmnopqrstuvwxyzæøå
Finnish:    0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖabcdefghijklmnopqrstuvwxyzåäö
Italian:    0123456789AÀBCÇDEÉÈFGHIÌJKLMNOÒPQRSTUÙVWXYZaàbcçdeéèfghiìjklmnoòpqrstuùvwxyz
Norwegian:  0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅabcdefghijklmnopqrstuvwxyzæøå
Spanish:    0123456789AÁaáBbCcDdEÉeéFfGgHhIÍiíJjKkLlMmNnÑñOÓoóPpQqRrSsTtUÚuúÜüVvWwXxYyZz
Swedish:    0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖabcdefghijklmnopqrstuvwxyzåäö
```

制限事項 PROC SORT ステップでは *collating-sequence-option* は 1 つしか指定できません。

操作 SORTSEQ=オプションを指定した場合、データベース内処理行われません。

ヒント SORTSEQ= *collating-sequence* オプションはかっこなしで指定され、引数は関連付けられません。照合シーケンスの指定例は次のとおりです。

```
proc sort data=mydata SORTSEQ=ASCII;
```

encoding-value

エンコーディング値を指定します。結果は、指定エンコーディングで表される文字データのバイナリ照合と同じです。サポートされているエンコーディング値については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

制限事項 PROC SORT は、SORTSEQ=オプションに指定されたエンコーディングを認識する唯一のプロシジャまたは SAS システムの一部です。

ヒント エンコーディング値に英数字やアンダースコア以外の文字が含まれる場合は、その値を引用符で囲む必要があります。

参照項目 *SAS 各国語サポート(NLS): リファレンスガイド*で指定可能な エンコーディングのリスト

LINGUISTIC*<(collating-options)>*

言語照合を指定します。言語照合では、言語とロケールに関連付けられたルールに従い、文化的に配慮した方法で文字が並べ替えられます。ルールとデフォルトの *collating-sequence* オプションは、現在のロケール設定で指定された言語に基づきます。この実装は、International Components for Unicode (ICU)ライブラリにより提供されます。その結果、Unicode 照合アルゴリズム(UCA)との

互換性が高くなります。詳細については、“ データセットの言語ソートと ICU” (1790 ページ)を参照してください。

注: SAS のメンテナンスリリース 3 では、SORTSEQ=システムオプションで言語照合がサポートされるようになりました。ただし、システムオプションで言語照合を指定した場合、PROC SORT と PROC SQL のみ影響を受けます。

SORTSEQ=LINGUISTIC を指定するときに使用できるオプションを次に示します。これらのオプションで、言語照合シーケンスを変更します。

ALTERNATE_HANDLING=SHIFTED
スペース、句読点、記号などの可変文字の処理を制御します。このオプションを指定しない場合(デフォルト値 Non-Ignorable を使用)、これらの可変文字間の違いと通常の文字間の違いの重要性が同じになります。ALTERNATE_HANDLING オプションを指定すると、これらの可変文字の重要性が低くなります。

デフォルト NON_IGNORABLE

ヒント SHIFTED 値は、多くの場合、Quaternary に設定した STRENGTH=と組み合わせて使用されます。このような場合、文字列の比較時に、スペース、句読点および記号が考慮されますが、それは文字列のその他の側面(基本文字、アクセント、および大文字と小文字の区別)がすべて一致する場合に限ります。

参照項目 “例 5: ALTERNATE_HANDLING=を使用した言語ソート” (1822 ページ) および “例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語ソート” (1824 ページ)

CASE_FIRST=
大文字と小文字の順序を指定します。この引数は、TERTIARY、QUATERNARY または IDENTICAL レベルにのみ有効です。次の表に、CASE_FIRST 引数の値と情報を示します。

表 59.3 *CASE_FIRST=の引数*

値	説明
UPPER	大文字、小文字の順に並べ替えます。
LOWER	小文字、大文字の順に並べ替えます。

COLLATION=
文字の順序を指定します。次の表に、指定可能な COLLATION=値を示します。

注: 照合値を選択しない場合は、ユーザーの locale-default 照合値が選択されます。

表 59.4 *COLLATION=の値*

値	説明
BIG5HAN	ラテン語の場合はピンイン順序を指定し、中国語、日本語、韓国語文字の場合は bug5 文字セット順序を指定します。

値	説明
DIRECT	ヒンディー語の異形を指定します。
GB21312HAN	ラテン語の場合はピンイン順序を指定し、中国語、日本語、韓国語文字の場合は gb2312han 文字セット順序を指定します。
PHONEBOOK	文字の順序について電話帳スタイルを指定します。PHONEBOOK はドイツ語でのみ選択します。
PINYIN	文字ごとのピンインへの音訳に基づいて、中国語、日本語、韓国語文字の順序を指定します。通常、この順序は簡体字中国語で使用されます。
POSIX	Portable Operating System Interface です。このオプションでは、“C”ロケールの文字順序を指定します。
STROKE	非アルファベット書き込みスタイルの文字順序を指定します。STROKE は中国語、日本語、韓国語、またはベトナム語で選択します。通常、この順序は繁体字中国語で使用されます。
TRADITIONAL	文字の順序に対して従来のスタイルを指定します。たとえば、スペイン語で TRADITIONAL を選択します。

LOCALE= ***locale_name***
POSIX 名の形式(ja_JP など)でロケール名を指定します。PROC SORT でサポートされているロケール値と POSIX 値のリストについては、“LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。

制限事項 次のロケールは PROC SORT でサポートされません。

- Afrikaans_SouthAfrica, af_ZA
- Cornish_UnitedKingdom, kw_GB
- ManxGaelic_UnitedKingdom, gv_GB

NUMERIC_COLLATION=
テキスト内の整数値を、数字を表すために使用されている文字の代わりに、数値によって並べ替えます。

表 59.5 *NUMERIC_COLLATION の値*

値	説明
ON	数字を数値順に並べ替えます。たとえば、"8 Main St."を"45 Main St."より前に並べ替えます。
OFF	数字を文字値順に並べ替えます。たとえば、"45 Main St."を"8 Main St."より前に並べ替えます。

デフォルト OFF

STRENGTH=
強さの値は、照合レベルと関連しています。照合レベル値は 5 つあります。次の表に、この 5 レベルに関する情報を示します。強さのデフォルト値は、ロケールと関連しています。

表 59.6 *STRENGTH=の値*

値	照合の種類	説明
PRIMARY または 1	PRIMARY は、基本文字の違いを指定します("a" < "b"など)。	これは最も大きな違いです。たとえば、辞書は基本文字ごとに異なるセクションに分かれています。
SECONDARY または 2	文字のアクセントが第 2 レベルの違いとみなされます("as" < "às" < "at"など)。	文字列に第 1 レベルの違いがある場合、第 2 レベルの違いは無視されます。言語によっては、その他の文字の違いも第 2 レベルの違いとみなされることがあります。
TERTIARY または 3	第 3 レベルでは、大文字と小文字の違いが区別されます("ao" < "Ao" < "aò"など)。例については、“例 5: ALTERNATE_HANDLING=を使用した言語ソート” (1822 ページ)を参照してください。	文字列に第 1 レベルや第 2 レベルの違いがある場合、第 3 レベルの違いは無視されます。もう 1 つの例としては、仮名の大文字と小文字の違いがあります。
QUATERNARY または 4	レベル 1 から 3 までで句読点を無視する場合、追加のレベルを使用して句読点のある単語とない単語を区別できます("a-b" < "ab" < "aB"など)。例については、“例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語ソート” (1824 ページ)を参照してください。	句読点を無視する必要がある場合や、日本語のテキストを処理する場合に、第 4 レベルを使用してください。第 1 レベル、第 2 レベルまたは第 3 レベルの違いがある場合、この違いは無視されます。

値	照合の種類	説明
IDENTICAL または 5	これ以外のすべてのレベルが等しい場合、同一レベルをタイブレーカーとして使用します。レベル 1 から 4 までに違いがない場合は、各文字列の正規化形式 D (NFD)形式の Unicode コードポイント値がこのレベルで比較されます。	2 つの文字列でコードポイント値が異なることはほとんどないため、このレベルは慎重に使用する必要があります。たとえば、ヘブライ語の朗誦記号はこのレベルでのみ区別できます。

別名 LEVEL=

別名 UCA

制限事項 SORTSEQ=LINGUISTIC オプションは、PROC SORT SORTSEQ=オプションでのみ使用可能で、SAS システム SORTSEQ=オプションには使用できません。

言語照合は、プラットフォーム VMS on Itanium (VMI)または 64-bit Windows on Itanium (W64)ではサポートされていません。

操作 ICU バージョンは新しい SAS リリースで変更される可能性があります。2 つのリリースが別のバージョンの ICU を使用しているときに、一方の SAS リリースを使用してデータセットの言語ソート時に作成されるオブザベーションの順序は、もう一方のリリースで作成される順序と異なる場合があります。新しいリリースの SAS に移行するときは、PRESORTED オプションを指定した PROC SORT を使用して永久データセットの並べ替え順序を再確立することを検討してください。詳細については、“ データセットの言語ソートと ICU” (1790 ページ)を参照してください。

ヒント CONTENTS プロシジャまたは CONTENTS ステートメントの出力に、言語ソートされたデータセットの ICU バージョン番号が示されます。

collating-options は、かっこで囲む必要があります。複数の照合オプションを指定できます。

言語照合によって並べ替えられたデータセットで BY 処理を実行する場合、データセットを適切に処理するために NOBYSORTED システムオプションの指定が必要になる場合があります。BY 処理の実行内容は、照合シーケンス処理とは異なります。

参照項目 ICU ライセンス契約については、付録 4, “ICU ライセンス” (2187 ページ)を参照してください。

詳細については、“Specifying Linguistic Collation” (*SAS National Language Support (NLS): Reference Guide*)を参照してください。詳細につ

いては“ データセットの言語ソートと ICU” (1790 ページ)を参照してください。

注意:
データの並べ替えにホストソートユーティリティを使用する場合は、SORTSEQ=オプションで変換テーブルベースの照合シーケンスを指定すると、文字の BY 変数が破損する場合があります。 詳細については、動作環境に関する PROC SORT のドキュメントを参照してください

その他のオプション

オプションには、*collating-sequence-option* とその他複数のオプションが含まれます。この 2 種類のオプションの順序は問われません。

DATA=*SAS-data-set*
入力 SAS データセットを識別します。

制限事項 データベース内処理を実行するためには、データセットが DBMS に存在するテーブルを参照する必要があります。

注 PROC SORT は、拡張属性を入力データセットから出力データセットにコピーして拡張属性をサポートします。

参照項目 “入力データセット” (25 ページ)

SAS データセットオプション: リファレンス

DATECOPY
SAS データセットが作成された SAS 内部日時と並べ替え前に最後に変更を加えた日時を、並べ替えた結果のデータセットにコピーします。動作環境の日時は保持されません。

制限事項 DATECOPY は、結果のデータセットで V8 エンジンまたは V9 エンジンが使用される場合にのみ使用できます。

ヒント ファイル作成日時は、PROC DATASETS の MODIFY ステートメントの DTC=オプションで変更できます。詳細については、“MODIFY ステートメント” (508 ページ)を参照してください。

DUPOUT=*SAS-data-set*
重複オブザベーションの書き込み先となる出力データセットを指定します。

操作 DUPOUT=オプションを指定した場合、データベース内処理は行われません。

DUPOUT=オプションと UNIQUEOUT=オプションは両立しないため、同時には指定できません。

ヒント DUPOUT=オプションは、NODUPKEY オプションとのみ使用できます。DUPOUT=オプションを NOUNIQUEKEY オプションと組み合わせることはできません。

指定した DUPOUT=データセット名が INPUT データセット名と同じ場合、INPUT データセットの並べ替えも上書きも行われません。かわりに、SAS でエラーメッセージが生成されます。INPUT データセットを同名の DUPOUT=データセットで上書きするには、FORCE オプションを指定する必要があります。

参照項目 *SAS データセットオプション: リファレンス*

EQUALS | NOEQUALS
出力データセットのオブザベーションの順序を指定します。BY 変数値が同一のオブザベーションについて、EQUALS は、出力データセットで入力データセット内のオブザベーションの相対順序を維持します。NOEQUALS は、必ずしも出力データセットの相対順序を保持するとは限りません。

デフォルト EQUALS

操作 NODUPKEY を使用して出力データセットのオブザベーションを削除する場合、EQUALS または NOEQUALS の選択によって、削除されるオブザベーションが影響を受ける可能性があります。

EQUALS | NOEQUALS プロシジャオプションは、SORTEQUALS | NOSORTEQUALS システムオプションで設定されたデフォルトの並べ替え安定動作よりも優先されます。

EQUALS オプションは、スレッド化された並べ替えによってサポートされています。ただし、スレッド化された並べ替えで EQUALS オプションを使用すると、区分データセットが単一区分構成のように処理されるため、I/O パフォーマンスが低下する場合があります。

NOEQUALS オプションは、スレッド化された並べ替えでサポートされています。スレッド化された並べ替えによって返される BY グループ内のオブザベーションの順序は、実行間で一致しない場合があります。

ヒント NOEQUALS を使用すると、CPU 時間とメモリを節約できます。

FORCE
OUT=オプションを指定しない場合に、インデックス付きデータセットの並べ替えと置き換えを行います。FORCE オプションを指定しない場合、PROC SORT はインデックス付きデータセットの並べ替えや置き換えは行いません。並べ替えによってデータセットのユーザー作成インデックスが破損するためです。FORCE を指定すると、PROC SORT がデータセットの並べ替えと置き換えを行い、データセットのユーザー作成インデックスがすべて破損します。一貫性制約によって作成された、または必要とされたインデックスは保持されます。

制限事項 バージョン 5 互換性エンジンや、テープ形式エンジンなどの順次エンジンで作成されたデータセットに対して、PROC SORT と FORCE オプションを使用する場合、OUT=オプションも指定する必要があります。

ヒント PROC SORT は、データセットを並べ替える前に、データの不要な再並べ替えを行わないように、ソートインジケータをチェックします。デフォルトでは、PROC SORT は、並べ替え情報と要求された並べ替えが一致する場合、データセットを並べ替えません。FORCE オプションを使用すると、この動作を無効にできます。データセットオプション SORTEDBY=の並べ替え指定を確認できない場合は、FORCE の使用が必要になることがあります。SORTEDBY=の詳細については、*SAS データセットオプション: リファレンス* の SAS データセットオプションの章を参照してください。

NODUPKEY
BY 値が重複するオブザベーションのチェックと除外を行います。このオプションを指定すると、PROC SORT は各オブザベーションのすべての BY 値を出力データセットに先に書き込まれたオブザベーションの BY 値と比較します。完全に一致する値が見つかると、そのオブザベーションは出力データセットに書き込まれません。

動作環境の情報
VMS 動作環境で VMS ホストソートを使用している場合、出力データセットに書き込まれるオブザベーションが常に BY グループの最初のオブザベーションとは限りません。

操作 NODUPKEY を使用して BY 値の重複するオブザベーションを削除する際は、EQUALS または NOEQUALS の選択により、削除されるオブザベーションが影響を受ける可能性があります。

NODUPKEY オプションが指定され、システムオプション SQLGENERATION=に DBMS が割り当てられ、システムオプション SORTPGM=BEST の場合に、データベース内並べ替えが行われます。

オプション NODUPKEY と NOUNIQUEKEY は互換しません。これらのオプションを一緒に指定すると、SAS ログにエラーが書き込まれます。

ヒント 出力データセットで整合性のある結果を得るためには、EQUALS オプションと NODUPKEY オプションを一緒に使用します。

DUPOUT=オプションは NODUPKEY オプションと一緒に使用できます。DUPOUT=オプションを NOUNIQUEKEY オプションと組み合わせることはできません。

例 “例 4: 各 BY グループの最初のオブザベーションを維持する” (1820 ページ)

NOEQUALS
“EQUALS | NOEQUALS” (1802 ページ)を参照してください。

NOTHREADS
“THREADS | NOTHREADS” (1806 ページ)を参照してください。

NOUNIQUEKEY
一意のソートキーがある出力データセットのオブザベーションのチェックと除外を行います。キーを含むオブザベーションが BY グループ内の唯一のオブザベーションである場合、そのソートキーは一意です。オブザベーションが BY グループ内に 1 つしかない場合、そのオブザベーションには一意のソートキーがあります。

注: NODUPKEY は、BY グループの 1 変数を出力データセットに書き込み、BY グループの他のすべてのオブザベーションを破棄しますが、NOUNIQUEKEY はそれとは異なり、BY グループの一貫性を維持します。BY グループが 2 つ以上のオブザベーションから構成されている場合に BY グループのすべてのオブザベーションが出力データセットに書き込まれるか、BY グループが単一のオブザベーションから構成されている場合に BY グループのすべてのオブザベーションが破棄されるかのいずれかになります。

別名 NOUNIKEY | NOUNIKEYS | NOUNIQUEKEYS

操作 オプション NODUPKEY と NOUNIQUEKEY は互換しません。NODUPKEY と NOUNIQUEKEY を同時に指定すると、SAS ログにエラーが書き込まれます。

ヒント UNIQUEOUT= オプションは NOUNIQUEKEY オプションと一緒に使用できます。UNIQUEOUT= オプションを NODUPKEY オプションと組み合わせることはできません。

参照項目 除外されたオブザベーションを出力データセットに向ける UNIQUEOUT=

OUT= ***SAS-data-set***
出力データセットを指定します。*SAS-data-set* が存在しない場合、PROC SORT が作成します。

注意:
OUT=を指定せずに PROC SORT を使用する際は、注意が必要です。 OUT=オプションを指定しない場合、PROC SORT では、プロシジャがエラーなしで実行されると、元のデータセットが並べ替えたオブザベーションに置き換えられます。

デフォルト OUT=を指定しない場合、PROC SORT は元のデータセットを上書きします。

ヒント データベース内並べ替えでは、出力データセットは、DBMS の入力テーブルを参照できません。

データセットオプションは OUT=と使用できます。

参照項目 *SAS データセットオプション: リファレンス*

例 “例 1: 複数の変数の値によって並べ替える” (1813 ページ)

OVERWRITE
同じ名前の置換出力データセットにオブザベーションが作成される前に、入力データセットを削除できます。

注意:
OVERWRITE オプションは、バックアップされているデータセットか、再構築が可能なデータセットにのみ使用してください。 入力データセットは削除されるため、出力データセットの書き込み中にエラーが発生した場合はデータが失われます。

制限事項 OVERWRITE および OUT=オプションを指定し、OUT=データセット名が INPUT データセット名と同じではない場合、SAS で INPUT データセットは上書きされません。

TAGSORT オプションも指定した場合、OVERWRITE オプションは無効になります。TAGSORT は出力データセットの作成時に入力データセットを再度読み込む必要があるため、入力データセットは上書きできません。

OVERWRITE オプションは、SAS 並べ替えおよび SAS スレッド化された並べ替えによってのみサポートされています。このオプションは、ホストソートを使用する場合は無効になります。

ヒント OVERWRITE オプションを使用すると、必要なディスク容量を減らすことができます。

PRESORTED
並べ替えの前に、入力データセット内がチェックされ、オブザベーション順序が順番どおりかが判定されます。データセットがすでに BY ステートメントで指定したキー変数に従った順序であることがわかっている、またはそう思われる場合に、

PRESORTED オプションを使用します。このオプションを指定すると、データセットの並べ替えのコストを回避できます。

操作 “FORCE” (1802 ページ) オプションが指定された場合、順序チェックは実行されません。

ヒント SORT オプションと BY ステートメントに含められたキー変数との組み合わせによって指定された並べ替え順序に従って、SAS 処理に対応した順序で、外部テキストファイルからデータをインポートするには、DATA ステップを使用します。次に、データが適切に並べ替えられているとわかっている、またはそう思われる場合に、PRESORTED オプションを指定します。

PRESORTED オプションを ACCESS エンジンおよび DBMS データと使用することはお勧めしません。これらの外部データベースでは、クエリで ORDER BY 句を指定しない限り、オブザベーションが並べ替えられた順序で返される保証はありません。通常、外部データベースでは、物理的な順序という概念は使用されません。そのため、これらのデータベースでは、クエリを複数回実行した場合、オブザベーションが同一の順序で返される保証はありません。データの処理時に整合性のある反復可能な結果を得るためには、物理的な順序が重要になります。PROC SORT では、反復可能なデータ取得順序を指定しなければ、並べ替えの安定性を要求して “EQUALS | NOEQUALS” (1802 ページ) オプションを使用したとしても、PROC SORT を実行するたびに同一の順序でオブザベーションが返される保証はありません。反復可能なデータ取得順序を指定しなければ、PROC SORT による隣接する重複レコードの検出と除外も、PROC SORT を実行するたびに変わる可能性があります。

参照項目 システムオプション “SORTVALIDATE System Option” (*SAS System Options: Reference*).

REVERSE
通常の照合シーケンスを逆にした照合シーケンスを使用して、文字変数を並べ替えます。

動作環境の情報
動作環境の通常の照合シーケンスについては、“EBCDIC 順序” (1788 ページ), “ASCII 順序” (1788 ページ), 、および動作環境の SAS ドキュメントを参照してください。

制限事項 REVERSE オプションは *collating-sequence-option* とは併用できません。PROC SORT では、*collating-sequence-option* と REVERSE オプションのどちらかは指定できますが、両方は指定できません。

操作 REVERSE を BY ステートメントの DESCENDING オプションとともに使用すると、順序が通常の順序に戻ります。

参照項目 BY ステートメントの “DESCENDING” (1808 ページ) オプション。相違点は、DESCENDING オプションが文字変数と数値変数の両方に使用できる点です。

SORTSIZE=*memory-specification*
PROC SORT で利用可能な最大メモリ量を指定します。*memory-specification* の有効値は次のとおりです。

MAX
利用可能なメモリをすべて使用できるよう指定します。

n
メモリ量をバイト単位で指定します。この場合、*n* は実数です。

*n*K
メモリ量をキロバイト単位で指定します。この場合、*n* は実数です。

*n*M
メモリ量をメガバイト単位で指定します。*n* は実数です。

*n*G
メモリ量をギガバイト単位で指定します。*n* は実数です。

PROC SORT ステートメントで SORTSIZE=オプションを指定すると、一時的に SAS システムオプションよりも優先されます。詳細については、“SORTSIZE= System Option” (*SAS System Options: Reference*)を参照してください。

動作環境の情報
システムの並べ替えユーティリティによっては、このオプションの扱いが異なる場合もあります。動作環境の SAS ドキュメントを参照してください。

別名 SIZE=

デフォルト SAS システムオプション SORTSIZE=の値

ヒント PROC SORT ステートメントで SORTSIZE=オプションを MAX または 0 に設定するか、もしくは SORTSIZE=オプションを設定しない場合、PROC SORT の利用可能な物理メモリは、SAS システムオプション REALMEMSIZE および MEMSIZE の設定に基づいて制限されます。

SAS システムオプション REALMEMSIZE および MEMSIZE については、動作環境の SAS ドキュメントを参照してください。

TAGSORT
BY 変数とオブザベーション番号のみを一時ファイルに格納します。この BY 変数とオブザベーション番号は *tags* と呼ばれます。PROC SORT は、並べ替え処理の完了時に、タグを使用して入力データセットから並べ替え順序でレコードを取得します。

注: 作成されるユーティリティファイルは、TAGSORT オプションを指定しなかった場合よりもずっと小さくなります。

制限事項 TAGSORT オプションと OVERWRITE オプションは両立しません。

操作 TAGSORT オプションは、スレッド化された並べ替えではサポートされていません。

ヒント BY 変数の合計長がレコード長に比べて小さい場合、TAGSORT によって一時的なディスク使用量が大幅に削減されます。ただし、処理時間がずっと長くなる場合があります。

THREADS | NOTHREADS
スレッド化された並べ替えの有効化を可能にするか行わないようにします。

デフォルト THREADS | NOTHREADS SAS システムオプションの値。デフォルトは、SORT プロシジャの THREADS | NOTHREADS オプションを使用して無効にできます。

制限事項 サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

SPD エンジンによる THREADS | NOTHREADS プロシジャオプションの追加時にエラーが発生した場合、PROC SORT は処理を停止して、SAS ログにメッセージを書き込みます。

操作 システムオプションが制限されない限り、PROC SORT THREADS | NOTHREADS オプションは SAS システム THREADS | NOTHREADS オプションを無効にします(制約を参照)。詳細については、“THREADS System Option” (*SAS System Options: Reference*)を参照してください。

PROC SORT がスレッド処理が有効であると判断した場合、THREADS システムオプションが有効になります。SAS システムオプションの値が CPUCOUNT=1 の場合、スレッド処理は有効ではありません。ただし、システムオプションが NOTHREADS に設定されている場合、またはシステムオプションが THREADS でプロシジャオプションが NOTHREADS の場合は、PROC SORT THREADS オプションを指定してスレッド処理を強制できます。このオプションの組み合わせによって、スレッド処理は実行されず、システムオプションに基づくアクションが無効になります。スレッド化された並べ替えが有効で、NOEQUALS が指定されている場合、BY グループ内のオブザベーションが予想外の順序で返されることがあります。

スレッド化された SAS 並べ替えを使用している場合、UTILLOC=システムオプションがユーティリティファイルの配置に影響します。スレッドの有効な SAS アプリケーションでは、別々のスレッドからの並列アクセスが可能な一時ファイルを作成できます。詳細については、“UTILLOC= System Option” (*SAS System Options: Reference*)を参照してください。

PROC SORT で使用されるユーティリティファイルのページサイズは、新しい STRIPESIZE=システムオプションの影響を受けます。詳細については、“STRIPESIZE= System Option” (*SAS System Options: Reference*)を参照してください。

TAGSORT オプションは、スレッド化された並べ替えではサポートされていません。TAGSORT オプションを指定すると、スレッド処理は実行されません。

参照項目 “スレッド化された並べ替え” (1787 ページ) と “Support for Parallel Processing” (*SAS Language Reference: Concepts*).

UNIQUEOUT= ***SAS-data-set***

NOUNIQUEKEY オプションで除外されたオブザベーションの出力データセットを指定します。

別名 UNIOUT=

操作 DUPOUT=オプションと UNIOUT=オプションは両立しないため、同時には指定できません。

ヒント UNIQUEOUT= オプションは NOUNIQUEKEY オプションと一緒に使用できます。UNIQUEOUT= オプションを NODUPKEY オプションと組み合わせることはできません。

参照項目 “NOUNIQUEKEY” (1803 ページ)

SAS データセットオプション: リファレンス

BY ステートメント

並べ替え変数を指定します。

例: “例 1: 複数の変数の値によって並べ替える” (1813 ページ)
“例 2: 降順の並べ替え” (1815 ページ)
“例 4: 各 BY グループの最初のオブザベーションを維持する” (1820 ページ)

構文

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …>;

必須引数

variable
PROC SORT がオブザベーションの並べ替えの基準にする変数を指定します。PROC SORT はデフォルトで、まずデータセットを最初の BY 変数の値を基準にして昇順で並べます。PROC SORT は、次に、最初の BY 変数の値が同じオブザベーションを、2 番目の BY 変数の値を基準にして昇順で並べます。指定したすべての BY 変数に対して、この並べ替えが続行されます。

オプション引数

DESCENDING
ステートメントの直後に続く変数の並べ替え順序を逆順にして、オブザベーションを降順に並べ替えるようにします。DESCENDING キーワードは、後に続く変数を変更します。

ヒント PROC SORT BY ステートメントでは、DESCENDING キーワードは後に続く変数を変更します。

PROC SORT THREADS | NOTHREADS オプションが指定されていない限り、THREADS SAS システムオプションがデフォルトになります。

例 “例 2: 降順の並べ替え” (1815 ページ)

KEY ステートメント

並べ替えキーと変数を指定します。KEY ステートメントは、BY ステートメントの代替です。KEY ステートメント構文では、将来 KEY 変数ごとに異なる照合オプションを指定する可能性が考慮されています。現在使用できるオプションは、ASCENDING と DESCENDING のみです。

制限事項: BY ステートメントは、KEY ステートメントと使用できません。

ヒント: KEY ステートメントは、複数指定できます。

構文

KEY *変数* </ *オプション*>;

必須引数

variable(s)
PROC SORT でオブザベーションの並べ替えの基準にする変数を指定します。変数は複数指定できます。その変数は、それぞれスペースで区切る必要があります。変数の範囲も指定できます。たとえば、次のコードは、複数の変数と値の範囲の指定方法を示しています。

```
data sortKeys;
   input x1 x2 x3 x4 ;
cards;
   7 8 9 8
   0 0 0 0
   1 2 3 4 ;
run;
proc sort data=sortKeys out=sortedOutput;
   key x1 x2-x4;
run;
```

複数の KEY ステートメントも指定できます。すべてのソートキーのうち、最初に発生したソートキーが第 1 ソートキーとみなされます。指定されているすべての KEY ステートメントとその変数に対して、並べ替えが続行されます。たとえば、次のコードは、複数の KEY ステートメントの指定方法を示しています。

```
proc sort data=sortKeys out=sortedOutput;
   key x2;
   key x3;
run;
```

次のコード例では、BY ステートメントを使用して、前述の例と同じ種類の並べ替えを実行しています。

```
proc sort data=sortKeys out=sortedOutput;
   by x2 x3;
run;
```

オプション引数

ASCENDING
この後に続く 1 つまたは複数の変数を昇順で並べ替えます。オブザベーションは、昇順に並べ替えられます。ASCENDING キーワードは、KEY ステートメントで前に置かれる変数をすべて変更します。

別名 ASC

デフォルト ASCENDING がデフォルト並べ替え順序です。

ヒント PROC SORT KEY ステートメントでは、ASCENDING オプションは後に続く変数をすべて変更します。このオプションは、/ の後に置く必要があります。次の例では、入力データセットの x1 変数が昇順で並べ替えられます。

```
proc sort data=sortVar out=sortedOutput;
   key x1 / ascending;
run;
```

DESCENDING
ステートメントで後に続く変数の並べ替え順序を逆順にして、オブザベーションを降順に並べ替えるようにします。DESCENDING キーワードは、KEY ステートメントで前に置かれる変数をすべて変更します。

別名 DESC

デフォルト ASCENDING (ASC)がデフォルト並べ替え順序です

ヒント PROC SORT KEY ステートメントでは、DESCENDING オプションは後に続く変数を変更します。このオプションは、/の後に置く必要があります。次の例では、入力データセットの x1 変数と x2 変数が降順で並べ替えられます。

```
proc sort data=sortVar out=sortedOutput;
   key x1 x2 / descending;
run;
```

次の例では、BY ステートメントを使用して、前述の例と同じ種類の並べ替えを実行しています。

```
proc sort data=sortVar out=sortedOutput;
   by descending x1 descending x2 ;
run;
```

データベース内処理:PROC SORT

データベータ内処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。セキュリティの強化は、機密データを DBMS から抽出する必要がないため可能です。より迅速な処理は、データが比較的遅いネットワーク接続から転送される代わりに高速二次記憶装置を使用して DBMS でローカルで操作されるため、DBMS に自由に使えるより多くの処理リソースがあるため、DBMS は高度に並列かつスケーラブルな方法で実行するクエリを最適化できるため可能です。

DATA=入力データセットがデータベース管理システム(DBMS)のテーブルまたはビューとして保存される場合、PROC SORT プロシジャは In-Database 処理を使用して、データを並べ替えられます。データベース内処理は、より迅速な処理と、データベースと SAS ソフトウェア間のデータ転送の減少という利点を提供できます。

PROC SORT のデータベース内処理では、次のデータベース管理システムがサポートされます。

- Aster
- DB2
- Greenplum

- HAWQ
- Impala
- Netezza
- Oracle
- SAP HANA
- Teradata

PROC SORT は、SQL 明示パススルーを使用してデータベース内処理を実行します。パススルー機能は、SAS/ACCESS を使用して DBMS に接続し、ステートメントを実行するために DMBS に直接送信します。この機能により、DMBS の SQL 構文を使用できます。詳細については、*SAS/ACCESS for Relational Databases: Reference* の"Pass-Through Facility for Relational Dabases"を参照してください。

データベース内処理は、プロシジャとシステムオプションの組み合わせが正しく設定されている場合に PROC SORT によって使用されます。システムオプション SORTPGM=BEST を指定し、システムオプション SQLGENERATION=を In-Database 処理を行うように設定して、PROC SORT NODUPKEY オプションを指定した場合、PROC SORT はデータを並べ替える DBMS SQL クエリを生成します。並べ替え結果は、DBMS 内に新しいテーブルとして残すことも、SAS に返すこともできます。生成された SQL クエリを表示するには、SASTRACE=オプションを設定します。

また、SAS システムオプション SORTPGM=を使用すると、SQLGENERATION オプションを設定せずに、PROC SORT に対し DBMS、SAS、HOST のいずれかを使用して並べ替えを実行するように命令できます。SORTPGM=BEST を指定すると、DBMS、SAS、HOST のいずれかが並べ替えを実行します。PROC SORT により生成されるオブザベーション順序は、DBMS または SAS のどちらが並べ替えを実行するかによって異なります。

DBMS が並べ替えを実行する場合、DMBS 並べ替えプログラムの構成と特性が、結果のデータ順序に影響します。データ順序に影響する可能性がある DBMS の構成設定と特性には、文字照合、NULL 値の順序および並べ替えの安定性があります。ほとんどのデータベース管理システムでは、並べ替えの安定性が保証されていません。並べ替えは、SORTEQUALS/NOSORTEQUALS システムオプションおよび EQUALS/NOEQUALS プロシジャオプションの状態に関係なく、DMBS によって実行される場合があります。

SAS システムオプション SORTPGM=を SAS に設定した場合、並べ替え前のデータが DBMS から SAS に配信され、SAS で並べ替えが実行されます。ただし、DBMS からのオブザベーションの配信順序は保証されていません。そのため、DBMS データに対して安定した並べ替えを実行できるとしても、出力 BY グループ内のオブザベーションの順序が PROC SORT の実行のたびに同じになるという保証はありません。BY グループ内のオブザベーション順序の整合性を実現するには、まず DBMS データを含む SAS データセットを作成し、EQUALS または SORTEQUALS オプションを使用して安定したソートを実行します。

データベース内処理は、次の状況によって影響を受けます。

- PROC SORT オプションの SORTSEQ=または DUPOUT=を指定すると、データベース内処理は実行されません。
- データベース内処理では、OUT=プロシジャオプションの指定が必要です。このとき、出力データセットは DBMS の入力テーブルを参照できません。
- LIBNAME オプションとデータセットオプションもまたデーターベースの処理が起こるかどうかや、どんなタイプの問い合わせが来るかに影響されます。これらのすべてのオプションのリストについては、*SAS/ACCESS for Relational Databases: Reference* の"In-Database Procedures"を参照してください。SAS で OPTIONS

MSGLEVEL=I を設定すると、データベース内処理を行わないオプション、データベース内処理に影響するオプションを確認できます。

一貫性制約 SORT プロシジャ

入力データセットを並べ替え、並べ替えたデータセットと置き換えると、参照一貫性制約と一般一貫性制約の両方、および必要になる可能性があるすべてのインデックスが保持されます。新しいデータセットを作成する並べ替えでは、一貫性制約もインデックスも保持されません。FOUT=オプションを指定する場合としない場合の暗黙置換、明示置換および置換なしの詳細については、“出力データセット” (1812 ページ)を参照してください。一貫性制約の詳細については、“SAS Data Files” (*SAS Language Reference: Concepts*)の章を参照してください。

結果:SORT プロシジャ

プロシジャの出力

PROC SORT は、出力データセットのみを作成します。出力データセットを確認するには、PROC PRINT、PROC REPORT、または SAS で使用可能な多くの印刷方法のいずれかを使用します。

出力データセット

OUT=オプションを指定しない場合、PROC SORT では、プロシジャがエラーなしで実行されると、元のデータセットが並べ替えられたオブザベーションに置き換えられます。新しいデータセット名を使用して OUT=オプションを指定すると、PROC SORT は並べ替えられたオブザベーションを含む新しいデータセットを作成します。

表 59.7 *データセット置換オプション*

タスク	オプション
入力データセットの暗黙置換	proc sort data=names;
入力データセットの明示置換	proc sort data=names out=names;
入力データセットの置換なし	proc sort data=names out=namesbyid;

3 つの置換オプション(暗黙置換、明示置換および置換なし)すべてに関して、出力ライブラリで、少なくとも元のデータセットをコピーするのに十分なスペースが必要になります。

また、圧縮されているデータセットも並べ替えできます。圧縮データセットを入力データセットとして指定し、OUT=オプションを省略した場合、入力データセットは並べ替えられ、圧縮された状態は維持されます。OUT=データセットを指定すると、COMPRESS=データセットオプションで圧縮方法を選択した場合に限り、結果のデータセットが圧縮

されます。詳細については、“COMPRESS= Data Set Option” (*SAS Data Set Options: Reference*) を参照してください。

また、PROC SORT はメモリ内の圧縮されていないオブザベーションを操作し、並べ替えを完了するためのメモリが不足している場合は、ユーティリティファイルに解凍データを格納します。これらの理由から圧縮されたデータセットの分類は集中的でかつ予想よりより多くのストレージが必要です。TAGSORTの使用を考慮すると、圧縮データセットの分類のときのオプションです。

注: SAS システムオプション NOREPLACE が有効な場合は、元の永久データセットを並べ替えられたバージョンに置き換えることはできません。OUT=オプションを使用するか、OPTIONS ステートメントで SAS システムオプション REPLACE を指定する必要があります。SAS システムオプション NOREPLACE は、一時 SAS データセットには影響しません。

例: SORT プロシジャ

例 1: 複数の変数の値によって並べ替える

要素: PROC SORT ステートメントオプション
OUT=
BY ステートメント

他の要素: PROC PRINT

詳細

この例では、次を行います。

- 2 つの変数を基準にしてオブザベーションを並べ替えます。
- 並べ替えられたオブザベーションの出力データセットを作成します。
- 結果を印刷します。

プログラム

```
data account;
   input Company $ 1-22 Debt 25-30 AccountNumber 33-36
         Town $ 39-51;
   datalines;
Paul's Pizza              83.00  1019  Apex
World Wide Electronics   119.95  1122  Garner
Strickland Industries    657.22  1675  Morrisville
Ice Cream Delight        299.98  2310  Holly Springs
Watson Tabor Travel       37.95  3131  Apex
Boyd & Sons Accounting   312.49  4762  Garner
Bob's Beds               119.95  4998  Morrisville
Tina's Pet Shop           37.95  5108  Apex
Elway Piano and Organ     65.79  5217  Garner
Tim's Burger Stand       119.95  6335  Holly Springs
Peter's Auto Parts        65.79  7288  Apex
```

```
Deluxe Hardware         467.12  8941  Garner
Pauline's Antiques      302.05  9112  Morrisville
Apex Catering            37.95  9923  Apex
;

proc sort data=account out=bytown;

   by town company;
run;

proc print data=bytown;

   var company town debt accountnumber;

   title  'Customers with Past-Due Accounts';
   title2 'Listed Alphabetically within Town';
run;
```

プログラムの説明

入力データセット ACCOUNT を作成します。 ACCOUNT には、借金をしている各企業の名前、そのアカウントの借金額、アカウント番号、企業が所在する町が含まれます。

```
data account;
   input Company $ 1-22 Debt 25-30 AccountNumber 33-36
         Town $ 39-51;
   datalines;
Paul's Pizza             83.00  1019  Apex
World Wide Electronics  119.95  1122  Garner
Strickland Industries   657.22  1675  Morrisville
Ice Cream Delight       299.98  2310  Holly Springs
Watson Tabor Travel      37.95  3131  Apex
Boyd & Sons Accounting  312.49  4762  Garner
Bob's Beds              119.95  4998  Morrisville
Tina's Pet Shop          37.95  5108  Apex
Elway Piano and Organ    65.79  5217  Garner
Tim's Burger Stand      119.95  6335  Holly Springs
Peter's Auto Parts       65.79  7288  Apex
Deluxe Hardware         467.12  8941  Garner
Pauline's Antiques      302.05  9112  Morrisville
Apex Catering            37.95  9923  Apex
;
```

出力データセット BYTOWN を作成します。 OUT=は並べ替えられたオブザベーションに対し、新しいデータセットを作成します。

```
proc sort data=account out=bytown;
```

---

2 つの変数を基準にして並べ替えます。 BY ステートメントは、オブザベーションが町のアルファベット順、次に会社のアルファベット順で並べ替えられるよう指定します。

```
   by town company;
run;
```

出力データセット BYTOWN を印刷します。 PROC PRINT は、データセット BYTOWN を印刷します。

```
proc print data=bytown;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
var company town debt accountnumber;
```

タイトルを指定します。

```
title  'Customers with Past-Due Accounts';
title2 'Listed Alphabetically within Town';
run;
```

出力:出力:HTML

アウトプット 59.3 *複数の変数の値によって並べ替える*

Customers with Past-Due Accounts
Listed Alphabetically within Town

Obs	Company	Town	Debt	AccountNumber
1	Apex Catering	Apex	37.95	9923
2	Paul's Pizza	Apex	83.00	1019
3	Peter's Auto Parts	Apex	65.79	7288
4	Tina's Pet Shop	Apex	37.95	5108
5	Watson Tabor Travel	Apex	37.95	3131
6	Boyd & Sons Accounting	Garner	312.49	4762
7	Deluxe Hardware	Garner	467.12	8941
8	Elway Piano and Organ	Garner	65.79	5217
9	World Wide Electronics	Garner	119.95	1122
10	Ice Cream Delight	Holly Springs	299.98	2310
11	Tim's Burger Stand	Holly Springs	119.95	6335
12	Bob's Beds	Morrisville	119.95	4998
13	Pauline's Antiques	Morrisville	302.05	9112
14	Strickland Industries	Morrisville	657.22	1675

例 2: 降順の並べ替え

要素: この例の BY ステートメントオプション
DESCENDING

他の要素: PROC PRINT

データセット: Account

詳細

この例では、次を行います。

- 3 つの変数の値を基準にしてオブザベーションを並べ替えます。
- 変数の 1 つは、降順で並べ替えます。
- 結果を印刷します。

プログラム

```
proc sort data=account out=sorted;

   by town descending debt accountnumber;
run;

proc print data=sorted;

   var company town debt accountnumber;

   title  'Customers with Past-Due Accounts';
   title2 'Listed by Town, Amount, Account Number';
run;
```

プログラムの説明

出力データセット SORTED を作成します。 OUT=は並べ替えられたオブザベーションに対し、新しいデータセットを作成します。

```
proc sort data=account out=sorted;
```

---

3 つの変数を基準に、そのうち 1 つを降順にして並べ替えます。 BY ステートメントは、オブザベーションが町のアルファベット順、借金額の降順、アカウント番号の昇順の順番で並べ替えられるよう指定します。

```
   by town descending debt accountnumber;
run;
```

出力データセット SORTED を印刷します。 PROC PRINT は、データセット SORTED を印刷します。

```
proc print data=sorted;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
   var company town debt accountnumber;
```

タイトルを指定します。

```
   title  'Customers with Past-Due Accounts';
   title2 'Listed by Town, Amount, Account Number';
run;
```

出力:出力:HTML

最後に AccountNumber 順に並べ替えると、$37.95 の債務がある Apex の企業がアカウント番号順に並べられます。

アウトプット 59.4 *降順の並べ替え*

Customers with Past-Due Accounts
Listed by Town, Amount, Account Number

Obs	Company	Town	Debt	AccountNumber
1	Paul's Pizza	Apex	83.00	1019
2	Peter's Auto Parts	Apex	65.79	7288
3	Watson Tabor Travel	Apex	37.95	3131
4	Tina's Pet Shop	Apex	37.95	5108
5	Apex Catering	Apex	37.95	9923
6	Deluxe Hardware	Garner	467.12	8941
7	Boyd & Sons Accounting	Garner	312.49	4762
8	World Wide Electronics	Garner	119.95	1122
9	Elway Piano and Organ	Garner	65.79	5217
10	Ice Cream Delight	Holly Springs	299.98	2310
11	Tim's Burger Stand	Holly Springs	119.95	6335
12	Strickland Industries	Morrisville	657.22	1675
13	Pauline's Antiques	Morrisville	302.05	9112
14	Bob's Beds	Morrisville	119.95	4998

例 3: BY グループ内でオブザベーションの相対順序を維持する

要素: PROC SORT ステートメントオプション
EQUALS | NOEQUALS

他の要素: PROC PRINT

詳細

この例では、次を行います。

- 最初の変数の値を基準にしてオブザベーションを並べ替えます。
- EQUALS オプションを使用して、相対順序を維持します。
- NOEQUALS オプションを使用して相対順序を維持しないようにします。

プログラム

```
data insurance;
   input YearsWorked 1 InsuranceID 3-5;
   datalines;
5 421
```

```
5 336
1 209
1 564
3 711
3 343
4 212
4 616
;

proc sort data=insurance out=byyears1 equals;

   by yearsworked;
run;

proc print data=byyears1;

   var yearsworked insuranceid;

   title 'Sort with EQUALS';
run;

proc sort data=insurance out=byyears2 noequals;

   by yearsworked;
run;

proc print data=byyears2;

   var yearsworked insuranceid;

   title 'Sort with NOEQUALS';
run;
```

プログラムの説明

入力データセット INSURANCE を作成します。 INSURANCE には、全被保険従業員の勤続年数とその保険証 ID が含まれます。

```
data insurance;
   input YearsWorked 1 InsuranceID 3-5;
   datalines;
5 421
5 336
1 209
1 564
3 711
3 343
4 212
4 616
;
```

EQUALS オプションを使用して、出力データセット BYYEARS1 を作成します。 OUT=は並べ替えられたオブザベーションに対し、新しいデータセットを作成します。EQUALS オプションは、相対的なオブザベーション順序を維持します。

```
proc sort data=insurance out=byyears1 equals;
```

最初の変数順に並べ替えます。 BY ステートメントは、オブザベーションが勤続年数の数値順で並べ替えられるよう指定します。

```
   by yearsworked;
```

```
run;
```

出力データセット BYYEARS1 を印刷します。 PROC PRINT は、データセット BYYEARS1 を印刷します。

```
proc print data=byyears1;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
   var yearsworked insuranceid;
```

タイトルを指定します。

```
   title 'Sort with EQUALS';
run;
```

出力データセット BYYEARS2 を作成します。 OUT=は並べ替えられたオブザベーションに対し、新しいデータセットを作成します。NOEQUALS オプションは、相対的なオブザベーション順序を維持しません。

```
proc sort data=insurance out=byyears2 noequals;
```

最初の変数順に並べ替えます。 BY ステートメントは、オブザベーションが勤続年数の数値順で並べ替えられるよう指定します。

```
   by yearsworked;
run;
```

出力データセット BYYEARS2 を印刷します。 PROC PRINT は、データセット BYYEARS2 を印刷します。

```
proc print data=byyears2;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
   var yearsworked insuranceid;
```

タイトルを指定します。

```
   title 'Sort with NOEQUALS';
run;
```

出力:出力:HTML

EQUALS オプションを指定した並べ替えと、NOEQUALS オプションを指定した並べ替えとを比較すると、YearsWorked=3 のオブザベーションの並べ替え順序が異なることに注意してください。

アウトプット 59.5 *EQUALS オプションでの並べ替え*

Sort with EQUALS

Obs	YearsWorked	InsuranceID
1	1	209
2	1	564
3	3	711
4	3	343
5	4	212
6	4	616
7	5	421
8	5	336

アウトプット 59.6 *NOEQUALS オプションでの並べ替え*

Sort with NOEQUALS

Obs	YearsWorked	InsuranceID
1	1	209
2	1	564
3	3	343
4	3	711
5	4	212
6	4	616
7	5	421
8	5	336

例 4: 各 BY グループの最初のオブザベーションを維持する

要素: PROC SORT ステートメントオプション
NODUPKEY
BY ステートメント

他の要素: PROC PRINT

データセット: Account

注: EQUALS オプションは、各 BY グループの最初のオブザベーションが NODUPKEY オプションによって維持されるように、有効である必要があります。EQUALS オプションはデフォルトです。NOEQUALS オプションが指定されている場合、NODUPKEY オプションによ

って BY グループごとに 1 つのオブザベーションが維持されますが、それが必ずしも最初のオブザベーションとは限りません。

詳細

この例では、PROC SORT は各 BY グループの最初のオブザベーションのみを含む出力データセットを作成します。NODUPKEY オプションは、BY 値が出力データセットに書き込まれた最後のオブザベーションの BY 値と同じ場合に、そのオブザベーションが出力データセットに書き込まれないようにします。結果レポートには、オブザベーションが企業の所在する町ごとに 1 つずつ含まれます。

プログラム

```
proc sort data=account out=towns nodupkey;
   by town;
run;
proc print data=towns;
   var town company debt accountnumber;
   title 'Towns of Customers with Past-Due Accounts';
run;
```

プログラムの説明

出力データセット TOWNS を作成します。 ただし、ここに含まれるのは各 BY グループの最初のオブザベーションのみです。VMS 動作環境の並べ替えを使用している場合は、出力データセットに書き込まれるオブザベーションが常に BY グループの最初のオブザベーションとは限りません。

```
proc sort data=account out=towns nodupkey;
```

1 つの変数を基準にして並べ替えます。 BY ステートメントは、オブザベーションが町順で並べ替えられるよう指定します。

```
   by town;
run;
```

出力データセット TOWNS を印刷します。 PROC PRINT は、データセット TOWNS を印刷します。

```
proc print data=towns;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
   var town company debt accountnumber;
```

タイトルを指定します。

```
   title 'Towns of Customers with Past-Due Accounts';
run;
```

出力:出力:HTML

出力データセットには、入力データセットの町ごとに 1 つずつ、全部で 4 つのオブザベーションのみが含まれます。

アウトプット 59.7 *各 BY グループの最初のオブザベーションを維持する*

Towns of Customers with Past-Due Accounts

Obs	Town	Company	Debt	AccountNumber
1	Apex	Paul's Pizza	83.00	1019
2	Garner	World Wide Electronics	119.95	1122
3	Holly Springs	Ice Cream Delight	299.98	2310
4	Morrisville	Strickland Industries	657.22	1675

例 5: ALTERNATE_HANDLING=を使用した言語ソート

要素: PROC SORT ステートメントオプション
sortseq=linguistic
ALTERNATE_HANDLING=SHIFTED
STRENGTH=3

BY ステートメント

VAR ステートメント

他の要素: PROC PRINT

注: 文字列の言語ソートの強さの指定に関する詳細については、“例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語ソート” (1824 ページ)を参照してください。

詳細

この例では、PROC SORT は各 BY グループの最初のオブザベーションのみを含む出力データセットを作成します。"a-b" を"ab"および"aB"の近くになるように並べ替えるため、ALTERNATE_HANDLING=SHIFTED を指定しました。つまり、"a-b"がそのハイフンのせいで"ab"および"aB"と離れた場所に表示されないようにします。

注: この例では、このロケールのデフォルトの STRENGTH は 3 です。

次の例では、"a-b"と"ab"は同じものと見なされている点に注意してください。比較の最初の 3 つのレベル(アルファベット、付加記号、および大文字小文字)以外で並べ替える場合は、比較の 4 つ目のレベルを使用し、STRENGTH=4 と指定できます。“例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語ソート” (1824 ページ)は、文字列をさらに区別する方法を示しています。

プログラム

```
data a;
   length x $ 10;
   x='a-b'; output;
```

```
   x='ab'; output;
   x='a-b'; output;
   x='aB'; output;
run;

proc sort data=a sortseq=linguistic( ALTERNATE_HANDLING=SHIFTED );
   by x;
run;

title1 "Linguistic Collation with ALTERNATE_HANDLING=SHIFTED";
proc print data=a;
run;

title1 "Linguistic Collation with ALTERNATE_HANDLING=SHIFTED and BY Processing";
proc print data=a;
   var x;
   by x;
run;
```

プログラムの説明

データセットを作成します。

```
data a;
   length x $ 10;
   x='a-b'; output;
   x='ab'; output;
   x='a-b'; output;
   x='aB'; output;
run;
```

言語ソートを使用してデータセットを並べ替えます。 言語ソートと ALTERNATE_HANDLING=SHIFTED オプションを使用してデータセットを並べ替えます。このロケールのデフォルトの STRENGTH は 3 です。また、BY ステートメントも使用してオブザベーションを x で並べ替えます。

```
proc sort data=a sortseq=linguistic( ALTERNATE_HANDLING=SHIFTED );
   by x;
run;
```

データセット A を印刷します。 TITLE1 ステートメントで、PRINT プロシジャに出力に使用するタイトルを伝えます。PROC PRINT で、データセット A が印刷されます。

```
title1 "Linguistic Collation with ALTERNATE_HANDLING=SHIFTED";
proc print data=a;
run;
```

By 処理を使用してデータセット A を印刷します。 TITLE1 ステートメントで、PRINT プロシジャに出力に使用するタイトルを伝えます。PROC PRINT では、By 処理を使用してデータセット A が印刷されます。

```
title1 "Linguistic Collation with ALTERNATE_HANDLING=SHIFTED and BY Processing";
proc print data=a;
   var x;
   by x;
run;
```

出力:出力:HTML

1 つ目の PROC PRINT では、"a-b"と"ab"の順序が適切に定義されていないことが示されます。2 つ目の PROC PRINT では、BY 処理を使用してこれらの値が等価と見なされることが示されます。“例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語ソート” (1824 ページ)は、文字列をさらに区別する方法を示しています。

アウトプット 59.8 *ALTERNATE_HANDLING オプションを使用した言語ソート*

Linguistic Collation with ALTERNATE_HANDLING=SHIFTED

Obs	x
1	a-b
2	ab
3	a-b
4	aB

Linguistic Collation with ALTERNATE_HANDLING=SHIFTED and BY Processing

x=a-b

Obs	x
1	a-b
2	ab
3	a-b

x=aB

Obs	x
4	aB

例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語ソート

要素: PROC SORT ステートメントオプション
sortseq=linguistic
ALTERNATE_HANDLING=SHIFTED
STRENGTH=4

BY ステートメント

VAR ステートメント

他の要素: PROC PRINT

詳細

この例では、PROC SORT は各 BY グループの最初のオブザベーションのみを含む出力データセットを作成します。この例では、"a-b"をハイフンに関係なく"ab"および"aB"に近づくように並べ替えるため、ALTERNATE_HANDLING=SHIFTED が指定されています。

次の例では、"a-b"と"ab"は同じものと見なされている点に注意してください。ただし、これらをさらに区別して、2 つの個別の BY グループに表示する場合は、文字列をさらに並べ替える必要があります。比較の最初の 3 つのレベル(アルファベット、付加記号、および大文字小文字)以外で並べ替える場合は、比較の 4 つ目のレベル、STRENGTH=4 を指定できます。

プログラム

```
data a;
   length x $ 10;
   x='a-b'; output;
   x='ab'; output;
   x='a-b'; output;
   x='aB'; output;
run;

proc sort data=a sortseq=linguistic( ALTERNATE_HANDLING=SHIFTED STRENGTH=4);
   by x;
run;

title1 "Linguistic Collation with STRENGTH=4";
proc print data=a;
run;

Title1 "Linguistic Collation with STRENGTH=4 and BY Processing";
proc print data=a;
   var x;
   by x;
run;
```

プログラムの説明

データセットを作成します。

```
data a;
   length x $ 10;
   x='a-b'; output;
   x='ab'; output;
   x='a-b'; output;
   x='aB'; output;
run;
```

言語ソートを使用してデータセットを並べ替えます。 言語ソートと ALTERNATE_HANDLING=SHIFTED オプションを使用してデータセットを並べ替えます。このロケールのデフォルトの STRENGTH は 4 です。BY ステートメントでは、オブザベーションが x 順に並べ替えられるよう指定します。

```
proc sort data=a sortseq=linguistic( ALTERNATE_HANDLING=SHIFTED STRENGTH=4);
   by x;
run;
```

出力データセット A を印刷します。 TITLE1 ステートメントで、PRINT プロシジャに出力に使用するタイトルを伝えます。PROC PRINT で、データセット A が印刷されます。

```
title1 "Linguistic Collation with STRENGTH=4";
proc print data=a;
run;
```

By 処理を使用して出力データセット A を印刷します。 TITLE ステートメントで、PRINT プロシジャにこの出力に使用するタイトルを伝えます。PROC PRINT では、By 処理を使用してデータセット A が印刷されます。

```
Title1 "Linguistic Collation with STRENGTH=4 and BY Processing";
proc print data=a;
   var x;
```

```
    by x;
run;
```

出力:出力:HTML

1 つ目の PROC PRINT では、"a-b"と"ab"の順序が適切に定義されていないことが示されます。STRENGTH=4 を設定してこの 2 つを区別します。2 番目の PROC PRINT では、BY 処理を使用して優先順位とその区別の方法を示します。

アウトプット 59.9 *ALTERNATE_HANDLING および STRENGTH オプションを使用した言語ソート*

Linguistic Collation with STRENGTH=4

Obs	x
1	a-b
2	a-b
3	ab
4	aB

Linguistic Collation with STRENGTH=4 and BY Processing

x=a-b

Obs	x
1	a-b
2	a-b

x=ab

Obs	x
3	ab

x=aB

Obs	x
4	aB

60 章

SQOOP プロシジャ

概要: SQOOP プロシジャ

Apache Sqoop と SQOOP プロシジャについて

Hadoop とリレーショナルデータベース管理システム(RDBMS)との間でのデータ転送に、Apache Sqoop (*scoop* と発音します)を使用できます。SQOOP プロシジャを使用すると、SAS セッションから Apache Sqoop にアクセスしてデータベースと HDFS 間でデータを転送できます。これにより、SAS アプリケーション内から Hadoop クラスタに Sqoop コマンドをサブミットできます。

Sqoop コマンドは、Hadoop 向けの Apache Oozie Workflow Scheduler を使ってクラスタに渡されます。PROC SQOOP で Sqoop タスク用の Oozie ワークフローを定義すると、そのワークフローは RESTful API を使用して Oozie サーバーにサブミットされます。

PROC SQOOP は Apache Sqoop コマンド行インターフェイス(CLI)と同様に機能します。同じ構文を使って、SQOOP プロシジャの COMMAND ステートメントで Sqoop CLI コマンドをサブミットできます。このプロシジャは、ジョブが正常に完了したかどうか、および Sqoop タスクが失敗した場合に Hadoop クラスタからの詳細を確認できる場所についてフィードバックを提供します。

Hadoop ディストリビューションによっては、Sqoop コマンドのさまざまなバージョンがサポートされています。特定の Sqoop コマンド構文については、ディストリビューションのドキュメントを参照してください。

Apache Sqoop の詳細については、http://sqoop.apache.org および http://sqoop.apache.org/docs/1.4.4/index.html にあるオンラインドキュメントを参照してください。

Sqoop に関する考慮事項と使用法については、*Apache Sqoop Cookbook* を参照してください。

Oozie の詳細については、http://oozie.apache.org を参照してください。

構文: SQOOP プロシジャ

要件 Sqoop コマンドではなく、PROC SQOOP ステートメントの DBUSER および DBPWD オプションとともに個別に--username または--password オプションを指定します。

例: SQL クエリを使用した Teradata から HDFS へのインポート

PROC SQOOP <*Sqoop-options*>;

ステートメント	タスク	例
“PROC SQOOP ステートメント”	データ転送のための Apache Sqoop へのアクセスを可能にします。	Ex. 1

PROC SQOOP ステートメント

Apache Sqoop へアクセスし、データベースと HDFS の間でデータ転送を可能にするオプションを使います。

構文

PROC SQOOP
DBUSER='*database-user-name*'
DBPWD='*database-password*'
HADOOPUSER='*hadoop-user-name*'
HADOOPPWD='*hadoop-password*'
OOZIEURL='*oozie-URL*'
NAMENODE='*name-node-URL*'
JOBTRACKER='*job-tracker-URL*'
WFHDFSPATH='*Oozie-workflow-path*'
<PASSWORDFILE='*password-file*'*Sqoop-options*>
<DELETEWF>
COMMAND='*command-to-sqoop*';

必須引数

DBUSER='*database-user-name*'
インポートまたはエクスポートに使用するデータベースユーザー名を指定します。

DBPWD='*database-password*'
DBUSER オプションに関連付けられているデータベースパスワードを指定します。このオプションは PASSWORDFILE オプションと相互排他的であるため、どちらか一方のオプションのみ指定する必要があります。

HADOOPUSER='*hadoop-user-name*'
インポートまたはエクスポートに使用する Hadoop ユーザー名を指定します。

HADOOPPWD='*hadoop-password*'
HADOOPUSER オプションに関連付けられている Hadoop パスワードを指定します。

OOZIEURL='*oozie-URL*'
Oozie サーバーへの URL を指定します。

NAMENODE='*name-node-URL*'
HDFS NameNode サービスへの URL を指定します。

JOBTRACKER='*job-tracker-URL*'
HDFS JobTracker または ResourceManager サービスへの URL を指定します。

WFHDFSPATH='*Oozie-workflow-path-and-filename*'
HDFS 内の Oozie ワークフローがアップロードされるパスとファイル名を指定します。

COMMAND='*command-to-sqoop*'
Apache Sqoop コマンドを指定します。次のようにコマンドを指定する必要があります。

```
SQOOP-command --option … --option
```

制限事項 `sqoop` 起動コマンドは使用しないでください。

注 SQOOP プロシジャを使用して Sqoop コマンドをサブミットする場合、$CONDITIONS でエスケープ文字のスラッシュ(\)を使用する必要はありません。

ヒント Apache Sqoop コマンドの詳細については、http://sqoop.apache.org にあるオンラインドキュメントを参照してください。

オプション引数

PASSWORDFILE='*password-file*'
インポートまたはエクスポート用のデータベースパスワードを含む、HDFS 内のファイルの名前を指定します。このパスワードファイルはこのプロシジャを実行する前に存在している必要があります。また、常に安全に保管するよう注意してください。

DELETEWF
WFHDFSPATH 内の場所で指定された Oozie ワークフローファイルが存在する場合に、そのファイルを削除する必要があるかどうかを指定します。SQOOP プロシジャで、その場所に新しいワークフローファイルが作成されます。

SQOOP プロシジャの使用

一般的な使用方法

Oracle JDBC Connector の要件として、Sqoop の--table オプションに使用する値は大文字で指定する必要があります。テーブルでの大文字と小文字の区別の詳細については、特定の DBMS のドキュメントを参照してください。

接続文字列には、インポートするデータに適した文字セットオプションを含める必要があります。詳細については、コネクタのドキュメントを参照してください。

SQOOP プロシジャの使用要件

SQOOP プロシジャの使用を開始する前に必要な要件は次のとおりです。

必要な内容	PROC SQOOP オプション	確認する場所
データベースコネクタ	COMMAND:Sqoop 構文には、データベースに固有の--connection オプションが必要です。	Hadoop ディストリビューションのドキュメントを参照してください。
データベースユーザー ID	DBUSER	データベース管理者にお問い合わせください。
データベースパスワード	DBPWD	
データベースパスワードを含む HDFS ファイル	PASSWORDFILE	データベース管理者と Hadoop クラスタ管理者にお問い合わせください。
Hadoop ユーザー ID	HADOOPUSER	Hadoop クラスタ管理者にお問い合わせください。
Hadoop パスワード	HADOOPPWD	
Oozie URL	OOZIEURL	
名前ノード	NAMENODE	
Job Tracker (MR1)または Resource Manager (MR2)	JOBTRACKER	
Oozie ワークフロー出力パス	WFHDFSPATH	Hadoop クラスタ管理者にお問い合わせください。一般的に、Oozie ワークフローは HDFS 内のユーザーディレクトリに書き込まれます。
Sqoop コマンド	COMMN	Hadoop ディストリビューションの Sqoop ドキュメントを参照してください。

例: SQL クエリを使用した Teradata から HDFS へのインポート

この例では、既存のワークフローをこのタスクの新しいワークフローに置き換えるために、DELETEWF が含まれています。

注: 環境に適したデフォルトの NAMENODE および JOBTRACKER ポート値については、特定のディストリビューションの構成を確認するか、Hadoop ドキュメントを参照してください。

```
proc sqoop dbuser='mydbusr1' dbpwd='mydbpwd1'
   hadoopuser='sashdpusr1' hadooppwd='sashdppwd1'
   oozieurl='http://myoozie-04:55000/oozie'
   namenode='hdfs://myoozie-04.unx.srvr.com:8020'
   jobtracker='myoozie-04.unx.srvr.com:8032'
   wfhdfspath='hdfs://myoozie-04.unx.srvr.com:8020/user/mydbusr1/myworkflow.xml'
   deletewf
   command='import
      --connection-manager com.cloudera.connector.teradata.TeradataManager
      --connect jdbc:teradata://myconnecti/Database=sqoop
      --query "SELECT * FROM sales where ($CONDITIONS and I < 25)" -m 1
      --split-by i --delete-target-dir
      --target-dir /user/mydbusr1/sales2';
run;
```

61 章

STANDARD プロシジャ

概要:STANDARD プロシジャ

STANDARD プロシジャの動作について

STANDARD プロシジャは、SAS データセットの変数を指定の平均と標準偏差に標準化し、標準化された値を含む新しい SAS データセットを作成します。

データの標準化

次の出力に、標準化された学生の試験のスコアが出力データセットに含まれる単一の標準化を示します。出力を生成するステートメントは次のとおりです。

```
proc standard data=score mean=75 std=5
              out=stndtest;
run;

proc print data=stndtest;
run;
```

アウトプット 61.1 *PROC STANDARD を使用して標準化されたテストのスコア*

```
The SAS System                          1 Obs    Student       Test1 1
Capalleti    80.5388 2    Dubose       64.3918 3    Engles       80.9143 4
Grant        68.8980 5    Krupski      75.2816 6    Lundsford    79.7877 7
McBane       73.4041 8    Mullen       78.6612 9    Nguyen       74.9061 10
Patel        71.9020 11    Si           73.4041 12    Tanaka       77.9102
```

次の出力に、BY グループ処理を使用した、より複雑な例を示します。PROC STANDARD は平均寿命データを平均 0、標準偏差 1 に標準化して、2 つの BY グループに対し別々に Z スコアを計算します。データは、16 か国の 1950 年と 1993 年の出生時平均寿命です。stable または rapid に分類される各国の出生率が、2 つの BY グループを形成しています。分析を生成するステートメントは、次も行います。

- 標準化する変数ごとに統計量を印刷します。
- 欠損値を指定の平均と置き換えます。
- 指定の平均と標準偏差を使用して標準化された値を計算します。
- データセットを標準化された値とともに印刷します。

この出力を生成するプログラムの説明については、“例 2: BY グループの標準化と欠損値の置換” (1844 ページ)を参照してください。

アウトプット 61.2 *PROC STANDARD を使用した BY グループごとの Z スコア*

```
Life Expectancies by Birth Rate               2 --------------------
PopulationRate=Stable ---------------------- The STANDARD Procedure Standard
Name              Mean       Deviation               N Label Life50
67.400000        1.854724               5 1950 life expectancy Life93
74.500000        4.888763               6 1993 life expectancy
--------------------- PopulationRate=Rapid ---------------------- Standard
Name              Mean       Deviation               N Label Life50
42.000000        5.033223               8 1950 life expectancy Life93
59.100000        8.225300              10 1993 life expectancy
```

```
Standardized Life Expectancies at Birth           3 by a Country's Birth Rate
Population Rate       Country            Life50      Life93 Stable
France             -0.21567     0.51138 Stable      Germany            0.32350
0.10228 Stable      Japan              -1.83316     0.92048 Stable
Russia              0.00000    -1.94323 Stable      United Kingdom     0.86266
0.30683 Stable      United States       0.86266     0.10228 Rapid
Bangladesh          0.00000    -0.74161 Rapid       Brazil             1.78812
0.96045 Rapid       China              -0.19868     1.32518 Rapid
Egypt               0.00000     0.10942 Rapid       Ethiopia          -1.78812
-1.59265 Rapid       India              -0.59604    -0.01216 Rapid
Indonesia          -0.79472    -0.01216 Rapid       Mozambique         0.00000
-1.47107 Rapid       Philippines         1.19208     0.59572 Rapid
Turkey              0.39736     0.83888
```

構文: STANDARD プロシジャ

ヒント: ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

PROC STANDARD <*option(s)*>;
BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …> <NOTSORTED>;
FREQ *variable*;
VAR *variable(s)*;
WEIGHT *variable*;

ステートメント	タスク	例
“PROC STANDARD ステートメント”	変数を指定の平均と標準偏差に標準化します。	Ex. 1, Ex. 2
“BY ステートメント”	BY グループごとに個別の標準化された値を計算します。	Ex. 2
“FREQ ステートメント”	値が各オブザベーションの度数を表す変数を識別します。	
“VAR ステートメント”	標準化する変数を選択して、印刷出力に表示される順序を決定します。	Ex. 1
“WEIGHT ステートメント”	値が統計量計算で各オブザベーションを重み付ける変数を識別します。	

PROC STANDARD ステートメント

SAS データセットの変数を指定の平均と標準偏差に標準化し、標準化された値を含む新しい SAS データセットを作成します。

例: “例 2: BY グループの標準化と欠損値の置換” (1844 ページ)
“例 1: 指定された平均と標準偏差への標準化” (1841 ページ)

構文

PROC STANDARD <*option(s)*>;

オプション引数の要約

DATA=*SAS-data-set*
入力データセットを指定します。
EXCLNPWGT
非正の重みが付いたオブザベーションを除外します。
MEAN=*mean-value*
平均値を指定します。
OUT=*SAS-data-set*
出力データセットを指定します。
REPLACE
欠損値を変数の平均または MEAN=値と置き換えます。
STD=*std-value*
標準偏差値を指定します。

VARDEF=*divisor*
分散の計算に使用する分母を指定します。

Control printed output

NOPRINT
すべての印刷出力を抑制します。

PRINT
標準化する変数ごとに統計量を印刷します。

Preserve values

PRESERVERAWBYVALUES
未加工の BY 値を保持します。

引数なし

MEAN=、REPLACE、STD=を指定しない場合、出力データセットは入力データセットの同一のコピーです。

オプション引数

DATA=*SAS-data-set*
入力 SAS データセットを指定します。

制限事項 PROC STANDARD は、別のユーザーが同時にデータセットを更新している場合に同時アクセス権をサポートするエンジンと使用できません。

参照項目 “入力データセット” (25 ページ)

EXCLNPWGT
非正(ゼロまたは負)の重みが付いたオブザベーションを除外します。プロシジャは平均と標準偏差の計算にオブザベーションを使用しませんが、オブザベーションは標準化されます。デフォルトで、プロシジャは負の重みの付いたオブザベーションをゼロの重みが付いたオブザベーションのように扱い、それらをオブザベーションの合計数にカウントします。

別名 EXCLNPWGTS

MEAN=*mean-value*
変数を *mean-value* の平均に標準化します。

デフォルト 入力値の平均

例 “例 1: 指定された平均と標準偏差への標準化” (1841 ページ)

NOPRINT
プロシジャ出力の印刷を抑制します。NOPRINT はデフォルト値です。

OUT=*SAS-data-set*
出力データセットを指定します。*SAS-data-set* が存在しない場合、PROC STANDARD が作成します。OUT=を省略すると、データセットの名前は DATA*n* となります。*n* は、名前を一意のものにする最小整数です。

デフォルト Data*n*

例 “例 1: 指定された平均と標準偏差への標準化” (1841 ページ)

PRESERVERAWBYVALUES
未加工の BY 値を保持します。BY 変数が出力データセットに伝搬する場合、すべての BY 変数の未加工値を保持します。

PRINT
標準化する変数ごとに元の度数、平均、標準偏差を印刷します。

例 “例 2: BY グループの標準化と欠損値の置換” (1844 ページ)

REPLACE
欠損値を変数の平均と置き換えます。

操作 MEAN=を使用すると、PROC STANDARD は欠損値を指定の平均と置き換えます。

例 “例 2: BY グループの標準化と欠損値の置換” (1844 ページ)

STD=*std-value*
変数を標準偏差 *std-value* に標準化します。

デフォルト 入力値の標準偏差

例 “例 1: 指定された平均と標準偏差への標準化” (1841 ページ)

VARDEF=*divisor*
分散と標準偏差の計算に使用する分母を指定します。次のテーブルに、*divisor* に可能な値と、関連する分母を示します。

表 61.1 *VARDEF=に可能な値*

値	分母	分母の式
DF	自由度	$n - 1$
N	オブザベーション数	n
WDF	重みの合計 - 1	$(\Sigma_i w_i) - 1$
WEIGHT|WGT	重みの合計	$\Sigma_i w_i$

プロシジャは分散を*CSS*/*divisor*として計算します。*CSS*は修正平方和で、$\Sigma (x_i - \bar{x})^2$と等しくなります。分析変数を重み付けする場合、*CSS*は$\Sigma w_i(x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト DF

ヒント WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。i 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$、およびw_iは i 番目のオブザベーションの重みです。これにより、オブザベーションの分散の推定が単位重みとともに生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな n に対し)$\sigma^2/\bar{w}$の推定になります。$\bar{w}$は、平均重みで

す。これにより、オブザベーションの分散の漸近的推定が平均重みとともに生成されます。

参照項目 “WEIGHT” (74 ページ)

“キーワードと式” (2078 ページ)

BY ステートメント

BY グループごとに標準化された値を個別に計算します。

参照項目: “BY” (68 ページ)

例: “例 2: BY グループの標準化と欠損値の置換” (1844 ページ)

構文

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …> <NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは、指定するすべての変数別に並べ替える必要があります。並べ替えない場合は、適切にインデックス付けする必要があります。これらの変数は、*BY 変数*と呼ばれます。

オプション引数

DESCENDING
データセットが BY ステートメントで文字 DESCENDING の直後に続く変数別に降順で並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。データは、時系列などの別の方法でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ別の BY グループとして扱います。

FREQ ステートメント

値がオブザベーションの度数を表す数値変数を指定します。

ヒント: FREQ ステートメントと WEIGHT ステートメントの影響は、自由度の計算時以外は似ています。

参照項目: FREQ ステートメントを使用する例については、次を参照してください。“FREQ” (72 ページ)

構文

FREQ *variable*;

必須引数

variable
値がオブザベーションの度数を表す数値変数を指定します。FREQ ステートメントを使用する場合、プロシジャは各オブザベーションが *n* オブザベーションを表すとみなします。*n* は *variable* の値です。*n* は整数でない場合、切り捨てられます。*n* が 1 未満の場合、プロシジャはそのオブザベーションを統計量の計算に使用しませんが、オブザベーションは標準化されます。

度数変数の合計は、オブザベーションの合計数を表します。

VAR ステートメント

標準化する変数と、印刷出力での順序を指定します。

デフォルト: VAR ステートメントを省略すると、PROC STANDARD はその他のステートメントでリストされないすべての数値変数を標準化します。

例: “例 1: 指定された平均と標準偏差への標準化” (1841 ページ)

構文

VAR *variable(s)*;

必須引数

variable(s)
標準化する変数を識別します。

WEIGHT ステートメント

統計量計算の分析変数に対し重みを指定します。

参照項目: 重み付き統計量の計算と、WEIGHT ステートメントを使用する例については、“WEIGHT” (74 ページ)を参照してください。

構文

WEIGHT *variable*;

必須引数

variable
分析変数の値に重みをつける数値変数を指定します。変数の値は、整数である必要はありません。次のテーブルに、重み値に基づいたアクションを示します。

表 61.2 *WEIGHT ステートメント値および PROC STANDARD アクション*

重み値	PROC STANDARD アクション
0	オブザベーションをオブザベーションの合計数にカウントします。
0 未満	重み値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントする
欠損値	オブザベーションを平均および標準偏差の計算から除外します

負およびゼロの重みを含むオブザベーションを平均および標準偏差の計算から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

ヒント WEIGHT ステートメントを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。詳細については、“VARDEF=*divisor* ” (1837 ページ) および重み付き統計量の計算(“キーワードと式” (2078 ページ))を参照してください。

詳細

注: バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

統計量の計算:STANDARD プロシジャ

値の標準化により、場所とスケールの属性がデータのセットから削除されます。標準化された値を計算する式は、

$$x_i' = \frac{S^*\left(x_i - \bar{x}\right)}{s_x} + M$$

この場合、

x_i'
新しい標準化された値です。

S
STD=の値です

M
MEAN=の値です。

x_i
オブザベーションの値です。

$\bar{x}$
変数の平均です。

s_x
変数の標準偏差です。

PROC STANDARD は、平均($\bar{x}$)と標準偏差(s_x)を入力データセットから計算します。結果として標準化された変数は、平均が **M**、標準偏差が **S** となります。

データが正規分布している場合、結果のデータはスチューデントの **t** 分布になるため、標準化はスチューデント化でもあります。

結果:STANDARD プロシジャ

欠損値

デフォルトで、PROC STANDARD は分析変数に対する欠損値を標準化プロセスから除外します。値は出力データセットで欠損のままです。REPLACE オプションを指定すると、プロシジャは欠損値を変数の平均または MEAN=値と置き換えます。

WEIGHT 変数または FREQ 変数の値が欠損している場合、プロシジャは平均と標準偏差の計算にオブザベーションを使用しません。ただし、オブザベーションは標準化されます。

出力データセット

PROC STANDARD は、OUT=オプションを指定するかどうかに関係なく、標準化された値を VAR ステートメント変数に保存する出力データセットを常に作成します。出力データセットには、標準化されていないものを含む、すべての入力データセット変数が含まれています。PROC STANDARD は、出力データセットを印刷しません。出力データセットを印刷するには、PROC PRINT、PROC REPORT または別の SAS レポートツールを使用します。

例: STANDARD プロシジャ

例 1: 指定された平均と標準偏差への標準化

要素: PROC STANDARD ステートメントオプション
MEAN=
OUT=
STD=
VAR ステートメント

他の要素: PRINT プロシジャ

詳細

この例では、次を行います。

- 2 つの変数を平均 75、標準偏差 5 に標準化します。
- 出力データセットを指定します。
- 標準化された変数を元の変数と結合します。
- 出力データセットを印刷します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data score;
   length Student $ 9;
   input Student $ StudentNumber Section $
         Test1 Test2 Final @@;
   format studentnumber z4.;
   datalines;
Capalleti 0545 1 94 91 87  Dubose    1252 2 51 65 91
Engles    1167 1 95 97 97  Grant     1230 2 63 75 80
Krupski   2527 2 80 69 71  Lundsford 4860 1 92 40 86
McBane    0674 1 75 78 72  Mullen    6445 2 89 82 93
Nguyen    0886 1 79 76 80  Patel     9164 2 71 77 83
Si        4915 1 75 71 73  Tanaka    8534 2 87 73 76
;

proc standard data=score mean=75 std=5 out=stndtest;

   var test1 test2;
run;

proc sql;
   create table combined as
   select old.student, old.studentnumber,
          old.section,
          old.test1, new.test1 as StdTest1,
          old.test2, new.test2 as StdTest2,
          old.final
   from score as old, stndtest as new
   where old.student=new.student;

proc print data=combined noobs round;
   title 'Standardized Test Scores for a College Course';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションで、SAS ジョブの開始日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Score データセットを作成します。 このデータセットには、2 つのテストと期末試験を受けた学生のテストの成績が含まれます。FORMAT ステートメントは、*Zw.d* 出力形式を

StudentNumber に割り当てます。この出力形式は、右寄せ出力にブランクではなく、0 を埋め込みます。LENGTH ステートメントは、Student の値の保存に使用するバイト数を指定します。

```
data score;
   length Student $ 9;
   input Student $ StudentNumber Section $
         Test1 Test2 Final @@;
   format studentnumber z4.;
   datalines;
Capalleti 0545 1 94 91 87  Dubose    1252 2 51 65 91
Engles    1167 1 95 97 97  Grant     1230 2 63 75 80
Krupski   2527 2 80 69 71  Lundsford 4860 1 92 40 86
McBane    0674 1 75 78 72  Mullen    6445 2 89 82 93
Nguyen    0886 1 79 76 80  Patel     9164 2 71 77 83
Si        4915 1 75 71 73  Tanaka    8534 2 87 73 76
;
```

標準化されたデータを生成し、Stndtest 出力データセットを作成します。 PROC STANDARD は平均 75 と標準偏差 5 を使用して、値を標準化します。OUT=は、標準化された値を含むデータセットとして Stndtest を識別します。

```
proc standard data=score mean=75 std=5 out=stndtest;
```

標準化する変数を指定します。 VAR ステートメントは標準化する変数と、その出力順序を指定します。

```
   var test1 test2;
run;
```

元の値と標準化された値を結合するデータセットを作成します。 PROC SQL は Score と Stndtest を結合して、学生ごとの標準化されたテストの成績と元のテストの成績を含む Combined データセット(テーブル)を作成します。AS を使用して標準化された変数 NEW.TEST1 の名前を StdTest1 に、NEW.TEST2 の名前を StdTest2 に変更すると、変数名が一意のものになります。

```
proc sql;
   create table combined as
   select old.student, old.studentnumber,
          old.section,
          old.test1, new.test1 as StdTest1,
          old.test2, new.test2 as StdTest2,
          old.final
   from score as old, stndtest as new
   where old.student=new.student;
```

データセットを出力します。 PROC PRINT で、Combined データセットを印刷します。ROUND は、標準化された値を小数点以下 2 桁に丸めます。TITLE ステートメントでタイトルを指定します。

```
proc print data=combined noobs round;
   title 'Standardized Test Scores for a College Course';
run;
```

出力:リスト

次のデータセットには、標準化された値を持つ変数と元の値を持つ変数の両方が含まれています。StdTest1 と StdTest2 には、PROC STANDARD で計算される標準化されたテストの成績が保存されています。

アウトプット 61.3 大学の講座の標準化した試験得点

```
Standardized Test Scores for a College Course                1
Student                          Std              Std Student      Number
Section    Test1    Test1    Test2    Test2    Final Capalleti     0545
1          94     80.54      91     80.86      87 Dubose        1252
2          51     64.39      65     71.63      91 Engles        1167
1          95     80.91      97     82.99      97 Grant         1230
2          63     68.90      75     75.18      80 Krupski       2527
2          80     75.28      69     73.05      71 Lundsford     4860
1          92     79.79      40     62.75      86 McBane        0674
1          75     73.40      78     76.24      72 Mullen        6445
2          89     78.66      82     77.66      93 Nguyen        0886
1          79     74.91      76     75.53      80 Patel         9164
2          71     71.90      77     75.89      83 Si            4915
1          75     73.40      71     73.76      73 Tanaka        8534
2          87     77.91      73     74.47      76
```

例 2: BY グループの標準化と欠損値の置換

要素: PROC STANDARD ステートメントオプション
PRINT
REPLACE
BY ステートメント

他の要素: FORMAT プロシジャ
PRINT プロシジャ
SORT プロシジャ

詳細

この例では、次を行います。

- 平均 0 と標準偏差 1 を使用して BY グループごとに Z スコアを個別に計算します。
- 欠損値を指定の平均と置き換えます。
- 標準化する変数の平均と標準偏差を印刷します。
- 出力データセットを印刷します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc format;
   value popfmt 1='Stable'
                2='Rapid';
run;
```

```
data lifexp;
   input PopulationRate Country $char14. Life50 Life93 @@;
   label life50='1950 life expectancy'
         life93='1993 life expectancy';
   datalines;
2 Bangladesh     .  53 2 Brazil         51 67
2 China          41 70 2 Egypt          42 60
2 Ethiopia       33 46 1 France         67 77
1 Germany        68 75 2 India          39 59
2 Indonesia      38 59 1 Japan          64 79
2 Mozambique      . 47 2 Philippines    48 64
1 Russia          . 65 2 Turkey         44 66
1 United Kingdom 69 76 1 United States  69 75
;

proc sort data=lifexp;
   by populationrate;
run;

proc standard data=lifexp mean=0 std=1 replace
              print out=zscore;

   by populationrate;

   format populationrate popfmt.;
   title1 'Life Expectancies by Birth Rate';
run;

proc print data=zscore noobs;
   title 'Standardized Life Expectancies at Birth';
   title2 'by a Country''s Birth Rate';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションで、SAS ジョブの開始日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

文字列出力形式を数値に割り当てます。 PROC FORMAT で、文字値を含む出生率を識別するための出力形式 POPFMT を作成します。

```
proc format;
   value popfmt 1='Stable'
                2='Rapid';
run;
```

Lifeexp データセットを作成します。 このデータセットの各オブザベーションには、16 か国の 1950 年と 1993 年の出生時平均寿命に関する情報が含まれます。国ごとの出生率は、stable (1)または rapid (2)に分類されます。欠損データを含む国は、1950 年より後に独立した国です。データは、Vital Signs 1994:The Trends That Are Shaping Our Future, Lester R. Brown, Hal Kane, and David Malin Roodman, eds.Copyright © 1994 by Worldwatch Institute.Reprinted by permission of W.W.Norton & Company, Inc.)によるものです

```
data lifexp;
```

```
   input PopulationRate Country $char14. Life50 Life93 @@;
   label life50='1950 life expectancy'
         life93='1993 life expectancy';
   datalines;
2 Bangladesh     .  53 2 Brazil         51 67
2 China          41 70 2 Egypt          42 60
2 Ethiopia       33 46 1 France         67 77
1 Germany        68 75 2 India          39 59
2 Indonesia      38 59 1 Japan          64 79
2 Mozambique      . 47 2 Philippines    48 64
1 Russia          . 65 2 Turkey         44 66
1 United Kingdom 69 76 1 United States  69 75
;
```

Lifeexp データセットを並べ替えます。 PROC SORT で、オブザベーションを出生率別に並べ替えます。

```
proc sort data=lifexp;
   by populationrate;
run;
```

すべての数値変数に対し標準化されたデータを生成し、Z-score 出力データセットを作成します。 PROC STANDARD で、すべての数値変数を平均 1、標準偏差 0 に標準化します。REPLACE で欠損値を置き換えます。PRINT で統計量を印刷します。

```
proc standard data=lifexp mean=0 std=1 replace
              print out=zscore;
```

BY グループごとに標準化された値を作成します。 BY ステートメントで、出生率別に値を個別に標準化します。

```
   by populationrate;
```

変数に出力形式を割り当て、レポートのタイトルを指定します。 FORMAT ステートメントで、PopulationRate に出力形式を割り当てます。出力データセットには、フォーマットされた値が含まれます。TITLE ステートメントでタイトルを指定します。

```
   format populationrate popfmt.;
   title1 'Life Expectancies by Birth Rate';
run;
```

データセットを出力します。 PROC PRINT で、ZSCORE データセットを標準化された値で印刷します。TITLE ステートメントで、印刷する 2 つのタイトルを指定します。

```
proc print data=zscore noobs;
   title 'Standardized Life Expectancies at Birth';
   title2 'by a Country''s Birth Rate';
run;
```

出力:リスト

PROC STANDARD では、BY グループごとに標準化する各変数の変数名、平均、標準偏差、入力度数、ラベルを印刷します。バングラデシュ、モザンビーク、ロシアの平均寿命の欠損はなくなっています。欠損値は指定の平均(0)と置き換えられています。

アウトプット 61.4 出生率別の平均寿命

```
Life Expectancies by Birth Rate                        1
--------------------------- PopulationRate=Stable
----------------------------- Standard Name              Mean
Deviation               N    Label Life50       67.400000
1.854724                5    1950 life expectancy Life93       74.500000
4.888763                6    1993 life expectancy ----------------------------
PopulationRate=Rapid ---------------------------- Standard Name
Mean       Deviation               N    Label Life50       42.000000
5.033223                8    1950 life expectancy Life93       59.100000
8.225300               10    1993 life expectancy Standardized Life Expectancies
at Birth                     2 by a Country's Birth Rate Population Rate
Country            Life50      Life93 Stable      France
-0.21567     0.51138 Stable      Germany            0.32350     0.10228
Stable      Japan             -1.83316     0.92048 Stable
Russia              0.00000    -1.94323 Stable      United Kingdom
0.86266     0.30683 Stable       United States      0.86266     0.10228
Rapid       Bangladesh         0.00000    -0.74161 Rapid
Brazil              1.78812     0.96045 Rapid       China
-0.19868     1.32518 Rapid       Egypt              0.00000     0.10942
Rapid       Ethiopia          -1.78812    -1.59265 Rapid
India              -0.59604    -0.01216 Rapid       Indonesia
-0.79472    -0.01216 Rapid       Mozambique         0.00000    -1.47107
Rapid       Philippines        1.19208     0.59572 Rapid
Turkey              0.39736     0.83888
```

62 章

STREAM プロシジャ

STREAM プロシジャの動作について

STREAM プロシジャを使用すると、SAS マクロ指定を含む可能性のある任意のテキストで構成される入力ストリームを処理できます。マクロが実行され、展開されている間、入力ストリームの他のテキストは保持されます。テキストストリームは、SAS 構文として検証されません。出力ストリームは、ファイル参照で参照され、また従来の SAS 出力先を使用するように定義できる外部ファイルに送信されます。

概念:STREAM プロシジャ

テキストおよびマクロコードの解釈

次の例では、%DOIT という名前のマクロを使用して、テーブルを含む HTML ストリームを生成します。

```
%macro doit(nrows,ncols);
<table>
%do i=1 to &nrows;
<tr>
%do j=1 to &ncols;
<td>&j</td>
%end;
</tr>
%end;
</table>
%mend doit;
```

このマクロが SAS コードストリーム内で実行されると、HTML 出力が生成されますが、出力は有効な SAS 構文ではないため構文上は無効になります。ただし、このマクロが PROC STREAM を介して実行されると、HTML 出力はファイルに書き込まれ、SAS 構文として検証されません。例は、次のとおりです。

```
proc stream outfile=myfile; begin
%doit(2,3)
;;;;
```

次の出力が *myfile* に書き込まれます。

```
<table> <tr> <td>1</td> <td>2</td> <td>3</td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> </tr> </
table>
```

入力ストリームでは%INCLUDE ステートメントを使用できます。使用可能な入力ファイルの種類には、HTML ファイル、RTF ファイル、その他の種類のテキストベースファイルがあります。マクロの定義と呼び出しを除き、実際の SAS コードを含めるために%INCLUDE を使用することはありません。マクロ以外の SAS コードは実行されず、他のテキストと同様に扱われます。

%の後に続く、マクロで認識されるテキストが実行されます。%LET、%INCLUDE、%SYSFUNC などがあります。ユーザー定義マクロなど他のマクロも実行されます。

すべてのマクロ変数参照が解決されますが、マクロ名が認識されない場合は警告が発行されます。%PUT や%LIST などのマクロステートメントでは、出力がログに送信され、トークンは生成されません。これらのステートメントを入力ストリームで使用することは可能ですが、使用しないようにする必要があります。

トークナイザの制限

SAS トークナイザまたはワードスキャナには、SAS 構文入力ストリーム以外の構文入力ストリームを正しく処理する際に問題となる可能性のある多くの制限があります。これらの制限について、次のリストで説明します。

- %INCLUDE ステートメントからの入力レコードは、レコードサイズのデフォルト値である 32,767 バイトを超えることはできません。
- %INCLUDE からの入力レコードが 32,767 バイトを超えると、レコードサイズに関する正確な情報がトークナイザから PROC STREAM に提供されません。
- %INCLUDE および%LET ステートメントはステートメント境界から開始する必要があります。つまり、ステートメントの前にセミコロンを付け、ステートメントはセミコロンで終わる必要があります。マクロの呼び出しにはこの要件は適用されません。
- マクロステートメントはトークナイザですべて使用および実行されるため、PROC STREAM でマクロステートメントは認識されません。たとえば、**`%let xyz=2;`**はす

べて使用されます。この動作は、スコープ内でグローバルではない、%IF や%GOTO などのマクロステートメントにも当てはまります。これらのマクロステートメントにはエラーのフラグが立てられますが、PROC STREAM でそのエラーは認識されません。

- a～z (大文字または小文字)の文字、または&が前に付いているアンダースコアから始まるトークンは、解決するマクロ変数であるかどうかがマクロプロセッサによって判定されます。マクロ変数でない場合は、警告が発行されます。現在、この警告は非表示にすることができません。
- PL/I プログラミング言語のコメント(/* *comment* */)は、通常、トークナイザによって完全に取り込まれ、PROC STREAM では表示されません。PROC STREAM ステートメントで NOABSSCMT オプションを使用すると、/*と*/およびその間にあるすべてのテキストが表示されます。このオプションは、入力ストリームに引用符で囲まれていない/*が含まれている可能性がある場合にも必要になります。たとえば、UNIX のワイルドカード仕様を表す**`/tmp/xyz/*`**がこれに該当します。
- 1 つのトークンが 32,767 文字を超えることはできません。
- レコードの最後のトークンの後に、オプションのキャリッジリターンが前に付いた改行が続き、次のレコードの最初のトークンが列 1 から開始される場合、トークナイザはこれらのトークンの間にブランクを挿入します。このブランクがあると有効な SAS 構文になります。これらのトークンは、引用符で囲まれている文字列の一部でない場合連結されません。たとえば、レコードの長さが 80 文字であるとします。前にブランクの付いている文字列 *abc* は列 78～80 に配置されます。次のレコードでは、後ろにブランクの付いた *def* が列 1～3 に表示されます。この場合、PROC STREAM は、先頭のブランクがトークンストリームに表示されていない場合でも、*abc* を受け取った後に、先頭にブランクの付いた *def* を受け取ります。そうではなく、ブランクが前に付いた *'abc* が列 77～80 に表示され、後ろにブランクの付いた *def'* が列 1～4 に表示されている場合、PROC STREAM は *'abcdef'* を受け取ります。
- 印刷不可の文字(キャリッジリターン(CR)や改行(LF)を除く)は正しくトークン化できません。
- 入力ストリームを、PDF や JPG ファイルなどのバイナリストリームにすることはできません。
- トークナイザの通常の動作は、一重引用符または二重引用符の付いた文字列を単一のトークンとして提供することです。*don't do this* などのテキストが引用符で囲まれていない場合や、*don''t do this* のようにテキストにエスケープシーケンスが含まれている場合、これが問題になります。QUOTING=オプションを使用すると、引用符は、ハイフンやスラッシュなどの他の特殊文字と同様に処理されます。
- 1 つのマクロ変数を 64,000 文字まで展開できます。ただし、メモリやディスク領域などのリソースの制限を除き、マクロがモデルテキストとして返すことができる内容に制限はありません。

構文: STREAM プロシジャ

PROC STREAM OUTFILE=*fileref* <*options*>; BEGIN
text-1
<*text-n*>
;;;;

ステートメント	タスク
“PROC STREAM ステートメント”	SAS マクロ仕様を含むことができる任意のテキストで構成される入力ストリームを処理します。

PROC STREAM ステートメント

SAS マクロ仕様を含むことができる任意のテキストで構成される入力ストリームを処理できます。

構文

PROC STREAM OUTFILE= *fileref* <*option(s)*>; BEGIN
text-1
<*text-n*>
;;;;

オプション引数の要約

MOD
出力ファイルが上書きされるのではなく追加されることを指定します。

NOABSSCMT
コメントが出力ストリームに書き込まれるかどうかを指定します。

PRESCOL
元の入力ファイルの列を保持することを示します。

QUOTING=SINGLE | DOUBLE | BOTH
引用符の処理方法を指定します。

RESETDELIM='*label*'
特殊マーカートークンを示します。

必須引数

OUTFILE=*fileref*
すべてのトークンが書き込まれるファイルを指定します。

ファイル参照の LRECL 仕様が使用されます。LRECL を指定しない場合、グローバル LRECL=オプションのデフォルト値である 32,767 バイトが使用されます。PRESCOL オプションを使用しない場合、トークン間に適切な数のブランクが挿入されて、すべてのトークンがストリーム出力されます。レコード間でトークンは分割されません。また、ブランクがトークン内にない限り、トークンのストリームはレコード間で分割されません。たとえば、<table>X</table>はレコード間で分割されませんが、<table> X </table>はブランクが示されている場所(X の前後)で分割されます。

text
PROC STREAM で使用する SAS ステートメントまたはマクロを指定します。

オプション引数

MOD
出力ファイルが上書きされるのではなく追加されることを指定します。

NOABSSCMT
コメントが出力ストリームに書き込まれるかどうかを指定します。

PL/I プログラミング言語スタイルのコメントが表示される場合(/* comments */)、コメント文字(/*と*/)間のすべてのテキストが出力ストリームに表示されます。このオプションを指定しない場合、PL/I スタイルのコメントは出力ストリームに表示されません。NOABSSCMT を設定する場合は、通常一重引用符(word don't など)がコメントに表示される可能性があるため、QUOTING=も設定することを強くお勧めします。

PRESCOL
元の入力ファイルの列を保持することを示します。

マクロ代替がある場合やレコードサイズが 32,767 バイトを超える場合、このオプションを使用しても失敗します。この場合、マクロの展開が列の場所に影響する可能性があります。

PRESCOL オプションを指定すると、%INCLUDE マクロで含まれる RTF ファイルの有効性が向上します。

QUOTING=SINGLE | DOUBLE | BOTH
引用符の処理方法を指定します。

SINGLE
一重引用符を他の文字と同様に扱うことを指定します。SINGLE オプションを使用する場合で、'&hello'のように、マクロ参照が一重引用符内にある場合、マクロ参照が展開されます。

DOUBLE
二重引用符を他の文字と同様に扱うことを指定します。

BOTH
SINGLE および DOUBLE の両方のオプションを使用することを指定します。

RESETDELIM='*label*'
特殊マーカートークンを示します。

%INCLUDE や%LET ステートメントのように、ステートメントを展開する必要がある場合にこのオプションが使用されます。これらのステートメントはステートメント境界から開始する必要があります。構文でステートメント境界が許可されていない場合、指定されたラベルとその後に続くセミコロンを入力ストリームに挿入して、トークナイザ要件を満たすことができます。ラベルとセミコロンは出力ファイルに送信されません。

次の例では、%INCLUDE の前にセミコロンが付いておらず、コードは正しくありません。

```
x y %include myfile; z
```

ただし、RESETDELIM=オプションを使用すると、展開は予想どおりに実行されます。

```
resetdelim='mylabel'
x y mylabel;
%include myfile; z
```

*Mylabel;*は出力ファイルに表示されません。

Mylabel は有効な SAS 名であることが必要です。つまり、先頭は文字またはアンダースコアで、それ以降のすべての文字は文字、アンダースコア、数値である必要があります。*mylabel* の長さは 32 文字以内であることが必要です。

仕様と使用法では、大文字と小文字が区別されません。RESETDELIM=にはデフォルト値がありません。

SAS 9.3 の 2 つ目のメンテナンスリリースでは、PROC STREAM が、&STREAMDELIM という名前のマクロ変数の有無をチェックします。このマクロ変

数が存在する場合は、そのまま使用されます。このマクロ変数が存在しない場合、PROC STREAM は、RESETDELIM=オプションが指定されているかどうかを検証します。このオプションが指定されている場合、マクロ変数&STREAMDELIM は、RESETDELIM=オプションの値に設定されます。RESETDELIM=オプションが指定されていない場合、PROC STREAM は、現在の日時値に基づいて &STREAMDELIM マクロ変数の一意の値を作成します。

この点を考慮して、&STREAMDELIM を PROC STREAM の入力に追加できます。このマクロ変数が表示される場合、オプションのキーワードが続くものと見なされます。終了のセミコロンが必要です。&STREAMDELIM 値からセミコロンまで(セミコロンを含む)のすべてのトークンは出力ストリームに送信されず、PROC STREAM の特別な管理情報項目になります。

入力で%INCLUDE ステートメントを指定する必要がある場合でも、その前にセミコロンを付けない場合は、%INCLUDE ステートメントの前に次のステートメントを追加する必要があります。

```
&STREAMDELIM;
```

このステートメントを指定すると、セミコロンはトークナイザに認識されますが、PROC STREAM に取り込まれます。

&STREAMDELIM の後に次のオプションのキーワードを続けることができます。

NEWLINE
新しい行が出力ファイルに送信されることを指定します。

READFILE *filename*
指定されたファイル名が開かれ、その内容がそのまま読み込まれて出力ファイルに書き込まれることを指定します。このファイル内のマクロの展開は行われず、新しい行が保持されます。この動作は、マクロの展開が実行され、新しい行が無視される%INCLUDE の動作と異なります。READFILE キーワードの使用例については、“READFILE キーワードの使用” (1856 ページ)を参照してください。

入力ストリームでのマクロベースコードの使用

入力ストリームのマクロベースコードは、次の例のように展開されます。

```
%let abc=123;
proc stream outfile=out; begin
<title>Run &abc</title>
<table>
<tr> <td>1<td>2</tr>
<tr> <td>3<td>4</tr>
</table>
;;;;
```

出力ファイルには次の結果が含まれます。

```
<title>Run 123</title> <table> <tr> <td>1<td>2</tr> <tr> <td>3<td>4</tr> </
table>
```

改行はありませんが、ブランクは保持され、改行ごとにブランクが挿入されます。

入力ストリームの%と&が引用符で囲まれておらず、エスケープシーケンスではないため、問題が発生する可能性があります。たとえば、*&*は、HTML ストリームでアン

パサンドを表すエスケープシーケンスとして表示されます。*&AMP* という名前のマクロ変数を使用すると、その値が*&amp* のかわりに使用されます。この種類のマクロ変数が存在しない場合は、警告が発行されます。これらの問題を回避するためには、次の例に示す%STR(&)のように、&をエスケープ文字として使用します。

```
<title>A %str(&)amp; B</title>
```

次の出力が出力ストリームに書き込まれます。

```
<title>A &amp; B</title>
```

%についても同様に、マクロで正しく解釈されないものは、次の例に示すように**%STR(%)**として示されます。

```
<a href="www.sas.com/x%str(%20y)">the link</a>
```

次の出力が出力ストリームに書き込まれます。

```
<a href="www.sas.com/x%20y">the link</a>
```

%STR のエスケープシーケンスが一重引用符内にある場合は、QUOTING=オプションを使用するか、%STR(%')のように、エスケープシーケンスと一重引用符を使用します。

SAS コードの実行

前のセクションで説明したように、存在するすべての SAS コードが実行されるわけではありません。トークンは他のトークンと同様にストリーム出力されます。この動作の例外は、%SYSFUNC または%SYSCALL マクロ関数が存在している場合に起こります。これらのマクロ関数は、次の例に示すように指定された関数を実行します。

```
%let abc=%sysfunc(getoption(obs));
&abc
```

このステートメントで GETOPTION 関数が呼び出され、OBS オプションの値が取得されます。GETOPTION は、値を ABC マクロ変数に配置する文字列として返します。次に、その値が出力ストリームに書き込まれます。

%SYSFUNC で DOSUB 関数を使用して、SAS コードのストリームを実行することができます。DOSUB 関数ではファイル参照が指定され、そのファイル内のすべての SAS コードが実行されます。

この例では、ファイル参照 MYCODE が、次の SAS コードを指し示しています。

```
filename myhtml "c:\temp\temp.txt";

data _null_;
   file myhtml;
   put '<table>';
   put '<tr><td>1</td></tr>';
   put '<tr><td>2</td></tr>';
   put '</table>';
run;
```

次の STREAM プロシジャを実行すると、PROC STREAM が *mycode* を%SYSFUNC 関数の DOSUB に対する引数として使用していることがわかります。

```
filename myhtml "c:\temp\temp.txt";
filename new "c:\temp\new.html";

proc stream outfile=new; begin
```

```
%let abc=%sysfunc(dosub(mycode));
%include myhtml;
;;;;
```

出力ストリームには次の結果が含まれます。

```
<table><tr><td>1</td></tr><tr><td>2</td></tr></table>
```

注: DOSUB 関数は DOSUBL 関数と似ていますが、DOSUB には、SAS コードを含むファイルのファイル参照が渡されます。DOSUBL にはテキスト文字列が渡され、その値が SAS コードとして実行されます。詳細については、“DOSUBL Function” (*SAS Functions and CALL Routines: Reference*)を参照してください。

Rich Text Format (RTF)ファイル出力

Microsoft Word を使用してデータを入力し、RTF 出力を作成できます。**Today is &sysdate..**および **My name is &name..**と入力すると、Microsoft Word ではデータが Mytest.rtf という名前の RTF ファイルとして保存されます。このファイルのサイズは約 30K で、大量のマークアップデータが含まれています。そのデータの 1 つのセクションに、入力された実際のテキストが次のように含まれます。

```
\fs20\lang1033\langfe1033\cgrid\langnp1033\langfenp1033 {\rtlch\fcs1 \af0 \ltrch\fcs0 \insrsid8922096
Today is &sysdate.. }{\rtlch\fcs1 \af0 \ltrch\fcs0 \insrsid14092174
\par }{\rtlch\fcs1 \af0 \ltrch\fcs0 \insrsid8922096
\par My name is &name..
\par
```

入力されたテキストにはマクロ置換が含まれています。

この SAS プログラムを実行すると、Newrtf.rtf という名前の新しい RTF ファイルが作成されます。この出力ファイルにおける違いは、マクロ置換は行われていても、マークアップ言語が変わらないことです。

```
%let name=John;
filename oldrtf 'mytest.rtf' recfm=v lrecl=32767;
filename newrtf 'newrtf.rtf' recfm=v lrecl=32767;
proc stream outfile=newrtf quoting=both asis; begin
&streamdelim;
%include oldrtf;
;;;;
```

Newrtf.rtf ファイルが Microsoft Word で読み込まれると、次のような内容が表示されます。

```
Today is 30NOV12.


My name is John.
```

READFILE キーワードの使用

HTML では、<PRE> ...</PRE>タグが、事前にフォーマットされたテキストのストリームをカプセル化し、改行やスペースを尊重して、そのまま表示されます。READFILE キ

ーワードを使用すると、ファイルの内容を読み込み、そのまま保持できます。
READFILE キーワードは、&STREAMDELIM マクロ変数の後に続きます。

```
filename fixed temp;
data _null_; file fixed;
     input; put _infile_;
     datalines4;
This is the first line of fixed text.
This is another line to be fixed.
This is the last line of fixed text.
;;;;

%macro doit(nrows,ncols);
<table>
%do i=1 %to &nrows;
<tr>
%do j=1 %to &ncols;
<td>&j</td>
%end;
</tr>
%end;
%mend;

filename new temp;
proc stream outfile=new; begin
<PRE>
&streamdelim readfile fixed;
</PRE>
%doit(2,3)
;;;;

data _null_; infile new; input; put _infile_; run;
filename fixed temp;
data _null_; file fixed;
     input; put _infile_;
     datalines4;
This is the first line of fixed text.
This is another line to be fixed.
This is the last line of fixed text.
;;;;

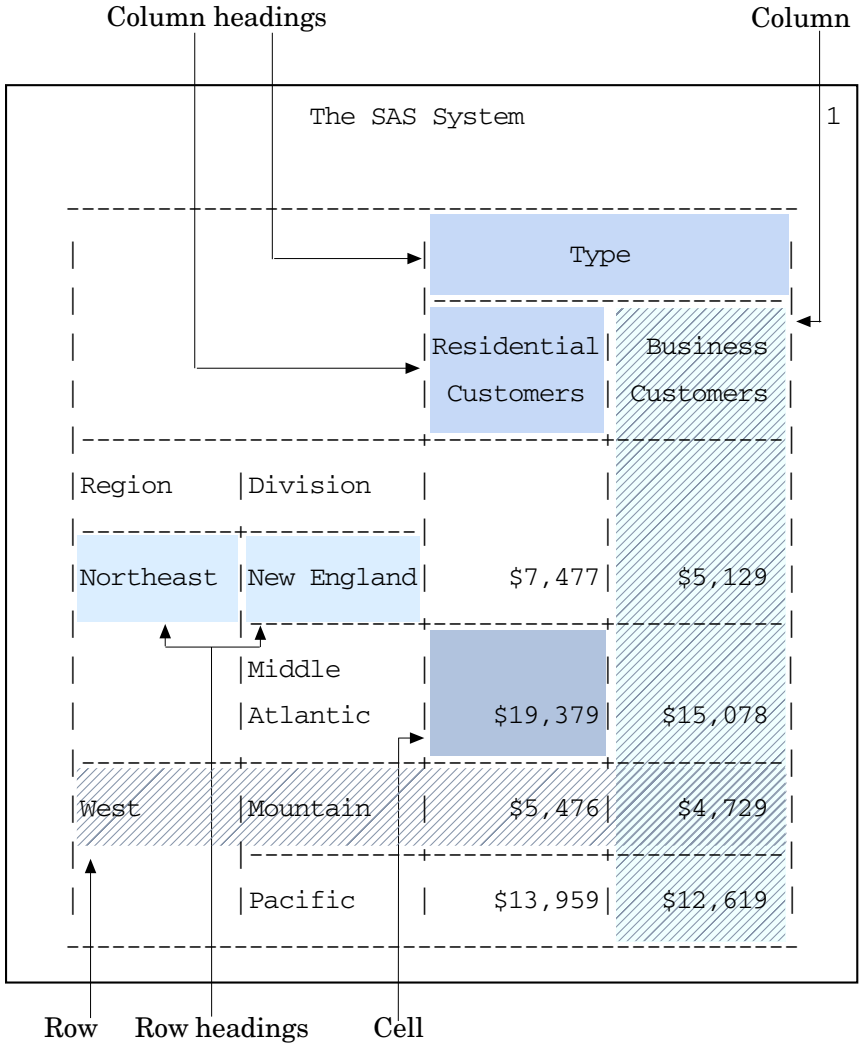
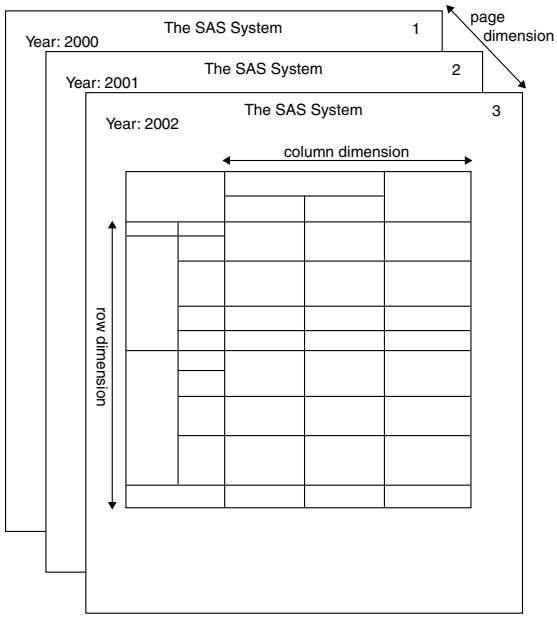
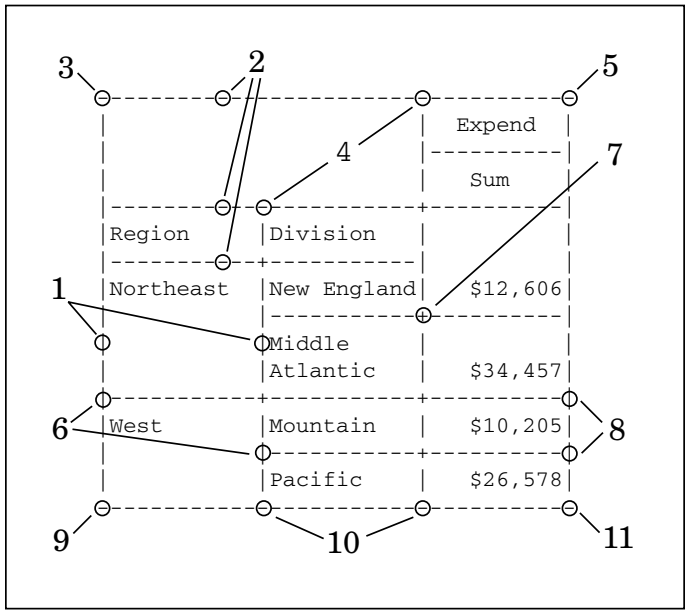
%macro doit(nrows,ncols);
<table>
%do i=1 %to &nrows;
<tr>
%do j=1 %to &ncols;
<td>&j</td>
%end;
</tr>
%end;
%mend;

filename new temp;
proc stream outfile=new; begin
<PRE>
&streamdelim readfile fixed;
```

```
</PRE>
%doit(2,3)
;;;;

data _null_; infile new; input; put _infile_; run;
```

結果のファイルは次のようになります。

```
<PRE>
This is the first line of fixed text.
This is another line to be fixed.
This is the last line of fixed text.
</PRE> <table> <tr> <td>1</td> <td>2</td> <td>3</dt> </tr>
<tr> <td>1</td> <td>2</td> <td>3</dt> </tr> </table>
```

%INCLUDE を使用した PROC STREAM へのファイルの挿入

複雑な HTML および Javascript コードをストリーミングするには、別のファイルでコードを作成します。次に、%INCLUDE ステートメントを使用して、ファイルをプログラムに含めます。

次の例は、このアプローチを示しています。

```
filename temp1 temp;

data _null_;
   infile datalines;
   file temp1;
   input;
   l=length(_infile_);
   put @1 _infile_ $varying200. l;
   datalines4;
<table>
<tr>
<td>abc</td><td>def</td>
</tr>
<tr>
<td>ghi</td><td>jkl</td>
</tr>
</table>
;;;;

filename myfile "c:\temp\test1.html";

proc stream outfile=myfile prescol resetdelim='label'; begin
<html>
<h2>Test Example</h2>
label;
%include temp1;
</html>
;;;;
```

出力ストリームへの新しい行の挿入

新しい行を出力ストリームに挿入するには、次の例に示すように、デリミタの後ろのセミコロンの前にキーワード NEWLINE を追加します。

```
proc stream outfile=abc resetdelim='mylabel'; begin
my item here;
mylabel newline;
my next item here
;;;;
```

この例では、次の出力が生成されます。

```
my item here;
my next item here
```

他の方法では予測どおりに新しい行を挿入できません。

STREAM プロシジャの終了

PROC STREAM ステップの終了は、プロシジャの最後の行として記述される、間にブランクを含まない 4 つのセミコロンで示されます。入力ストリームの最後の項目がセミコロンで終了する場合は、最後の項目の直後に、RESETDELIM=で指定されたラベルと、それに続くセミコロンを使用します。

次の例では、**here** の後のセミコロンが認識されないため、正しくコード化されません。

```
proc stream outfile=abc; begin
my item here;
;;;;
```

次の例は正しくコード化されます。

```
proc stream outfile=abc resetdelim='mylabel'; begin
my item here;
mylabel;
;;;;
```

この例では、**here** の後のセミコロンを認識させるため、RESETDELIM=が使用されています。

63 章

SUMMARY プロシジャ

概要: SUMMARY プロシジャ

SUMMARY プロシジャは、すべてのオブザベーションにわたる変数、またはオブザベーショングループ内の変数に対する記述統計量を計算するためのデータ要約ツールを提供します。SUMMARY プロシジャは、MEANS プロシジャに酷似しています。構文の詳細については、37 章, “MEANS プロシジャ,” (1092 ページ) を参照してください。ここで説明されている相違を除くすべての PROC MEANS 情報は、PROC SUMMARY にも適用されます。

構文: SUMMARY プロシジャ

ヒント: Output Delivery System をサポートします。詳細については、“Output Delivery System: Basic Concepts” (*SAS Output Delivery System: User's Guide*)を参照してください。

ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。

構文の説明については、“構文” (1098 ページ)を参照してください。

PROC SUMMARY <*option(s)*> <*statistic-keyword(s)*>;
BY <DESCENDING> *variable-1*<<DESCENDING> *variable-2 …*>
<NOTSORTED>;
CLASS *variable(s)*</ *option(s)*>;
FREQ *variable*;
ID *variable(s)*;
OUTPUT <OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
<*id-group-specification(s)*> <*maximum-id-specification(s)*>
<*minimum-id-specification(s)*> </ *option(s)*> ;
TYPES *request(s)*;
VAR *variable(s)* </ WEIGHT=*weight-variable*>;
WAYS *list*;
WEIGHT *variable*;

ステートメント	タスク
“PROC SUMMARY ステートメント”	すべてのオブザベーションにわたる変数、またはオブザベーショングループ内の変数に対して記述統計量を計算します。
“BY ステートメント”	BY グループごとに別の統計量を計算します。
“CLASS ステートメント”	値が分析対象のサブグループを定義する変数を識別します。
“FREQ ステートメント”	値が各オブザベーションの度数を表す変数を識別します。
“ID ステートメント”	追加の ID 変数を出力データセットに含めます。
“OUTPUT ステートメント”	指定されている統計量と ID 変数を含む出力データを作成します。
“TYPES ステートメント”	データの分割に使用する分類変数の特定の組み合わせを識別します。
“VAR ステートメント”	分析変数と結果での順序を識別します。
“WAYS ステートメント”	分類変数の一意の組み合わせを作成するための方法の数を指定します。
“WEIGHT ステートメント”	値が統計量計算で各オブザベーションを重み付ける変数を識別します。

PROC SUMMARY ステートメント

すべてのオブザベーションにわたる変数とオブザベーショングループ内の変数に対し記述統計量を計算します。

参照項目: 完全な構文の詳細については、“PROC MEANS ステートメント” (1100 ページ).

詳細

PRINT | NOPRINT を参照してください。

PROC SUMMARY が記述統計量を表示するかどうかを指定します。デフォルトで、PROC SUMMARY は表示出力を生成しませんが、PROC MEANS は生成します。

デフォルト NOPRINT

VAR ステートメント

分析変数と結果での順序を識別します。

デフォルト: VAR ステートメントを省略すると、PROC SUMMARY はオブザベーションの単純なカウントを生成します。PROC MEANS はその他のステートメントにリストされていないすべての数値変数を分析しようとします。

操作: PROC SUMMARY ステートメントで統計量を指定し、VAR ステートメントが省略された場合、または数値変数が OUTPUT ステートメントの統計量と関連付けられていない場合、PROC SUMMARY は処理を停止し、エラーメッセージが SAS ログに書き込まれます。

注: VAR ステートメントの説明については、PROC MEANS の VAR ステートメントを参照してください。

参照項目: 構文の詳細については、“VAR ステートメント” (1127 ページ)を参照してください。

64 章

TABULATE プロシジャ

概要: TABULATE プロシジャ

TABULATE プロシジャの動作について

TABULATE プロシジャは、データセットの一部またはすべての変数を使用して、記述統計量を表形式で表示します。単純なものから高度にカスタマイズされたものまで、さまざまなテーブルを作成できます。

PROC TABULATE は、MEANS、FREQ、REPORT などのその他の記述統計プロシジャによって計算される統計量と同じ統計量の多くを計算します。PROC TABULATE には、次のような機能があります。

- 表レポートを作成するための簡単で強力な方法を提供する
- 変数値の分類および変数間の階層関係の構築を柔軟に行える
- 変数とプロシジャ生成統計量のラベル付けおよびフォーマットを行うための手法を提供する

単純なテーブル

次の出力に、PROC TABULATE によって生成された単純なテーブルを示します。データセット"ENERGY" (2152 ページ)には、米国の地域、Northeast (1)、West (4)の各州における、2 種類の顧客、住宅と企業のエネルギー支出に関するデータが含まれています。テーブルでは、地域区分内の州に対する支出が合計されます。RTS オプションにより、列ヘッダーをハイフンを利用せずに表示するための十分なスペースが提供されます。

```
proc tabulate data=energy;
```

```
    class region division type;
    var expenditures;
    table region*division, type*expenditures /
          rts=20;
run;
```

アウトプット 64.1 *PROC TABULATE によって生成される単純なテーブル*

The SAS System

		Type	
		1	2
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
1	1	7477.00	5129.00
	2	19379.00	15078.00
4	3	5476.00	4729.00
	4	13959.00	12619.00

複合テーブル

次の出力は、アウトプット 64.1 (1867 ページ)の作成に使用されたものと同じデータセットを使用した、より複雑なテーブルです。このレポートを作成するステートメントを使用して、次のことを行います。

- 列ヘッダーと行ヘッダーをカスタマイズする
- すべてのテーブルセルに出力形式を適用する
- 住居顧客と企業顧客の支出を合計する
- 各区分の小計を計算する
- すべての地域の合計を計算する

このレポートを作成するプログラムの説明については、“例 6: 共通分類変数 ALL を使用した情報の要約” (1960 ページ)を参照してください。

アウトプット 64.2 *PROC TABULATE によって生成される複合テーブル*

```
Energy Expenditures for Each Region              2 (millions of dollars)
------------------------------------------------------------------- | | Customer
Base | | | |-------------------------| | | |Residential | Business | All | | |
Customers | Customers | Customers | |-----------------------+------------
+------------+------------| |Region |Division | | | | |-----------+-----------|
| | | |Northeast |New England| 7,477| 5,129| 12,606| | |-----------+------------
+------------+------------| | |Middle | | | | | |Atlantic | 19,379| 15,078|
34,457| | |-----------+------------+------------+------------| | |Subtotal |
26,856| 20,207| 47,063| |-----------+-----------+------------+------------
+------------| |West |Division | | | | | |-----------| | | | | |Mountain |
5,476| 4,729| 10,205| | |-----------+------------+------------+------------| | |
Pacific | 13,959| 12,619| 26,578| | |-----------+------------+------------
+------------| | |Subtotal | 19,435| 17,348| 36,783| |-----------------------
+------------+------------+------------| |Total for All Regions | $46,291|
$37,555| $83,846|
--------------------------------------------------------------------
```

PROC TABULATE と ODS (アウトプットデリバリシステム)

次の出力は、ハイパーテキストマークアップ言語(HTML)で作成されるテーブルを示します。Output Delivery System を PROC TABULATE と使用して、カスタマイズした出力を HTML、Rich Text Format (RTF)、Portable Document Format (PDF)などの出力形式で作成できます。このテーブルを生成するプログラムの説明については、“例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)を参照してください。

アウトプット 64.3 PROC TABULATE によって生成される HTML テーブル

Energy Expenditures
(millions of dollars)

Region by Division by Type		Type		Total
		Residential Customers	Business Customers	
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
West	Division			
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$19,435	$17,348	$36,783
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$46,291	$37,555	$83,846

概念: TABULATE プロシジャ

用語:TABULATE プロシジャ

次の図で、PROC TABULATE の説明で一般に使用される用語の一部について説明します。

図 64.1 PROC TABULATE テーブルの一部



図 64.2 PROC TABULATE テーブル次元



また、次の用語が PROC TABULATE の説明で頻出します。PROC TABULATE:

カテゴリ
分類変数の一意の値の組み合わせ。TABULATE プロシジャは、データセットのオブザベーションに存在する値の一意の組み合わせに対してそれぞれ別のカテゴリを作成します。PROC TABULATE によって作成されるカテゴリはそれぞれ、カテゴリを説明するページ、行、列がクロスするテーブルの 1 つ以上のセルによって表されます。

図 64.1 (1870 ページ)のテーブルには、Region、Division、Type の 3 つの分類変数が含まれています。これらの分類変数は、次のテーブルに表示される 8 つのカテゴリを形成します。(便宜上、カテゴリはフォーマットされた値について説明されます)。

表 64.1 3 つの分類変数から作成されるカテゴリ

部分	Division	Type
Northeast	New England	Residential Customers
Northeast	New England	Business Customers
Northeast	Middle Atlantic	Residential Customers
Northeast	Middle Atlantic	Business Customers
West	Mountain	Residential Customers
West	Mountain	Business Customers
West	Pacific	Residential Customers
West	Pacific	Business Customers

継続メッセージ
テーブルが複数の物理ページに及ぶ場合にテーブル下に表示されるテキスト。

ネストされた変数
値が別の変数の各値とともにテーブルに表示される変数。

図 64.1 (1870 ページ)では、Division が Region の下にネストされています。

ページ次元テキスト
テーブルにページ次元がある場合にテーブル上に表示されるテキスト。ただし、TABLE ステートメントで BOX=_PAGE_を指定すると、テーブル上に表示されるテキストがボックスに表示されます。図 64.2 (1871 ページ)では、値に続く単語 **Year:**がページ次元テキストです。

ページ次元テキストには、スタイルがあります。デフォルトのスタイルは *Beforecaption* です。スタイル使用の詳細については、"基本的なレポート記述プロシジャを使用した ODS スタイルの使用" (1918 ページ)を参照してください。

サブテーブル
1 つ以上の次元に連結要素が含まれている場合に TABLE ステートメントの各次元から単一の要素をクロスすることによって生成されるセルグループ。

図 64.1 (1870 ページ) にはサブテーブルは含まれていません。複数のサブテーブルで構成されるテーブルの説明については、"例 13: 分母定義を使用し、基本的な度数カウントとパーセントを表示する" (1984 ページ)を参照してください。

入力データセットのスレッド処理

THREADED オプションにより、入力データセットの並列処理の有効化または無効化が可能になります。スレッド処理により、処理操作における一定の並列処理が達成されます。この並列処理の目的は、所定の操作を完了するための処理時間を削減し、それによって追加 CPU リソースのコストを抑えることです。詳細については、"Support for Parallel Processing" (*SAS Language Reference: Concepts*)を参照してください。

SAS システムオプション CPUCOUNT=の値はスレッド化された並べ替えのパフォーマンスに影響します。CPUCOUNT=は、スレッド化されたプロシジャで使用できるシステム CPU の数を示します。

詳細については、"THREADS System Option" (*SAS System Options: Reference*) および "CPUCOUNT= System Option" (*SAS System Options: Reference*)を参照してください。

構文: TABULATE プロシジャ

要件: PROC TABULATE プロシジャステップでは、最低 1 つの TABLE ステートメントが必要です。

TABLE ステートメントで使用される変数によって、CLASS ステートメント、VAR ステートメント、またはその両方が必要です。

ヒント: ステートメント ATTRIB、FORMAT、LABEL および WHERE は、PROC TABULATE プロシジャと使用できます。詳細については、"複数のプロシジャで同じ機能を提供するステートメント" (67 ページ)を参照してください。

発生するデータベース内処理に対し、データは SAS データベース内処理用に適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、"PROC TABULATE のデータベース内処理" (1932 ページ)を参照してください。

PROC TABULATE <*option(s)*> <STYLE=*style-override(s)*>;
BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* ...><NOTSORTED>;
CLASS *variable(s)* </ *option(s)*> <STYLE=*style-override(s)* >;
CLASSLEV *variable(s)* </ STYLE=*style-override(s)*>;
FREQ *variable*;
KEYLABEL *keyword-1*='*description-1*' <*keyword-2*='*description-2*' ...>;
KEYWORD *keyword(s)* </ STYLE=*style-override(s)*>;
TABLE <<*page-expression*,> *row-expression*,>
column-expression </ *table-option(s)*>;
VAR *analysis-variable(s)* </ *option(s)*> <STYLE=*style-override(s)*>;
WEIGHT *variable*;

ステートメント	タスク	例
“PROC TABULATE ステートメント”	記述統計量を表形式で表示します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 6, Ex. 8, Ex. 12, Ex. 14, Ex. 15
“BY ステートメント”	BY グループごとに別のテーブルを作成します	
“CLASS ステートメント”	入力データセットの変数を分類変数として識別します	Ex. 3, Ex. 4
“CLASSLEV ステートメント”	分類変数の水準値ヘッダーのスタイルを指定します	Ex. 14, Ex. 15
“FREQ ステートメント”	値が各オブザベーションの度数を表す入力データセットの変数を識別します	
“KEYLABEL ステートメント”	キーワードのラベルを指定します	
“KEYWORD ステートメント”	キーワードヘッダーのスタイルを指定します	Ex. 14
“TABLE ステートメント”	作成するテーブルを記述します	Ex. 1, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10, Ex. 11, Ex. 12, Ex. 13, Ex. 14, Ex. 15
“VAR ステートメント”	入力データセットの変数を分析変数として識別します	Ex. 14
“WEIGHT ステートメント”	値が統計量計算で各オブザベーションに重みを付ける入力データセットの変数を識別します	

PROC TABULATE ステートメント

記述統計量を表形式で表示します。

構文

PROC TABULATE <*option(s)*>;

オプション引数の要約

CONTENTS=*link-name*
出力への HTML コンテンツリンクをカスタマイズします。

DATA=*SAS-data-set*
入力データセットを指定します。

NOTHREADS
入力データセットの並列処理を無効化します。

OUT=*SAS-data-set*
出力データセットを指定します。

THREADS | NOTHREADS
SAS システムオプション THREADS | NOTHREADS を無効にします。

TRAP
浮動小数点例外の復旧を可能にします。

Control the statistical analysis

ALPHA=*value*
信頼限界に対する信頼水準を指定します。

EXCLNPWGT
非正の重みが付いたオブザベーションを除外します。

PCTLDEF=
分位点を計算する算術定義を指定します。

QMARKERS=*number*
P^2 分位点推定方法に使用するサンプルサイズを指定します。

QMETHOD=OS | P2 | HIST
分位点推定方法を指定します。

QNTLDEF=1 | 2 | 3 | 4 | 5
分位点を計算する算術定義を指定します。

VARDEF=*divisor*
分散の計算のための分母を指定します。

Customize the appearance of the table

FORMAT=*format-name*
テーブルの各セルに対しデフォルトの出力形式を指定します。

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
テーブルの外枠と分割線の構成に使用する文字を定義します。

NOSEPS
行タイトルとテーブルの本文から水平区切り線を削除します。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
分類変数の値を指定した順序に従って並べ替えます。

STYLE=*style-override(s)*
テーブルの特定の領域に適用する 1 つ以上のスタイルの上書きを指定します。

Identify categories of data that are of interest

CLASSDATA=*SAS-data-set*
テーブルと出力データセットに含める分類変数の値の組み合わせを含む 2 次データセットを指定します。

EXCLUSIVE
テーブルと出力データセットから、CLASSDATA=データセットにない分類変数値のすべての組み合わせを除外します。

MISSING
欠損値を分類変数の有効値とみなします。

オプション引数

ALPHA=*value*
平均値に対する信頼限界を計算する信頼水準を指定します。信頼限界のパーセントは(1–*value*)×100 です。たとえば、**ALPHA=.05** の場合、信頼限界は 95%となります。

デフォルト .05

範囲 0 から 1

操作 信頼限界を計算するには、*statistic-keyword* LCLM または UCLM を指定します。

CLASSDATA=*SAS-data-set*
出力に表示が必要な分類変数の値の組み合わせを含むデータセットを指定します。CLASSDATA=データセットで発生し、入力データセットでは発生しない分類変数の値の組み合わせが各テーブルまたは出力データセットに表示され、度数はゼロとなります。

制限事項 CLASSDATA=データセットにすべての分類変数を含める必要があります。このデータの種類と出力形式は、入力データセットの対応する分類変数と一致する必要があります。

操作 EXCLUSIVE オプションを使用すると、PROC TABULATE は、分類変数の値の組み合わせが CLASSDATA=データセットにない入力データセットのオブザベーションを除外します。

ヒント CLASSDATA=データセットを使用して、入力データセットをフィルタまたは補足します。

例 “例 2: テーブルに表示する分類変数の組み合わせを指定する” (1947 ページ)

CONTENTS=*link-name*
TABULATE プロシジャを使用して生成された最初のテーブルの ODS 出力を指す HTML 目次のリンクを指定できるようになります。

注: CONTENTS=は、ODS HTML 出力のコンテンツファイルにのみ影響します。実際の TABULATE プロシジャレポートには影響しません。

DATA=*SAS-data-set*
入力データセットを指定します。

参照項目 “入力データセット” (25 ページ)

EXCLNPWGT
非正(ゼロまたは負)の重みがついたオブザベーションを分析から除外します。デフォルトでは、PROC TABULATE は、負の重みが付いたオブザベーションを重みが 0 のオブザベーションとして処理し、オブザベーション合計数に含めます。

別名 EXCLNPWGTS

参照項目 “WEIGHT=*weight-variable*” (1910 ページ)および“WEIGHT ステートメント” (1910 ページ)

EXCLUSIVE
テーブルと出力データセットから、CLASSDATA=データセットにない分類変数のすべての組み合わせを除外します。

要件 CLASSDATA=データセットが指定されていない場合、EXCLUSIVE オプションは無視されます。

例 “例 2: テーブルに表示する分類変数の組み合わせを指定する” (1947 ページ)

FORMAT=*format-name*
各テーブルセルの値に対しデフォルトの出力形式を指定します。SAS 出力形式またはユーザー定義の出力形式はどれでも使用できます。

別名 F=

デフォルト FORMAT=を省略すると、PROC TABULATE は BEST12.2 をデフォルトの出力形式として使用します。

操作 TABLE ステートメントで指定される出力形式は、FORMAT=で指定される出力形式に優先します。

ヒント FORMAT=オプションは、テーブルの印刷に使用される印刷位置数の制御に特に便利です。

例 “例 1: 基本的な 2 次元表の作成” (1944 ページ)

“例 6: 共通分類変数 ALL を使用した情報の要約” (1960 ページ)

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
テーブルの外枠と分割線を構成するために使用する文字を定義します。

position(s)
SAS フォーマット文字列における 1 つ以上の文字の位置を識別します。スペースまたはカンマで位置を区切ります。

デフォルト *position(s)*の省略は、20 のすべての可能な SAS フォーマット文字を順番に指定することと同じです。

formatting-character(s)
指定位置に使用する文字をリストします。PROC TABULATE は、*formatting-character(s)*の文字を表示されている順序で *position(s)*に割り当てます。たとえば、次のオプションではアスタリスク(*)を 3 番目のフォーマット文字に、数字記号(#)を 7 番目の文字に割り当てます。その他の文字は変更されません。

```
formchar(3,7)='*#'
```

PROC TABULATE では、SAS で提供している 20 のフォーマット文字のうち 11 が使用されます。表 64.2 (1877 ページ)に、PROC TABULATE で使用されるフォーマット文字を示します。図 64.3 (1877 ページ)に、PROC TABULATE からの出力における各フォーマット文字の使用を示します。

表 64.2 *PROC TABULATE によって使用されるフォーマット文字*

Position	デフォルト	表示に使用
1	|	右枠、左枠、列間の区切り
2	-	上枠、下枠、および行間の水平区切り線
3	-	左枠の先頭文字
4	-	列を区切る文字の行の先頭文字
5	-	右枠の先頭文字
6	|	水平区切り線の行の最左文字
7	+	垂直文字の列と水平文字の行の交点
8	|	水平区切り線の行の最右文字
9	-	左枠の末尾文字
10	-	列を区切る文字の行の末尾文字
11	-	右枠の末尾文字

図 64.3 *PROC TABULATE 出力のフォーマット文字*



制限事項 FORMCHAR=プションは、従来の SAS モノスペース出力の出力先にのみ影響します。

操作 SAS システムオプション FORMCHAR=では、デフォルトのフォーマット文字を指定します。システムオプションは、フォーマット文字の全体の文字列を定義します。プロシジャの FORMCHAR=オプションでは、選択した文字を再定義できます。

ヒント 16 進数文字を含む *formatting-characters* の文字を使用できます。16 進数文字を使用する場合、**x** を終了引用符の後に付ける必要があります。たとえば、次のオプションでは、16 進文字 2D が 3 番目のフォーマット文字に、16 進文字 7C が 7 番目の文字にそれぞれ割り当てられます。その他の文字は変わりません。

```
formchar(3,7)='2D7C'x
```

*formatting-character(s)*に対しすべてブランクを指定すると、外枠または区切り線のないテーブルが作成されます。

```
formchar(1,2,3,4,5,6,7,8,9,10,11)          ='           '
```

(11 のブランク)

参照項目 フォーマット出力を使用した例については、*PROC TABULATE by Example, Second Edition* を参照してください。

どの 16 進コードをどの文字に使用するかについては、ハードウェアのドキュメントを参照してください。

MISSING

欠損値を有効値とみなし、分類変数の組み合わせを作成します。数値を表すために使用される特殊欠損値(文字 A から Z、アンダースコア(_)文字)がそれぞれ個別の値としてみなされます。各欠損値に対するヘッダーがテーブルに表示されます。

デフォルト MISSING を省略すると、PROC TABULATE は分類変数が欠損値のオブザベーションをレポートに含みません。

参照項目 “欠損分類変数を含むオブザベーションを包含” (1937 ページ)

“Creating Special Missing Values” (*SAS Language Reference: Concepts*) (特別な意味を持つ欠損値の説明)

NOSEPS

行タイトルとテーブルの本文から水平区切り線を削除します。水平区切り線は、ネストされた列ヘッダーの間に残ります。

制限事項 NOSEPS オプションは、従来の SAS monospace 出力の出力先にのみ影響します。

ヒント 区切り線を削除するのではなくブランクと置き換える場合、オプション “FORMCHAR <(*position(s)*)>='*formatting-character(s)*' ” (1876 ページ)を使用します。

例 “例 8: 行ヘッダーをインデントし、水平区切り線を削除する” (1964 ページ)

NOTHREADS
“THREADS | NOTHREADS” (1883 ページ)を参照してください。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
テーブルのヘッダーを形成する分類変数の値の一意の組み合わせを指定した順序に従って作成するための並べ替え順序を指定します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 CLASS ステートメントで PRELOADFMT を使用すると、各分類変数の値の順序が、PROC FORMAT が関連するユーザー定義出力形式の値の格納に使用する順序と一致します。CLASSDATA=オプションを使用する場合、PROC TABULATE は CLASSDATA= データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC TABULATE は最初にユーザー定義出力形式を使用して出力を並べ替えます。EXCLUSIVE を省略すると、PROC TABULATE はユーザー定義の出力形式と CLASSDATA=値の後に入力データセットの分類変数の一意の値を発生順に追加します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

データベース内プロシジャを ORDER=DATA オプションとともに使用している場合、結果がばらつく可能性があります。DBMS テーブルの行には固有の順序はありません。SAS プロシジャのデータベーステーブルに書き込まれた行の順序は保持されない可能性があります。

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。出力形式が数値分類変数に割り当てられていない場合、デフォルトの出力形式 BEST12.が使用されます。この順序は、使用している動作環境によって異なります。

別名 FMT | EXTERNAL

FREQ
度数カウントの降順で値を並べ替えます。

操作 CLASS ステートメントで ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

UNFORMATTED
値をフォーマットされていない値別に並べ替えます。これにより、PROC SORT と同じ順序が作成されます。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を時系列順に表示する場合に特に便利です。

別名 UNFMT | INTERNAL

デフォルト UNFORMATTED

操作 CLASS ステートメントで PRELOADFMT オプションを使用すると、PROC TABULATE はレベルをユーザー定義出力形式の値の順序で並べ替えます。

例 “ORDER=DATA を使用したヘッダーの順序について” (1943 ページ)

OUT=*SAS-data-set*
出力データセットを指定します。*SAS-data-set* は存在しない場合、PROC TABULATE が作成します。

出力データセットのオブザベーション数は、テーブルで使用されるデータのカテゴリ数と、生成されるサブテーブル数によって異なります。出力データセットには、これらの変数が次の順序で含まれています。

BY 変数
BY ステートメントで表示される変数。

分類変数
CLASS ステートメントで表示される変数。

TYPE
オブザベーションの要約統計量を生成した分類変数の組み合わせを表示する文字変数。_TYPE_の位置はそれぞれ、CLASS ステートメントの 1 つの変数を表します。その変数が統計量を生成したカテゴリ内にある場合、位置には 1 が含まれます。その他の場合、位置には 0 が含まれます。共通分類変数 ALL を使用しない単純な PROC TABULATE ステップでは、_TYPE_のすべての値には 1 のみ含まれます。これは、検討中のカテゴリのみすべての分類変数を含むためです。変数 ALL を使用すると、テーブルには一部の分類変数を含まないカテゴリに対するデータが含まれるため、_TYPE_の位置には 1 と 0 が含まれます。

PAGE
オブザベーションを含む論理ページ。

TABLE
オブザベーションを含むテーブル数。

統計量
データセットの各オブザベーションに対して計算される統計量。

例 “例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (1950 ページ)

PCTLDEF=
“QNTLDEF=1 | 2 | 3 | 4 | 5” (1881 ページ)を参照してください。

QMARKERS=*number*
P^2 分位点推定方法に使用するマーカーのデフォルト数を指定します。マーカー数によって固定メモリ空間のサイズを制御します。

デフォルト デフォルト値は、要求する分位点によって異なります。中央値(P50)の場合、*number* は 7 です。分位点(P25 と P75)の場合、*number* は 25 です。分位点 P1、P5、P10、P90、P95 または P99 の場合、*number* は 105 です。複数の分位点を要求すると、PROC TABULATE は *number* の最大デフォルト値を使用します。

範囲 3 より大きい奇数の整数

ヒント デフォルト設定を超えるマーカー数を増やして推定の正確度を向上し、マーカー数を減らしてメモリと計算時間を削減します。

参照項目 “分位点” (1131 ページ)

QMETHOD=OS | P2 | HIST
PROC TABULATE が分位点の計算時に入力データの処理に使用する方法を指定します。オブザベーション数が QMARKERS=値および QNTLDEF=5 以下の場合、両方の方法による結果は同じになります。

OS
順序統計量を使用します。PROC UNIVARIATE は、この方法を使用します。

注: この方法は、非常に多くのメモリを必要とする可能性があります。

P2 | HIST
P^2 方法を使用して、分位点を概算します。

デフォルト OS

制限事項 QMETHOD=P2 の場合、PROC TABULATE は MODE または重み付き分位点を計算しません。

ヒント QMETHOD=P2 の場合、一部の分位点(P1、P5、P95、P99)に対して信頼できる推定は、一部の種類のデータに対して信頼できないことがあります。

参照項目 “分位点” (1131 ページ)

QNTLDEF=1 | 2 | 3 | 4 | 5
QMETHOD=OS の指定時にプロシジャが分位点の計算に使用する算術定義を指定します。QMETHOD=P2 の場合、QNTLDEF=5 を使用する必要があります。

別名 PCTLDEF=

デフォルト 5

参照項目 “分位数と関連統計量” (2083 ページ)

STYLE=*style-override(s)*
テーブルのデータセルに適用する 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、データセルの背景色が赤になるように指定します。

```
proc tabulate data=one style=[backgroundcolor=red];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override これは次の形式を取ります。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1* <*style-attribute-name-2=style-attribute-value-2 …*>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で大かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用してスタイル要素を割り当てる” (1930 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

別名 S=

制限事項 STYLE=オプションは、LISTING 以外の出力先にのみ影響します。

ヒント STYLE=オプションをその他のステートメントまたは次元式で使用して、テーブルのその他の部分にスタイル要素を指定できます。

欠損値を含むデータセルにスタイル要素を指定するには、TABLE ステートメント MISSTEXT=オプションで STYLE=を使用します。

角かっこ([と])の代わりに中かっこ({と})を使用できます。

参照項目 PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

例 “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

THREADS | NOTHREADS
入力データセットの並列処理を有効化または無効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします(詳細については、“Support for Parallel Processing” (*SAS Language Reference: Concepts*)を参照)。

デフォルト SAS システムオプション THREADS | NOTHREADS の値。

制限事項 サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

操作 PROC TABULATE は、BY ステートメントが指定されている、または SAS システムオプション CPUCOUNT の値が 2 未満の場合を除いて、SAS システムオプション THREADS の値を使用します。この場合、PROC TABULATE ステートメントで THREADS オプションを指定して、PROC TABULATE が並列処理を使用するように指定できます。並列処理でもあるマルチスレッド処理が有効な場合、オブザベーションは予想外の順序で返されることがあります。ただし、BY ステートメントが指定されている場合、オブザベーションは正しく並べ替えられます。

TRAP
データ処理中に通常の SAS FPE 処理によって復旧できない浮動小数点例外(FPE)の復旧を可能にします。TRAP オプションを使用しない場合、算術例外の場合に PROC TABULATE が終了するように、通常の SAS FPE 処理はそのまま有効です。

VARDEF=*divisor*
分散と標準偏差の計算に使用する分母を指定します。次のテーブルに、*divisor* に可能な値と、関連する分母を示します。

表 64.3 VARDEF=に可能な値

値	分母	分母の式
DF	自由度	$n - 1$
N	オブザベーション数	n
WDF	重みの合計 - 1	$(\Sigma_i w_i) - 1$
WEIGHT | WGT	重みの合計	$\Sigma_i w_i$

プロシジャは分散を*CSS*/*divisor*として計算します。*CSS*は修正平方和で、$\Sigma (x_i - \bar{x})^2$と等しくなります。分析変数を重み付けする場合、*CSS*は$\Sigma w_i(x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト DF

要件 平均値の標準誤差を計算するには、VARDEF=のデフォルト値を使用します。

ヒント WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。i 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$、およびw_iは i 番目のオブザベーションの重みです。これにより、オブザベーションの分散の推定が単位重みとともに生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな n に対し)$\sigma^2/\bar{w}$の推定になります。$\bar{w}$は、平均重みです。これにより、オブザベーションの分散の漸近的推定が平均重みとともに生成されます。

参照項目 “重み付き統計量の例” (76 ページ)

BY ステートメント

BY グループごとに別のページに別のテーブルを作成します。

参照項目: “BY” (68 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* ...><NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、データセットのオブザベーションは指定するすべての変数別に並べ替えるか、適切にインデックス付けする必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
オブザベーションが BY ステートメントの文字 DESCENDING の直後に続く変数で降順に並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。たとえば、オブザベーションは時系列でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

CLASS ステートメント

テーブルの分類変数を識別します。分類変数によって、PROC TABULATE が統計量の計算に使用するカテゴリが決定されます。

注: オプションがない CLASS ステートメントでは、内部デフォルト、または PROC TABULATE ステートメントのオプションにより指定された値が使用されます。たとえば次のコードでは、変数 **c** と **d** には内部デフォルトが使用されることがあります。PROC TABULATE ステートメントにおいて ORDER=オプションを指定したとすると、変数 c と d には、PROC TABULATE ステートメントの ORDER=オプションにより指定された値が使用されます。

```
class a b / order=data;
class c d;
```

ヒント: 複数の CLASS ステートメントを使用できます。

一部の CLASS ステートメントオプションも PROC TABULATE ステートメントで使用できます。これらは、CLASS ステートメントで指定するものだけでなく、すべての CLASS 変数に影響します。

例: “例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (1950 ページ)
“例 4: マルチラベル出力形式の使用” (1955 ページ)

構文

CLASS *variable(s)* </ *option(s)*>;

オプション引数の要約

ASCENDING
分類変数値を昇順で並べ替えるように指定します。

DESCENDING
分類変数値を降順で並べ替えるように指定します。

EXCLUSIVE
ユーザー定義の出力形式の事前にロードされた範囲にない分類変数のすべての組み合わせをテーブルと出力データセットから除外します。

GROUPINTERNAL
PROC TABULATE が値をグループ化し、分類変数の組み合わせを作成する際に分類変数に出力形式を適用しないように指定します。

MISSING
欠損値を有効な分類変数水準とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

MLF
マルチラベル出力形式が分類変数に割り当てられている場合に、PROC TABULATE が指定範囲または重複する範囲に出力形式ラベルを使用して、サブグループの組み合わせを作成できるようになります。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
出力の分類変数の水準をグループ化する順序を指定します。

PRELOADFMT
すべての出力形式が分類変数に対して事前にロードされることを示します。

STYLE=*style-override*
ページ次元テキストと分類変数名ヘッダーに使用する 1 つ以上のスタイルの上書きを指定します。

必須引数

variable(s)
プロシジャがデータのグループ化に使用する 1 つ以上の変数を指定します。CLASS ステートメントの変数は、*分類変数*といいます。分類変数は、数値か文字です。分類変数には連続値を含めることができますが、通常は変数の分類を定義するいくつかの不連続値が含まれます。データを分類変数別に並べ替える必要はありません。

操作 変数名と統計量名が同じ場合、統計量名を一重引用符か二重引用符で囲みます。

オプション引数

ASCENDING
分類変数値を昇順で並べ替えるように指定します。

別名 ASCEND

操作 ASCENDING と DESCENDING の両方を指定すると、PROC TABULATE は警告メッセージを発行して、両方のオプションを無視します。

DESCENDING
分類変数値を降順で並べ替えるように指定します。

別名 DESCEND

デフォルト ASCENDING

操作 ASCENDING と DESCENDING の両方を指定すると、PROC TABULATE は警告メッセージを発行して、両方のオプションを無視します。

EXCLUSIVE
ユーザー定義の出力形式の事前にロードされた範囲にない分類変数のすべての組み合わせをテーブルと出力データセットから除外します。

要件 分類変数出力形式を事前にロードするには、CLASS ステートメントで PRELOADFMT オプションを指定する必要があります。

例 “例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (1950 ページ)

GROUPINTERNAL
PROC TABULATE が値をグループ化し、分類変数の組み合わせを作成する際に分類変数に出力形式を適用しないように指定します。

操作 CLASS ステートメントで PRELOADFMT オプションを指定すると、PROC TABULATE は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ORDER=FORMATTED オプションを指定すると、PROC TABULATE は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ヒント GROUPINTERNAL オプションは、数値変数に不連続値が含まれている場合にコンピュータリソースを節約します。

MISSING
欠損値を有効な分類変数水準とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING オプションを省略すると、PROC TABULATE により、CLASS 変数値が欠損しているオブザベーションがテーブルと出力データセットから除外されます。

参照項目 “Creating Special Missing Values” (*SAS Language Reference: Concepts*) (特別な意味を持つ欠損値の詳細)

MLF
マルチラベル出力形式が分類変数に割り当てられている場合に、PROC TABULATE が指定範囲または重複する範囲に出力形式ラベルを使用して、サブグループの組み合わせを作成できるようになります。

注: フォーマットされた値が重複する場合、1 つの内部分類変数値が複数の分類変数サブグループの組み合わせにマップします。そのため、すべてのサブグループの N 統計量の合計がデータセットのオブザベーション数(N 統計量全体)を超えます。

要件 VALUE ステートメントで PROC FORMAT と MULTILABEL オプションを使用して、マルチラベル出力形式を作成する必要があります。

操作 MLF と ORDER=FREQ を使用すると、フォーマットされた値に使用する順序が生成されない場合があります。

MLF を指定すると、分類変数のフォーマットされた値は内部値になります。そのため、ORDER=FORMATTED を指定すると、ORDER=UNFORMATTED を指定した場合と同じ結果になります。

ヒント MLF を省略すると、PROC TABULATE は最初の外部出力形式値に対応する 1 次出力形式ラベルを使用して、サブグループの組み合わせを決定します。

参照項目 “MULTILABEL” (870 ページ) (FORMAT プロシジャの VALUE ステートメント)

例 “例 4: マルチラベル出力形式の使用” (1955 ページ)

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
出力の分類変数の水準をグループ化する順序を指定します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 PRELOADFMT を使用すると、各分類変数の値の順序は、PROC FORMAT が関連するユーザー定義出力形式の値の保存に使用する順序に一致します。PROC ステートメントで CLASSDATA=オプションを使用すると、PROC TABULATE は CLASSDATA=データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC TABULATE は最初にユーザー定義出力形式を使用して出力を並べ替えます。PROC ステートメントで EXCLUSIVE を省略すると、PROC TABULATE はユーザー定義出力形式と CLASSDATA=値の後に、入力データセットにある分類変数の一意の値を発生順序で配置します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。この順序は、使用している動作環境によって異なります。

別名 FMT | EXTERNAL

FREQ
度数カウントの降順で値を並べ替えます。

操作 ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

UNFORMATTED
値をフォーマットされていない値別に並べ替えます。これにより、PROC SORT と同じ順序が作成されます。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を時系列順に表示する場合に特に便利です。

別名 UNFMT | INTERNAL

デフォルト UNFORMATTED

操作 CLASS ステートメントで PRELOADFMT オプションを使用すると、PROC TABULATE はレベルをユーザー定義出力形式の値の順序で並べ替えます。

ヒント デフォルトでは、FREQ 以外のすべての順序は昇順です。降順の順序の場合は、DESCENDING オプションを使用します。

例 “ORDER=DATA を使用したヘッダーの順序について” (1943 ページ)

PRELOADFMT
すべての出力形式が分類変数に対して事前にロードされることを示します。

要件 PRELOADFMT は、EXCLUSIVE、ORDER=DATA または PRINTMISS を指定して、出力形式を分類変数に割り当てる場合のみ有効です。EXCLUSIVE、ORDER=DATA または PRINTMISS も指定せずに PRELOADFMT を指定すると、警告メッセージが SAS ログに書き込まれます。

操作 PROC TABULATE 出力を入力データセットに存在するフォーマットされた分類変数値の組み合わせに制限するには、CLASS ステートメントで EXCLUSIVE オプションを使用します。

ユーザー定義出力形式のすべての範囲と値を出力に含めるには、TABLE ステートメントで PRINTMISS オプションを使用します。PRELOADFMT を PRINTMISS と使用する際は注意が必要です。この機能により、フォーマットされた分類変数の可能なすべての組み合わせが作成されます。これらの組み合わせの一部が意味をなさない場合があります。

例 “例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (1950 ページ)

STYLE=*style-override*
ページ次元テキストと分類変数名ヘッダーに使用する 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、ページ次元テキストの背景色と分類変数名ヘッダーが薄緑色になるように指定します。

```
class region division prodtype / style=[background=lightgreen];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override これは次の形式を取ります。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1 <style-attribute-name-2=style-attribute-value-2 …>*]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で大かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用してスタイル要素を割り当てる” (1930 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

別名 S=

制限事項 STYLE=オプションは、LISTING 以外の出力先にのみ影響します。

ヒント CLASS ステートメントのページ次元テキストに指定されるスタイル要素を無効にする場合、TABLE ステートメントページ次元式でスタイル要素を指定できます。

CLASS ステートメントで分類変数名ヘッダーに指定されるスタイル要素を無効にする場合、関連する TABLE ステートメント次元式でスタイル要素を指定できます。

ページ次元式にネストされた要素が複数含まれている場合、Beforecaption スタイル要素がネスティングの最初の要素のスタイル要素となります。

CLASS ステートメントでの STYLE=の使用は、PROC TABULATE ステートメントでの使用とはやや異なります。CLASS ステートメントでは、継承は行と列では異なります。行の場合、親ヘッダーは現在のヘッダーの左側に置かれます。列の場合、親ヘッダーは現在のヘッダーの上に置かれます。

参照項目 PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

例 “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

詳細

分類変数の欠損値の PROC TABULATE による処理方法

デフォルトでは、オブザベーションに分類変数に対する欠損値が含まれている場合、PROC TABULATE は作成したすべてのテーブルからそのオブザベーションを除外します。CLASS ステートメントは、PROC TABULATE ステップのすべての TABLE ステートメントに適用されます。そのため、変数を分類変数として定義すると、PROC TABULATE は、変数が 1 つ以上のテーブルに対し TABLE ステートメントに記述されない場合でも、変数に欠損値があるすべてのオブザベーションをすべてのテーブルから除外します。

PROC TABULATE ステートメントで MISSING オプションを指定すると、プロシジャは欠損値をすべての分類変数に対する有効水準とみなします。CLASS ステートメントで MISSING オプションを指定すると、PROC TABULATE は欠損値を CLASS ステートメントで指定される分類変数に対する有効水準とみなします。

CLASSLEV ステートメント

分類変数水準値ヘッダーにスタイル要素を指定します。

制限事項: このステートメントは、LISTING 以外の出力先にのみ影響します。

例: “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)
“例 15: スタイル優先” (2004 ページ)

構文

CLASSLEV *variable(s)* </ STYLE=*style-override(s)*>;

必須引数

variable(s)
スタイル要素を指定する CLASS ステートメントから 1 つ以上の分類変数を指定します。

オプション引数

STYLE=*style-override*
分類変数水準値ヘッダーに 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、分類変数水準名ヘッダーの背景色が黄色になるように指定します。

```
classlev region division prodtype  / style=[background=yellow];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override これは次の形式を取ります。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1* <*style-attribute-name-2=style-attribute-value-2 …*>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で大かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用してスタイル要素を割り当てる” (1930 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

別名 S=

制限事項 STYLE=オプションは、LISTING 以外の出力先にのみ影響します。

ヒント CLASSLEV ステートメントでの STYLE=の使用は、PROC TABULATE ステートメントでの使用とはやや異なります。CLASSLEV ステートメントでは、継承は行と列では異なります。行の場合、親ヘッダーは現在のヘッダーの左側に置かれます。列の場合、親ヘッダーは現在のヘッダーの上に置かれます。

CLASSLEV ステートメントで指定されるスタイル要素を無効にする場合、関連する TABLE ステートメント次元式でスタイル要素を指定できます。

角かっこ([と])の代わりに中かっこ({と})を使用できます。

参照項目 PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

例 “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

FREQ ステートメント

各オブザベーションの度数を含む数値変数を指定します。

ヒント: FREQ ステートメントと WEIGHT ステートメントの影響は、自由度の計算時以外は似ています。

例: “FREQ” (72 ページ)

構文

FREQ *variable*;

必須引数

variable
値がオブザベーションの度数を表す数値変数を指定します。FREQ ステートメントを使用する場合、プロシジャは各オブザベーションが *n* オブザベーションを表すとみなします。*n* は *variable* の値です。*n* は整数でない場合、切り捨てられます。*n* が 1 未満または欠損値の場合、プロシジャはそのオブザベーションを統計量の計算に使用しません。

度数変数の合計は、オブザベーションの合計数を表します。

KEYLABEL ステートメント

PROC TABULATE ステップで使用するキーワードのラベルを指定します。PROC TABULATE は、キーワードの表示にこのラベルを使用します。ラベルが指定されない場合はキーワードが表示されます。

構文

KEYLABEL *keyword-1*='*description-1*' <*keyword-2*='*description-2*' ...>;

必須引数

keyword
“PROC TABULATE で使用できる統計量” (1911 ページ)で述べている統計量に対するキーワードの 1 つ、または共通分類変数 ALL です(“次元式で使用できる要素” (1904 ページ)を参照)。

description
ラベルとして使用する最大 256 文字です。構文にあるように、*description* を引用符で囲む必要があります。

制限事項 各キーワードには、特定の PROC TABULATE ステップにラベルを 1 つだけ含めることができます。同じキーワードに対し複数のラベルを要求すると、PROC TABULATE はステップで指定される最後のラベルを使用します。

KEYWORD ステートメント

キーワードヘッダーにスタイル要素を指定します。

制限事項: このステートメントは、LISTING 以外の出力にのみ影響します。

例: “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

構文

KEYWORD *keyword(s)* </ STYLE=*style-override(s)*>;

必須引数

keyword

“PROC TABULATE で使用できる統計量” (1911 ページ)で述べている統計量に対するキーワードの 1 つ、または共通分類変数 ALL です(“次元式で使用できる要素” (1904 ページ)を参照)。

オプション引数

STYLE=*style-override*

キーワードヘッダーに 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、キーワードヘッダーの背景色が淡い黄色になるように指定します。

```
keyword all sum / style=[background=linen];
```

style-override

レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override これは次の形式を取ります。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1* <*style-attribute-name-2=style-attribute-value-2 …*>]

<PARENT>

データセルがその親ヘッダーのスタイル要素を使用することを指定します。データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で大かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目

PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目

PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用してスタイル要素を割り当てる” (1930 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目

PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

別名 S=

制限事項 STYLE=オプションは、LISTING 以外の出力先に影響します。

ヒント KEYWORD ステートメントでの STYLE=の使用は、PROC TABULATE ステートメントでの使用とはやや異なります。KEYWORD ステートメントでは、継承は行と列では異なります。行の場合、親ヘッダーは現在のヘッダーの左側に置かれます。列の場合、親ヘッダーは現在のヘッダーの上に置かれます。

KEYWORD ステートメントで指定されるスタイル要素を無効にする場合、関連する TABLE ステートメント次元式でスタイル要素を指定できます。

参照項目 PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

例 “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

TABLE ステートメント

印刷するテーブルを記述します。

要件 TABLE ステートメントのすべての変数を VAR ステートメントまたは CLASS ステートメントに記述する必要があります。

ヒント: 複数のテーブルを作成するには、複数の TABLE ステートメントを使用します。

変数名リストショートカットの使用が、TABLE ステートメント内でサポートされています。詳細については、“変数名リストを示すショートカット” (57 ページ)を参照してください。

構文

TABLE <<*page-expression*,> *row-expression*,>
column-expression </ *table-option(s)*>;

オプション引数の要約

BOX={<*label=value*> <STYLE=*style-override(s)*>}
行タイトルの上の空のボックスにテキストとスタイルの上書きを指定します。

BOX=*value*
行タイトルの上の空のボックスにテキストを指定します。

CONDENSE
1 枚の印刷ページにできるだけ多くの論理ページを印刷します。または可能な場合は、1 ページには広すぎるテーブルの複数のページを別のページに印刷するのではなく、1 ページに相互に重ねて印刷します。

CONTENTS=*link-name*
TABLE ステートメントを使用して生成されるテーブルの ODS 出力を指す HTML 目次のリンクを指定できるようになります。

FORMAT_PRECEDENCE=PAGE|ROW|COLUMN|COL

ページ次元(PAGE)、行次元(ROW)、列次元(COLUMN または COL)に対して指定される出力形式がテーブルセルのコンテンツに適用されるかどうかを指定します。

FUZZ=*number*
分析変数値と、度数カウント以外のテーブルセル値が比較される数値を提供し、自明な値(FUZZ=値未満の絶対値)を計算と印刷から削除します。

INDENT=*number-of-spaces*
ネストされている行ヘッダーをインデントするスペース数を指定し、分類変数に対する行ヘッダーを非表示にします。

MISSTEXT='*text*'
欠損値を含むテーブルセルに対して、印刷される最大 256 文字のテキストを提供します。

MISSTEXT={<*label*='*text*'> <STYLE=*style-override(s)*>}
印刷される最大 256 文字のテキストを提供し、欠損値を含むテーブルセルにスタイルの上書きを指定します。

NOCELLMERGE
データセルがテーブルのその他のデータセルと結合されないように指定します。

NOCONTINUED
複数ページにわたってテーブルの下部に表示される続行メッセージ **continued** を非表示にします。

page-expression
テーブルのページを定義します。

PRINTMISS
変数のヘッダーが印刷されるたびに分類変数に対して発生するすべての値を印刷します。これらのヘッダーが作成するセルの一部に対しデータがない場合でも同様です。

ROW=*spacing*
行クロスのすべてのタイトル要素が、ブランクの場合でも割り当てられたスペースかどうかを指定します。

row-expression
テーブルの行を定義します。

RTSPACE=*number*
行ヘッダーの外枠文字の印刷に使用されるスペースを含む、行次元のすべてのヘッダーに割り当てる印刷位置の数を指定します。

STYLE_PRECEDENCE=PAGE | ROW | COLUMN | COL
ページ次元(PAGE)、行次元(ROW)または列次元(COLUMN または COL)に対して指定されるスタイルがテーブルセルのコンテンツに適用されるかどうかを指定します。

STYLE=*style-override*
テーブルセル以外のテーブル部分に使用する 1 つ以上のスタイルの上書きを指定します。

必須引数

column-expression
テーブルの列を定義します。次元式構成の詳細については、“詳細” (1904 ページ)を参照してください。

制限事項 列次元は、TABLE ステートメントの最後の次元です。行次元または行次元とページ次元は、列次元の前に置くことができます。

オプション引数

page-expression
テーブルのページを定義します。次元式構成の詳細については、“詳細” (1904 ページ)を参照してください。

制限事項 ページ次元は、テーブルステートメントの最初の次元です。行次元と列次元はページ次元の後に置く必要があります。

例 “例 9: 複数ページテーブルの作成” (1967 ページ)

row-expression
テーブルの行を定義します。次元式構成の詳細については、“詳細” (1904 ページ)を参照してください。

制限事項 行次元は、テーブルステートメントの最後の次元の隣です。列次元は、行次元の後に置く必要があります。ページ次元は行次元の前に置くことができます。

テーブルオプション

BOX=*value*
行タイトルの上の空のボックスにテキストを指定します。

Value は、次のうちいずれかになります。

PAGE
ページ次元テキストをボックスに書き込みます。ボックスに合わないページ次元テキストはボックスの上のデフォルトの位置に置かれ、ボックスは空のままです。

BOX={*<label=value>* <STYLE=*style-override(s)*>}
行タイトルの上の空のボックスにテキストとスタイルの上書きを指定します。

Value は、次のうちいずれかになります。

PAGE
ページ次元テキストをボックスに書き込みます。ボックスに合わないページ次元テキストはボックスの上のデフォルトの位置に置かれ、ボックスは空のままです。

'*string*'
引用符で囲まれた文字列をボックスに書き込みます。ボックスに合わない文字列は、切り捨てられます。

variable
変数の名前(または変数に含まれている場合はラベル)をボックスに書き込みます。ボックスに合わない名前またはラベルは、切り捨てられます。

STYLE=オプションの引数とその使用方法については、TABLE ステートメントの STYLE= (1902 ページ)を参照してください。

例 “例 9: 複数ページテーブルの作成” (1967 ページ)

“例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

CONDENSE
1 枚の印刷ページにできるだけ多くの論理ページを印刷します。または可能な場合は、1 ページには広すぎるテーブルの複数のページを別のページに印刷するのではなく、1 ページに相互に重ねて印刷します。*logical page* は、次のうちいずれかにあるすべての行と列です。

- ページ次元カテゴリ(BY グループ処理なし)
- BY グループ(ページ次元なし)
- 単一の BY グループ内のページ次元カテゴリ

制限事項 CONDENSE オプションは、BY ステートメントによって生成されるページでは無効です。BY グループの最初のテーブルは、常に新しいページで始まります。

例 “例 9: 複数ページテーブルの作成” (1967 ページ)

CONTENTS=*link-name*
TABLE ステートメントを使用して生成されるテーブルの ODS 出力を指す HTML 目次のリンクを指定できるようになります。

注: CONTENTS=は、ODS HTML 出力のコンテンツファイルにのみ影響します。実際の TABULATE プロシジャレポートには影響しません。

FORMAT_PRECEDENCE=PAGE|ROW|COLUMN|COL
ページ次元(PAGE)、行次元(ROW)、列次元(COLUMN または COL)に対して指定される出力形式がテーブルセルのコンテンツに適用されるかどうかを指定します。

デフォルト COLUMN

FUZZ=*number*
分析変数値と、度数カウント以外のテーブルセル値が比較される数値を提供し、自明な値(FUZZ=値未満の絶対値)を計算と印刷から削除します。絶対値が FUZZ=値未満である数値は、計算と印刷でゼロとして扱われます。デフォルト値は、使用しているコンピュータ上の表示可能な最小浮動小数点数です。

INDENT=*number-of-spaces*
ネストされている行ヘッダーをインデントするスペース数を指定し、分類変数に対する行ヘッダーを非表示にします。

制限事項 HTML、RTF、Printer 出力先で、INDENT=オプションは分類変数の行ヘッダーを非表示にしますが、ネストされている行ヘッダーをインデントしません。

ヒント 行次元にクロスがない場合は、インデントは実行されません。そのため、*number-of-spaces* の値は無効になります。ただしそのような場合、INDENT=は分類変数に対する行ヘッダーを非表示にします。

参照項目 “例 8: 行ヘッダーをインデントし、水平区切り線を削除する” (1964 ページ) (クロスあり) “例 9: 複数ページテーブルの作成” (1967 ページ)(クロスなし)

MISSTEXT='*text*'
欠損値を含むテーブルセルに対して、印刷される最大 256 文字のテキストを提供します。

MISSTEXT={<*label*='*text*'> <STYLE=*style-override(s)*>}
印刷される最大 256 文字のテキストを提供し、欠損値を含むテーブルセルにスタイルの上書きを指定します。STYLE=オプションの引数とその使用方法については、TABLE ステートメントの STYLE= (1902 ページ)を参照してください。

操作 次元式で指定されるスタイル要素は、指定セルに対し MISSTEXT=オプションで指定されるスタイル要素に優先します。

例 “欠損値を含むセルに対するテキストの提供” (1940 ページ)

“例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

NOCELLMERGE
データセルがテーブルのその他のデータセルと結合されないように指定します。

注: NOCELLMERGE オプションは、ODS 形式の出力先と機能します。これらには、ODS MARKUP グループ、ODS RTF および ODS PRINTER グループの出力先が含まれます。

制限事項 NOCELLMERGE は、従来の monospace 出力とは機能しません。

操作 ROW=FLOAT または INDENT=0 を指定すると、PROC TABULATE は単一の結合されていないデータ行を生成します。NOCELLMERGE オプションは無視されます。結合が必要な行がないためです。

NOCELLMERGE オプションが有効な場合、空のデータセルのスタイルはデータセルのデフォルトのスタイルとなります。空のデータセルのスタイルは、フォーマットされたデータセルのスタイルと異なることがあります。

例 “例 16: NOCELLMERGE オプションの使用” (2008 ページ)

NOCONTINUED
複数ページにわたってテーブルの下部に表示される続行メッセージ `continued` を非表示にします。テキストは、AFTERCAPTION スタイル要素で表示されます。

注: HTML ブラウザはページを分割しないため、NOCONTINUED は HTML 出力先に影響しません。

PRINTMISS
変数のヘッダーが印刷されるたびに分類変数に対して発生するすべての値を印刷します。これらのヘッダーが作成するセルの一部に対しデータがない場合でも同様です。その結果、PRINTMISS は、テーブルのすべての論理ページに対して同じ行および列ヘッダーを単一の BY グループ内に作成します。

デフォルト RINTMISS オプションを省略すると、PROC TABULATE は、PROC TABULATE ステートメントで CLASSDATA=オプションを使用しない限り、データがない行または列を非表示にします。

制限事項 論理ページ全体に欠損値のみが含まれている場合、PRINTMISS オプションが指定されていてもそのページは印刷されません。

参照項目 “CLASSDATA=*SAS-data-set*” (1875 ページ)

例 “すべてのカテゴリに対するヘッダーの提供” (1939 ページ)

ROW=*spacing*
行クロスのすべてのタイトル要素が、ブランクの場合でも割り当てられたスペースかどうかを指定します。*spacing* の可能な値は、次のとおりです。

CONSTANT
タイトルがブランクの場合でも、スペースをすべての行タイトルに割り当てます。(N=' 'など)。

別名 CONST

FLOAT
行タイトルスペースをクロスのブランク以外の行タイトル間に均等に分配します。

デフォルト CONSTANT

例 “例 7: 行ヘッダーの削除” (1962 ページ)

RTSPACE=*number*
行ヘッダーの外枠文字の印刷に使用されるスペースを含む、行次元のすべてのヘッダーに割り当てる印刷位置の数を指定します。PROC TABULATE は、このスペースを行ヘッダーのすべてのレベル間に均等に分配します。

別名 RTS=

デフォルト SAS システムオプション LINESIZE=の値の 4 分の 1

制限事項 RTSPACE=オプションは、従来の SAS monospace 出力先にのみ影響します。

操作 デフォルトでは、PROC TABULATE はブランクの行タイトルにスペースを割り当てます。TABLE ステートメントで ROW=FLOAT を使用して、ブランク以外のタイトル間でのみスペースを分配します。

参照項目 行タイトルのスペースを制御する例については、*PROC TABULATE by Example, Second Edition* を参照してください。

例 “例 1: 基本的な 2 次元表の作成” (1944 ページ)

STYLE=*style-override*
テーブルセル以外のテーブル部分に使用する 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、欠損値の背景色が赤になり、ボックスの背景色がオレンジ色になるように指定します。

```
table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
        misstext=[label='Missing']style=[background=red]
        box=[label='Region by Division and Type'  style=[backgroundcolor=orange]];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override これは次の形式を取ります。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1* <*style-attribute-name-2=style-attribute-value-2 …*>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で大かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用してスタイル要素を割り当てる” (1930 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

別名 S=

制限事項 STYLE=オプションは、LISTING 以外の出力先にのみ影響します。

ヒント TABLE ステートメントでオプションとして行われるスタイル要素指定を無効にする場合、STYLE=を TABLE ステートメントの次元式で指定します。

角かっこ([と])の代わりに中かっこ({と})を使用できます。

参照項目 PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

例 “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

STYLE_PRECEDENCE=PAGE | ROW | COLUMN | COL
ページ次元(PAGE)、行次元(ROW)または列次元(COLUMN または COL)に対して指定されるスタイルがテーブルセルのコンテンツに適用されるかどうかを指定します。

デフォルト COLUMN

例 “例 15: スタイル優先” (2004 ページ)

詳細

次元式について

次元式は、次元を形成する変数、変数値、統計量の組み合わせを指定することにより、次元(テーブルの列、行、ページ)のコンテンツと表示を定義します。TABLE ステートメントは、カンマで区切られた 1 つから 3 つの次元式から構成されています。オプションは、次元式の後に続きます。

3 つの次元がすべて指定される場合、最左の次元式がページを、中央の次元式が行を、最右の次元式が列をそれぞれ定義します。2 つの次元が指定される場合、左の次元式が行を、右の次元式が列をそれぞれ定義します。1 つの次元が指定される場合、次元式は列を定義します。

次元式は、1 つ以上の要素と演算子から構成されています。

次元式で使用できる要素

分析変数
(VAR ステートメント (1907 ページ)を参照)。

分類変数
(CLASS ステートメント (1885 ページ)を参照)。

共通分類変数 ALL
同じ挿入グループまたは次元(変数 ALL が挿入グループに含まれていない場合)の分類変数のすべてのカテゴリを要約します。

注: 入力データセットに ALL という変数が含まれている場合、共通分類変数の名前を引用符で囲みます。

例 “例 6: 共通分類変数 ALL を使用した情報の要約” (1960 ページ)

“例 9: 複数ページテーブルの作成” (1967 ページ)

“例 13: 分母定義を使用し、基本的な度数カウントとパーセントを表示する” (1984 ページ)

統計量のキーワード

利用可能な統計量のリストについては、“PROC TABULATE で使用できる統計量” (1911 ページ)を参照してください。アスタリスク(*)演算子を使用して、統計量キーワードと変数を関連付けます。N 統計量(非欠損値の数)は、変数と関連付けずに次元式で指定できます。

デフォルト 分析変数の場合、デフォルトの統計量は SUM です。その他の場合、デフォルトの統計量は N です。

制限事項 N 以外の統計量キーワードは、分析変数と関連付ける必要があります。

操作 キーワード要素が変数としてではなく統計量キーワードとして扱われるように、統計量キーワードは一重引用符または二重引用符で囲む必要があります。デフォルトでは、これらのキーワードは変数として扱われます。

例
```
n
Region*n
Sales*max
```

例 “例 10: 複数回答式の調査データに基づくレポート作成” (1970 ページ)

“例 13: 分母定義を使用し、基本的な度数カウントとパーセントを表示する” (1984 ページ)

出力形式修飾子

セルの値をフォーマットする方法を定義します。アスタリスク(*)演算子を使用して、出力形式修飾子とフォーマットするセルを生成する要素(分析変数または統計量)を関連付けます。出力形式修飾子の形式は、次のとおりです。

```
f=format
```

ヒント 出力形式修飾子は、CLASS 変数に影響しません。

参照項目 テーブルにおける出力形式指定の詳細については、“テーブルの値のフォーマッティング” (1914 ページ)を参照してください。

例
```
Sales*f=dollar8.2
```

例 “例 6: 共通分類変数 ALL を使用した情報の要約” (1960 ページ)

ラベル

変数と統計量の名前を一時的に置き換えます。ラベルは、ラベルの直前の変数または統計量にのみ影響します。ラベルの形式は、次のとおりです。

```
statistic-keyword-or-variable-name='label-text'
```

ヒント PROC TABULATE はブランクの列ヘッダーのスペースをテーブルから削除しますが、デフォルトではすべての行ヘッダーがブランクでない限り、ブランクの行ヘッダーのスペースは削除しません。ブランクの行ヘッダーのスペースを削除するには、TABLE ステートメントで ROW=FLOAT を使用します。

例
```
Region='Geographical Region'
Sales*max='Largest Sale'
```

例 “例 5: 行と列のヘッダーのカスタマイズ” (1958 ページ)

“例 7: 行ヘッダーの削除” (1962 ページ)

style の指定
ページ次元テキスト、ヘッダー、データセルにスタイル要素とスタイル属性を指定します。詳細については、“次元式でのスタイル属性およびスタイル要素の指定” (1907 ページ)を参照してください。

次元式で使用できる演算子

アスタリスク*
分類変数の値の組み合わせからカテゴリを作成し、次元に適切なヘッダーを構成します。要素のうちいずれかが分析変数である場合、分析変数の統計量が分類変数によって作成されるカテゴリに対し計算されます。このプロセスをクロスといいます。

例
```
Region*Division
Quarter*Sales*f=dollar8.2
```

例 “例 1: 基本的な 2 次元表の作成” (1944 ページ)

(ブランク)
各要素の出力が、先行する要素の出力の直後に置かれます。このプロセスを連結といいます。

例
```
n Region*Sales ALL
```

例 “例 6: 共通分類変数 ALL を使用した情報の要約” (1960 ページ)

かっこ ()
要素をグループ化し、演算子とグループの連結要素をそれぞれ関連付けます。

例
```
Division*(Sales*max Sales*min)
(Region ALL)*Sales
```

例 “例 6: 共通分類変数 ALL を使用した情報の要約” (1960 ページ)

山かっこ<>
分母定義を指定します。これによってパーセントの計算での分母の値が決定されます。分母定義を構成する方法については、“パーセントの計算 ” (1914 ページ)を参照してください。

例 “例 10: 複数回答式の調査データに基づくレポート作成” (1970 ページ)

“例 13: 分母定義を使用し、基本的な度数カウントとパーセントを表示する” (1984 ページ)

次元式でのスタイル属性およびスタイル要素の指定

1 つ以上のスタイル要素またはスタイル属性を次元式に指定して、LISTING 以外の出力先の表示を制御できます。次の領域の表示を変更できます。

- 分析変数名のヘッダー
- 分類変数名ヘッダー
- 分類変数の水準値ヘッダー
- データセル
- キーワードのヘッダー
- ページ次元テキスト

スタイル属性またはスタイル要素を次元式に指定すると、PROC TABULATE、CLASS、CLASSLEV、KEYWORD、TABLE、または VAR ステートメントなど、別のステートメントに指定したスタイル属性またはスタイル要素を無効にする場合に便利です。

スタイル要素とスタイル属性を次元式に指定する構文は次のとおりです。

[STYLE<(CLASSLEV)>=<*style-element-name* | PARENT>

[*style-attribute-name-1=style-attribute-value-1< style-attribute-name-2=style-attribute-value-2 ...*>]]

これらは、次元式にあるスタイル属性の例の一部です。

- ```
  dept={label='Department'
        style=[color=red]}, N
  ```
- **dept*[style=*MyDataStyle*], N**
- **dept*[format=12.2 style=*MyDataStyle*], N**

注: 次元式で使用する場合、STYLE=オプションを角括弧([and])または括弧({ and }) で囲む必要があります。

(CLASSLEV)
分類変数水準値ヘッダーにスタイル要素を割り当てます。たとえば、次の TABLE ステートメントでは、分類変数 DEPT の水準値ヘッダーの前景色が黄色になるように指定します。

```
table dept=[style(classlev)=
            [color=yellow]]*sales;
```

注: CLASSLEV オプションは、次元式でのみ使用されます。

次元式でのスタイル要素の指定方法の例については、“例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)を参照してください。PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

VAR ステートメント

分析変数として使用する数値変数を識別します。

別名: VARIABLES

ヒント: 複数の VAR ステートメントを使用できます。

例: “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)
```

構文

VAR *analysis-variable(s)* </ *option(s)*>;

必須引数

analysis-variable(s);
テーブルの分析変数を識別します。分析変数は、PROC TABULATE が統計量を計算する数値変数です。分析変数の値は、連続または不連続となります。

オブザベーションに分析変数に対する欠損値が含まれている場合、PROC TABULATE は N (非欠損変数値を含むオブザベーション数)と NMISS (欠損変数値を含むオブザベーション数)以外のすべての統計量の計算からの値を省略します。たとえば、欠損値によって SUM は増加しません。欠損値は MEAN などの統計量の計算時にカウントされません。

操作 変数名と統計量名が同じ場合、統計量名を一重引用符か二重引用符で囲みます。

オプション引数

STYLE=*style-override(s)*
分類変数名ヘッダーに 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、分析変数名ヘッダーの背景色が黄褐色になるように指定します。

```
var actual  / style=[background=tan];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override これは次の形式を取ります。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1* <*style-attribute-name-2=style-attribute-value-2 …*>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素

- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で大かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用してスタイル要素を割り当てる” (1930 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目
PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素の表については、“テーブル領域のスタイル要素とスタイル属性” (1927 ページ)を参照してください。

別名 S=

制限事項 STYLE=オプションは、LISTING 以外の出力先にのみ影響します。

ヒント VAR ステートメントで指定されるスタイル要素を無効にする場合、スタイル要素を関連する TABLE ステートメント次元式で指定できます。

VAR ステートメントでの STYLE=の使用は、PROC TABULATE ステートメントでの使用とはやや異なります。VAR ステートメントでは、継承は行と列では異なります。行の場合、親ヘッダーは現在のヘッダーの左側に置かれます。列の場合、親ヘッダーは現在のヘッダーの上に置かれます。

参照項目 PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

例 “例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)

WEIGHT=*weight-variable*
その値が VAR ステートメントで指定される変数の値に重みを付ける数値変数を指定します。重み変数が整数である必要がないのは、重み変数の値が次である場合です。

重み値	PROC TABULATE 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 未満	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

制限事項 重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。

注 バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

ヒント WEIGHT=オプションを使用する場合、適切な VARDEF=オプションの値を考慮します。“VARDEF=*divisor* ” (1883 ページ)の説明を参照してください。

複数の VAR ステートメントで WEIGHT オプションを使用して、分析変数に対し異なる重みを指定します。

WEIGHT ステートメント

統計量計算の分析変数に対し重みを指定します。

参照項目: 重み付き統計量の計算と、WEIGHT ステートメントを使用する例については、“重み付き統計量の計算” (75 ページ)を参照してください。

構文

WEIGHT *variable*;

必須引数

variable
分析変数の値に重みをつける数値変数を指定します。変数の値は、整数である必要はありません。PROC TABULATE は、次のテーブルに従って重み値に応答します。

重み値	PROC TABULATE 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 未満	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

注: バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

制限事項

重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。

重み変数がアクティブの場合、PROC TABULATE は MODE を計算しません。代わりに、MODE の計算が必要で、重み変数がアクティブの場合は、PROC UNIVARIATE を使用しようとします。

操作

VAR ステートメントで WEIGHT=オプションを使用して重み変数を指定すると、PROC TABULATE はこの変数を代わりに使用して、これらの VAR ステートメント変数に重みを付けます。

ヒント

WEIGHT ステートメントを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。“VARDEF=*divisor* ” (1883 ページ)と重み付き統計量の計算の説明については、このドキュメントの“キーワードと式” (2078 ページ)セクションを参照してください。

PROC TABULATE で使用できる統計量

次のキーワードを使用して、TABLE ステートメントで統計量を要求するか、KEYWORD ステートメントまたは KEYLABEL ステートメントで統計量キーワードを指定します。

注: 変数名(分類または分析)と統計量名が同じ場合、統計量名を一重引用符で囲みます。(例:**'MAX'**)

記述統計量キーワード	
COLPCTN	PCTSUM
COLPCTSUM	RANGE
CSS	REPPCTN
CV	REPPCTSUM
KURTOSIS | KURT	ROWPCTN
LCLM	ROWPCTSUM
MAX	SKEWNESS | SKEW
MEAN	STDDEV | STD
MIN	STDERR
MODE	SUM
N	SUMWGT
NMISS	UCLM
PAGEPCTN	USS
PAGEPCTSUM	VAR
PCTN	
四分位範囲統計量キーワード	
MEDIAN | P50	Q3 | P75
P1	P70
P5	P80
P10	P90
P20	P95
P30	P99
P40	
P60	
Q1|P25	QRANGE

仮説検定キーワード	
PROBT | PRT	T

これらの統計量、その計算に使用される式、データ要件については、このドキュメントの“キーワードと式” (2078 ページ)セクションを参照してください。

平均値の標準誤差(STDERR)またはスチューデントの *t* 検定を計算するには、VARDEF=オプションのデフォルト値、DF を使用する必要があります。VARDEF=オプションは、PROC TABULATE ステートメントで指定されます。

重み付き分位点を計算するには、PROC TABULATE ステートメントで QMETHOD=OS を使用する必要があります。

LCLM と UCLM を使用して、平均の両側信頼限界を計算します。LCLM または UCLM のみを使用して、片側信頼限界を計算します。PROC TABULATE ステートメントで ALPHA=オプションを使用して、信頼水準を指定します。

分類変数のフォーマッティング

FORMAT ステートメントを使用して、出力形式を PROC TABULATE ステップの間隔に対する分類変数に割り当てます。出力形式を分類変数に割り当てると、PROC TABULATE はフォーマットされた値を使用してカテゴリを作成し、フォーマットされた値がヘッダーで使用されます。分類変数に出力形式を指定せず、変数にその他の出力形式が割り当てられていない場合、GROUPINTERNAL オプションが指定されていない限り、デフォルトの出力形式、BEST12.が使用されます。

ユーザー定義の出力形式は、値を少数のカテゴリにグループ化する場合に特に便利です。たとえば、値が 1 から 99 までの分類変数 Age がある場合、テーブルに扱いやすい数のカテゴリが含まれるように、年齢をグループ化するユーザー定義の出力形式を作成できます。次の PROC FORMAT ステップでは、年齢の可能な値をすべて 6 つの値グループに要約する出力形式を作成します。

```
proc format;
   value agefmt  0-29='Under 30'
                 30-39='30-39'
                 40-49='40-49'
                 50-59='50-59'
                 60-69='60-69'
                 other='70 or over';
run;
```

ユーザー定義の出力形式作成の詳細については、27 章, “FORMAT プロシジャ,” (834 ページ)を参照してください。

デフォルトでは、PROC TABULATE は度数カウントがゼロでなく、値が欠損していない出力形式のみテーブルに含めます。すべての分類変数の欠損値を出力に含めるには、PROC TABULATE ステートメントで MISSING オプションを使用します。選択した分類変数の欠損値を含めるには、CLASS ステートメントで MISSING オプションを使用します。度数カウントがゼロの出力形式を含めるには、CLASS ステートメントで PRELOADFMT オプション、TABLE ステートメントで PRINTMISS オプション、または PROC TABULATE ステートメントで CLASSDATA=オプションを使用します。

テーブルの値のフォーマッティング

テーブルセルのデータ用出力形式には、2 つの目的があります。PROC TABULATE による値の表示方法の決定と列幅の決定です。テーブルセルの値に対するデフォルトの出力形式は、12.2 です。次を実行して、テーブルセルの値を印刷するための出力形式を変更できます。

- PROC TABULATE ステートメントの FORMAT=オプションでデフォルトの出力形式を変更する
- F=出力形式修飾子を使用して TABLE ステートメントで要素をクロスする

注: FORMAT ステートメントまたは ATTRIB ステートメントを使用して、テーブルセルの値を印刷するための出力形式を変更することはできません。これらのステートメントを使用すると、それらに含まれる分析変数が無視されます。

PROC TABULATE は、出力形式に対する次のデフォルトの優先順位から特定のセルに使用する出力形式を決定します。

1. その他の出力形式が指定されていない場合、PROC TABULATE はデフォルトの出力形式(12.2)を使用します。
2. PROC TABULATE ステートメントの FORMAT=オプションは、デフォルトの出力形式を変更します。出力形式修飾子がセルに影響しない場合、PROC TABULATE はこの出力形式をセルの値に使用します。
3. ページ次元の出力形式修飾子は、行次元または列次元のセルに対し別の出力形式修飾子を指定しない限り、論理ページのすべてのテーブルセルの値に適用されます。
4. 行次元の出力形式修飾子は、列次元のセルに別の出力形式修飾子を指定しない限り、行のすべてのテーブルセルの値に適用されます。
5. 列次元の出力形式修飾子は、列のすべてのテーブルセルの値に適用されます。

この優先順位は、TABLE ステートメントで FORMAT_PRECEDENCE=オプションを使用して、変更できます。詳細については、“TABLE ステートメント” (1897 ページ)を参照してください。たとえば、FORMAT_PRECEDENCE=ROW を指定し、行次元で出力形式修飾子を指定すると、その出力形式はテーブルセルに指定されているその他すべての出力形式に優先します。

パーセントの計算

単一のテーブルセルの値のパーセントを計算する

次の統計量は、セルグループの値の合計に対する単一のテーブルセルの値のパーセントを印刷します。分母定義は不要です。ただし、分析変数がパーセント合計統計量の分母定義として使用できます。

- REPPCTN 統計量と REPPCTSUM 統計量 — レポートの値の合計に対する単一のテーブルセルの値のパーセントを印刷します。
- COLPCTN 統計量と COLPCTSUM 統計量 — 列の値の合計に対する単一のテーブルセルの値のパーセントを印刷します。

- ROWPCTN 統計量と ROWPCTSUM 統計量 — 行の値の合計に対する単一のテーブルセルの値のパーセントを印刷します。
- PAGEPCTN 統計量と PAGEPCTSUM 統計量 — ページの値の合計に対する単一のテーブルセルの値のパーセントを印刷します。

これらの統計量は、通常使用されるパーセントを計算します。例については、“例 12: さまざまなパーセント表示の統計量の計算” (1981 ページ)を参照してください。

PCTN と PCTSUM の使用

PCTN 統計量と PCTSUM 統計量は、これらの同じパーセントの計算に使用できます。これらを使用して、分母を手動で定義できます。PCTN 統計量と PCTSUM 統計量は、別のテーブルセルの値(パーセントの計算の分母で使用される)、またはセルグループの値の合計に対する単一テーブルセルの値のパーセントを印刷します。デフォルトでは、PROC TABULATE はすべての N セル(PCTN 統計量の場合)またはすべての SUM セル(PCTSUM 統計量の場合)の値を要約し、要約した値を分母に使用します。PROC TABULATE が分母に使用する値を分母定義によって制御できます。

分母定義は、PCTN 統計量または PCTSUM 統計量の隣に山かっこ(<と>)で囲みます。分母定義は、分母に対して合計するカテゴリを指定します。

このセクションでは、単純なテーブルで分母定義を指定する方法について説明します。“例 13: 分母定義を使用し、基本的な度数カウントとパーセントを表示する” (1984 ページ)では、複数のサブテーブルで構成されるテーブルで分母定義を指定する方法について説明します。分母定義の例については、*PROC TABULATE by Example, Second Edition* を参照してください。

PCTN 統計量に分母を指定する

次の PROC TABULATE ステップでは、データセット“ENERGY” (2152 ページ)を使用して N 統計量と PCTN の 3 つの異なるバージョンを計算します。

```
proc tabulate data=energy;
   class division type;
   table division*
           (n='Number of customers'
            pctn<type>='% of row' 1
            pctn<division>='% of column' 2
            pctn='% of all customers'), 3
          type/rts=50;
   title 'Number of Users in Each Division';
run;
```

TABLE ステートメントは、Division の値ごとに 1 行、Type の値ごとに 1 列を作成します。各行内で、TABLE ステートメントは N と、PCTN の 3 つの異なる計算の合計 4 つの統計量をネストします。(次の図を参照)。PCTN の発生ごとに異なる分母定義が使用されます。

図 64.4 度数カウントがハイライト表示された PCTN 統計量の 3 つの異なる使用

Number of Users in Each Division

		Type	
		1	2
Division			
1	Number of customers	6.00	6.00
	% of row 1	50.00	50.00
	% of column 2	27.27	27.27
	% of all customers 3	13.64	13.64
2	Number of customers	3.00	3.00
	% of row	50.00	50.00
	% of column	13.64	13.64
	% of all customers	6.82	6.82
3	Number of customers	8.00	8.00
	% of row	50.00	50.00
	% of column	36.36	36.36
	% of all customers	18.18	18.18
4	Number of customers	5.00	5.00
	% of row	50.00	50.00
	% of column	22.73	22.73
	% of all customers	11.36	11.36

1 **<type>**は、Division の同じ値内の Type のすべての発生に対する度数カウントを合計します。したがって、Division=1 の場合、分母は 6 + 6、または 12 となります。

2 **<division>**は、Type の同じ値内の Division のすべての発生に対し度数カウントを合計します。したがって、Type=1 の場合、分母は 6 + 3 + 8 + 5、または 22 となります。

3 PCTN の 3 番目の使用には、分母定義はありません。分母定義の省略は、すべての分類変数を分母定義に含めることと同じです。したがって、すべてのセルに対し、分母は 6 + 3 + 8 + 5 + 6 + 3 + 8 + 5、または 44 となります。

PCTSUM 統計量に分母を指定する

次の PROC TABULATE ステップでは、Type と Division の各組み合わせに対する支出額を合計し、PCTSUM の 3 つの異なるバージョンを計算します。

```
proc tabulate data=energy format=8.2;
   class division type;
   var expenditures;
   table division*
           (sum='Expenditures'*f=dollar10.2
            pctsum<type>='% of row' 1
            pctsum<division>='% of column' 2
            pctsum='% of all customers'), 3
         type*expenditures/rts=40;
   title 'Expenditures in Each Division';
run;
```

TABLE ステートメントは、Division の値ごとに 1 行、Type の値ごとに 1 列を作成します。Type が Expenditures とクロスするため、各セルの値が、セルに影響するすべてのオブザベーションに対する Expenditures の値の合計となります。各行内で、TABLE ステートメントは SUM と、PCTSUM の 3 つの異なる計算の合計 4 つの統計量をネストします。(次の図を参照)。PCTSUM の発生ごとに異なる分母定義が使用されます。

図 64.5 *合計がハイライト表示された PCTSUM 統計量の 3 つの異なる使用*

Expenditures in Each Division

		Type	
		1	2
		Expenditu-res	Expenditu-res
Division			
1	Expenditures	$7,477.00	$5,129.00
	% of row 1	59.31	40.69
	% of column 2	16.15	13.66
	% of all customers 3	8.92	6.12
2	Expenditures	$19,379.00	$15,078.00
	% of row	56.24	43.76
	% of column	41.86	40.15
	% of all customers	23.11	17.98
3	Expenditures	$5,476.00	$4,729.00
	% of row	53.66	46.34
	% of column	11.83	12.59
	% of all customers	6.53	5.64
4	Expenditures	$13,959.00	$12,619.00
	% of row	52.52	47.48
	% of column	30.15	33.60
	% of all customers	16.65	15.05

1 **<type>**は、Division の同じ値内の Type のすべての発生に対し Expenditures の値を合計します。したがって、Division=1 の場合、分母は$7,477 + $5,129 となります。

2 **<division>**は、Type の同じ値内の Division のすべての発生に対し度数カウントを合計します。したがって、Type=1 の場合、分母は$7,477 + $19,379 + $5,476 + $13,959 となります。

3 PCTN の 3 番目の使用には、分母定義はありません。分母定義の省略は、すべての分類変数を分母定義に含めることと同じです。したがって、すべてのセルに対し、分母は$7,477 + $19,379 + $5,476 + $13,959 + $5,129 + $15,078 + $4,729 + $12,619 となります。

基本的なレポート記述プロシジャを使用した ODS スタイルの使用

概要

ODS をサポートする Base SAS プロシジャの多くは、1 つ以上のテーブルテンプレートを使用して出力オブジェクトを生成します。これらのテーブルテンプレートには、テーブル要素(列、ヘッダー、フッター)に対するテンプレートが含まれています。各テーブル要素で、出力のさまざまな部分に 1 つ以上のスタイル要素を使用することを指定できます。これらのスタイル要素はプロシジャの構文内で指定することはできませんが、使用する ODS 出力先に合わせてカスタマイズされたスタイルを使用できます。テーブルとスタイルのカスタマイズの詳細については、“TEMPLATE Procedure: Creating a Style Template” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。

Base SAS レポートプロシジャの PROC PRINT、PROC REPORT および PROC TABULATE を使用してデータをすばやく分析し、読みやすいテーブルに整理することができます。これらのプロシジャステートメントで STYLE=オプションを使用してレポートの表示を変更できます。STYLE=オプションを使用すると、すべての出力のデフォルトスタイルを変更せずに、出力の各セションに変更を加えることができます。プロシジャ内の特定のステートメントで STYLE=オプションを指定して、プロシジャ出力の特定のセクションをカスタマイズすることができます。

次のプログラムでは、STYLE=オプションを使用して以下の PROC REPORT 出力の背景色を作成します。

```
title "Height and Weight by Gender and Age";
proc report nowd data=sashelp.class
   style(header)=[background=white];
   col age (('Gender' sex),(weight height));
   define age / style(header)=[background=lightgreen];
   define sex / across style(header)=[background=yellow] ' ';
   define weight / style(header)=[background=orange];
   define height / style(header)=[background=tan];
run;
```

図 64.6 *PROC REPORT 出力の拡張*

Height and Weight by Gender and Age

	Gender			
	F		M	
Age	Weight	Height	Weight	Height
253	811	545.3	1089.5	639.1

次のプログラムでは、STYLE=オプションを使用して以下の PROC TABULATE 出力の色を作成します。

```
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

proc tabulate data=prdsale;
   class region division prodtype / style=[background=lightgreen];
   classlev region division prodtype  / style=[background=yellow];
   var actual  / style=[background=tan];
   keyword all sum / style=[background=linen color=blue];
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         box=[label='Region by Division and Type' style=[backgroundcolor=orange]];

title 'Actual Product Sales';
title2 '(millions of dollars)';
run;
```

図 64.7 PROC TABULATE 出力の拡張

Actual Product Sales
(millions of dollars)

Region by Division and Type		Product type		Total
		FURNITURE	OFFICE	
		Actual Sales	Actual Sales	Actual Sales
		Sum	Sum	Sum
Region	Division			
EAST	CONSUMER	$72,570	$108,686	$181,256
	EDUCATION	$73,901	$115,104	$189,005
	Total	$146,471	$223,790	$370,261
WEST	Division			
	CONSUMER	$76,209	$105,020	$181,229
	EDUCATION	$67,945	$110,902	$178,847
	Total	$144,154	$215,922	$360,076
Total	Division			
	CONSUMER	$148,779	$213,706	$362,485
	EDUCATION	$141,846	$226,006	$367,852
	Total	$290,625	$439,712	$730,337

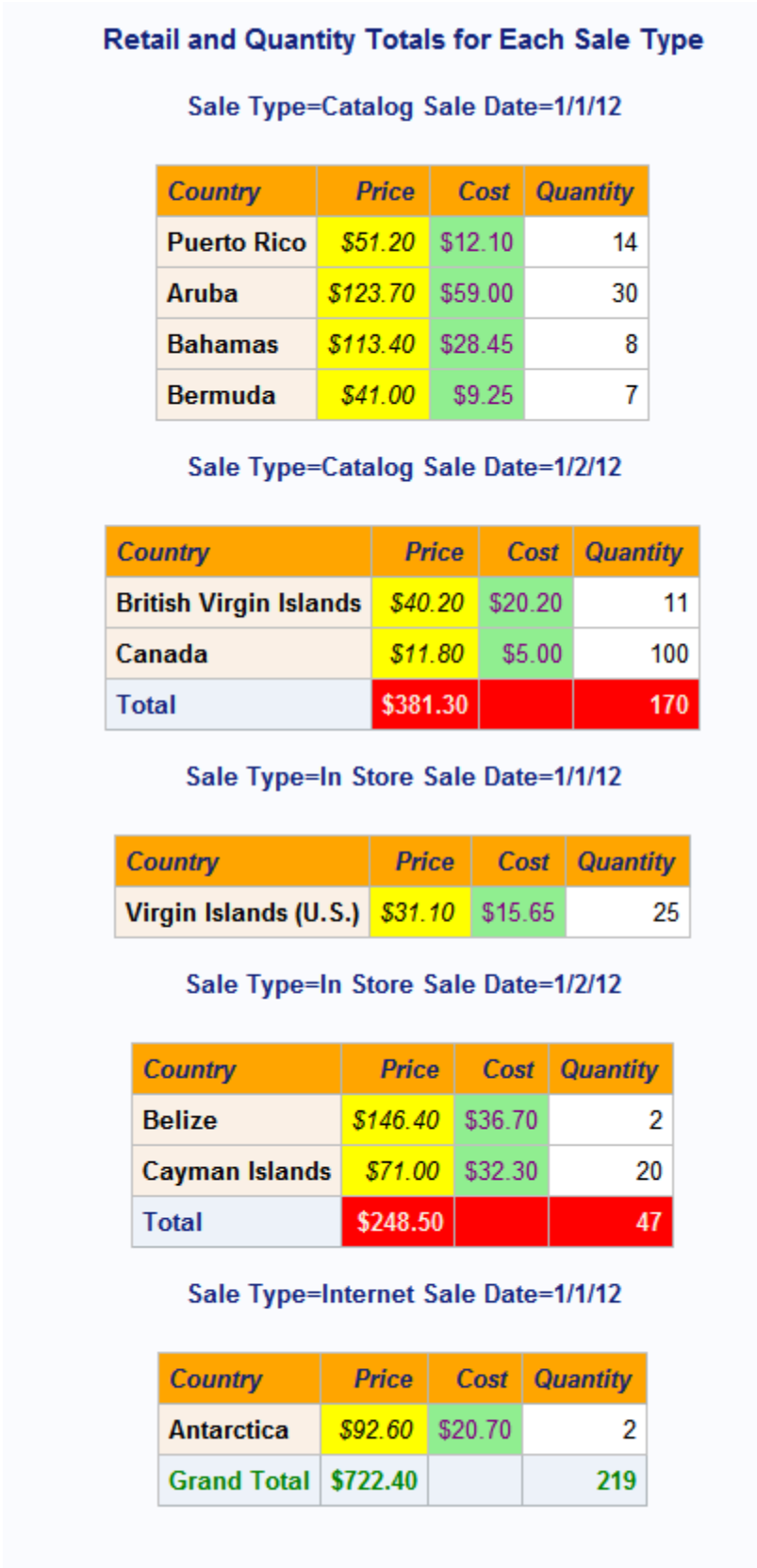
次のプログラムでは、STYLE=オプションを使用して以下の PROC PRINT 出力の色を作成します。

```
proc print data=exprev noobs sumlabel='Total' GRANDTOTAL_LABEL="Grand Total"
```

```
      style(table)=[frame=box rules=groups]
      style(bysumline)=[background=red foreground=linen]
      style(grandtotal)=[foreground=green]
      style(header)=[font_style=italic background=orange];
   by sale_type order_date;
   sum price quantity;
   sumby sale_type;
   label sale_type='Sale Type' order_date='Sale Date';
format price dollar10.2 cost dollar10.2;
   var Country / style(data)=[font_face=arial font_weight=bold background=linen];
   var Price / style(data)=[font_style=italic background=yellow];
   var Cost / style(data)=[foreground=hgt. background=lightgreen];
title 'Retail and Quantity Totals for Each Sale Type';
run;
```

完全な入力データセットについては、“EXPREV” (2154 ページ)を参照してください。

図 64.8 PROC PRINT 出力の拡張



Retail and Quantity Totals for Each Sale Type

Sale Type=Catalog Sale Date=1/1/12

Country	Price	Cost	Quantity
Puerto Rico	$51.20	$12.10	14
Aruba	$123.70	$59.00	30
Bahamas	$113.40	$28.45	8
Bermuda	$41.00	$9.25	7

Sale Type=Catalog Sale Date=1/2/12

Country	Price	Cost	Quantity
British Virgin Islands	$40.20	$20.20	11
Canada	$11.80	$5.00	100
Total	$381.30		170

Sale Type=In Store Sale Date=1/1/12

Country	Price	Cost	Quantity
Virgin Islands (U.S.)	$31.10	$15.65	25

Sale Type=In Store Sale Date=1/2/12

Country	Price	Cost	Quantity
Belize	$146.40	$36.70	2
Cayman Islands	$71.00	$32.30	20
Total	$248.50		47

Sale Type=Internet Sale Date=1/1/12

Country	Price	Cost	Quantity
Antarctica	$92.60	$20.70	2
Grand Total	$722.40		219

スタイル、スタイル要素およびスタイル属性

SAS 出力の表示はスタイルテンプレート(スタイル)で制御されます。スタイルは、SAS 出力の視覚側面(色、フォント、罫線、マーカーなど)を定義する ODS テンプレートの一種です。スタイルにより、そのスタイルを使用するドキュメントの全体的な表示が決まります。スタイルテンプレートは、スタイル要素とスタイル属性で構成されています。

- スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性の名前付きコレクションです。ODS 出力の各領域には、その領域に関連付けられているスタイル要素名があります。スタイル要素名で、スタイル属性が適用される場所を指定します。たとえば、スタイル要素に、列ヘッダーの表示またはセル内のデータの表示に関する指示を含めることができます。スタイル要素で、スタイルを使用する出力のデフォルトの色およびフォントも指定することもできます。
- スタイル属性は、色、フォントのプロパティ、罫線の特性などの視覚プロパティで、予約された名前と値を使用して ODS で定義されます。スタイル属性は、スタイルテンプレート内のスタイル要素によりまとめて参照されます。各*スタイル属性*は、表示の 1 つの側面の値を指定します。たとえば、BACKGROUNDCOLOR=属性では、HTML テーブルまたは印刷出力の色付きテーブルの背景色を指定します。FONTSTYLE=属性では、Roman フォントとイタリックフォントのどちらを使用するか指定します。

注: スタイルはデータの表示を制御するので、LISTING、DOCUMENT または OUTPUT の出力先に移動する出力オブジェクトには影響しません。

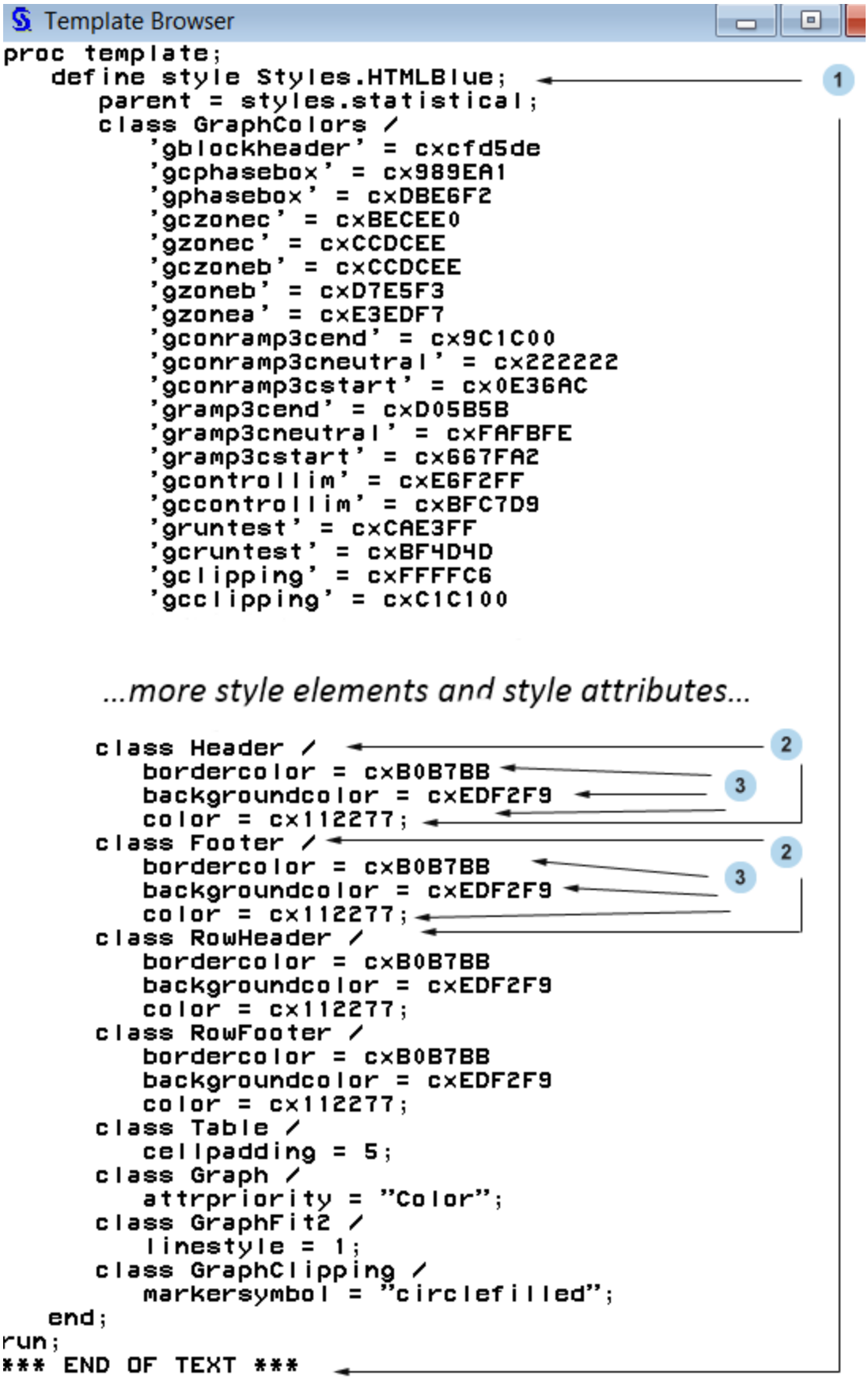
使用可能なスタイルは、SASHELP.TMPLMST アイテムストアに含まれています。SAS Enterprise Guide では、スタイルシートのリストがスタイルウィザードで表示されます。バッチモードまたは SAS Studio では、次のコードをサブミットして使用可能なスタイルテンプレートのリストを表示できます。

```
proc template;
list styles / store=sashelp.tmplmst;
run;
```

ODS スタイルの表示に関する詳細は、“Viewing ODS Styles Supplied by SAS” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

デフォルトでは、HTML 出力は HTMLBlue スタイルテンプレートを使用します。スタイル、スタイル要素およびスタイル属性を熟知するために、これらの関係を確認します。以下の図は、スタイル、スタイル要素およびスタイル属性の関係を示しています。次の図は、スタイルの構造を示す例です。

図 64.9 HtmlBlue スタイルの図



以下のリストは、上記の図の番号の付いた項目に対応しています。

1 Styles.HtmlBlue は*スタイル*です。スタイルは、SAS 出力の表示側面(色、フォント、フォントのサイズなど)の表示方法を記述します。スタイルにより、そのスタイルを使用する ODS ドキュメントの全体的な表示が決まります。HTML 出力のデフォルトのスタイルは HtmlBlue です。各スタイルはスタイル要素で構成されています。各出力先には、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルが 1 つあります。

- HTML 出力のデフォルトのスタイルは HTMLBlue です。
- PRINTER の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

を使用して新しいスタイルを作成できます。“DEFINE STYLE Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*).新しいスタイルは既存のスタイルとは別に作成することも、既存のスタイルに基づいて作成することもできます。既存のスタイルから新しいスタイルを作成するには、“PARENT= Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*) を使用できます。ODS スタイルに関する詳細は、“Style Templates” (*SAS Output Delivery System: User's Guide*)を参照してください。

2 *スタイル要素*の例として、ヘッダーとフッターがあります。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。たとえば、スタイル要素に、列ヘッダーの表示またはテーブルセル内のデータの表示に関する指示を含めることができます。スタイル要素で、スタイルを使用する出力のデフォルトの色およびフォントも指定することもできます。スタイル要素はスタイル内に存在し、1 つ以上のスタイル属性で構成されています。スタイル要素はユーザーが定義することも、SAS から提供されるものを使用することもできます。ユーザー定義のスタイル要素は “STYLE Statement” (*SAS 9.4 Output Delivery System: Procedures Guide*)で作成できます。

注: HTML およびマークアップ言語とその継承に使用されるデフォルトのスタイル要素のリストについては、“Style Elements” (*SAS Output Delivery System: User's Guide*)を参照してください。

3 *スタイル属性*の例として、BORDERCOLOR=、BACKGROUNDCOLOR=、COLOR=があります。スタイル属性では、スタイル要素が適用される出力の領域の 1 つの側面の値を指定します。たとえば、COLOR=属性では、フォントの色に **cx112277** の値を指定します。SAS で提供されるスタイル属性のリストについては、“Style Attributes” (*SAS Output Delivery System: User's Guide*)を参照してください。

スタイル属性はスタイル参照を使用して参照できます。スタイル参照の詳細については、“style-reference” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

次の表に、PROC PRINT、PROC TABULATE および PROC REPORT.ステートメントの STYLE=オプションで設定できる一般的に使用されているスタイル属性を示します。これらの属性のうちほとんどは、セル以外のテーブルの各部分(テーブルの罫線、列と行の間の罫線など)に適用されます。すべての属性がすべての出力先で有効であるわけではありません。これらのスタイル属性、有効な値および適用可能な出力先に関する詳細は、“Style Attributes Tables ” (*SAS 9.4 Output Delivery System: Procedures Guide*)を参照してください。

表 64.4 PROC REPORT、PROC TABULATE および PROC PRINT のスタイル属性

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
ASIS=	X	X		X		X
BACKGOUNDCOLOR=	X	X	X	X	X	X
BACKGOUNDIMAGE=	X	X	X	X	X	X
BORDERBOTTOMCOLOR=	X	X		X		
BORDERBOTTOMSTYLE=	X	X	X	X		
BORDERBOTTOMWIDTH=	X	X	X	X		
BORDERLEFTCOLOR=	X	X		X		
BORDERLEFTSTYLE=	X	X	X	X		
BORDERLEFTWIDTH=	X	X	X	X		
BORDERCOLOR=	X	X		X	X	X
BORDERCOLORDARK=	X	X	X	X	X	X
BORDERCOLORLIGHT=	X	X	X	X	X	X
BORDERRIGHTCOLOR=	X	X		X		
BORDERRIGHTSTYLE=	X	X	X	X		
BORDERRIGHTWIDTH=	X	X	X	X		
BORDERTOPCOLOR=	X	X		X		
BORDERTOPSTYLE=	X	X	X	X		
BORDERTOPWIDTH=	X	X	X	X		
BORDERWIDTH=	X	X	X	X	X	X

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
CELLPADDING=	X		X		X	
CELLSPACING=	X		X		X	
CELLWIDTH=	X	X	X	X		X
CLASS=	X	X	X	X	X	X
COLOR=	X	X	X			
FLYOVER=	X	X		X		X
FONT=	X	X	X	X	X	X
FONTFAMILY=	X	X	X	X	X	X
FONTSIZE=	X	X	X	X	X	X
FONTSTYLE=	X	X	X	X	X	X
FONTWEIGHT=	X	X	X	X	X	X
FONTWIDTH=	X	X	X	X		X
FRAME=	X		X		X	
HEIGHT=	X	X		X	X	X
HREFTARGET=		X		X		X
HTMLSTYLE=	X	X	X	X	X	
NOBREAKSPACE=	X	X		X		X
OUTPUTWIDTH=	X	X	X	X	X	
POSTHTML=*	X	X	X	X	X	X
POSTIMAGE=	X	X	X	X	X	X
POSTTEXT=*	X	X	X	X	X	X
PREHTML=*	X	X	X	X	X	X
PREIMAGE=	X	X	X	X	X	X
PRETEXT=*	X	X	X	X	X	X

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域:CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT:テーブル以外のすべての場所
PROTECTSPECIALCHARS=		X		X	X	X
RULES=	X		X		X	
TAGATTR=	X	X	X	X	X	X
TEXTALIGN=	X	X	X	X	X	X
URL=		X		X		X
VERTICALALIGN=		X		X		X
WIDTH=	X	X	X	X	X	

* これらの属性をこの場所で使用する場合には、属性 PRETEXT=、POSTTEXT=、PREHTML=、POSTHTML=で指定されるテキストにのみ影響します。表に表示されるテキストの前景色またはフォントを変更するには、表ではなくセルに影響する場所で対応する属性を設定する必要があります。スタイル属性とその値の詳細な説明については、“Style Attributes” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

テーブル領域のスタイル要素とスタイル属性

次の表に、PROC TABULATE テーブルのさまざまな部分に対するデフォルトのスタイル要素およびスタイル属性を示します。この表には、最も一般的に使用される ODS 出力先である、HTML、PDF および RTF のデフォルトがリストされています。それぞれの出力先には、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルテンプレートが含まれます。

- HTML 出力のデフォルトスタイルは HTMLBlue です。
- PRINTER の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

ODS 出力先とそのデフォルトのスタイルに関する詳細なドキュメントは、“Style Templates” (*SAS Output Delivery System: Advanced Topics*)を参照してください。

表 64.5 テーブル部分のデフォルトのスタイル要素およびスタイル属性

部分	スタイル要素	HTML スタイル属性	PDF スタイル属性	RTF スタイル属性
列ヘッダーとボックス	ヘッダー	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT’" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb
ページ次元テキスト	Beforecaption	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxfafbfe	FONTFAMILY = "Arial, 'Albany AMT’" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000
行ヘッダー	Rowheader	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT’" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb
データセル	Data	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="’Albany AMT’, Albany" FONTSIZE = 8pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000 BORDERWIDTH = NaN COLOR = cx000000	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000

PROC TABULATE ステートメントで STYLE=オプションを使用する

PROC TABULATE スタイルの上書きは、テーブルを作成するステートメントに基づくテーブルに適用されます。STYLE=オプションを使用すると、変更する領域を制御するステートメント内に STYLE=オプションを指定することにより、出力のその領域の表示を制御するスタイル要素またはスタイル属性を変更できます。次の表に、変更可能なテーブルの領域と、それらに対応するステートメントを示します。

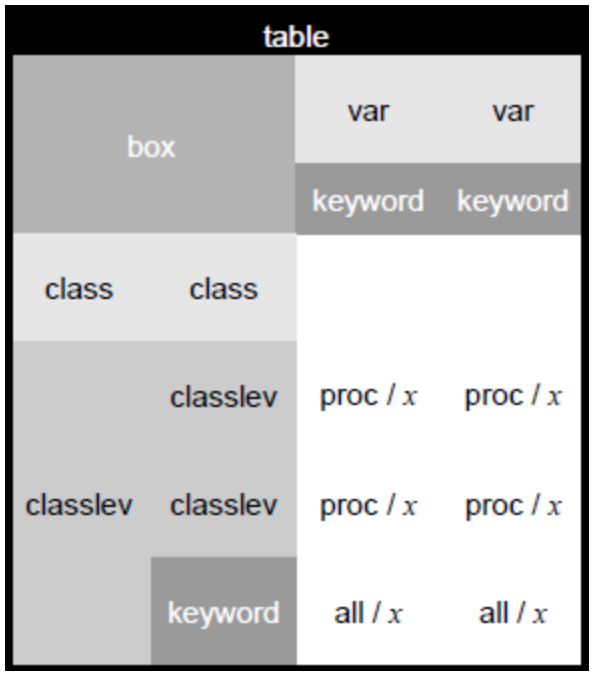
TABLE ステートメントでの指定が、PROC TABULATE、CLASS、CLASSLEV、VAR、および KEYWORD ステートメントでの同じ指定に優先します。これにより、複数の TABLE ステートメントで異なるスタイル動作を行えます。ただし、PROC TABULATE ステートメントで指定し、TABLE ステートメントで無効にしないスタイル属性が継承されます。たとえば、PROC TABULATE ステートメントですべてのデータセルに対し青い背景、白い前景を指定し、TABLE ステートメントで特定のクロスのデータセルに対し灰色の背景を指定すると、これらのデータセルの背景が灰色に、前景が白(PROC TABULATE ステートメントでの指定どおり)になります。

STYLE=オプションに関する詳細情報については、各ステートメントのドキュメントを参照してください。

表 64.6 PROC TABULATE で STYLE=オプションを使用する

変更する領域	使用する STYLE オプション
データセル	PROC TABULATE ステートメントまたは次元式 (p. 1874)
ページ次元テキストと分類変数名ヘッダー	“CLASS ステートメント” (p. 1885)
分類水準値ヘッダー	“CLASSLEV ステートメント” (p. 1891)
キーワードのヘッダー	“KEYWORD ステートメント” (p. 1895)
他の場所で指定されていない境界線、ルール、その他の部分	“TABLE ステートメント” (p. 1897)
ボックステキスト	TABLE ステートメントの BOX=オプション (p. 1897)
欠損値	TABLE ステートメントの MISSTEXT=オプション (p. 1897)
分析変数名ヘッダー	“VAR ステートメント” (p. 1907)

図 64.10 PROC TABULATE 領域と対応するステートメント



スタイル属性をテーブルセルに適用する

PROC TABULATE により、特定のセルに使用するスタイル属性がスタイルの次のデフォルトの優先順位から決定されます。

1. その他のスタイル属性が指定されない場合、PROC TABULATE ではデフォルトのスタイル(Data)からのデフォルトのスタイル属性が使用されます。
2. PROC TABULATE ステートメントの STYLE=オプションにより、デフォルトのスタイル属性が変更されます。その他の STYLE=オプション指定がセルに影響しない場合、PROC TABULATE ではこれらのスタイル属性がセルに使用されます。
3. ページ次元で指定される STYLE=オプションは、行次元または列次元のセルに別の STYLE=オプションを指定しない限り、論理ページのすべてのテーブルセルに適用されます。
4. 行次元で指定される STYLE=オプションは、列次元のセルに別の STYLE=オプションを指定しない限り、行のすべてのテーブルセルに適用されます。
5. 列次元で指定される STYLE=オプションは、列のすべてのテーブルセルに適用されます。

この優先順位を変更するには、TABLE ステートメント (1904 ページ)で STYLE_PRECEDENCE=オプションを使用します。たとえば、STYLE_PRECEDENCE=ROW を指定し、行次元で STYLE=オプションを指定すると、これらのスタイル属性値はテーブルセルに指定されているその他すべてに優先します。

出力形式を使用してスタイル要素を割り当てる

出力形式を使用して、コンテンツが分類変数または分析変数の値によって決定されるセルにスタイル属性値を割り当てることができます。たとえば、次のコードは、値が

10,000 未満のセルに赤い背景を割り当て、値が最低 10,000 で 20,000 未満のセルに黄色い背景を割り当て、値が最低 20,000 のセルに緑の背景を割り当てます。

```
proc format;
   value expfmt low-<10000='red'
                      10000-<20000='yellow'
                      20000-high='green';
run;

ods html body='external-HTML-file';
proc tabulate data=energy style=[backgroundcolor=expfmt.];
   class region division type;
   var expenditures;
   table (region all)*(division all),
           type*expenditures;
run;
ods html close;
```

次元式でのスタイル要素の指定

1 つ以上のスタイル要素またはスタイル属性を次元式に指定して、LISTING 以外の出力先の表示を制御できます。次の領域の表示を変更できます。

- 分析変数名のヘッダー
- 分類変数名ヘッダー
- 分類変数の水準値ヘッダー
- データセル
- キーワードのヘッダー
- ページ次元テキスト

スタイル属性またはスタイル要素を次元式に指定すると、PROC TABULATE、CLASS、CLASSLEV、KEYWORD、TABLE、または VAR ステートメントなど、別のステートメントに指定したスタイル属性またはスタイル要素を無効にする場合に便利です。

スタイル要素とスタイル属性を次元式に指定する構文は次のとおりです。

[STYLE<(CLASSLEV)>=<*style-element-name* | PARENT>

[*style-attribute-name-1=style-attribute-value-1< style-attribute-name-2=style-attribute-value-2 ...>*]]

これらは、次元式にあるスタイル属性の例の一部です。

- `dept={label='Department'`
  `style=[color=red]}, N`
- **`dept*[style=MyDataStyle], N`**
- **`dept*[format=12.2 style=MyDataStyle], N`**

注: 次元式で使用する場合、STYLE=オプションを角括弧([and])または括弧({ and }) で囲む必要があります。

(CLASSLEV)
分類変数水準値ヘッダーにスタイル要素を割り当てます。たとえば、次の TABLE ステートメントでは、分類変数 DEPT の水準値ヘッダーの前景色が黄色になるように指定します。

```
table dept=[style(classlev)=
            [color=yellow]]*sales;
```

注: CLASSLEV オプションは、次元式でのみ使用されます。

次元式でのスタイル要素の指定方法の例については、“例 14: ODS 出力にスタイル要素を指定する” (1999 ページ)を参照してください。PROC TABULATE でのスタイルの使用の詳細については、“基本的なレポート記述プロシジャを使用した ODS スタイルの使用” (1918 ページ)を参照してください。

PROC TABULATE のデータベース内処理

データベース内処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。セキュリティの強化は、機密データをデータベース管理システム(DBMS)から抽出する必要がないため可能です。より迅速な処理は、データが比較的遅いネットワーク接続から転送される代わりに高速二次記憶装置を使用して DBMS でローカルで操作されるため、DBMS に自由に使えるより多くの処理リソースがあるため、DBMS は高度に並列かつスケーラブルな方法で実行するクエリを最適化できるため可能です。

DATA=入力データセットが DBMS にテーブルまたはビューとして保存される場合、PROC TABULATE プロシジャはデータベース内処理を使用して、データベース内の操作のほとんどを実行できます。データベース内処理は、より迅速な処理と、データベースと SAS ソフトウェア間のデータ転送の減少という利点を提供できます。

PROC TABULATE は、SQL 暗黙パススルーを使用して、データベース内処理を実行します。プロシジャは、TABLE ステートメントで指定する分類および統計量に基づく SQL クエリを生成します。データベースはこれらの SQL クエリを実行し、初期要約テーブルを構成します。これらは次に PROC TABULATE に送信されます。

分類変数が指定されると、プロシジャは N 次元の種類を表す SQL GROUP BY 句を作成します。DBMS では、N 次元のクラスツリーのみ生成されます。集計クエリがデータベースで実行される際に作成される結果セットが、SAS によって内部 PROC TABULATE データ構造に読み込まれます。

SAS 出力形式定義がデータベースで配置される際に、分類変数のフォーマットがデータベースで実行されます。SAS 出力形式定義がデータベースで配置されていない場合、データベース内集計が未加工値で発生し、結果のセットが PROC TABULATE 内部構造に結合される際に関連する出力形式が適用されます。マルチラベルフォーマットは常に、データベースによって返される最初に集計された結果セットを使用して実行されます。

データベース内処理に対し、次の統計量がサポートされています。N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、CSS、USS、VAR、STD、STDERR、UCLM、LCLM、CV。

SQLGENERATION システムオプションまたは LIBNAME ステートメントオプションは、データベース内プロシジャがデータベース内で実行されるかどうか、およびその実行方法を制御します。デフォルトで、データベース内プロシジャは可能な場合はデータベース内で実行されます。データベース内処理を実行しない、多くのデータセットオプションがあります。完全なリストについては、*SAS/ACCESS for Relational Databases: Reference* の“データベース内プロシジャ”を参照してください。

PROC TABULATE には、次のような機能があります。

- Aster
- DB2
- Greenplum

- HADOOP
- HAWQ
- IMPALA
- Netezza
- Oracle
- SAP HANA
- Teradata

結果:TABULATE プロシジャ

欠損値

PROC TABULATE による欠損値の処理

入力データセットの変数に対する欠損値が出力にどのように影響するかは、PROC TABULATE ステップで変数をどのように使用するかによって異なります。次のテーブルに、プロシジャが欠損値をどのように処理するかを要約します。

表 64.7 PROC TABULATE による欠損値の処理方法の要約

条件	PROC TABULATE デフォルト	デフォルトに優先
オブザベーションに分析変数に対する欠損値が含まれている	その特定の変数に対し統計量(N と NMISS 以外)の計算からそのオブザベーションを除外する	代替なし
オブザベーションに分類変数に対する欠損値が含まれている	テーブルからオブザベーションを除外する [1]	MISSING を PROC TABULATE ステートメントまたは CLASS ステートメントで使用する
カテゴリに対しデータがない	テーブルにカテゴリを表示しない	TABLE ステートメントで PRINTMISS を、PROC TABULATE ステートメントで CLASSDATA=を使用する
テーブルセルに影響するすべてのオブザベーションに分析変数に対する欠損値が含まれている	セルの統計量(N と NMISS 以外)に対する欠損値を表示する	TABLE ステートメントで MISSTEXT=を使用する

[1] CLASS ステートメントは、PROC TABULATE ステップのすべての TABLE ステートメントに適用されます。そのため、変数を分類変数として定義すると、PROC TABULATE は変数に対する欠損値を含むオブザベーションを省略します。TABLE ステートメントでその変数を使用しない場合も同様です。

条件	PROC TABULATE デフォルト	デフォルトに優先
フォーマットされた値に対しデータがない	テーブルにフォーマットされた値を表示しない	CLASS ステートメントで PRELOADFMT、TABLE ステートメントで PRINTMISS を使用する。または PROC TABULATE ステートメントで CLASSDATA=を使用する。あるいは各フォーマットされた値に対しデータが含まれるように入力データセットにダミーオブザベーションを追加する
FREQ 変数値が欠損値、または 1 未満である	統計量の計算にオブザベーションを使用しない	代替なし
WEIGHT 変数値が欠損値、または 0 である	値 0 を使用する	代替なし

This section presents a series of PROC TABULATE steps that illustrate how PROC TABULATE treats missing values. The following program creates the data set and formats that are used in this section and prints the data set. The data set COMPREV contains no missing values. (See the output below.)

```
proc format;
   value cntryfmt 1='United States'
                  2='Japan';
   value compfmt  1='Supercomputer'
                  2='Mainframe'
                  3='Midrange'
                  4='Workstation'
                  5='Personal Computer'
                  6='Laptop';
run;

data comprev;
   input Country Computer Rev90 Rev91 Rev92;
   datalines;
1 1 788.8 877.6 944.9
1 2 12538.1 9855.6 8527.9
1 3 9815.8 6340.3 8680.3
1 4 3147.2 3474.1 3722.4
1 5 18660.9 18428.0 23531.1
2 1 469.9 495.6 448.4
2 2 5697.6 6242.4 5382.3
2 3 5392.1 5668.3 4845.9
2 4 1511.6 1875.5 1924.5
2 5 4746.0 4600.8 4363.7
;

proc print data=comprev noobs;
   format country cntryfmt. computer compfmt.;
   title 'The Data Set COMPREV';
run;
```

アウトプット 64.4 データセット COMPREV

The Data Set COMPREV

Country	Computer	Rev90	Rev91	Rev92
United States	Supercomputer	788.8	877.6	944.9
United States	Mainframe	12538.1	9855.6	8527.9
United States	Midrange	9815.8	6340.3	8680.3
United States	Workstation	3147.2	3474.1	3722.4
United States	Personal Computer	18660.9	18428.0	23531.1
Japan	Supercomputer	469.9	495.6	448.4
Japan	Mainframe	5697.6	6242.4	5382.3
Japan	Midrange	5392.1	5668.3	4845.9
Japan	Workstation	1511.6	1875.5	1924.5
Japan	Personal Computer	4746.0	4600.8	4363.7

欠損値なし

次の PROC TABULATE ステップにより、次の出力が生成されます。

```
proc tabulate data=comprev;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32;
   format country cntryfmt. computer compfmt.;
   title 'Revenues from Computer Sales';
   title2 'for 1990 to 1992';
run;
```

データセットに欠損値が含まれていないため、テーブルにはすべてのオブザベーションが含まれます。すべてのヘッダーとセルには、非欠損値が含まれています。

アウトプット 64.5 コンピュータ売上データ:欠損値なし

Revenues from Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
	Japan	5392.10	5668.30	4845.90
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

欠損分類変数

次のプログラムでは COMPREV をコピーし、8 番目のオブザベーションに Computer の欠損値が含まれるようにデータを変更します。この新しいデータセットの指定を除けば、次の出力“Computer Sales Data:Midrange, Japan, Deleted”を生成するプログラムは、前述の出力“Computer Sales Data:No Missing Values”を生成するプログラムと同じです。PROC TABULATE は分類変数に対する欠損値を含むオブザベーションを無視します。

```
data compmiss;
   set comprev;
   if _n_=8 then computer=.;
run;

proc tabulate data=compmiss;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32;
   format country cntryfmt. computer compfmt.;
   title 'Revenues from Computer Sales';
   title2 'for 1990 to 1992';
run;
```

Computer に対する欠損値を含むオブザベーションのカテゴリは、**`Midrange, Japan`** でした。このカテゴリは、現在存在しません。デフォルトでは、PROC TABULATE は分類変数に対する欠損値を含むオブザベーションを無視するため、このテーブルには出力“Computer Sales Data:No Missing Values”よりも 1 行少ない行が含まれています。

アウトプット 64.6 コンピュータ売上データ:Midrange、Japan 削除済み

Revenues from Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

欠損分類変数を含むオブザベーションを包含

このプログラムは、MISSING オプションを前のプログラムに追加します。MISSING は、PROC TABULATE ステートメント、CLASS ステートメントで使用できます。MISSING を選択した分類変数にのみ適用する場合、選択した変数を含む別の CLASS ステートメントで MISSING を指定します。MISSING オプションにより、分類変数の欠損値を含むオブザベーションがレポートに含まれます。(次の出力を参照してください。)

```
proc tabulate data=compmiss missing;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32;
   format country cntryfmt. computer compfmt.;
   title 'Revenues from Computer Sales';
   title2 'for 1990 to 1992';
run;
```

このテーブルには、Computer の欠損値を含むカテゴリが含まれます。このカテゴリは、テーブルの最初のデータ行を作成します。

アウトプット 64.7 *コンピュータ売上データ:Computer の欠損値*

Revenues from Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
.	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

欠損分類変数を含むオブザベーションのヘッダーのフォーマット

デフォルトでは、出力“Computer Sales Data:Missing Values for Computer”にあるように、PROC TABULATE は分類変数の欠損値を欠損値に対する標準の SAS 文字の 1 つとして表示します(ピリオド、ブランク、アンダースコア、または A から Z までの文字のいずれか)。それ以外を表示する場合、次のプログラムのように、欠損値を含む分類変数に出力形式を割り当てる必要があります。(次の出力を参照してください。)

```
proc format;
   value misscomp 1='Supercomputer'
                  2='Mainframe'
                  3='Midrange'
                  4='Workstation'
                  5='Personal Computer'
                  6='Laptop'
                  .='No type given';
run;

proc tabulate data=compmiss missing;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32;
   format country cntryfmt. computer misscomp.;
   title 'Revenues for Computer Sales';
   title2 'for 1990 to 1992';
```

```
run;
```

このテーブルでは、欠損値が MISSCOMP.出力形式によって指定されるテキストとして表示されます。

アウトプット 64.8 *コンピュータ売上データ:欠損 Computer 値に提供されるテキスト*

Revenues for Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
No type given	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

すべてのカテゴリに対するヘッダーの提供

デフォルトでは、PROC TABULATE は存在しないカテゴリに対し列と行を印刷および省略する各ページを評価します。たとえば、“Computer Sales Data:Text Supplied for Missing Computer Value ”には **`No type given`**、**`United States`**、**`Midrange`**、**`Japan`** の行は含まれません。これらのカテゴリにデータがないためです。可能なすべてのカテゴリをテーブルに表示する場合、次のプログラムのように TABLE ステートメントで PRINTMISS オプションを使用します。(次の出力を参照してください。)

```
proc tabulate data=compmiss missing;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32 printmiss;
   format country cntryfmt. computer misscomp.;
   title 'Revenues for Computer Sales';
   title2 'for 1990 to 1992';
run;
```

このテーブルには、カテゴリ **No type given**、**United States** とカテゴリ **Midrange, Japan** の行が含まれています。これらのカテゴリにデータがないため、統計量の値はすべて欠損値となります。

アウトプット 64.9 *コンピュータ売上データ:欠損統計量値*

Revenues for Computer Sales for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
No type given	United States	.	.	.
	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
	Japan	.	.	.
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

欠損値を含むセルに対するテキストの提供

カテゴリの一部のオブザベーションに分析変数に対する欠損値が含まれている場合、PROC TABULATE は統計量(N と NMISS 以外)の計算にこれらのオブザベーションは使用しません。ただし、カテゴリのオブザベーションにそれぞれ欠損値が含まれている場合、PROC TABULATE は統計量の値に対する欠損値を表示します。分析変数の欠損値をテキストと置き換えるには、次のプログラムのように TABLE ステートメントで MISSTEXT=オプションを使用して、使用するテキストを指定します。(次の出力を参照してください。)

```
proc tabulate data=compmiss missing;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32 printmiss misstext='NO DATA!';
   format country cntryfmt. computer misscomp.;
   title 'Revenues for Computer Sales';
   title2 'for 1990 to 1992';
run;
```

このテーブルでは、欠損値の表示に通常使用されるピリオドを MISSTEXT=オプションのテキストと置き換えます。

アウトプット 64.10 *コンピュータ売上データ:欠損統計量値に対し提供されるテキスト*

Revenues for Computer Sales for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
No type given	United States	NO DATA!	NO DATA!	NO DATA!
	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
	Japan	NO DATA!	NO DATA!	NO DATA!
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

出力形式のすべての値に対するヘッダーの提供

PROC TABULATE は、入力データセットに表示される値に対してのみヘッダーを印刷します。たとえば、出力形式 COMPFMT.は、Computer について 6 つの可能な値を提供しています。これらの値のうち 5 つのみが、データセット COMPREV で発生します。データセットには、ラップトップコンピュータに対するデータは含まれていません。

Computer の可能なすべての値に対するヘッダーを含める場合(出力と、ラップトップに関するデータがない場合に後で作成されるテーブルとを比較しやすくするため)、そのようなテーブルの作成には 3 つの方法があります。

- CLASS ステートメントの PRELOADFMT オプションを TABLE ステートメントの PRINTMISS オプションとともに使用します。PRELOADFMT を使用する別の例については、“例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (1950 ページ)を参照してください。
- PROC TABULATE ステートメントで CLASSDATA=オプションを使用します。CLASSDATA=オプションを使用する例については、“例 2: テーブルに表示する分類変数の組み合わせを指定する” (1947 ページ)を参照してください。
- 出力形式で処理される各値がデータセットに少なくとも一度表示されるように、ダミー値を入力データセットに追加します。

次のプログラムは、PRELOADFMT オプションを関連する変数を含む CLASS ステートメントに追加します。

結果は、次の出力のとおりです。

```
proc tabulate data=compmiss missing;
   class country;
   class computer / preloadfmt;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32 printmiss misstext='NO DATA!';
   format country cntryfmt. computer compfmt.;
   title 'Revenues for Computer Sales';
   title2 'for 1990 to 1992';
run;
```

このテーブルには、Computer のそれぞれの可能な値に対するヘッダーが含まれます。

アウトプット 64.11 *コンピュータ売上データ:すべての可能な Computer 値を含む*

Revenues for Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
.	United States	NO DATA!	NO DATA!	NO DATA!
	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
	Japan	NO DATA!	NO DATA!	NO DATA!
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70
Laptop	United States	NO DATA!	NO DATA!	NO DATA!
	Japan	NO DATA!	NO DATA!	NO DATA!

ORDER=DATA を使用したヘッダーの順序について

ORDER=オプションは、すべての分類変数に適用されます。異なる変数のヘッダーを別に並べ替えることが必要な場合があります。ヘッダーを並べ替える 1 つの方法は、データを表示によってグループ化し、ORDER=DATA を指定することです。

この方法を使用できるようにするには、最初の分類変数の最初の値が、その他すべての分類変数の可能なすべての値を含むデータで発生する必要があります。この条件に一致しない場合、ヘッダーの順序が希望と異なることがあります。

次のプログラムは、オブザベーションが最初に Animal の値、次に Food の値によって並べ替えられる単一のデータセットを作成します。PROC TABULATE ステートメントの ORDER=オプションは、分類変数のヘッダーをデータセットで表示される順序によって並べ替えます。(次の出力を参照してください。)**bones** は Animal=**dog** のオブザベーショングループの Food の最初の値ですが、Food のその他すべての値はデータセットの **bones** の前に表示されます。これは、**bones** が Animal=**cat** の場合に表示されないためです。そのため、次の出力のテーブルの **bones** のヘッダーは、アルファベット順ではありません。

つまり、PROC TABULATE は、前のカテゴリによって構築された順序を後続のカテゴリに対して保持します。Animal の各値に対して Food の順序を再構築する場合、BY グループ処理を使用します。PROC TABULATE は BY グループに対しそれぞれ別のテーブルを作成するため、並べ替えは BY グループ間で異なる可能性があります。

```
data foodpref;
   input Animal $ Food $;
   datalines;
cat fish
cat meat
cat milk
dog bones
dog fish
dog meat
;

proc tabulate data=foodpref format=9.
              order=data;
   class animal food;
   table animal*food;
run;
```

アウトプット 64.12 *分類変数のヘッダーの並べ替え*

Animal Food Preference

Animal					
cat			dog		
Food			Food		
fish	meat	milk	fish	meat	bones
N	N	N	N	N	N
1	1	1	1	1	1

PROC TABULATE による ODS 出力のポータビリティ

特定の状況で PROC TABULATE をアウトプットデリバリシステムと使用すると、ポータブルでないファイルが生成されます。SAS セッションの SAS システムオプション FORMCHAR=で非標準の線描文字が使用されると、SAS Monospace フォントがインストールされていない動作環境では線の代わりに不正な文字が出力に含まれていることがあります。この問題を回避するため、PROC TABULATE を実行する前に次の OPTIONS ステートメントを指定します。

```
options formchar="|----|+|---+=|-/\<>*";
```

例: TABULATE プロシジャ

例 1: 基本的な 2 次元表の作成

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメントオプション
- クロス(*)演算子
- RTS=

VAR ステートメント

他の要素: DATA ステップ

FORMAT プロシジャ

FORMAT statement
TITLE statement

データセット: ENERGY

詳細

次のサンプルプログラムでは、次を実行します。

- 各地域の各区分の各種のユーザー(住居または企業)に対するカテゴリを作成する
- テーブルのすべてのセルに同じ出力形式を適用する
- 各分類変数に出力形式を適用する
- 行ヘッダーのスペースを拡張する

プログラム

```
data energy;
   length State $2;
   input Region Division state $ Type Expenditures;
   datalines;
1 1 ME 1 708
1 1 ME 2 379

 ... more data lines ...

4 4 HI 1 273
4 4 HI 2 298
;

proc format;
   value regfmt 1='Northeast'
                2='South'
                3='Midwest'
                4='West';
   value divfmt 1='New England'
                2='Middle Atlantic'
                3='Mountain'
                4='Pacific';
   value usetype 1='Residential Customers'
                 2='Business Customers';
run;

proc tabulate data=energy format=dollar12.;

   class region division type;


   var expenditures;

    table region*division,
          type*expenditures

          / rts=25;

   format region regfmt. division divfmt. type usetype.;

   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
```

```
run;
```

プログラムの説明

ENERGY データセットを作成します。 ENERGY には、米国の Northeast 地域と West 地域の各州の企業顧客および住居顧客のエネルギー支出に関するデータが含まれています。A データステップは (2152 ページ) データセットを作成します。

```
data energy;
   length State $2;
   input Region Division state $ Type Expenditures;
   datalines;
1 1 ME 1 708
1 1 ME 2 379

 ... more data lines ...

4 4 HI 1 273
4 4 HI 2 298
;
```

出力形式 REGFMT.、DIVFMT.および USETYPE.を作成します。 PROC FORMAT は、Region、Division、Type に対する出力形式を作成します。

```
proc format;
   value regfmt 1='Northeast'
                2='South'
                3='Midwest'
                4='West';
   value divfmt 1='New England'
                2='Middle Atlantic'
                3='Mountain'
                4='Pacific';
   value usetype 1='Residential Customers'
                 2='Business Customers';
run;
```

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Region、Division、Type の値別に分類します。

```
   class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
   var expenditures;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Region のフォーマットされた値に対しそれぞれ行を作成します。各行内でネストされるのは、Division の各フォーマットされた値に対する行です。TABLE ステートメントは、Type のフォーマットされた値に対してもそれぞれ列を作成します。これらの行と列によって作成される各セルには、

セルに影響するすべてのオブザベーションに対する分析変数 Expenditures の合計が含まれます。

```
      table region*division,
            type*expenditures
```

行タイトルスペースを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。

```
            / rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
      format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
      title 'Energy Expenditures for Each Region';
      title2 '(millions of dollars)';
   run;
```

出力

アウトプット 64.13 *基本的な 2 次元表*

Energy Expenditures for Each Region
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

例 2: テーブルに表示する分類変数の組み合わせを指定する

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- CLASSDATA=
- EXCLUSIVE
- FORMAT=

TABLE ステートメントオプション

クロス(*)演算子
RTS=

VAR ステートメント

他の要素: DATA ステップ
FORMAT statement
TITLE statement

データセット: ENERGY

詳細

この例では、次を行います。

- CLASSDATA=オプションを使用して、テーブルに表示する分類変数の組み合わせを指定する。
- EXCLUSIVE オプションを使用して、出力を CLASSDATA=データセットで指定される組み合わせにのみ制限する。EXCLUSIVE オプションを使用しない場合、出力は“例 1: 基本的な 2 次元表の作成” (1944 ページ)と同じになります。

プログラム

```
data classes;
   input region division type;
   datalines;
1 1 1
1 1 2
4 4 1
4 4 2
;

proc tabulate data=energy format=dollar12.
              classdata=classes exclusive;


   class region division type;

   var expenditures;

   table region*division,
          type*expenditures

        / rts=25;

   format region regfmt. division divfmt. type usetype.;

   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

プログラムの説明

CLASSES データセットを作成します。 CLASSES には、PROC TABULATE がテーブルの作成に使用する分類変数値の組み合わせが含まれます。

```
data classes;
   input region division type;
   datalines;
```

```
1 1 1
1 1 2
4 4 1
4 4 2
;
```

テーブルオプションを指定します。 CLASSDATA=と EXCLUSIVE は、分類変数の水準の組み合わせを CLASSES データセットで指定される組み合わせに制限します。

```
proc tabulate data=energy format=dollar12.
              classdata=classes exclusive;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Region、Division、Type の値別に分類します。

```
   class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
   var expenditures;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Region のフォーマットされた値に対しそれぞれ行を作成します。各行内でネストされるのは、Division の各フォーマットされた値に対する行です。TABLE ステートメントは、Type のフォーマットされた値に対してもそれぞれ列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Expenditures の合計が含まれます。

```
   table region*division,
         type*expenditures
```

行タイトルスペースを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。

```
        / rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
   format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

出力

アウトプット 64.14 *地域ごとのエネルギー支出*

Energy Expenditures for Each Region
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
West	Pacific	$13,959	$12,619

例 3: すでに読み込まれている出力形式を分類変数とともに使用する

要素: CLASS ステートメントオプション
- EXCLUSIVE
- PRELOADFMT

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=
- OUT=

TABLE ステートメントオプション
- クロス(*)演算子
- PRINTMISS
- RTS

VAR ステートメント

他の要素: FORMAT statement

PRINT プロシジャ

TITLE statement

データセット: ENERGY

詳細

この例では、次を行います。

- フォーマットされた分類変数値の可能なすべての組み合わせを含むテーブルを作成する(PRELOADFMT と PRINTMISS) (これらの組み合わせにゼロの度数が含まれていても、これらの組み合わせが意味をなさなくても同様)
- ユーザー定義出力形式のすでに読み込まれた範囲のみを分類変数の水準として使用する(PRELOADFMT と EXCLUSIVE)
- 出力を出力データセットに書き込み、そのデータセットを印刷する

プログラム

```
proc tabulate data=energy format=dollar12.;

   class region division type / preloadfmt;

   var expenditures;

   table region*division,
         type*expenditures / rts=25 printmiss;

   format region regfmt. division divfmt. type usetype.;

   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;

proc tabulate data=energy format=dollar12. out=tabdata;

   class region division type / preloadfmt exclusive;

   var expenditures;

   table region*division,
         type*expenditures / rts=25;

   format region regfmt. division divfmt. type usetype.;

   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;

proc print data=tabdata;
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Region、Division、Type の値別に分類します。PRELOADFMT は、PROC TABULATE が分類変数に対するユーザー定義出力形式のすでに読み込まれた値を使用するように指定します。

```
      class region division type / preloadfmt;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
      var expenditures;
```

テーブルの行と列を定義し、行と列のオプションを指定します。 PRINTMISS は、ユーザー定義出力形式のすべての可能な組み合わせが分類変数の水準として使用されるように指定します。

```
      table region*division,
            type*expenditures / rts=25 printmiss;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

テーブルオプションと出力データセットを指定します。 OUT=オプションは、PROC TABULATE がデータを書き込む出力データセットの名前を指定します。

```
proc tabulate data=energy format=dollar12. out=tabdata;
```

分析対象となるサブグループを指定します。 EXCLUSIVE オプションは PRELOADFMT と使用されると、ユーザー定義出力形式のすでに読み込まれた範囲のみを分類変数の水準として使用します。

```
class region division type / preloadfmt exclusive;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;
```

テーブルの行と列を定義し、行と列のオプションを指定します。 この場合、PRINTMISS オプションは指定されません。指定された場合、CLASS ステートメントの EXCLUSIVE オプションに優先します。

```
table region*division,
      type*expenditures / rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

出力データセット WORK.TABDATA を印刷します。

```
proc print data=tabdata;
run;
```

出力

PRELOADFMT オプションと PRINTMISS オプションで作成されたこの出力には、分類変数値に対するすでに読み込まれたユーザー定義出力形式のすべての可能な組

み合わせが含まれています。ゼロの度数を含む組み合わせや、`Northeast` と `Pacific` などの意味をなさない組み合わせが含まれます。

アウトプット 64.15 *地域ごとのエネルギー支出*

Energy Expenditures for Each Region
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
	Mountain	.	.
	Pacific	.	.
South	New England	.	.
	Middle Atlantic	.	.
	Mountain	.	.
	Pacific	.	.
Midwest	New England	.	.
	Middle Atlantic	.	.
	Mountain	.	.
	Pacific	.	.
West	New England	.	.
	Middle Atlantic	.	.
	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

PRELOADFMT オプションと EXCLUSIVE オプションで作成されたこの出力には、入力データセットで表示される分類変数値に対するすでに読み込まれたユーザー定義

出力形式の組み合わせのみ含まれます。この出力は、“例 1: 基本的な 2 次元表の作成” (1944 ページ)からの出力と同じです。

アウトプット 64.16 *地域ごとのエネルギー支出*

Energy Expenditures for Each Region
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

この出力には、PROC TABULATE ステートメントの OUT=オプションによって作成された出力データセット TABDATA が表示されます。TABDATA には、PRELOADFMT オプションと EXCLUSIVE オプションを指定して作成されるデータが含まれます。

アウトプット 64.17 *地域ごとのエネルギー支出*

Energy Expenditures for Each Region
(millions of dollars)

Obs	Region	Division	Type	_TYPE_	_PAGE_	_TABLE_	Expenditures_Sum
1	Northeast	New England	Residential Customers	111	1	1	7477
2	Northeast	New England	Business Customers	111	1	1	5129
3	Northeast	Middle Atlantic	Residential Customers	111	1	1	19379
4	Northeast	Middle Atlantic	Business Customers	111	1	1	15078
5	West	Mountain	Residential Customers	111	1	1	5476
6	West	Mountain	Business Customers	111	1	1	4729
7	West	Pacific	Residential Customers	111	1	1	13959
8	West	Pacific	Business Customers	111	1	1	12619

例 4: マルチラベル出力形式の使用

要素: CLASS ステートメントオプション
- MLF

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメント
- ALL 分類変数
- 連結(ブランク)演算子
- クロス(*)演算子
- 要素グループ化(かっこ)演算子
- ラベル
- 変数リスト

VAR ステートメント

他の要素: DATA ステップ

FORMAT プロシジャ

FORMAT statement

TITLE statement

データセット: CARSURVEY

詳細

この例では、次を行います。

- PROC FORMAT の VALUE ステートメントでマルチラベル出力形式を指定する方法を示す
- MLF オプションを CLASS ステートメントと使用して、マルチラベル出力形式処理を有効化する方法を示す
- マルチラベル出力形式処理が有効化される場合の N 統計量の動作を説明する

プログラム

```
data carsurvey;
   input Rater Age Progressa Remark Jupiter Dynamo;
   datalines;
1   38  94  98  84  80
2   49  96  84  80  77
3   16  64  78  76  73
4   27  89  73  90  92

 ... more data lines ...

77   61  92  88  77  85
78   24  87  88  88  91
79   18  54  50  62  74
80   62  90  91  90  86
;

proc format;
```

```
   value agefmt (multilabel notsorted)
         15 - 29 = 'Below 30 years'
         30 - 50 = 'Between 30 and 50'
       51 - high = 'Over 50 years'
         15 - 19 = '15 to 19'
         20 - 25 = '20 to 25'
         25 - 39 = '25 to 39'
         40 - 55 = '40 to 55'
       56 - high = '56 and above';
run;

proc tabulate data=carsurvey format=10.;

   class age / mlf;

   var progressa remark jupiter dynamo;

   table age all, n all='Potential Car Names'*(progressa remark
      jupiter dynamo)*mean;


   title1 "Rating Four Potential Car Names";
   title2 "Rating Scale 0-100 (100 is the highest rating)";

   format age agefmt.;
run;
```

プログラムの説明

CARSURVEY データセットを作成します。 CARSURVEY には、自動車メーカーによって新車名を評価するために集められた潜在顧客のフォーカスグループに配信された調査からのデータが含まれます。データセットの各オブザベーションには、ID 番号、参加者の年齢、参加者の 4 つの車名の評価が含まれます。DATA ステップは、データセットを作成します。

```
data carsurvey;
   input Rater Age Progressa Remark Jupiter Dynamo;
   datalines;
1    38   94   98   84   80
2    49   96   84   80   77
3    16   64   78   76   73
4    27   89   73   90   92

 ... more data lines ...

77    61   92   88   77   85
78    24   87   88   88   91
79    18   54   50   62   74
80    62   90   91   90   86
;
```

AGEFMT.出力形式を作成します。 FORMAT プロシジャは“MULTILABEL” (870 ページ)を使用して、年齢に対するマルチラベル出力形式を作成します。マルチラベル出力形式は、複数のラベルを同じ値に割り当てることができる出力形式です。この場合は、範囲が重複するためです。各値は発生する各範囲のテーブルに表されます。NOTSORTED オプションは、範囲を範囲が定義された順序で保存します。

```
proc format;
   value agefmt (multilabel notsorted)
```

```
          15 - 29 = 'Below 30 years'
          30 - 50 = 'Between 30 and 50'
        51 - high = 'Over 50 years'
          15 - 19 = '15 to 19'
          20 - 25 = '20 to 25'
          25 - 39 = '25 to 39'
          40 - 55 = '40 to 55'
        56 - high = '56 and above';
run;
```

テーブルオプションを指定します。 FORMAT=オプションは、最大 10 桁を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=carsurvey format=10.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは Age を分類変数として識別し、MLF オプションを使用してマルチラベル出力形式処理を有効化します。

```
    class age / mlf;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が変数 Progressa、Remark、Jupiter、Dynamo の統計量を計算するように指定します。

```
    var progressa remark jupiter dynamo;
```

テーブルの行と列を定義します。 TABLE ステートメントの行次元は、Age のフォーマットされた値に対しそれぞれ行を作成します。マルチラベル出力形式により、オブザベーションが複数の行または年齢のカテゴリに含まれるようになります。行次元は ALL 分類変数を使用して、すべての行に関する情報を要約します。列次元は N 統計量を使用して、各年齢グループに対するオブザベーション数を計算します。行次元の ALL 分類変数とクロスする N 統計量の結果は、行に対する N 統計量の合計ではなく、オブザベーションの合計数です。列次元はクロスの開始時に ALL 分類変数を使用して、ラベル **`Potential Car Names`** を割り当てます。4 つのネストされた列が、各年齢グループに対する車名の平均評価を計算します。

```
    table age all, n all='Potential Car Names'*(progressa remark
       jupiter dynamo)*mean;
```

タイトルを指定します。

```
    title1 "Rating Four Potential Car Names";
    title2 "Rating Scale 0-100 (100 is the highest rating)";
```

出力をフォーマットします。 FORMAT ステートメントは、ユーザー定義出力形式 AGEFMT.をこの分析用の Age に割り当てます。

```
    format age agefmt.;
run;
```

出力

アウトプット 64.18 *可能性のある 4 つの車名の評価*

Rating Four Potential Car Names
Rating Scale 0-100 (100 is the highest rating)

	N	Potential Car Names			
		Progressa	Remark	Jupiter	Dynamo
		Mean	Mean	Mean	Mean
Age					
15 to 19	14	75	78	81	73
20 to 25	11	89	88	84	89
25 to 39	26	84	90	82	72
40 to 55	14	85	87	80	68
56 and above	15	84	82	81	75
Below 30 years	36	82	84	82	75
Between 30 and 50	25	86	89	81	73
Over 50 years	19	82	84	80	76
All	80	83	86	81	74

例 5: 行と列のヘッダーのカスタマイズ

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメントオプション
- クロス(*)演算子
- ラベル
- RTS=

VAR ステートメント

他の要素: FORMAT statement

TITLE statement

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、行ヘッダーと列ヘッダーをカスタマイズする方法を示します。ラベルは、ヘッダー用テキストを指定します。ブランクのラベルはブランクのヘッダーを作成します。PROC TABULATE は、テーブルからブランクの列ヘッダーのスペースを削除します。

プログラム

```
proc tabulate data=energy format=dollar12.;

   class region division type;

   var expenditures;

    table region*division,
         type='Customer Base'*expenditures=' '*sum=' '

         / rts=25;

   format region regfmt. division divfmt. type usetype.;

   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
   class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
   var expenditures;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Region のフォーマットされた値に対しそれぞれ行を作成します。各行内でネストされるのは、Division の各フォーマットされた値に対する行です。TABLE ステートメントは、Type のフォーマットされた値に対してもそれぞれ列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Expenditures の合計が含まれます。引用符内のテキストは、対応する変数または統計量に対するヘッダーを指定します。Sum はデフォルトの統計量ですが、そのヘッダーに対しブランクを指定できるようにここで指定されます。

```
    table region*division,
         type='Customer Base'*expenditures=' '*sum=' '
```

行タイトルスペースを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。

```
         / rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を Region、Division、Type に割り当てます。

```
    format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
    title 'Energy Expenditures for Each Region';
    title2 '(millions of dollars)';
run;
```

出力

Type のヘッダーには、TABLE ステートメントで指定されるテキストが含まれます。TABLE ステートメントは、Expenditures と Sum のヘッダーを削除しました。

アウトプット 64.19 *地域ごとのエネルギー支出*

Energy Expenditures for Each Region
(millions of dollars)

		Customer Base	
		Residential Customers	Business Customers
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

例 6: 共通分類変数 ALL を使用した情報の要約

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメント
- ALL 分類変数
- 連結(ブランク演算子)
- 出力形式修飾子
- 要素グループ化(かっこ演算子)
- RTS=

VAR ステートメント

他の要素: FORMAT statement

TITLE statement

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、共通分類変数 ALL を使用して複数のカテゴリからの情報を要約する方法を示します。

プログラム

```
proc tabulate data=energy format=comma12.;

   class region division type;

   var expenditures;

table region*(division all='Subtotal')
         all='Total for All Regions'*f=dollar12.,

         type='Customer Base'*expenditures=' '*sum=' '
         all='All Customers'*expenditures=' '*sum=' '

      / rts=25;

   format region regfmt. division divfmt. type usetype.;

   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、COMMA12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=comma12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
   class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
   var expenditures;

table region*(division all='Subtotal')
         all='Total for All Regions'*f=dollar12.,

         type='Customer Base'*expenditures=' '*sum=' '
         all='All Customers'*expenditures=' '*sum=' '
```

行タイトルスペースを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。

```
      / rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
   format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
    title 'Energy Expenditures for Each Region';
    title2 '(millions of dollars)';
run;
```

出力

共通分類変数 ALL により、このテーブルの小計と合計が提供されます。

アウトプット 64.20 *地域ごとのエネルギー支出*

Energy Expenditures for Each Region
(millions of dollars)

		Customer Base		All Customers
		Residential Customers	Business Customers	
Region	Division			
Northeast	New England	7,477	5,129	12,606
	Middle Atlantic	19,379	15,078	34,457
	Subtotal	26,856	20,207	47,063
West	Division			
	Mountain	5,476	4,729	10,205
	Pacific	13,959	12,619	26,578
	Subtotal	19,435	17,348	36,783
Total for All Regions		$46,291	$37,555	$83,846

例 7: 行ヘッダーの削除

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメントオプション
- クロス(*)演算子
- ラベル
- ROW=FLOAT
- RTS=

VAR ステートメント

他の要素: FORMAT statement

TITLE statement

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、ブランクの行ヘッダーをテーブルから削除する方法を示します。これを行うには、行ヘッダーに対しブランクのラベルを指定し、TABLE ステートメントで ROW=FLOAT を指定する必要があります。

プログラム

```
proc tabulate data=energy format=dollar12.;
   class region division type;
   var expenditures;
   table region*division*expenditures=' '*sum=' ',
         type='Customer Base'
         / rts=25 row=float;
   format region regfmt. division divfmt. type usetype.;
   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
   class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
   var expenditures;
```

テーブル行を定義します。 TABLE ステートメントの行次元は、Region のフォーマットされた値に対しそれぞれ行を作成します。これらの行内でネストされるのは、Division の各フォーマットされた値に対する行です。分析変数 Expenditures と Sum 統計量も行次元に含まれるため、PROC TABULATE はそれらの統計量の行ヘッダーも作成します。引用符内のテキストは、対応する変数または統計量に対するヘッダーを指定します。Sum はデフォルトの統計量ですが、そのヘッダーに対しブランクを指定できるようにここで指定されます。

```
   table region*division*expenditures=' '*sum=' ',
```

テーブル列を定義します。 TABLE ステートメントの列次元は、Type のフォーマットされた値に対しそれぞれ列を作成します。

```
         type='Customer Base'
```

行タイトルのスペースを指定し、ブランクの行ヘッダーを削除します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。ROW=FLOAT は、ブランクの行ヘッダーを削除します。

```
           / rts=25 row=float;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
      format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
      title 'Energy Expenditures for Each Region';
      title2 '(millions of dollars)';
   run;
```

出力

このテーブルと“例 5: 行と列のヘッダーのカスタマイズ” (1958 ページ)の出力を比較します。2 つのテーブルは同じですが、このテーブルを作成するプログラムでは行次元の Expenditures と Sum が使用されます。PROC TABULATE によりブランクのヘッダーが列次元から自動的に削除されますが、ROW=FLOAT を指定して、ブランクのヘッダーを行次元から削除する必要があります。

アウトプット 64.21 *地域ごとのエネルギー支出*

Energy Expenditures for Each Region
(millions of dollars)

		Customer Base	
		Residential Customers	Business Customers
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

例 8: 行ヘッダーをインデントし、水平区切り線を削除する

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=
- NOSEPS

TABLE ステートメントオプション
- クロス(*)演算子
- ラベル

INDENT=
RTS=

VAR ステートメント

他の要素: FORMAT statement
ODS LISTING ステートメント
ODS LISTING CLOSE ステートメント
OPTIONS ステートメント
TITLE statement

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、次を行ってテーブルの構造を簡略化する方法を示します。

- 分類変数の行ヘッダーを削除する
- 親行下にネストされた行を並べて配置せずにインデントする
- 水平区切り線を行タイトルとテーブルの本文から削除する

プログラム

```
options nodate nonumber;
ods listing;

proc tabulate data=energy format=dollar12. noseps;

   class region division type;

   var expenditures;

    table region*division,
          type='Customer Base'*expenditures=' '*sum=' '

         / rts=25 indent=4;

   format region regfmt. division divfmt. type usetype.;

   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;

ods listing close;
```

プログラムの説明

LISTING 出力先を開きます。 INDENT 引数は、HTML 出力のネストされた行ヘッダーをインデントしません。出力は、ページ番号と日付が表示されないリストとして取得されます。

```
options nodate nonumber;
ods listing;
```

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。NOSEPS は、行タイトルとテーブルの本文から水平区切り線を削除します。

```
proc tabulate data=energy format=dollar12. noseps;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Region のフォーマットされた値に対しそれぞれ行を作成します。各行内でネストされるのは、Division の各フォーマットされた値に対する行です。TABLE ステートメントは、Type のフォーマットされた値に対してもそれぞれ列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Expenditures の合計が含まれます。すべての次元の引用符内のテキストは、対応する変数または統計量に対するヘッダーを指定します。Sum はデフォルトの統計量ですが、そのヘッダーに対しブランクを指定できるようにここで指定されます。

```
table region*division,
      type='Customer Base'*expenditures=' '*sum=' '
```

行タイトルスペースとインデント値を指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。INDENT=は分類変数の行ヘッダーを削除し、Division の値を Region の値の横ではなく下に置き、Division の 4 つのスペースの値をインデントします。

```
           / rts=25 indent=4;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

LISTING 出力先を閉じます。

```
ods listing close;
```

出力

NOSEPS は、区切り線を行タイトルとテーブルの本文から削除します。INDENT=は Region と Division の行ヘッダーを削除し、Division の値を Region の値の下にインデントします。

アウトプット 64.22　地域ごとのエネルギー支出

Energy Expenditures for Each Region
(millions of dollars)

	Customer Base	
	Residential Customers	Business Customers
Northeast		
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078
West		
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619

例 9: 複数ページテーブルの作成

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメントオプション
- ALL 分類変数
- BOX=
- CONDENSE
- INDENT=
- ページ式
- RTS=

VAR ステートメント

他の要素: FORMAT statement

TITLE statement

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、各地域に対する個別のテーブルと、すべての地域に対する 1 つのテーブルを作成します。デフォルトでは、PROC TABULATE は個別のページに各テーブルを作成しますが、CONDENSE オプションはそれらのテーブルをすべて同じページに置きます。

プログラム

```
proc tabulate data=energy format=dollar12.;

   class region division type;
```

```
    var expenditures;
    table region='Region: ' all='All Regions',
    division all='All Divisions',
          type='Customer Base'*expenditures=' '*sum=' '
          / rts=25 box=_page_ condense indent=1;
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures for Each Region and All Regions';
    title2 '(millions of dollars)';
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
    class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
    var expenditures;
```

テーブルページを定義します。 TABLE ステートメントのページ次元は、Region のフォーマットされた値に対しそれぞれ 1 つのテーブルを、すべての地域に対し 1 つのテーブルを作成します。引用符内のテキストは、各ページに対しヘッダーを提供します。

```
    table region='Region: ' all='All Regions',
```

テーブル行を定義します。 行次元は、Division のフォーマットされた値に対しそれぞれ 1 行、すべての区分に対し 1 行を作成します。引用符内のテキストは、行ヘッダーを提供します。

```
    division all='All Divisions',
```

テーブル列を定義します。 TABLE ステートメントの列次元は、Type のフォーマットされた値に対しそれぞれ列を作成します。これらのページ、行および列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Expenditures の合計が含まれます。引用符内のテキストは、対応する変数または統計量に対するヘッダーを指定します。Sum はデフォルトの統計量ですが、そのヘッダーに対しブランクを指定できるようにここで指定されます。

```
          type='Customer Base'*expenditures=' '*sum=' '
```

追加のテーブルオプションを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。BOX=は、ページヘッダーを行ヘッダーの上のボックス内に置きます。CONDENSE は、1 つの物理ページに可能な限り多くのテーブルを置きます。INDENT=は、Division の行ヘッダーを削除します。(行次元にはネスティングがないため、インデントは実行されません)。

```
          / rts=25 box=_page_ condense indent=1;
```

出力をフォーマットします。FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region and All Regions';
title2 '(millions of dollars)';
run;
```

出力

アウトプット 64.23 *各地域および全地域のエネルギー支出*

Energy Expenditures for Each Region and All Regions
(millions of dollars)

Region: Northeast	Customer Base	
	Residential Customers	Business Customers
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078
All Divisions	$26,856	$20,207

Region: West	Customer Base	
	Residential Customers	Business Customers
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619
All Divisions	$19,435	$17,348

All Regions	Customer Base	
	Residential Customers	Business Customers
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619
All Divisions	$46,291	$37,555

例 10: 複数回答式の調査データに基づくレポート作成

要素:
PROC TABULATE ステートメントオプション
- DATA=

TABLE ステートメント
- 分母定義(山かっこ演算子)
- N 統計量
- PCTN 統計量
- 変数リスト

VAR ステートメント

他の要素:
DATA ステップ

FORMAT プロシジャ

FOOTNOTE statement

OPTIONS ステートメントオプション
- FORMDLIM=
- NONUMBER

SYMPUT ルーチン

TITLE statement

データセット: CUSTOMER_RESPONSE

詳細

この例の 2 つのテーブルに、次を示します。

- 製品を購入する顧客の決定に最も影響している要因
- 企業に関する顧客の情報源

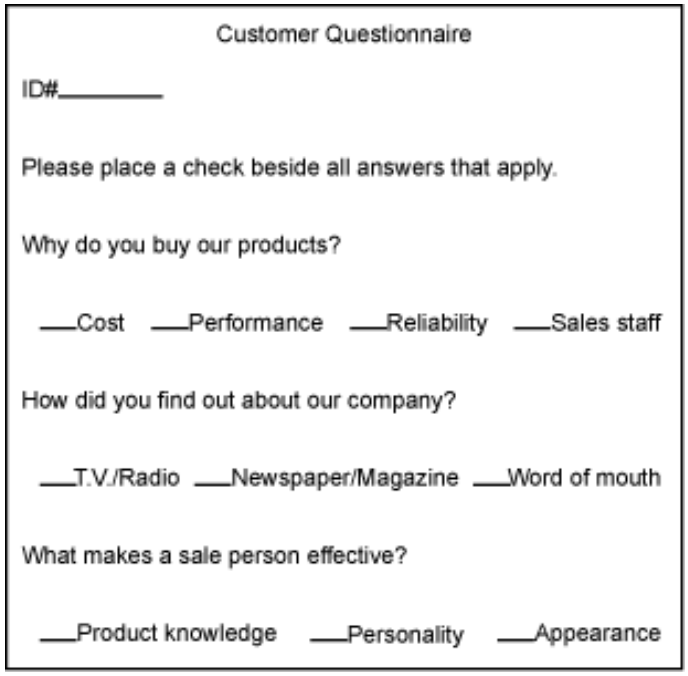
レポートは、1 つのページ番号のみ付いた 1 つの物理ページに表示されます。デフォルトでは、レポートは別々のページに表示されます。

この例では、これらのテーブルを作成する方法だけでなく、次を行う方法も示されます。

- DATA ステップを使用して、データセットのオブザベーション数をカウントする
- 値をマクロ変数に保存する
- その値に後で SAS セッションでアクセスする

次の図に、データの収集に使用される調査用紙を示します。

図 64.11 記入式調査用紙



プログラム

```
data customer_response;
   input Customer Factor1-Factor4 Source1-Source3
         Quality1-Quality3;
   datalines;
1 . . 1 1 1 1 . 1 . .
2 1 1 . 1 1 1 . 1 1 .
3 . . 1 1 1 1 . . . .

. . . more data lines . . .

119 . . . 1 . . . 1 . .
120 1 1 . 1 . . . . 1 .
;

data _null_;
   if 0 then set customer_response nobs=count;
   call symput('num',left(put(count,4.)));
   stop;
run;

proc format;
   picture pctfmt low-high='009.9 %';
run;

proc tabulate data=customer_response;

   var factor1-factor4 customer;

   table factor1='Cost'
         factor2='Performance'
```

```
          factor3='Reliability'
          factor4='Sales Staff',
          (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;

   title 'Customer Survey Results: Spring 1996';
   title3 'Factors Influencing the Decision to Buy';
run;

proc tabulate
data=customer_response;

   var source1-source3 customer;

   table source1='TV/Radio'
          source2='Newspaper'
          source3='Word of Mouth',
          (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;

   title 'Source of Company Name';
   footnote "Number of Respondents: &num";
run;

options formdlim='' number;
```

プログラムの説明

CUSTOMER_RESPONSE データセットを作成します。 CUSTOMER_RESPONSE には、顧客調査からのデータが含まれています。データセットの各オブザベーションには、1 人の回答者の製品購入の決定に影響する要因に関する情報が含まれます。データステップは (2147 ページ) エータセットを作成します。0 ではなく欠損値を使用することは、PROC TABULATE での度数カウントの計算に重要です。

```
data customer_response;
   input Customer Factor1-Factor4 Source1-Source3
         Quality1-Quality3;
   datalines;
1 . . 1 1 1 1 . 1 . .
2 1 1 . 1 1 1 . 1 1 .
3 . . 1 1 1 1 . . . .

. . . more data lines . . .

119 . . . 1 . . . 1 . .
120 1 1 . 1 . . . . 1 .
;
```

オブザベーション数をマクロ変数に保存します。 SET ステートメントはコンパイル時に CUSTOMER_RESPONSE のディスクリプタ部分を読み込み、オブザベーション数(回答者数)を COUNT に保存します。SYMPUT ルーチンは、COUNT の値をマクロ変数 NUM に保存します。その他のプロシジャと DATA ステップは、この変数をその他の SAS セッションに使用できます。常に false の IF 0 条件により、オブザベーションを読み込む SET ステートメントが実行されなくなります。(オブザベーションの読み込みは不要です)。STOP ステートメントにより、DATA ステップは 1 回だけ実行されるようになります。

```
data _null_;
   if 0 then set customer_response nobs=count;
   call symput('num',left(put(count,4.)));
   stop;
```

```
run;
```

PCTFMT.出力形式を作成します。 FORMAT プロシジャは、パーセント用の出力形式を作成します。PCTFMT.出力形式は、少なくとも 1 桁を含むすべての値を小数点の左に書き込み、1 桁を含むすべての値を小数点の右に書き込みます。ブランクとパーセント記号が桁の後に続きます。

```
proc format;
   picture pctfmt low-high='009.9 %';
run;
```

レポートを作成し、デフォルトのテーブルオプションを使用します。

```
proc tabulate data=customer_response;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が変数 Factor1、Factor2、Factor3、Factor4 および Customer の統計量を計算するように指定します。変数 Customer は、TABLE ステートメントで定義される Percent 列の計算に使用されるため、表示する必要があります。

```
   var factor1-factor4 customer;
```

テーブルの行と列を定義します。 TABLE ステートメントは、係数ごとに 1 行、度数カウントごとに 1 列、パーセントごとに 1 列作成します。引用符内のテキストは、対応する行または列に対するヘッダーを提供します。出力形式修飾子 F=7.と F=PCTFMT9.は関連するセルの値に対し出力形式を提供し、列幅を列ヘッダーに合うように拡張します。

```
   table factor1='Cost'
         factor2='Performance'
         factor3='Reliability'
         factor4='Sales Staff',
         (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;
```

タイトルを指定します。

```
   title 'Customer Survey Results: Spring 1996';
   title3 'Factors Influencing the Decision to Buy';
run;
```

レポートを作成し、デフォルトのテーブルオプションを使用します。

```
proc tabulate
data=customer_response;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が変数 Source1、Source2、Source3 および Customer の統計量を計算するように指定します。変数 Customer は、分母定義で表示されるため、変数リストに表示する必要があります。

```
   var source1-source3 customer;
```

テーブルの行と列を定義します。 TABLE ステートメントは、会社名のソースごとに 1 行、度数カウントごとに 1 列、パーセントごとに 1 列作成します。引用符内のテキストは、対応する行または列に対するヘッダーを提供します。

```
   table source1='TV/Radio'
         source2='Newspaper'
         source3='Word of Mouth',
         (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;
```

タイトルとフットノートを指定します。 マクロ変数 NUM は、回答者数に解決されます。FOOTNOTE ステートメントは、マクロ変数が解決するように一重引用符ではなく二重引用符を使用します。

```
    title 'Source of Company Name';
    footnote "Number of Respondents: &num";
run;
```

SAS システムオプションをリセットします。 FORMDLIM=オプションは、ページ区切りをページ排出にリセットします。NUMBER オプションは、後続ページでのページ番号の表示を再開します。

```
options formdlim='' number;
```

出力

アウトプット 64.24 *顧客調査結果:1996 年春*

Customer Survey Results: Spring 1996

Factors Influencing the Decision to Buy

	Count	Percent
Cost	87	72.5 %
Performance	62	51.6 %
Reliability	30	25.0 %
Sales Staff	120	100.0 %

Source of Company Name

	Count	Percent
TV/Radio	92	76.6 %
Newspaper	69	57.5 %
Word of Mouth	26	21.6 %

Number of Respondents: 120

例 11: 複数選択式の調査データに基づくレポート作成

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメント
- N 統計量

他の要素: DATA ステップ
FORMAT プロシジャ
FORMAT statement
TRANSPOSE プロシジャ
データセットオプション
RENAME=
TITLE statement

データセット: RADIO

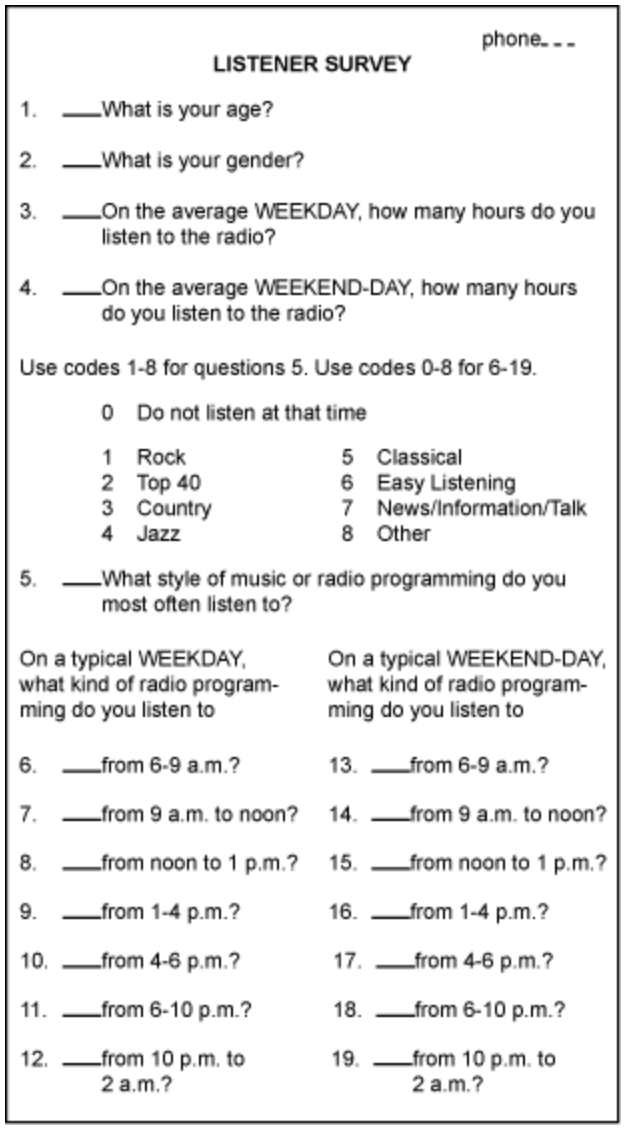
詳細

視聴者の好みに関するこのレポートには、通常の平日の 7 つの各期間に各種の番組を選択する視聴者数を示します。データは調査によって収集され、結果は SAS データセットに保存されています。このデータセットにはこのレポートに必要なすべての情報が含まれていますが、その情報は PROC TABULATE が使用できるようには調整されていません。

時間帯とラジオ番組選択のクロス集計表を作成するには、時間帯に対する変数と番組の好みに対する変数を含むデータセットがなければなりません。PROC TRANSPOSE は、データをこれらの変数を含む新しいデータセットに再形成します。データが適切な形式になると、PROC TABULATE はレポートを作成します。

次の図に、データの収集に使用される調査用紙を示します。

図 64.12 記入式調査用紙



phone_ _ _

LISTENER SURVEY

1. ___What is your age?

2. ___What is your gender?

3. ___On the average WEEKDAY, how many hours do you listen to the radio?

4. ___On the average WEEKEND-DAY, how many hours do you listen to the radio?

Use codes 1-8 for questions 5. Use codes 0-8 for 6-19.

0 Do not listen at that time

1	Rock	5	Classical
2	Top 40	6	Easy Listening
3	Country	7	News/Information/Talk
4	Jazz	8	Other

5. ___What style of music or radio programming do you most often listen to?

On a typical WEEKDAY, what kind of radio programming do you listen to	On a typical WEEKEND-DAY, what kind of radio programming do you listen to
6. ___from 6-9 a.m.?	13. ___from 6-9 a.m.?
7. ___from 9 a.m. to noon?	14. ___from 9 a.m. to noon?
8. ___from noon to 1 p.m.?	15. ___from noon to 1 p.m.?
9. ___from 1-4 p.m.?	16. ___from 1-4 p.m.?
10. ___from 4-6 p.m.?	17. ___from 4-6 p.m.?
11. ___from 6-10 p.m.?	18. ___from 6-10 p.m.?
12. ___from 10 p.m. to 2 a.m.?	19. ___from 10 p.m. to 2 a.m.?

外部場イルには (2173 ページ) 調査に対する生データが含まれています。そのファイルからの複数行がここに表示されます。

```
967 32 f 5 3 5 7 5 5 5 7 0 0 0 8 7 0 0 8 0 781 30 f 2 3 5 5 0 0 0 5 0 0 0 4 7 5 0 0 0 85
```

プログラム

```
data radio;
   infile 'input-file' missover;

   input /(Time1-Time7) ($1. +1);
   listener=_n_;
run;

proc format;
   value $timefmt 'Time1'='6-9 a.m.'
                  'Time2'='9 a.m. to noon'
```

```
                  'Time3'='noon to 1 p.m.'
                  'Time4'='1-4 p.m.'
                  'Time5'='4-6 p.m.'
                  'Time6'='6-10 p.m.'
                  'Time7'='10 p.m. to 2 a.m.'
                    other='*** Data Entry Error ***';
   value $pgmfmt      '0'="Don't Listen"
                  '1','2'='Rock and Top 40'
                     '3'='Country'
              '4','5','6'='Jazz, Classical, and Easy Listening'
                     '7'='News/ Information /Talk'
                     '8'='Other'
                    other='*** Data Entry Error ***';
run;

proc transpose data=radio
               out=radio_transposed(rename=(col1=Choice))
               name=Timespan;
   by listener;
   var time1-time7;
run;

proc tabulate data=radio_transposed format=12.;

format timespan $timefmt. choice $pgmfmt.;

   class timespan choice;

   table timespan='Time of Day',
         choice='Choice of Radio Program'*n='Number of Listeners';

   title 'Listening Preferences on Weekdays';
run;
```

プログラムの説明

RADIO データセットを作成し、入力ファイルを指定します。 RADIO には、336 人の視聴者の調査からのデータが含まれています。データセットには、視聴者と、視聴者のラジオ番組の好みに関する情報が含まれています。INFILE ステートメントは、そのデータを含む外部ファイルを指定します。MISSOVER は、INPUT ステートメントに表示されるすべての変数に対する現在の行に値が見つからない場合、入力ポインタが次のレコードに移動しないようにします。

```
data radio;
   infile 'input-file' missover;

   input /(Time1-Time7) ($1. +1);
   listener=_n_;
run;
```

$TIMEFMT.出力形式と$PGMFMT.出力形式を作成します。 PROC FORMAT は、時間帯用と番組選択用の出力形式を作成します。

```
proc format;
   value $timefmt 'Time1'='6-9 a.m.'
                  'Time2'='9 a.m. to noon'
                  'Time3'='noon to 1 p.m.'
                  'Time4'='1-4 p.m.'
                  'Time5'='4-6 p.m.'
```

```
                    'Time6'='6-10 p.m.'
                    'Time7'='10 p.m. to 2 a.m.'
                      other='*** Data Entry Error ***';
   value $pgmfmt      '0'="Don't Listen"
                    '1','2'='Rock and Top 40'
                      '3'='Country'
                '4','5','6'='Jazz, Classical, and Easy Listening'
                      '7'='News/ Information /Talk'
                      '8'='Other'
                      other='*** Data Entry Error ***';
run;
```

RADIO データセットを転置して、データを再形成します。 PROC TRANSPOSE は、RADIO_TRANSPOSED を作成します。このデータセットには、元のデータセットからの変数 Listener が含まれています。また、2 つの転置された変数、Timespan と Choice も含まれています。Timespan には、出力データセットのオブザベーションを形成するために転置される入力データセットからの変数名(Time1-Time7)が含まれています。選択には、これらの変数の値が含まれています(PROC TRANSPOSE ステップの説明については、“詳細” (1979 ページ)を参照)。

```
proc transpose data=radio
               out=radio_transposed(rename=(col1=Choice))
               name=Timespan;
   by listener;
   var time1-time7;
run;
```

レポートを作成し、テーブルオプションを指定します。 FORMAT=オプションは、各テーブルセルの値に対するデフォルトの出力形式を指定します。

```
proc tabulate data=radio_transposed format=12.;
```

転置された変数をフォーマットします。 FORMAT ステートメントは、これらの出力形式を出力データセットの変数と常に関連付けます。

```
format timespan $timefmt. choice $pgmfmt.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Timespan と Choice を分類変数として識別します。

```
   class timespan choice;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Timespan のフォーマットされた値ごとに 1 行、Choice のフォーマットされた値ごとに 1 列を作成します。各列には、N 統計量の値が表示されます。引用符内のテキストは、対応する行または列に対するヘッダーを提供します。

```
   table timespan='Time of Day',
         choice='Choice of Radio Program'*n='Number of Listeners';
```

タイトルを指定します。

```
   title 'Listening Preferences on Weekdays';
run;
```

出力

アウトプット 64.25 平日の視聴傾向

Listening Preferences on Weekdays

	Choice of Radio Program					
	Don't Listen	Rock and Top 40	Country	Jazz, Classical, and Easy Listening	News/ Information /Talk	Other
	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners
Time of Day						
6-9 a.m.	34	143	7	39	96	17
9 a.m. to noon	214	59	5	51	3	4
noon to 1 p.m.	238	55	3	27	9	4
1-4 p.m.	216	60	5	50	2	3
4-6 p.m.	56	130	6	57	69	18
6-10 p.m.	202	54	9	44	20	7
10 p.m. to 2 a.m.	264	29	3	36	2	2

詳細

データを再形成する

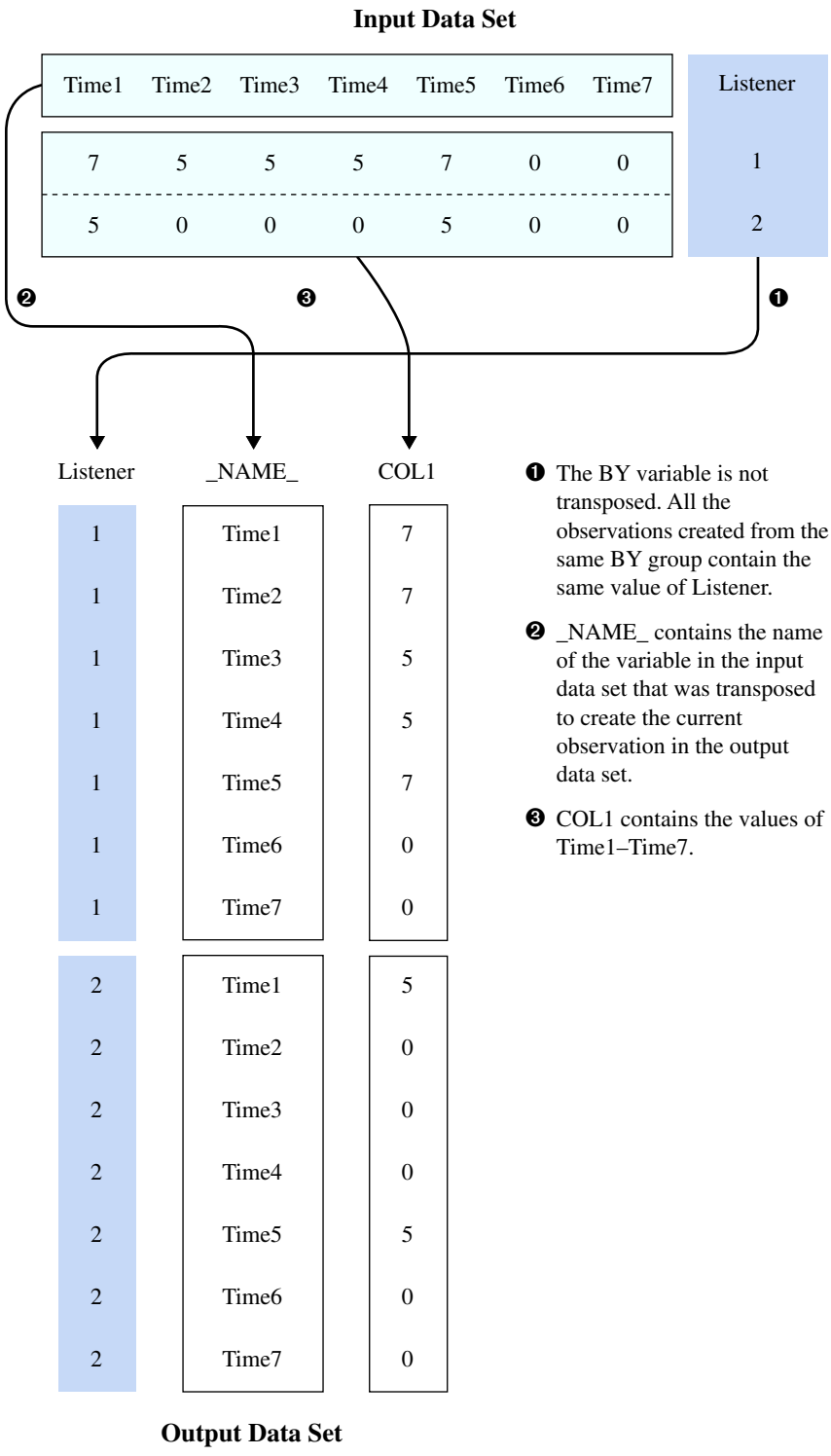
元の入力データセットには、クロス集計表の作成に必要なすべての情報が含まれていますが、PROC TABULATE はその形式の情報を使用できません。PROC TRANSPOSE は、新しいデータセットの各オブザベーションに変数 Listener、時間帯に対する変数、番組の好みに対する変数が含まれるように、データを再配置します。次の図に、転置について説明します。PROC TABULATE はこの新しいデータセットを使用して、クロス集計表レポートを作成します。

PROC TRANSPOSE は、1 つのオブザベーションに保存された値が 1 つの変数に書き込まれるように、データを再構築します。転置する変数を指定できます。

この例のように BY 処理を使用して転置する場合、転置する変数ごとに 1 つのオブザベーションを各 BY グループから作成します。この例では、Listener が BY 変数です。入力データセットの各オブザベーションは、BY グループです。Listener の値が各オブザベーションに対して一意であるためです。

この例では、Time1 から Time7 までの 7 つの変数を転置します。そのため、出力データセットには、入力データセットの各 BY グループ(各オブザベーション)からの 7 つのオブザベーションが含まれます。

図 64.13 2 つのオブザベーションの転置



PROC TRANSPOSE ステップについて

Here is a detailed explanation of the PROC TRANSPOSE step that reshapes the data:

```
proc transpose data=radio    1
               out=radio_transposed(rename=(col1=Choice))    2
```

```
                name=Timespan;    3
   by listener;    4
   var time1-time7;    5
   format timespan $timefmt. choice $pgmfmt.;    6
run;
```

1 DATA=オプションは、入力データセットを指定します。

2 OUT=オプションは、出力データセットを指定します。RENAME=データセットオプションは、配置変数の名前を COL1 (デフォルト名)から Choice に変更します。

3 NAME=オプションは、現在のオブザベーションを作成するために転置中の変数の名前を含む出力データセットの変数の名前を指定します。デフォルトでは、この変数の名前は _NAME_ です。

4 BY ステートメントは Listener を BY 変数として識別します。

5 VAR ステートメントは Time1 から Time7 までを転置する変数として識別します。

6 FORMAT ステートメントは出力形式を Timespan と Choice に割り当てます。レポートを作成する PROC TABULATE ステップは Timespan と Choice をフォーマットする必要はありません。出力形式はこれらの変数とともに保存されるためです。

例 12: さまざまなパーセント表示の統計量の計算

要素: CLASS statement

PROC TABULATE ステートメントオプション
- FORMAT=

TABLE ステートメントオプション
- ALL 分類変数
- COLPCTSUM 統計量
- 連結(ブランク)演算子
- クロス(*)演算子
- 出力形式修飾子
- 要素グループ化(かっこ)演算子
- ラベル
- REPPCTSUM 統計量
- ROWPCTSUM 統計量
- 変数リスト
- ROW=FLOAT
- RTS=

VAR ステートメント

他の要素: DATA ステップ

FORMAT プロシジャ

TITLE statement

詳細

この例では、COLPCTSUM、REPPCTSUM および ROWPCTSUM の 3 つのパーセント合計統計量の使用方法を示します。

プログラム

```
data fundrais;
```

```
   length name $ 8 classrm $ 1;
   input @1 team $ @8 classrm $ @10 name $
         @19 pencils @23 tablets;
   sales=pencils + tablets;
   datalines;
BLUE   A ANN       4   8
RED    A MARY      5  10
GREEN  A JOHN      6   4
RED    A BOB       2   3
BLUE   B FRED      6   8
GREEN  B LOUISE   12   2
BLUE   B ANNETTE   .   9
RED    B HENRY     8  10
GREEN  A ANDREW    3   5
RED    A SAMUEL   12  10
BLUE   A LINDA     7  12
GREEN  A SARA      4   .
BLUE   B MARTIN    9  13
RED    B MATTHEW   7   6
GREEN  B BETH     15  10
RED    B LAURA     4   3
;

proc format;
   picture pctfmt low-high='009 %';
run;

title "Fundraiser Sales";

proc tabulate format=7.;

   class team classrm;

   var sales;

   table (team all),

         classrm='Classroom'*sales=' '*(sum
         colpctsum*f=pctfmt9.
         rowpctsum*f=pctfmt9.
         reppctsum*f=pctfmt9.)
         all*sales*sum=' '

         /rts=20;
run;
```

プログラムの説明

FUNDRAIS データセットを作成します。 FUNDRAIS には、基金活動での学生の売上に関するデータが含まれています。DATA ステップは、データセットを作成します。

```
data fundrais;
   length name $ 8 classrm $ 1;
   input @1 team $ @8 classrm $ @10 name $
         @19 pencils @23 tablets;
   sales=pencils + tablets;
   datalines;
BLUE   A ANN       4   8
RED    A MARY      5  10
```

```
GREEN  A JOHN      6   4
RED    A BOB       2   3
BLUE   B FRED      6   8
GREEN  B LOUISE   12   2
BLUE   B ANNETTE   .   9
RED    B HENRY     8  10
GREEN  A ANDREW    3   5
RED    A SAMUEL   12  10
BLUE   A LINDA     7  12
GREEN  A SARA      4   .
BLUE   B MARTIN    9  13
RED    B MATTHEW   7   6
GREEN  B BETH     15  10
RED    B LAURA     4   3
;
```

PCTFMT.出力形式を作成します。 FORMAT プロシジャは、パーセント用の出力形式を作成します。PCTFMT.出力形式は、少なくとも 1 つの桁、ブランク、パーセント記号を含むすべての値を書き込みます。

```
proc format;
   picture pctfmt low-high='009 %';
run;
```

タイトルを指定します。

```
title "Fundraiser Sales";
```

レポートを作成し、テーブルオプションを指定します。 FORMAT=オプションは、最大 7 桁を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate format=7.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Team と Classrm を分類変数として識別します。

```
   class team classrm;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Sales 変数の統計量を計算するように指定します。

```
   var sales;
```

テーブル行を定義します。 TABLE ステートメントの行次元は、Team のフォーマットされた値に対しそれぞれ行を作成します。レポートの最終行には、すべてのチームの売上が要約されます。

```
   table (team all),
```

テーブル列を定義します。 TABLE ステートメントの列次元は、Classrm のフォーマットされた値に対しそれぞれ列を作成します。Classrm の各値内でクロスするのは、ブランクのラベルを含む分析変数(**sales**)です。各列内でネストされるのは、クラスの売上を要約する列です。**sum** とラベル付けされている最初のネストされた列は、クラスルーム行の売上合計です。**ColPctSum** とラベル付けされている 2 番目のネストされた列は、クラスルームのすべてのチームの売上合計に対するクラスルーム行の売上合計です。**RowPctSum** とラベル付けされている 3 番目のネストされた列は、すべてのクラスルーム行の売上合計に対するクラスルーム行の売上合計のパーセントです。**RepPctSum** とラベル付けされている 4 番目のネストされた列は、すべてのクラスルームのすべて

のチームの売上合計に対するクラスルーム行の売上合計のパーセントです。レポートの最終列に、すべてのクラスルーム行の売上が要約されます。

```
            classrm='Classroom'*sales=' '*(sum
            colpctsum*f=pctfmt9.
            rowpctsum*f=pctfmt9.
            reppctsum*f=pctfmt9.)
            all*sales*sum=' '
```

行タイトルのスペースを指定し、ブランクの行ヘッダーを削除します。 RTS=は、行ヘッダーに対する行ごとに 20 文字を提供します。

```
            /rts=20;
   run;
```

出力

アウトプット 64.26 *基金活動の売上*

Fundraiser Sales

	Classroom								All
	A				B				
	Sum	ColPctSum	RowPctSum	RepPctSum	Sum	ColPctSum	RowPctSum	RepPctSum	sales
team									
BLUE	31	34 %	46 %	15 %	36	31 %	53 %	17 %	67
GREEN	18	19 %	31 %	8 %	39	34 %	68 %	19 %	57
RED	42	46 %	52 %	20 %	38	33 %	47 %	18 %	80
All	91	100 %	44 %	44 %	113	100 %	55 %	55 %	204

詳細

次に、クラスルーム A の Blue チーム用の出力の生成に使用されるパーセント合計統計量計算を示します。

- COLPCTSUM=31/91*100=34%
- ROWPCTSUM=31/67*100=46%
- REPPCTSUM=31/204*100=15%

同様の計算が、その他のチームとクラスルーム用の出力の生成に使用されます。

例 13: 分母定義を使用し、基本的な度数カウントとパーセントを表示する

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメントオプション
- ALL 分類変数
- 分母定義(山かっこ演算子)

N 統計量
PCTN 統計量
RTS=

他の要素: DATA ステップ
FORMAT プロシジャ
FORMAT statement
TITLE statement

詳細

クロス集計表テーブル(分割表、スタブアンドバナーレポートとも呼ばれます)には、2 つ以上の変数に対して組み合わされた度数分布が表示されます。このテーブルには、4 つの職階における女性と男性に対する度数カウントがそれぞれ表示されます。度数カウントがそれぞれ次を表すパーセントも表示されます。

- 職階における女性と男性の合計(行パーセント)
- すべての職階における性別の合計(列パーセント)
- すべての従業員の合計

プログラム

```
data jobclass;
   input Gender Occupation @@;
   datalines;
1 1  1 1  1 1  1 1  1 1  1 1  1 1
1 2  1 2  1 2  1 2  1 2  1 2  1 2
1 3  1 3  1 3  1 3  1 3  1 3  1 3
1 1  1 1  1 1  1 2  1 2  1 2  1 2
1 2  1 2  1 3  1 3  1 4  1 4  1 4
1 4  1 4  1 4  1 1  1 1  1 1  1 1
1 1  1 2  1 2  1 2  1 2  1 2  1 2
1 2  1 3  1 3  1 3  1 3  1 4  1 4
1 4  1 4  1 4  1 1  1 3  2 1  2 1
2 1  2 1  2 1  2 1  2 1  2 2  2 2
2 2  2 2  2 2  2 3  2 3  2 3  2 4
2 4  2 4  2 4  2 4  2 4  2 1  2 3
2 3  2 3  2 3  2 3  2 4  2 4  2 4
2 4  2 4  2 1  2 1  2 1  2 1  2 1
2 2  2 2  2 2  2 2  2 2  2 2  2 2
2 3  2 3  2 4  2 4  2 4  2 1  2 1
2 1  2 1  2 1  2 2  2 2  2 2  2 3
2 3  2 3  2 3  2 4
;

proc format;
   value gendfmt 1='Female'
                 2='Male'
              other='*** Data Entry Error ***';
   value occupfmt 1='Technical'
                  2='Manager/Supervisor'
                  3='Clerical'
                  4='Administrative'
               other='*** Data Entry Error ***';
run;
```

```
proc tabulate data=jobclass format=8.2;

   class gender occupation;

    table (occupation='Job Class' all='All Jobs')
              *(n='Number of employees'*f=9.
              pctn<gender all>='Percent of row total'
              pctn<occupation all>='Percent of column total'
              pctn='Percent of total'),

    gender='Gender' all='All Employees'/ rts=50;

   format gender gendfmt. occupation occupfmt.;

   title 'Gender Distribution';
   title2 'within Job Classes';
run;
```

プログラムの説明

JOBCLASS データセットを作成します。JOBCLASS には、架空の会社の従業員の性別と職階に関するエンコード情報が含まれています。

```
data jobclass;
   input Gender Occupation @@;
   datalines;
1 1  1 1  1 1  1 1  1 1  1 1  1 1
1 2  1 2  1 2  1 2  1 2  1 2  1 2
1 3  1 3  1 3  1 3  1 3  1 3  1 3
1 1  1 1  1 1  1 2  1 2  1 2  1 2
1 2  1 2  1 3  1 3  1 4  1 4  1 4
1 4  1 4  1 4  1 1  1 1  1 1  1 1
1 1  1 2  1 2  1 2  1 2  1 2  1 2
1 2  1 3  1 3  1 3  1 3  1 4  1 4
1 4  1 4  1 4  1 1  1 3  2 1  2 1
2 1  2 1  2 1  2 1  2 1  2 2  2 2
2 2  2 2  2 2  2 3  2 3  2 3  2 4
2 4  2 4  2 4  2 4  2 4  2 1  2 3
2 3  2 3  2 3  2 3  2 4  2 4  2 4
2 4  2 4  2 1  2 1  2 1  2 1  2 1
2 2  2 2  2 2  2 2  2 2  2 2  2 2
2 3  2 3  2 4  2 4  2 4  2 1  2 1
2 1  2 1  2 1  2 2  2 2  2 2  2 3
2 3  2 3  2 3  2 4
;
```

GENDFMT.出力形式と OCCUPFMT.出力形式を作成します。PROC FORMAT は、変数 Gender と Occupation に対し出力形式を作成します。

```
proc format;
   value gendfmt 1='Female'
                 2='Male'
              other='*** Data Entry Error ***';
   value occupfmt 1='Technical'
                  2='Manager/Supervisor'
                  3='Clerical'
                  4='Administrative'
               other='*** Data Entry Error ***';
```

```
run;
```

レポートを作成し、テーブルオプションを指定します。 FORMAT=オプションは、8.2 出力形式を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=jobclass format=8.2;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Gender と Occupation を分類変数として識別します。

```
class gender occupation;
```

```
table (occupation='Job Class' all='All Jobs')
          *(n='Number of employees'*f=9.
          pctn<gender all>='Percent of row total'
          pctn<occupation all>='Percent of column total'
          pctn='Percent of total'),
```

テーブル列を定義して、行ヘッダーのスペース量を指定します。 列次元は、Gender のフォーマットされた値ごとに 1 列、全従業員に対し 1 列を作成します。引用符内のテキストは、対応する列に対してヘッダーを提供します。RTS=オプションは、行ヘッダーに対する行ごとに 50 文字を提供します。

```
gender='Gender' all='All Employees'/ rts=50;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Gender と Occupation に割り当てます。

```
format gender gendfmt. occupation occupfmt.;
```

タイトルを指定します。

```
title 'Gender Distribution';
title2 'within Job Classes';
run;
```

出力

アウトプット 64.27 *職階内の性別分布*

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

詳細

概要

テーブルの行を定義する TABLE ステートメントの部分には PCTN 統計量が使用され、3 つの異なるパーセントが計算されます。

PCTN のすべての計算では、分子は N で、テーブルの 1 つのセルに対する度数カウントです。PCTN の発生ごとの分母は、分母定義によって決定されます。分母定義は、

キーワード PCTN の後に山かっこで囲まれて表示されます。これは、1 つ以上の式のリストです。リストは、分母に対して合計する度数カウントを PROC TABULATE に認識させます。

テーブル構造を分析する

テーブルの構造をよく見ると、PROC TABULATE が分母定義をどのように使用するかがわかります。次の TABLE ステートメントの簡略バージョンで、テーブルの基本構造について説明します。

```
table occupation='Job Class' all='All Jobs',
      gender='Gender' all='All Employees';
```

テーブルは、4 つのサブテーブルの組み合わせです。このレポートでは、サブテーブルはそれぞれ行次元の 1 つの分類変数と列次元の 1 つの分類変数のクロスです。クロスはそれぞれ 1 つ以上のカテゴリを作成します。カテゴリは **`female, technical`**、**`all, clerical`** などの分類変数の一意の値の組み合わせです。

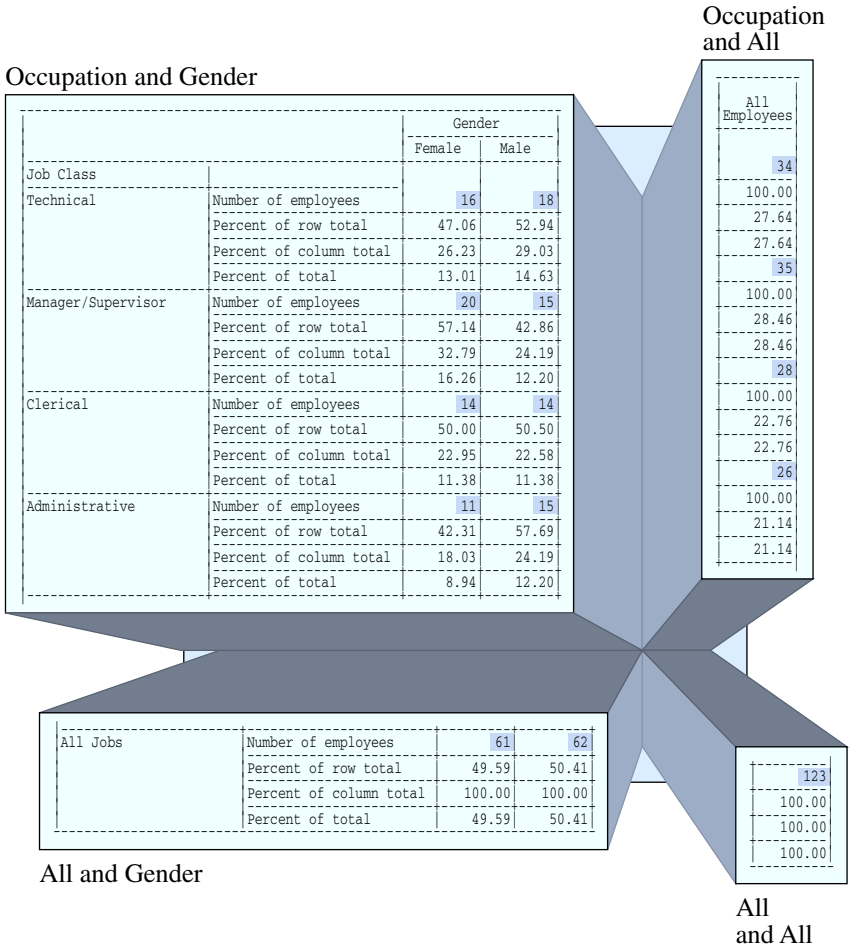
次のテーブルでは、それぞれのサブテーブルについて説明します。

表 64.8 *サブテーブルのコンテンツ*

サブテーブルに影響する分類変数	度数カウントの説明	カテゴリ名
Occupation と Gender	各職における女性数または各職における男性数	8
All と Gender	女性数または男性数	2
Occupation と All	各職の人数	4
All と All	すべての職の人数	1

次の図では、これらのサブテーブル、各カテゴリに対する度数カウントがハイライト表示されています。

図 64.14 4 つのサブテーブルの説明



Occupation and Gender

Job Class		Gender	
		Female	Male
Technical	Number of employees	16	18
	Percent of row total	47.06	52.94
	Percent of column total	26.23	29.03
	Percent of total	13.01	14.63
Manager/Supervisor	Number of employees	20	15
	Percent of row total	57.14	42.86
	Percent of column total	32.79	24.19
	Percent of total	16.26	12.20
Clerical	Number of employees	14	14
	Percent of row total	50.00	50.50
	Percent of column total	22.95	22.58
	Percent of total	11.38	11.38
Administrative	Number of employees	11	15
	Percent of row total	42.31	57.69
	Percent of column total	18.03	24.19
	Percent of total	8.94	12.20

Occupation and All

All Employees
34
100.00
27.64
27.64
35
100.00
28.46
28.46
28
100.00
22.76
22.76
26
100.00
21.14
21.14

All and Gender

All Jobs	Number of employees	61	62
	Percent of row total	49.59	50.41
	Percent of column total	100.00	100.00
	Percent of total	49.59	50.41

All and All

123
100.00
100.00
100.00

分母定義を解釈する

TABLE ステートメントの次のフラグメントは、このレポートに対する分母定義を定義します。PCTN キーワードと分母定義がハイライト表示されています。

```
table (occupation='Job Class' all='All Jobs')
            *(n='Number of employees'*f=5.
              pctn<gender all>='Row percent'
              pctn<occupation all>='Column percent'
              pctn='Percent of total'),
```

PCTN を使用するたびに Occupation と All の各値内の統計量の行がネストされます。各分母定義は、その行の分母に対して合計する度数カウントを PROC TABULATE に認識させます。このセクションでは、これらの分母定義を PROC TABULATE がどのように解釈するかについて説明します。

行のパーセント

The part of the TABLE statement that calculates the row percentages and that labels the row is

```
pctn<gender all>='Row percent'
```

各サブテーブルに対するこの分母定義を PROC TABULATE がどのように解釈するかを考えます。

アウトプット 64.28 サブテーブル 1:Occupation と Gender

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は、分母定義の最初の要素 Gender を参照し、Gender がサブテーブルに影響するかどうかを確認します。Gender がサブテーブルに影響しているため、PROC TABULATE は Gender を分母定義として使用します。この分母定義は、PROC TABULATE に対し、Occupation の同じ値内の Gender のすべての発生に対する度数カウントを合計するように命令します。

たとえば、カテゴリ **female, technical** の分母は、Occupation の値が **technical** であるこのサブテーブルのすべてのカテゴリに対するすべての度数カウントの合計です。そのようなカテゴリには、**female, technical** と **male, technical** の 2 つがあります。対応する度数カウントは 16 と 18 です。そのため、このカテゴリの分母は 16+18、または 34 となります。

アウトプット 64.29 *サブテーブル 2:All と Gender*

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は、分母定義の最初の要素 Gender を参照し、Gender がサブテーブルに影響するかどうかを確認します。Gender がサブテーブルに影響しているため、PROC TABULATE は Gender を分母定義として使用します。この分母定義は、PROC TABULATE に対し、サブテーブルの Gender のすべての発生に対する度数カウントを合計するように命令します。

たとえば、カテゴリ **all, female** の分母は、**all, female** と **all, male** に対する度数カウントの合計です。対応する度数カウントは 61 と 62 です。そのため、このサブテーブルのセルの分母は、61+62 または 123 になります。

アウトプット 64.30 サブテーブル 3:Occupation と All

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は、分母定義の最初の要素 Gender を参照し、Gender がサブテーブルに影響するかどうかを確認します。Gender がサブテーブルに影響しないため、PROC TABULATE は分母定義の次の要素、All を参照します。変数 All はこのサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。All は、カテゴリを 1 つだけ含む予約分類変数です。そのため、この分母定義は、PROC TABULATE に対し、All の度数カウントを分母として使用するように命令します。

たとえば、カテゴリ `clerical, all` の分母はそのカテゴリに対する度数カウント、28 となります。

注: これらのテーブルセルでは、分子と分母が同じため、このサブテーブルの行のパーセントはすべて 100 になります。

アウトプット 64.31 サブテーブル 4:All と All

Gender Distribution
within Job Classes

		Gender		All
		Female	Male	Employees
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は、分母定義の最初の要素 Gender を参照し、Gender がサブテーブルに影響するかどうかを確認します。Gender がサブテーブルに影響しないため、PROC TABULATE は分母定義の次の要素、All を参照します。変数 All はこのサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。All は、カテゴリを 1 つだけ含む予約分類変数です。そのため、この分母定義は、PROC TABULATE に対し、All の度数カウントを分母として使用するように命令します。

このサブテーブルには、カテゴリ `all, all` のみがあります。このカテゴリの分母は 123 となります。

注: このテーブルセルでは、分子と分母が同じため、このサブテーブルの行のパーセントは 100 となります。

列のパーセント

The part of the TABLE statement that calculates the column percentages and labels the row is

```
pctn<occupation all>='Column percent'
```

各サブテーブルに対するこの分母定義を PROC TABULATE がどのように解釈するかを考えます。

アウトプット 64.32 サブテーブル 1:Occupation と Gender

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は分母定義の最初の要素 Occupation を参照し、Occupation がサブテーブルに影響するかどうかを確認します。Occupation がサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。この分母定義は、PROC TABULATE に対し、Gender の同じ値内の Occupation のすべての発生に対する度数カウントを合計するように命令します。

たとえば、カテゴリ **manager/supervisor, male** の分母は、Gender の値が **male** であるこのサブテーブルのすべてのカテゴリに対するすべての度数カウントの合計です。そのようなカテゴリには、**technical, male**、**manager/supervisor, male**、**clerical, male**、**administrative, male** の 4 つがあります。対応する度数カウントは 18、15、14、15 です。そのため、このカテゴリの分母は、18+15+14+15 または 62 となります。

アウトプット 64.33 *サブテーブル 2:All と Gender*

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は分母定義の最初の要素 Occupation を参照し、Occupation がサブテーブルに影響するかどうかを確認します。Occupation がサブテーブルに影響しないため、PROC TABULATE は分母定義の次の要素、All を参照します。変数 All がこのサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。All は、カテゴリを 1 つだけ含む予約分類変数です。そのため、この分母定義は、PROC TABULATE に対し、All の度数カウントを分母として使用するように命令します。

たとえば、カテゴリ **all, female** の分母はそのカテゴリに対する度数カウント、61 となります。

注: これらのテーブルセルでは、分子と分母が同じため、このサブテーブルの列のパーセントはすべて 100 となります。

アウトプット 64.34 サブテーブル 3:Occupation と All

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は分母定義の最初の要素 Occupation を参照し、Occupation がサブテーブルに影響するかどうかを確認します。Occupation がサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。この分母定義は、PROC TABULATE に対し、サブテーブルの Occupation のすべての発生に対する度数カウントを合計するように命令します。

たとえば、カテゴリ **technical, all** の分母は **technical, all**、**manager/supervisor, all**、**clerical, all**、**administrative, all** に対する度数カウントの合計です。対応する度数カウントは 34、35、28、26 です。そのため、このカテゴリの分母は、34+35+28+26 または 123 となります。

アウトプット 64.35 *サブテーブル 4:All と All*

Gender Distribution
within Job Classes

		Gender		All
		Female	Male	Employees
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は分母定義の最初の要素 Occupation を参照し、Occupation がサブテーブルに影響するかどうかを確認します。Occupation がサブテーブルに影響しないため、PROC TABULATE は分母定義の次の要素、All を参照します。変数 All がこのサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。All は、カテゴリを 1 つだけ含む予約分類変数です。そのため、この分母定義は、PROC TABULATE に対し、All の度数カウントを分母として使用するように命令します。

このサブテーブルには、カテゴリ `all, all` のみがあります。このカテゴリの分母は 123 となります。

注: この計算では、分子と分母が同じため、このサブテーブルの列のパーセントは 100 となります。

合計パーセント

合計パーセントを計算し、行にラベル付けする TABLE ステートメントの一部は、次のとおりです。

```
pctn='Total percent'
```

分母定義を指定しない場合、PROC TABULATE はサブテーブルのすべての度数カウントを総計することによりセルに対する分母を取得します。次のテーブルに、この例のすべてのサブテーブルに対するプロセスを要約します。

表 64.9 *合計パーセントの分母*

サブテーブルに影響する分類変数	度数カウント	合計
Occupant と Gender	16, 18, 20, 15 14, 14, 11, 15	123

サブテーブルに影響する分類変数	度数カウント	合計
Occupant と All	34, 35, 28, 26	123
Gender と All	61, 62	123
All と All	123	123

結果として、合計パーセントの合計の分母は常に 123 になります。

例 14: ODS 出力にスタイル要素を指定する

要素: CLASS ステートメントオプション
STYLE=
CLASSLEV ステートメントオプション
STYLE=
KEYLABEL ステートメント
KEYWORD ステートメントオプション
STYLE=
PROC TABULATE ステートメントオプション
DATA=
STYLE=
TABLE ステートメントオプション
STYLE=
MISSTEXT=
BOX=

他の要素: ODS HTML ステートメント
ODS HTML CLOSE ステートメント
ODS PDF ステートメント
ODS PDF CLOSE ステートメント
ODS RTF ステートメント
ODS RTF CLOSE ステートメント
OPTIONS ステートメント
TITLE statement

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、HTML、RTF および PDF ファイルを作成して、さまざまなテーブル部分に対しスタイル要素を指定します。

プログラム

```
options nodate pageno=1;

ods html body='external-HTML-file';
ods pdf file='external-PDF-file';
ods rtf file='external-RTF-file';

proc tabulate data=energy style=[fontweight=bold];

   class region division type / style=[textalign=center];

   classlev region division type / style=[textalign=left];

   var expenditures / style=[fontsize=3];

   keyword all sum / style=[fontwidth=wide];
   keylabel all="Total";

   table (region all)*(division all*[style=[backgroundcolor=yellow]]),
         (type all)*(expenditures*f=dollar10.) /
         style=[bordercolor=blue]

         misstext=[label="Missing" style=[fontweight=light]]

         box=[label="Region by Division by Type"
            style=[fontstyle=italic]];

   format region regfmt. division divfmt. type usetype.;

   title 'Energy Expenditures';
   title2 '(millions of dollars)';
run;

ods html close;
ods pdf close;
ods rtf close;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=とPAGESIZE=は、HTML、RTF および Printer 出力に影響しないため、この例では設定しません。

```
options nodate pageno=1;
```

ODS 出力ファイル名を指定します。 複数の ODS 出力先を開くことによって、一度の実行で複数の出力ファイルを作成できます。ODS HTML ステートメントは、HTML で書き込まれる出力を生成します。ODS PDF ステートメントは出力を Portable Document Format (PDF)で作成します。ODS RTF ステートメントは出力を Rich Text Format (RTF)で作成します。PROC TABULATE からの出力は、これらのファイルにそれぞれ移動します。

```
ods html body='external-HTML-file';
ods pdf file='external-PDF-file';
ods rtf file='external-RTF-file';
```

テーブルオプションを指定します。 PROC TABULATE ステートメントの STYLE=オプションは、テーブルのデータセルに対しスタイル要素を指定します。

```
proc tabulate data=energy style=[fontweight=bold];
```

分析対象となるサブグループを指定します。 CLASS ステートメントの STYLE=オプションは、分類変数名ヘッダーに対しスタイル要素を指定します。

```
class region division type / style=[textalign=center];
```

分類変数値ヘッダーに対しスタイル属性を指定します。 CLASSLEV ステートメントの STYLE=オプションは、分類変数水準値ヘッダーに対しスタイル要素を指定します。

```
classlev region division type / style=[textalign=left];
```

分析変数と、そのスタイル属性を指定します。 VAR ステートメントの STYLE=オプションは、変数名ヘッダーに対しスタイル要素を指定します。

```
var expenditures / style=[fontsize=3];
```

キーワードに対しスタイル属性を指定し、"all"キーワードにラベル付けします。 KEYWORD ステートメントの STYLE=オプションは、キーワードに対しスタイル要素を指定します。KEYLABEL ステートメントは、ラベルをキーワードに割り当てます。

```
keyword all sum / style=[fontwidth=wide];
keylabel all="Total";
```

テーブル行とテーブル列、およびそのスタイル属性を定義します。 次元式の STYLE=オプションは、テーブルセルの属性を指定する PROC TABULATE のその他の STYLE=指定に優先します。スラッシュ(/)の後の STYLE=オプションは、テーブルセル以外のテーブルの部分に対し属性を指定します。

```
table (region all)*(division all*[style=[backgroundcolor=yellow]]),
      (type all)*(expenditures*f=dollar10.) /
      style=[bordercolor=blue]
```

欠損値を含むセルに対し、スタイル属性を指定します。 TABLE ステートメントの MISSTEX オプションの STYLE=オプションは、欠損値を含むテーブルセルのテキストに使用するスタイル要素を指定します。

```
      misstext=[label="Missing" style=[fontweight=light]]
```

行タイトルの上のボックスに対しスタイル属性を指定します。 TABLE ステートメントの BOX オプションの STYLE=オプションは、行タイトルの上のボックスのテキストに使用するスタイル要素を指定します。

```
      box=[label="Region by Division by Type"
         style=[fontstyle=italic]];
```

分類変数値をフォーマットします。 FORMAT ステートメントは、出力形式を Region、Division、Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
   title 'Energy Expenditures';
   title2 '(millions of dollars)';
run;
```

ODS 出力先をクローズします。

```
ods html close;
ods pdf close;
ods rtf close;
```

出力

アウトプット 64.36 *HTML 出力*

Energy Expenditures
(millions of dollars)

Region by Division by Type		Type		Total
		Residential Customers	Business Customers	
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
West	Division			
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$19,435	$17,348	$36,783
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$46,291	$37,555	$83,846

アウトプット 64.37 PDF 出力

Energy Expenditures
(millions of dollars)

Region by Division by Type		Type		
		Residential Customers	Business Customers	Total
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
West	Division			
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$19,435	$17,348	$36,783
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$46,291	$37,555	$83,846

アウトプット 64.38 RTF 出力

Energy Expenditures
(millions of dollars)

Region by Division by Type		Type		
		Residential Customers	Business Customers	Total
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
West	Division			
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$19,435	$17,348	$36,783
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Mountain	$5,476	$4,729	$10,205
	Pacific	$13,959	$12,619	$26,578
	Total	$46,291	$37,555	$83,846

例 15: スタイル優先

要素: CLASS ステートメント 0

CLASSLEV ステートメントオプション
STYLE=

KEYLABEL ステートメント

LABEL statement

PROC TABULATE ステートメントオプション
DATA=
FORMAT=

TABLE ステートメント
クロス(*)演算子
STYLE=
STYLE_PRECEDENCE=オプション

VAR ステートメント

他の要素: ODS HTML ステートメント
FORMAT プロシジャ
FORMAT statement
TITLE statement

データセット: SALES

詳細

この例では、次を行います。

- 地域別に、販売の種類(小売または卸売)ごとにカテゴリを作成する
- ドル出力形式をテーブルのすべてのセルに適用する
- 各地域と販売の種類に対し、イタリックのフォントスタイルを適用する
- スタイル(背景=赤、黄色、オレンジ)の色を STYLE_PRECEDENCE =オプションに基づいて適用する
- ODS HTML 出力を生成する

プログラム

```
proc format;
      value $saletypefmt 'R'='Retail'
                         'W'='WholeSale';
run;

ods html file="stylePrecedence.html";

title "Style Precedence";
title2 "First Table: no precedence, Orange";
title3 "Second Table: style_precedence=page, Red";

proc tabulate data=sales format=dollar10.;

class product region saletype;


classlev region saletype / style={font_style=italic};


var netsales;

label netsales="Net Sales";


keylabel all="Total";


table product *{style={background=red}},
region*{style={background=yellow}},
saletype*{style={background=orange}};

table product *{style={background=red}},
region*{style={background=yellow}},
saletype*{style={background=orange}} / style_precedence=page;

format saletype $saletypefmt.;


run;
```

プログラムの説明

SALETYPEFMT.出力形式を作成します。 PROC FORMAT は、SALETYPE に対し出力形式を作成します。

```
proc format;
      value $saletypefmt 'R'='Retail'
                         'W'='WholeSale';
run;
```

ODS 出力ファイル名を指定します。 ODS HTML ステートメントは、HTML で書き込まれる出力を生成します。

```
ods html file="stylePrecedence.html";
```

生成するテーブルのタイトルを指定します。 2 つのテーブルが生成されます。First Table にはスタイル優先は表示されません。Second Table には、優先する色が STYLE_PRECEDENCE オプションによって指定される色に基づいていることが表示されます。

```
title "Style Precedence";
title2 "First Table: no precedence, Orange";
title3 "Second Table: style_precedence=page, Red";
```

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR10.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=sales format=dollar10.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Product、Region、SaleType の値別に分類します。

```
class product region saletype;
```

サブグループにスタイルを指定します。 CLASSLEV ステートメントは、Region 要素と Saletype 要素にスタイルを指定します。

```
classlev region saletype / style={font_style=italic};
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Netsales 変数の統計量を計算するように指定します。

```
var netsales;
```

ラベルを指定します。 LABEL ステートメントは、Netsales 変数の名前を Net Sales に変更します。

```
label netsales="Net Sales";
```

キーラベルを指定します。 KEYLABEL ステートメントは、共通分類変数 ALL を Total にラベル付けします。

```
keylabel all="Total";
```

テーブルの行と列を定義します。 TABLE ステートメントは、ページごとに製品別のテーブルを作成します。この例では、1 つの製品、A100 があります。また、TABLE ステートメントは、Region のフォーマットされた値ごとに 1 行、SaleType のフォーマットされた値ごとに 1 列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Net Sales の合計が含まれます。次元式の STYLE=オプションは、テーブルセルに属性を指定する PROC TABULATE でのその他の STYLE=指定に優先します。この最初のテーブルでは、列次元はデフォルトで、列と関連付けられているスタイルが優先します。そのため、背景色としてオレンジがデフォルトで使用されます。

```
table product *{style={background=red}},
region*{style={background=yellow}},
saletype*{style={background=orange}};
```

STYLE_PRECEDENCE オプションを使用して、テーブル行とテーブル列を定義します。 TABLE ステートメントはページごとに製品別のテーブル、A100 を作成します。また、TABLE ステートメントは、Region のフォーマットされた値ごとに 1 行、SaleType のフォーマットされた値ごとに 1 列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Net Sales の合計が含まれます。次元式の STYLE=オプションは、テーブルセルに属性を指定する PROC TABULATE でのその他の STYLE=指定に優先します。2 番目のテーブルでは、STYLE_PRECEDENCE オプションがページ式で指定されます。そのため、背景に適用されるスタイルは赤となります。

```
table product *{style={background=red}},
region*{style={background=yellow}},
saletype*{style={background=orange}} / style_precedence=page;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を SaleType 変数に割り当てます。

```
format saletype $saletypefmt.;
```

プログラムを実行します。

```
run;
```

出力

アウトプット 64.39 *スタイル優先*

Style Precedence
First Table: no precedence, Orange
Second Table: style_precedence=page, Red

Product A100

	SaleType	
	Retail	*WholeSale*
	N	N
Region		
NC	$3	$3
TX	.	$2

Style Precedence
First Table: no precedence, Orange
Second Table: style_precedence=page, Red

Product A100

	SaleType	
	Retail	*WholeSale*
	N	N
Region		
NC	$3	$3
TX	.	$2

例 16: NOCELLMERGE オプションの使用

要素: CLASS statement

PROC TABULATE ステートメントオプション
- DATA=
- STYLE=

CLASS statement

TABLE ステートメント
- クロス(*)演算子
- STYLE=オプション
- NOCELLMERGE=オプション

他の要素: ODS HTML ステートメント

ODS HTML CLOSE ステートメント
TITLE statement

詳細

この例では、次を行います。

- セルスタイル動作が結合されたテーブルを作成する
- セルが結合されていない 2 つ目のテーブルを作成する
- 空のデータセルとフォーマットデータセルで異なるスタイルが使用されている場合にセルスタイルがどのように影響を受けるかを示す

プログラム

```
ods html file="tabstyle.html";

proc tabulate data=sashelp.class style={background=red};

class sex age;


table sex*{style={background=blue}} all, age;

title 'Data Cell Styles in Merged Cells';

run;


proc tabulate data=sashelp.class style={background=red};

class sex age;


table sex*{style={background=blue}} all, age/nocellmerge;

title1 'Data Cell Styles with NOCELLMERGE Option';

run;


ODS HTML close;
```

プログラムの説明

ODS 出力ファイル名を指定します。 ODS HTML ステートメントは、HTML で書き込まれる出力を生成します。

```
ods html file="tabstyle.html";
```

PROC TABULATE オプションを指定します。 STYLE=オプションは、テーブルのセルの背景色を赤に設定します。

```
proc tabulate data=sashelp.class style={background=red};
```

サブグループを指定します。 CLASS ステートメントは、データを性別および年齢別に分類します。

```
class sex age;
```

テーブルの行と列を定義します。 TABLE ステートメントはテーブルを作成します。次元式の STYLE=オプションは、テーブルセル属性に対する PROC TABULATE ステートメントからの STYLE=設定に優先します。

```
table sex*{style={background=blue}} all, age;
```

生成するテーブルのタイトルを指定します。 このテーブルでは、スタイル色の変更が結合されたセルにどのように影響するかを示します。

```
title 'Data Cell Styles in Merged Cells';
```

プログラムを実行します。

```
run;
```

PROC TABULATE オプションを指定します。 STYLE=オプションは、テーブルのセルの背景色を赤に設定します。

```
proc tabulate data=sashelp.class style={background=red};
```

サブグループを指定します。 CLASS ステートメントは、データを性別および年齢別に分類します。

```
class sex age;
```

テーブルの行と列を定義します。 TABLE ステートメントはテーブルを作成します。次元式の STYLE=オプションは、PROC TABULATE ステートメントからの STYLE=設定に優先しますが、フォーマットされたデータセルに限ります。

```
table sex*{style={background=blue}} all, age/nocellmerge;
```

生成するテーブルのタイトルを指定します。 このテーブルでは、スタイル色の変更が結合されていないフォーマットセルにどのように影響するかを示します。

```
title1 'Data Cell Styles with NOCELLMERGE Option';
```

プログラムを実行します。

```
run;
```

ODS HTML 出力の出力先をクローズします。

```
ODS HTML close;
```

出力

アウトプット 64.40 *NOCELLMERGE オプション*

Data Cell Styles in Merged Cells

	Age					
	11	12	13	14	15	16
	N	N	N	N	N	N
Sex						
F	1	2	2	2	2	.
M	1	3	1	2	2	1
All	2	5	3	4	4	1

Data Cell Styles with NOCELLMERGE Option

	Age					
	11	12	13	14	15	16
	N	N	N	N	N	N
Sex						
F	1	2	2	2	2	.
M	1	3	1	2	2	1
All	2	5	3	4	4	1

参考文献

Jain, Raj and Imrich Chlamtac. 1985. “The P² Algorithm for Dynamic Calculation of Quantiles and Histograms without Storing Observations.” *Communications of the Association of Computing Machinery* 28 (10): 1076-1085.

65 章

TIMEPLOT プロシジャ

概要: TIMEPLOT プロシジャ

TIMEPLOT プロシジャは、時間間隔に対して 1 つ以上の変数をプロットします。変数の値のリストが、プロットに伴って生成されます。プロットとリストは、PLOT プロシジャと PRINT プロシジャによって生成されるものと似ていますが、PROC TIMEPLOT 出力には、次に示す固有の特徴があります。

- 縦軸は、常にデータセット内のオブザベーションの順序を表します。そのため、オブザベーションが日付または時刻で順序付けられている場合、縦軸は時間の経過を表します。
- 横軸は、検証している変数の値を表します。PROC PLOT と同様に、PROC TIMEPLOT は 1 つの軸セット上に複数のプロットを重ね合わせることが可能なため、プロットの各行に複数の変数の値を含めることができます。
- PROC TIMEPLOT によって生成されるプロットは、複数のページに渡る場合があります。

- 各オブザベーションは、プロットの個別の行に順次表示されます。PROC TIMEPLOT では、PROC PLOT で発生することがあるようにオブザベーションが非表示になることはありません。
- プロットされる値のリストには、プロットに表示されない変数を含めることができます。

次の出力は、PROC TIMEPLOT で生成できる単純なレポートを示しています。このレポートは、年の最初の 6 週間の、2 人の営業担当者の冷蔵庫の売上を示します。出力を生成するステートメントは次のとおりです。“例 1: 単一変数のプロット” (2026 ページ)の DATA ステップにより、データセット Sales が作成されます。

```
title 'The SAS System';
options source;

options linesize=64 pagesize=60 nodate
        pageno=1;

proc timeplot data=sales;
   plot icebox;
   id month week;
   title 'Weekly Sales of Refrigerators';
   title2 'for the';
   title3 'First Six Weeks of the Year';
run;
```

アウトプット 65.1 *PROC TIMEPLOT で作成される単純なレポート*

Weekly Sales of Refrigerators
for the
First Six Weeks of the Year

Month	Week	Icebox	min 2520.04 … max 3550.43
1	1	3450.94	I
1	1	2520.04	I
1	2	3240.67	I
1	2	2675.42	I
1	3	3160.45	I
1	3	2805.35	I
1	4	3400.24	I
1	4	2870.61	I
2	1	3550.43	I
2	1	2730.09	I
2	2	3385.74	I
2	2	2670.93	I

次の出力は、アウトプット 65.1 (2014 ページ)を作成するのに使用されたものと同じデータセットのより複雑なレポートを示しています。このレポートを作成するステートメントを使用して、次の事を行います。

- 冷蔵庫の売上のプロットを 1 つ作成し、ストーブの売上のプロットを 1 つ作成します。
- 両方の営業担当者の売上を同一行にプロットします。

- 営業担当者の姓の最初の文字を使用して、プロット上のポイントを示します。
- 横軸のサイズを制御します。
- 出力形式とラベルを制御します。

このレポートを作成するプログラムの説明については、“例 5: 複数のオブザベーションをプロットの 1 行に表示する” (2036 ページ)を参照してください。

アウトプット 65.2 *PROC TIMEPLOT で作成されるより複雑なレポート*

Weekly Appliance Sales for the First Quarter

Month	Week	Kreitz Icebox	LeGrange Icebox	min $2,520.04 … max $3,550.43
January	1	$3,450.94	$2,520.04	L … K
January	2	$3,240.67	$2,675.42	L … K
January	3	$3,160.45	$2,805.35	L … K
January	4	$3,400.24	$2,870.61	L … K
February	1	$3,550.43	$2,730.09	L … K
February	2	$3,385.74	$2,670.93	L … K

Weekly Appliance Sales for the First Quarter

Month	Week	Seller :Kreitz Stove	Seller :LeGrange Stove	min $184.24 … max $2,910.37
January	1	$1,312.61	$728.13	L K
January	2	$222.35	$184.24	!
January	3	$2,263.33	$267.35	L … K
January	4	$1,787.45	$274.51	L … K
February	1	$2,910.37	$397.98	L … K
February	2	$819.69	$2,242.24	K … L

構文: TIMEPLOT プロシジャ

要件 最低 1 つの PLOT ステートメントが必要です。

ヒント: ATTRIB、FORMAT、LABEL、WHERE ステートメントを PROC TIMEPLOT とともに使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。

グローバルステートメントを使用することもできます。リストは、“グローバルステートメント” (24 ページ)および“Global Statements” (*SAS Statements: Reference*)を参照してください。

PROC TIMEPLOT <*option(s)*>;
BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …> <NOTSORTED>;
CLASS *variable(s)*;
ID *variable(s)*;
PLOT *plot-request(s)* </*option(s)*>;

ステートメント	タスク	例
"PROC TIMEPLOT ステートメント"	プロットの作成を要求します	Ex. 4
"BY ステートメント"	BY グループごとに別々のプロットを作成します	
"CLASS ステートメント"	データを分類変数の値に従ってグループ化します。	Ex. 5
"ID ステートメント"	指定される変数の値をリストで出力します。	Ex. 1, Ex. 2, Ex. 3
"PLOT ステートメント"	生成するプロットを指定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5

PROC TIMEPLOT ステートメント

プロットの作成を要求します。

例: "例 4: 2 つのプロットを重ね合わせる" (2033 ページ)

構文

PROC TIMEPLOT <*option(s)*>;

オプション引数

DATA=*SAS-data-set*
入力データセットを指定します。

ENCRYPTKEY=*key-value*
AES で暗号化されたデータセットをプロットする場合に必要となるキー値を指定します。入力データセットが ENCRYPT=AES で作成された場合は、ENCRYPTKEY=値を指定して対象データをプロットする必要があります。たとえば、DATA ステートメントを使用して secretPlot という名前のデータセットが作成される場合は、

```
data secretPlot(encrypt=AES encryptkey=Ib007)
```

次の PROC ステートメントを指定して secretPlot 内の対象データをプロットしてください。

```
proc timeplot data=secretPlot(encryptkey=Ib007);
```

参照項目 “ENCRYPTKEY= Data Set Option” (*SAS Data Set Options: Reference*)
ENCRYPTKEY=データセットオプションの詳細については、

MAXDEC=*number*
リストに印刷される小数点以下桁数の最大数を指定します。

デフォルト 2

範囲 0-12

操作 出力形式の小数点の指定は MAXDEC=の指定に優先します。

例 “例 4: 2 つのプロットを重ね合わせる” (2033 ページ)

SPLIT='*split-character*'
列ヘッダーの改行を制御する区切り文字を指定します。また、ラベルを列ヘッダーとして使用するように指定します。PROC TIMEPLOT は区切り文字に達すると列ヘッダーを改行し、次の行にヘッダーを続けます。区切り文字はブランクでない限り、列ヘッダーの一部ではありません。区切り文字が発生するたびにラベルの最大 256 文字に考慮されます。

別名 S=

デフォルト blank (' ')

注 列ヘッダーは最大 3 行です。列ラベルがこの固定数値より多い行に分割される場合、区切り文字はラベルの分割方法の推奨としてのみ使用されます。

UNIFORM
すべての BY グループにわたって横軸のスケールを均等に揃えます。デフォルトでは、PROC TIMEPLOT は各 BY グループの軸のスケールを個別に決定します。

操作 UNIFORM は参照線の平均の計算にも影響します。詳細については、“REF=*reference-value(s)* ” (2023 ページ)を参照してください。

BY ステートメント

BY グループごとに個別のプロットを作成します。

参照項目: “BY” (68 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* …>
<NOTSORTED>;

必須引数

variable
プロシジャが BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、デー

タは変数で並べ替えるか、インデックスを付ける必要があります。これらの変数は、*BY 変数*と呼ばれます。

オプション引数

DESCENDING
データセットが BY ステートメントで単語 DESCENDING の直後に続く変数を基準にして降順で並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。データは、時系列などの別の方法でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。同じ値の BY 変数を持つオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

CLASS ステートメント

データを分類変数の値に従ってグループ化します。

ヒント: PROC TIMEPLOT はクラスを形成するために CLASS 変数のフォーマットされた値を使用します。そのため、出力形式で値がグループ化される場合、プロシジャではそのグループが使用されます。

例: “例 5: 複数のオブザベーションをプロットの 1 行に表示する” (2036 ページ)

構文

CLASS *variable(s)*;

必須引数

variable(s)
プロシジャがデータのグループ化に使用する 1 つ以上の変数を指定します。CLASS ステートメントの変数は、*分類変数*といいます。分類変数は、数値か文字です。分類変数には連続値を含めることができますが、通常は変数の分類を定義するいくつかの不連続値が含まれます。データを分類変数別に並べ替える必要はありません。

分類変数の値がリストで表示されます。PROC TIMEPLOT は、分類変数の値の組み合わせが変わるたびに、1 行を出力してプロットします。そのため、データを分類変数の値に従って並べ替えまたはグループ化すると、出力はより意味のあるものとなります。

詳細

複数の CLASS ステートメントの使用
任意の数の CLASS ステートメントを使用できます。複数の CLASS ステートメントを使用する場合、PROC TIMEPLOT は、すべての CLASS ステートメントからの全変数を

単純に連結します。次の CLASS ステートメントの形式には、3 つの変数が含まれます。

CLASS *variable-1 variable-2 variable-3*;

これは次の形式と同じ結果になります。

CLASS *variable-1*;
CLASS *variable-2*;
CLASS *variable-3*;

シンボル変数の使用

通常は、シンボル変数がある CLASS ステートメントを使用します。この場合、プロット変数のリストには、シンボル変数の各値の列が含まれます。プロットの各行には、シンボル変数のそれぞれの値のポイントが含まれます。プロット記号は、シンボル変数のフォーマットされた値の最初の文字です。クラス内の複数のオブザベーションが同一のシンボル変数値を持つ場合、PROC TIMEPLOT はその値の最初の出現のみプロットおよび出力し、SAS ログへ警告メッセージを書き込みます。

プロット記号および CLASS ステートメントの詳細については、プロット要請 (2020 ページ)を参照してください。

ID ステートメント

指定される変数の値をリストで出力します。

例: “例 1: 単一変数のプロット” (2026 ページ)
“例 2: 軸とプロット記号のカスタマイズ” (2029 ページ)
“例 3: プロット記号に変数を使用する” (2031 ページ)

構文

ID *variable(s)*;

必須引数

variable(s)
リストで出力する 1 つ以上の *ID 変数*を特定します。

PLOT ステートメント

生成するプロットを指定します。

ヒント: 各 PLOT ステートメントにより個別のプロットが生成されます。

例: “例 1: 単一変数のプロット” (2026 ページ)
“例 2: 軸とプロット記号のカスタマイズ” (2029 ページ)
“例 3: プロット記号に変数を使用する” (2031 ページ)
“例 4: 2 つのプロットを重ね合わせる” (2033 ページ)
“例 5: 複数のオブザベーションをプロットの 1 行に表示する” (2036 ページ)

構文

PLOT *plot-request(s)* </*option(s)*>;

オプション引数の要約

Control the appearance of the plot

HILOC
: 最左のプロット記号と最右のプロット記号とをハイフン(-)の線で結び付けます。

JOINREF
: 記号が参照記号またはプロット記号かどうかに関係なく、プロットの各行の最左の記号と最右の記号とをハイフン(-)の線で結び付けます。

NOSYMNAME
: CLASS ステートメントの使用時に、列ヘッダーにシンボル変数名を表示しません。

NPP
: PLOT ステートメントに表示される変数の、値のリスト表示を抑制します。

POS=*print-positions-for-plot*
: 横軸に使用する出力位置の数を指定します。

Create and customize a reference line

REF=*reference-value(s)*
: プロットで横軸上の指定された値に対して直角を成す直線を引きます。

REFCHAR='*character*'
: 参照線を引くための文字を指定します。

Customize the axis

AXIS=*axis-specification*
: プロットする横軸の値の範囲と横軸上のそれぞれの出力位置によって表される間隔を指定します。

REVERSE
: 横軸上で最左の位置に最大値を置いて、値を並び替えます。

Display multiple plots on the same set of axes

OVERLAY
: 1 つの PLOT ステートメントにある要求をすべて、1 つの軸セット上にプロットします。

OVPCHAR='*character*'
: 複数のプロット記号が存在する場合に出力される文字を指定します。

必須引数

plot-request(s)
プロットする変数を指定します。(オプション)使用するプロット記号も指定します。デフォルトでは、各 PLOT 要求により個別のプロットが生成されます。

1 つの PLOT ステートメントにおいて、異なる形式の要求を組み合わせることができます。形式の組み合わせ例については、“例 4: 2 つのプロットを重ね合わせる” (2033 ページ)を参照してください。

variable(s)
プロットする数値変数を 1 つ以上指定します。PROC TIMEPLOT は、変数名の最初の文字をプロット記号として使用します。

例 “例 1: 単一変数のプロット” (2026 ページ)

(variable(s))='plotting-symbol'
プロットする数値変数を 1 つ以上指定し、リスト内のすべての変数に使用するプロット記号を使用します。1 つの変数のみ使用する場合は、かっこを省略できます。

例 “例 2: 軸とプロット記号のカスタマイズ” (2029 ページ)

(variable(s))=symbol-variable
プロットする数値変数を 1 つ以上指定し、*シンボル変数*を指定します。PROC TIMEPLOT は、シンボル変数のフォーマットされた値のブランク以外の最初の文字を、リストのすべての変数のプロット記号として使用します。プロット記号は、シンボル変数の値が異なる場合、オブザベーションごとに異なります。1 つの変数のみ使用する場合は、かっこを省略できます。

例 “例 3: プロット記号に変数を使用する” (2031 ページ)

オプション引数

AXIS=*axis-specification*
プロットする横軸の値の範囲と軸上の各出力位置によって表される間隔を指定します。PROC TIMEPLOT は、スペースがあれば軸の始端と終端をラベル付けします。

数値の場合、*axis-specification* は次のうちいずれか、または両方の組み合わせになります。

- *n*< …*n*>
- *n* **TO** *n* <BY *increment*>

値は、昇順または降順である必要があります。降順を指定するには、*increment* に負の値を使用します。指定される値は、値が均等に分布しない場合も、横軸に沿って均等にスペースが調整されます。数値は、次の方法で指定できます。

表 65.1 *AXIS=値の指定*

指定	コメント
`axis=1 2 10`	値は 1、2 および 10 です。
`axis=10 to 100 by 5`	値は、10 で開始して 100 で終了する 5 の増分として示されます。
`axis=12 10 to 100 by 5`	前述の 2 つの仕様の形式の組み合わせです。

日時値を含む軸の変数の場合、*axis-specification* は、明示的な値リストか、増分を指定した開始値と終了値のどちらかです。

- '*date-time-value*'*i* < …'*date-time-value*'*i*>
- '*date-time-value*'*i* **TO** '*date-time-value*'*i* <BY *increment*>

'date-time-value'i
SAS 関数 INTCK および INTNX に対して記述された任意の SAS 日付値、SAS 時間値または SAS 日時値。接尾辞 *i* は次のいずれかになります。

D 日付

T 時間

DT 日時

increment
INTCK または INTNX 関数の有効な引数のいずれかです。日付の場合、*increment* は次のいずれかになります。

- DAY
- WEEK
- MONTH
- QTR
- YEAR

日時の場合、*increment* は次のいずれかになります。

- DTDAY
- DTWEEK
- DTMONTH
- DTQTR
- DTYEAR

時刻の場合、*increment* は次のいずれかになります。

- HOUR
- MINUTE
- SECOND

次のステートメントは、日単位のインクリメントの使用例です。

```
axis='01JAN95'd to '01JAN96'd by month
axis='01JAN95'd to '01JAN96'd by qtr
```

個別の間隔の説明については、*SAS 言語リファレンス: 解説編*を参照してください。

注: わかりやすい出力形式で目盛値を出力するには、FORMAT ステートメントを使用する必要があります。

操作 POS=の値("POS=*print-positions-for-plot*" (2023 ページ)を参照してください) は AXIS=で設定される間隔に優先します。

ヒント データが範囲から外れ、プロットの軸領域外となる場合があります。これが発生した場合、PROC TIMEPLOT によりプロットの両側に山かっこ(<)または(>)が配置され、表示されていないデータが存在することを示します。

例 "例 2: 軸とプロット記号のカスタマイズ" (2029 ページ)

HILOC
最左のプロット記号と最右のプロット記号とをハイフン(-)の線で結び付けます。

操作 JOINREF が指定される場合、PROC TIMEPLOT は HILOC を無視します。

JOINREF
記号が参照記号またはプロット記号かどうかに関係なく、プロットの各行の最左の記号と最右の記号とをハイフン(-)の線で結び付けます。ただし、行に参照記号のみが含まれる場合、PROC TIMEPLOT は記号を結びつけません。

例 “例 3: プロット記号に変数を使用する” (2031 ページ)

NOSYMNAME
CLASS ステートメントの使用時に、列ヘッダーにシンボル変数名を表示しません。NOSYMNAME を使用すると、シンボル変数の値のみが列ヘッダーに表示されます。

例 “例 5: 複数のオブザベーションをプロットの 1 行に表示する” (2036 ページ)

NPP
PLOT ステートメントに表示される変数の、値のリスト表示を抑制します。

例 “例 3: プロット記号に変数を使用する” (2031 ページ)

OVERLAY
1 つの PLOT ステートメントにある要求をすべて、1 つの軸セット上にプロットします。指定しない場合、PROC TIMEPLOT はプロット要求に対してそれぞれプロットを生成します。

例 “例 4: 2 つのプロットを重ね合わせる” (2033 ページ)

OVPCHAR=*'character'*
複数のプロット記号が存在する場合に出力される文字を指定します。プロット記号と参照線の文字の両方が存在する場合、PROC TIMEPLOT はプロット記号を出力します。

デフォルト アットマーク(@)

例 “例 5: 複数のオブザベーションをプロットの 1 行に表示する” (2036 ページ)

POS=*print-positions-for-plot*
横軸に使用する出力位置の数を指定します。

デフォルト POS=と AXIS=の両方を省略する場合、PROC TIMEPLOT は最初は POS=20 であると見なします。ただし、スペースがあればこの値は増大され、利用可能なスペースにプロットが描かれます。

操作 POS=0 と AXIS=を指定する場合、利用可能なスペースにプロットが描かれます。POS=は AXIS=で設定される間隔に優先します。“AXIS=*axis-specification*” (2021 ページ)の説明を参照してください。

参照項目 “ページレイアウト” (2025 ページ)

例 “例 1: 単一変数のプロット” (2026 ページ)

REF=*reference-value(s)*
プロットで横軸上の指定された値に対して直角を成す直線を引きます。*reference-value(s)*の値は定数または次の形式で指定できます。

MEAN(*variable(s)*)

この形式の REF=を使用すると、PROC TIMEPLOT により指定の各変数の平均が評価され、各平均の参照線が引かれます。

操作 PROC TIMEPLOT ステートメントと UNIFORM オプションを使用すると、プロシジャはすべての BY グループのすべてのオブザベーションの変数の平均値を計算します。UNIFORM を使用しない場合、プロシジャは各 BY グループのそれぞれの変数の平均を計算します。

プロット記号と参照文字の両方が存在する場合、PROC TIMEPLOT はプロット記号を出力します。

例 “例 3: プロット記号に変数を使用する” (2031 ページ)

“例 4: 2 つのプロットを重ね合わせる” (2033 ページ)

REFCHAR=*'character'*
参照線を引くための文字を指定します。

デフォルト 縦棒(|)

操作 JOINREF オプションまたは HILOC オプションを使用する場合、プロット記号と同じ値を REFCHAR=に指定しないでください。これを行うと、PROC TIMEPLOT によりプロット記号は参照文字として解釈され、期待通りに結び付けられません。

例 “例 3: プロット記号に変数を使用する” (2031 ページ)

REVERSE
横軸上で最左の位置に最大値を置いて、値を並び替えます。

例 “例 4: 2 つのプロットを重ね合わせる” (2033 ページ)

結果:TIMEPLOT プロシジャ

データの考慮事項

入力データセットには、通常、分類変数または ID 変数のいずれかとして使用される日付変数が含まれています。PROC TIMEPLOT では、日付で並べ替えられた入力データセットは必須ではありませんが、通常、オブザベーションが時系列であると出力はより意味のあるものになります。また、CLASS ステートメントを使用する場合、入力データセットでオブザベーションが分類変数の値の組み合わせに従ってグループ化されていると、出力はより意味のあるものになります。詳細については、“CLASS ステートメント” (2018 ページ)を参照してください。

プロシジャの出力

ページレイアウト

各プロット要求について、PROC TIMEPLOT はリストとプロットを出力します。PROC TIMEPLOT により、次のようにページの配置が決定されます。

- POS=を使用する場合、プロシジャにより次が実行されます。
  - POS=値から、プロットのサイズを決定します。
  - 出力される値の列の幅から、均等になるよう配分して(列と列の間は最大 5 位置)、リストのスペースを決定します。
  - 出力をページの中央に配置します。
- POS=を省略する場合、プロシジャにより次が実行されます。
  - AXIS=オプションの値から、プロットの幅を決定します。
  - リストを展開して、残りのページに配置します。

リストとプロットを出力する十分なスペースがない場合、PROC TIMEPLOT は何も出力せず、次のエラーメッセージを SAS ログへ書き込みます。

```
ERROR:  Too many variables/symbol values
        to print.
```

エラーは、他のプロット要求には影響しません。

リストのコンテンツ

出力のリストに含まれる情報は、CLASS ステートメントを使用するかどうかによって異なります。CLASS ステートメントを使用しない場合、PROC TIMEPLOT により個別の行に各オブザベーションが出力(およびプロット)されます。CLASS ステートメントがない出力の例については、“例 1: 単一変数のプロット” (2026 ページ)を参照してください。

CLASS ステートメントを使用する場合、出力の形式は、シンボル変数を指定するかどうかによって異なります。CLASS ステートメントとシンボル変数との併用の詳細については、“シンボル変数の使用” (2019 ページ)を参照してください。

ODS テーブル名

TIMEPLOT プロシジャは、作成する各テーブルに名前を割り当てます。これらの名前を使用して、Output Delivery System (ODS)を使用してテーブルを選択し、出力データセットを作成する際にそのテーブルを参照できます。詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

表 65.2 *TIMEPLOT プロシジャによって生成される ODS テーブル*

テーブル名	説明	テーブル生成時の条件
Plot	単一プロット	OVERLAY オプションを指定しない場合
OverlaidPlot	単一軸セット上の 2 つ以上のプロット	OVERLAY オプションを指定する場合

欠損値

PROC TIMEPLOT から、次の 4 種類の変数がリストに表示されます。

- プロット変数
- ID 変数
- 分類変数
- シンボル変数(一部の列ヘッダーの部分として)。

プロット変数とシンボル変数はプロットにも表示されます。

分類変数の欠損値を含むオブザベーションは、オブザベーションのクラスを形成します。

リストでは、欠損値はピリオド(.)、ブランク、または特殊欠損値(文字 A～Z およびアンダースコア(_)文字)として表示されます。

プロットでは、PROC TIMEPLOT は次のように変数によって異なる方法で処理を行います。

- プロット変数の欠損値を含むオブザベーションまたはオブザベーションのクラスはプロットに表示されません。
- シンボル変数を使用すると、PROC TIMEPLOT は、ピリオド(.)を、シンボル変数の欠損値があるすべてのオブザベーションに対するプロット上のシンボル変数として使用します。シンボル変数の使用の詳細については、プロット要請 (2020 ページ)を参照してください。

例: TIMEPLOT プロシジャ

例 1: 単一変数のプロット

要素: PROC TIMEPLOT ステートメントオプション
DATA=
ID ステートメント
PLOT ステートメント
PLOT ステートメントオプション
POS=

他の要素: DATA ステップ

詳細

この例では、次のタスクについて説明します。

- DATA ステップを使用して、データセット SALES を作成します。
- 単一 PLOT ステートメントを使用して、冷蔵庫の売上をプロットします。
- プロットの横軸に使用される出力位置の数を指定します。

- プロットに含まれない 2 つの変数の値をリストで出力することにより、プロットポイントのコンテキストを提供します。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

data sales;
   input Month Week Seller $ Icebox Stove;
   datalines;
1 1 Kreitz   3450.94 1312.61
1 1 LeGrange 2520.04  728.13
1 2 Kreitz   3240.67  222.35
1 2 LeGrange 2675.42  184.24
1 3 Kreitz   3160.45 2263.33
1 3 LeGrange 2805.35  267.35
1 4 Kreitz   3400.24 1787.45
1 4 LeGrange 2870.61  274.51
2 1 Kreitz   3550.43 2910.37
2 1 LeGrange 2730.09  397.98
2 2 Kreitz   3385.74  819.69
2 2 LeGrange 2670.93 2242.24
;

proc timeplot data=sales;
   plot icebox / pos=50;

   id month week;

   title 'Weekly Sales of Iceboxes';
   title2 'for the';
   title3 'First Six Weeks of the Year';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

Sales データセットを作成します。 Sales には、2 人の営業担当者による冷蔵庫とストーブの売上に関する週単位の情報が含まれます。

```
data sales;
   input Month Week Seller $ Icebox Stove;
   datalines;
1 1 Kreitz   3450.94 1312.61
1 1 LeGrange 2520.04  728.13
1 2 Kreitz   3240.67  222.35
1 2 LeGrange 2675.42  184.24
1 3 Kreitz   3160.45 2263.33
1 3 LeGrange 2805.35  267.35
1 4 Kreitz   3400.24 1787.45
1 4 LeGrange 2870.61  274.51
2 1 Kreitz   3550.43 2910.37
2 1 LeGrange 2730.09  397.98
```

```
2 2 Kreitz   3385.74  819.69
2 2 LeGrange 2670.93 2242.24
;
```

冷蔵庫の売上をプロットします。プロット変数 Icebox がリストと出力の両方に表示されます。POS=は横軸に対し 50 の出力位置を指定します。

```
proc timeplot data=sales;
   plot icebox / pos=50;
```

プロットリストの行をラベル付けします。ID 変数、Month と Week の値がプロットのリストセクションのそれぞれの行を一意的に特定するために使用されます。

```
   id month week;
```

タイトルを指定します。

```
   title 'Weekly Sales of Iceboxes';
   title2 'for the';
   title3 'First Six Weeks of the Year';
run;
```

出力

リストの列ヘッダーは変数名です。プロットは、デフォルトのプロット記号としてプロット変数の名前の最初の文字を使用します。

アウトプット 65.3 *単一変数のプロット*

Weekly Sales of Refrigerators
for the
First Six Weeks of the Year

Month	Week	Icebox	min 2520.04 ... max 3550.43
1	1	3450.94	I
1	1	2520.04	I
1	2	3240.67	I
1	2	2675.42	I
1	3	3160.45	I
1	3	2805.35	I
1	4	3400.24	I
1	4	2870.61	I
2	1	3550.43	I
2	1	2730.09	I
2	2	3385.74	I
2	2	2670.93	I

例 2: 軸とプロット記号のカスタマイズ

要素: PROC PLOT ステートメントオプション
AXIS=
ID ステートメント
PLOT ステートメント

他の要素: LABEL ステートメント
PROC FORMAT
LIBNAME ステートメント
FMTSEARCH=システムオプション

データセット: Sales

詳細

この例では、次のタスクについて説明します。

- プロット記号として使用される文字を指定します。
- 横軸の最小値と最大値、各出力位置によって表される間隔を指定します。
- プロットに含まれない 2 つの変数の値をリストで出力することにより、プロットポイントのコンテキストを提供します。
- 変数のラベルをリストの列ヘッダーとして使用します。
- 永久出力形式を作成して使用します。

プログラム

```
libname proclib  'SAS-library';

options formchar="|----|+|---+=|-/\<>*";

proc format library=proclib;
   value monthfmt 1='January'
                  2='February';
run;

proc timeplot data=sales;
   plot icebox='R' / axis=2500 to 3600 by 25;

   id month week;

   label icebox='Refrigerator';

   format month monthfmt.;

title 'Weekly Refrigerator Sales';
title2 'for the First Six Weeks of the Year';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib  'SAS-library';
```

SAS システムオプションを設定します。FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

Month 変数の出力形式を作成します。PROC FORMAT は Month の永久出力形式を作成します。LIBRARY=オプションは、出力形式が後続の SAS セッションで使用できるように永久保管場所を指定します。この出力形式は、この章全体にわたって例に使用されています。

```
proc format library=proclib;
   value monthfmt 1='January'
                  2='February';
run;
```

冷蔵庫の売上をプロットします。プロット変数 Icebox がリストと出力の両方に表示されます。プロット記号は R です。AXIS=により軸の最小値が 2500 に、最大値が 3600 に設定されます。BY 25 により、軸上の各出力位置は 25 単位(この場合、ドル)を表すことが指定されます。

```
proc timeplot data=sales;
   plot icebox='R' / axis=2500 to 3600 by 25;
```

リストの行をラベル付けします。ID 変数、Month と Week の値がリストのそれぞれの行を一意的に特定するために使用されます。

```
   id month week;
```

ラベルをリストの売上の列に適用します。LABEL ステートメントは、ROC TIMEPLOT ステップの期間、ラベルと変数 Icebox とを関連付けます。PROC TIMEPLOT によりリストの列ヘッダーとしてラベルが使用されます。

```
   label icebox='Refrigerator';
```

MONTHFMT.出力形式を Month 変数に適用します。FORMAT ステートメントはレポートの Month に使用する出力形式を割り当てます。

```
   format month monthfmt.;
```

タイトルを指定します。

```
title 'Weekly Refrigerator Sales';
title2 'for the First Six Weeks of the Year';
run;
```

出力

リストには、ラベルのない変数 Month と Week、変数 Icebox に対するラベルであるラベル Refrigerator という 3 つの列ヘッダーがあります。プロット記号は `R` で、これは 変数ラベル Refrigerator を表しています。

アウトプット 65.4 *カスタマイズされた軸とプロット記号を使用したプロット*

Refrigerator Sales
for the First Six Weeks of the Year

Month	Week	Refrigerator	min 2500 ... max 3600
January	1	3450.94	R
January	1	2520.04	R
January	2	3240.67	R
January	2	2675.42	R
January	3	3160.45	R
January	3	2805.35	R
January	4	3400.24	R
January	4	2870.61	R
February	1	3550.43	R
February	1	2730.09	R
February	2	3385.74	R
February	2	2670.93	R

例 3: プロット記号に変数を使用する

要素: ID ステートメント
PLOT ステートメント
PLOT ステートメントオプション
- JOINREF
- NPP
- REF=
- REFCHAR=

データセット: Sales

出力形式: MONTHFMT.

詳細

この例では、次のタスクについて説明します。

- それぞれの営業担当者のポイントを区別するためにプロット記号として使用する変数を指定します。
- リストのプロット変数の値の出力を抑制します。
- 軸上の指定される値へ参照線を引くために使用する文字を指定し、線を引きます。
- プロットの各行の最左の記号と最右の記号とを結び付けます。

プログラム

```
libname proclib
'SAS-library';

options formchar="|----|+|---+=|-/\<>*" fmtsearch=(proclib);

proc timeplot data=sales;
   plot stove=seller /

                       npp

                       ref=1500 refchar=':'

                       joinref

                       axis=100 to 3000 by 50;

   id month week;

   format month monthfmt.;

   title 'Weekly Sales of Stoves';
   title2 'Compared to Target Sales of $1500';
   title3 'K for Kreitz; L for LeGrange';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib
'SAS-library';
```

SAS システムオプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options formchar="|----|+|---+=|-/\<>*" fmtsearch=(proclib);
```

ストーブの売上をプロットします。 PLOT ステートメントにより、プロット変数 Stove とシンボル変数 Seller の両方が指定されます。プロット記号は、Seller のフォーマットされた値の最初の文字です(この場合、`L` または `K`)。

```
proc timeplot data=sales;
   plot stove=seller /
```

プロット変数のリストでの表示を抑制します。 Stove 変数の値はリストに表示されません。

```
                       npp
```

プロットに参照線を作成します。 REF=と REFCHAR=により売上目標である$1500 にコロンで線が引かれます。

```
                       ref=1500 refchar=':'
```

プロットの各行の記号間に線を引きます。 このプロットでは、JOINREF により各プロット記号が参照線へ結び付けられます。

```
                       joinref
```

横軸をカスタマイズします。 AXIS=により横軸の最小値が 100 に、最大値が 3000 に設定されます。BY 50 により、軸上の各出力位置は 50 単位(この場合、ドル)を表すことが指定されます。

```
                        axis=100 to 3000 by 50;
```

リストの行をラベル付けします。 ID 変数、Month と Week の値がリストのそれぞれの行を特定するために使用されます。

```
      id month week;
```

MONTHFMT.出力形式を Month 変数に適用します。 FORMAT ステートメントはレポートの Month に使用する出力形式を割り当てます。

```
      format month monthfmt.;
```

タイトルを指定します。

```
      title 'Weekly Sales of Stoves';
      title2 'Compared to Target Sales of $1500';
      title3 'K for Kreitz; L for LeGrange';
   run;
```

出力

プロットでは、Seller の値の最初の文字がプロット記号として使用されます。

アウトプット 65.5 *プロット記号に変数を使用したプロット*

Weekly Sales of Stoves
Compared to Target Sales of $1500
K for Kreitz; L for LeGrange

```
    Month      Week       min                                                  max
                          100                                                 3000
                         *-----------------------------------------------------*
January            1     |                        K---:                        |
January            1     |             L-------------:                         |
January            2     |  K------------------------:                        |
January            2     |  L------------------------:                        |
January            3     |                           :-------------K            |
January            3     |   L-----------------------:                        |
January            4     |                           :-----K                    |
January            4     |   L-----------------------:                        |
February           1     |                           :-------------------------K|
February           1     |      L--------------------:                        |
February           2     |               K------------:                        |
February           2     |                           :-------------L            |
                         *-----------------------------------------------------*
```

例 4: 2 つのプロットを重ね合わせる

要素: PROC TIMEPLOT ステートメントオプション
MAXDEC=

PLOT ステートメント

PLOT ステートメントオプション
- OVERLAY
- REF=MEAN(*variable(s)*)
- REVERSE

データセット: Sales

詳細

この例では、次のタスクについて説明します。

- 2 つのプロットを 1 つの軸セット上に重ね合わせます。
- 一方のプロットのプロット記号として使用する変数と、他方のプロットのプロット記号として使用する文字を指定します。
- プロットされる 2 つの変数それぞれの平均値への参照線を引きます。
- 軸のラベル付けを逆順にし、最大値がプロットの最左に配置されるようにします。

プログラム

```
options formchar="|----|+|---+=|-/\<>*";

proc timeplot data=sales maxdec=0;

   plot stove=seller icebox='R' /

          overlay

            ref=mean(stove icebox)


reverse;

   label icebox='Refrigerators';

   title 'Weekly Sales of Stoves and Refrigerators';
   title2 'for the';
   title3 'First Six Weeks of the Year';
run;
```

プログラムの説明

FORMCHAR オプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。

```
options formchar="|----|+|---+=|-/\<>*";
```

表示する小数点以下桁数を指定します。 MAXDEC=リストに表示する小数点以下桁数を指定します。

```
proc timeplot data=sales maxdec=0;
```

ストーブと冷蔵庫の両方の売上をプロットします。 PLOT ステートメントは 2 つのプロットを要求します。1 つのプロットは Seller のフォーマットされた値の最初の文字を使用して Stove の値をプロットします。もう 1 つのプロットは文字 **R**(ラベル Refrigerators と結びつけるため)を使用して Icebox の値をプロットします。

```
      plot stove=seller icebox='R' /
```

両方のプロットを同一の軸セット上に出力します。

```
            overlay
```

プロットに 2 本の参照線を作成します。 REF=は、2 本の参照線を引きます。Stove の平均に対して直角を成す直線と、Icebox の平均に対して直角を成す直線です。

```
              ref=mean(stove icebox)
```

横軸上の値を、最大から最少へと並べ替えます。

```
   reverse;
```

ラベルをリストの売上の列に適用します。 LABEL ステートメントは、ROC TIMEPLOT ステップの期間、ラベルと変数 Icebox とを関連付けます。PROC TIMEPLOT によりリストの列ヘッダーとしてラベルが使用されます。

```
      label icebox='Refrigerators';
```

タイトルを指定します。

```
      title 'Weekly Sales of Stoves and Refrigerators';
      title2 'for the';
      title3 'First Six Weeks of the Year';
   run;
```

出力

リストの変数 Icebox の列ヘッダーは変数のラベル(Refrigerators)です。一方のプロットでは、Seller の値の最初の文字がプロット記号として使用されます。他方のプロットでは、文字 R が使用されます。

アウトプット 65.6 *異なるプロット記号を使用して重ね合わせた 2 つのプロット*

Weekly Sales of Stoves and Refrigerators
for the
First Six Weeks of the Year

```
Stove     Refrigerators      max                                          min
                             3550.43                                   184.24
                            *-----------------------------------------------*
 1313              3451     |R      |                          K   |         |
  728              2520     |       |        R                     |    L    |
  222              3241     |   R   |                              |       K |
  184              2675     |       |    R                         |        L|
 2263              3160     |      R|           K                  |         |
  267              2805     |       |   R                          |       L |
 1787              3400     | R     |                    K         |         |
  275              2871     |       |  R                           |       L |
 2910              3550     |R      | K                            |         |
  398              2730     |       |    R                         |      L  |
  820              3386     | R     |                              |  K      |
 2242              2671     |       |      R     L                 |         |
                            *-----------------------------------------------*
```

例 5: 複数のオブザベーションをプロットの 1 行に表示する

要素: CLASS ステートメント
PLOT ステートメント
PLOT ステートメントオプション
NOSYMNAME
OVPCHAR=
POS=

データセット: Sales

出力形式: MONTHFMT.

詳細

この例では、次のタスクについて説明します。

- 同じ月と週のオブザベーションをグループ化して、2 人の営業担当者の同一週の売上をプロットの同一行に表示します。
- プロット記号として使用される変数を指定します。
- 1 つのプロットでプロット変数の名前を非表示にします。

- プロットのサイズを指定して、両方のプロットの表示サイズが同じになるようにします。

プログラム

```
libname proclib
'SAS-library';

options formchar="|----|+|---+=|-/\<>*" fmtsearch=(proclib);

proc timeplot data=sales;
   class month week;

   plot stove=seller  / pos=25 ovpchar='!';
   plot icebox=seller / pos=25 ovpchar='!' nosymname;

   format stove icebox dollar10.2 month monthfmt.;

   title 'Weekly Appliance Sales for the First Quarter';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib
'SAS-library';
```

SAS システムオプションを設定します。 FORMCHAR をこの文字列どおりに設定すると、SAS Monospace フォントを使用できない SAS 以外の環境での HTML 出力の表示が改善されます。FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options formchar="|----|+|---+=|-/\<>*" fmtsearch=(proclib);
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Month と Week が同一の値であるすべてのオブザベーションをグループ化し、1 行に出力します。CLASS ステートメントとシンボル変数を使用すると、リストにシンボル変数の各値のプロット変数の列が 1 列生成されます。

```
proc timeplot data=sales;
   class month week;
```

ストーブと冷蔵庫の売上をプロットします。 各 PLOT ステートメントにより個別のプロットが生成されます。プロット記号は、次のようにシンボル変数のフォーマットされた値の最初の文字です。Kreitz の場合は **K**、LeGrange の場合は **L**.POS=は各プロットで横軸で 25 の出力位置を使用するように指定します。OVPCHAR=は複数のプロット記号が存在する場合に、感嘆符をプロット記号として指定します。NOSYMNAME は、2 番目のリストのシンボル変数 Seller の名前を非表示にします。

```
   plot stove=seller  / pos=25 ovpchar='!';
   plot icebox=seller / pos=25 ovpchar='!' nosymname;
```

リストの値に出力形式を適用します。 FORMAT ステートメントはレポートの Stove、Icebox、Month で使用する出力形式を割り当てます。TITLE ステートメントでタイトルを指定します。

```
   format stove icebox dollar10.2 month monthfmt.;
```

タイトルを指定します。

```
    title 'Weekly Appliance Sales for the First Quarter';
run;
```

出力

アウトプット 65.7 *アイスボックスの販売者別週間売上*

Weekly Appliance Sales for the First Quarter

Month	Week	Kreitz Icebox	LeGrange Icebox	min $2,520.04 — max $3,550.43
January	1	$3,450.94	$2,520.04	|L K |
January	2	$3,240.67	$2,675.42	| L K |
January	3	$3,160.45	$2,805.35	| L K |
January	4	$3,400.24	$2,870.61	| L K |
February	1	$3,550.43	$2,730.09	| L K|
February	2	$3,385.74	$2,670.93	| L K |

アウトプット 65.8 *ストーブの販売者別週間売上*

Weekly Appliance Sales for the First Quarter

Month	Week	Seller :Kreitz Stove	Seller :LeGrange Stove	min $184.24 — max $2,910.37
January	1	$1,312.61	$728.13	| L K |
January	2	$222.35	$184.24	|! |
January	3	$2,263.33	$267.35	| L K |
January	4	$1,787.45	$274.51	| L K |
February	1	$2,910.37	$397.98	| L K|
February	2	$819.69	$2,242.24	| K L |

66 章

TRANSPOSE プロシジャ

概要: TRANSPOSE プロシジャ

TRANSPOSE プロシジャの動作について

TRANSPOSE プロシジャは、SAS データセットの値を再構築し、選択した変数をオブザベーションに転置して、出力データセットを作成します。多くの場合、TRANSPOSE プロシジャを使用すると、長い DATA ステップを書かずに同じ結果を得ることができます。さらに、出力データセットは、分析、レポート作成、その他のデータ操作のための後続の DATA または PROC ステップでも使用できます。

PROC TRANSPOSE では、印刷出力は作成されません。PROC TRANSPOSE ステップから出力データセットを印刷するには、PROC PRINT、PROC REPORT または別の SAS レポートツールを使用します。

転置された変数を作成するために、このプロシジャで、入力データセットのオブザベーション値が出力データセットの変数値に転置されます。

PROC TRANSPOSE が実行可能な転置の種類について

単純な転置

次の例では、単純な転置について説明します。入力データセットでは、各変数が 1 人の試験者のスコアを表します。出力データセットでは、各オブザベーションが 1 人の試験者のスコアを表します。_NAME_の各値は、プロシジャで転置された入力データセットの変数名です。したがって、_NAME_の値によって、出力データセットの各オブザベーションのソースが識別されます。たとえば、出力データセットの最初のオブザベーションの値は、入力データセットの変数 Tester1 の値から生成されています。出力を生成するステートメントは次のとおりです。

```
proc print data=proclib.product noobs;
   title 'The Input Data Set';
run;

proc transpose data=proclib.product
               out=proclib.product_transposed;
run;

proc print data=proclib.product_transposed noobs;
   title 'The Output Data Set';
run;
```

アウトプット 66.1 単純な転置

```
The Input Data Set                              1 Tester1 Tester2 Tester3 Tester4 22         25
21         21 15         19         18         17 17         19         19         19 20
19         16          19 14         15         13          13 15         17         18          19
10         11           9          10 22         24          23          21
```

```
The Output Data Set 2 _NAME_ COL1 COL2 COL3 COL4 COL5 COL6 COL7 COL8 Tester1 22 15 17 20 14 15 10 22
Tester2 25 19 19 19 15 17 11 24 Tester3 21 18 19 16 13 18 9 23 Tester4 21 17 19 19 13 19 10 21
```

BY グループを使用する複合転置

BY グループを使用する次の例は、より複合的です。入力データセットは、2 つの湖の魚の重量と体長の測定値を表します。出力データセットを作成するステートメントでは、次が行われます。

- 体長の測定値を含む変数のみを転置します。
- 湖と日付の組み合わせごとに 1 つずつ、6 つの BY グループを作成します。
- データセットオプションを使用して、転置された変数の名前を指定します。

アウトプット 66.2 BY グループによる転置

```
Input Data Set 1 L o                    L      W      L      W      L      W      L      W
c                   e      e      e      e      e      e      e      e a                    n
i      n      i      n      i      n      i t              D     g      g      g      g      g
g      g      g i              a     t      h      t      h      t      h      t      h
o              t     h      t      h      t      h      t      h      t n              e     1
1      2      2      3      3      4      4 Cole Pond      02JUN95    31    0.25    32    0.30    32
0.25    33    0.30 Cole Pond      03JUL95    33    0.32    34    0.41    37    0.48    32    0.28 Cole
Pond      04AUG95    29    0.23    30    0.25    34    0.47    32    0.30 Eagle Lake    02JUN95    32
0.35    32    0.25    33    0.30     .      .Eagle Lake     03JUL95    30    0.20    36
0.45     .     .       .      .Eagle Lake     04AUG95    33    0.30    33    0.28    34    0.42     ..
```

```
Fish Length Data for Each Location and Date                2 Location        Date     _NAME_
Measurement Cole Pond      02JUN95    Length1          31 Cole Pond      02JUN95    Length2          32
Cole Pond      02JUN95    Length3          32 Cole Pond      02JUN95    Length4          33 Cole Pond
03JUL95    Length1          33 Cole Pond      03JUL95    Length2          34 Cole Pond      03JUL95
Length3          37 Cole Pond      03JUL95    Length4          32 Cole Pond      04AUG95    Length1
29 Cole Pond      04AUG95    Length2          30 Cole Pond      04AUG95    Length3          34 Cole
Pond      04AUG95    Length4          32 Eagle Lake     02JUN95    Length1          32 Eagle Lake
02JUN95    Length2          32 Eagle Lake     02JUN95    Length3          33 Eagle Lake     02JUN95
Length4           .Eagle Lake     03JUL95    Length1          30 Eagle Lake     03JUL95    Length2
36 Eagle Lake     03JUL95    Length3           .Eagle Lake     03JUL95    Length4           .Eagle Lake
04AUG95    Length1          33 Eagle Lake     04AUG95    Length2          33 Eagle Lake     04AUG95
Length3          34 Eagle Lake     04AUG95    Length4           .
```

これらの結果を生成する SAS プログラムの詳細な説明については、“例 4: BY グループの転置” (2056 ページ)を参照してください。

PROC TRANSPOSE のデータベース内処理

データベース内処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。機密データをデータソースから抽出する必要がないため、セキュリティを強化できます。データが、比較的低速なネットワーク接続を介して移送されるかわりに、高速の二次記憶装置を使用して、データソース上でローカル操作されます。データソースが使用されるのは、データソースに自由に使えるより多くの処理リソースがあり、クエリを最適化して高度に並列かつスケーラブルな方法で実行できるためです。

DATA=入力テーブルがデータベースでテーブルまたはビューとして保存される場合、PROC TRANSPOSE はデータベース内処理を使用して、データベース内の作業のほとんどを実行できます。データベース内処理は、より迅速な処理と、データベースと SAS ソフトウェア間のデータ転送の減少という利点を提供できます。

PROC TRANSPOSE のデータベース内処理では、次のデータベースプロバイダがサポートされています。

- Hadoop
- Teradata

TRANSPOSE プロシジャはデータの動的な変換を実行します。この変換では、出力テーブルの特性、特に変数の数と名前およびそれらの型が、変数値および入力テーブルの特性から判断されます。この動的な動作は、2 パス処理で実現されます。最初のパスで入力テーブルの行を調べて出力テーブルの特性を確認し、2 回目のパスでデータを転置する作業を実行します。大量並列処理(MPP)データベース内での並列処理により、最初のパスと 2 回目のパスの両方が高速化されます。MPP では、多数のプロセッサまたは別のコンピュータを使用して一連の協調型計算を並列で実行します。最初のパスでは、クラスタのノードにすでに分割されている行のとおりに、並列かつイン

プレースで調べられます。2 回目のパスで、行は再分割されて BY グループが作成され、それらのグループを個別に並列で処理します。

データベース内処理の最初のパスと 2 回目のパスはどちらも、クラスタのノード内にある SAS Embedded Process 内の DS2 プログラムを実行して行われます。DBMS により、SAS Embedded Process ではテーブルからデータを読み込み、データをテーブルに書き込むことができます。SAS Embedded Process は、DS2 プログラムの実行コンテキストを提供します。作業の 2 つのパスは DS2 言語で表されるため、テーブルの列は、DS2 データ型を持つ変数にキャストされます。DS2 内のデータ型サポートは、従来の SAS システムで提供されているサポートよりも範囲が広いため、データベース内で実行すると、転置された出力データ内で入力データのデータ型と値を維持する TRANSPOSE プロシジャの機能が拡張されます。

SQLGENERATION システムオプションまたは LIBNAME ステートメントオプションは、データベース内プロシジャがデータベース内で実行されるかどうか、およびその実行方法を制御します。INDB=YES と指定すると、PROC TRANSPOSE がデータベース内で実行されます。データベース内処理の妨げとなる可能性のあるプログラミングの考慮事項が多数あります。完全なリストについては、*SAS In-Database Products: User's Guide* の Procedure Considerations and Limitations を参照してください。

構文: TRANSPOSE プロシジャ

ヒント: Output Delivery System はサポートされていません。

発生するデータベース内処理に対し、データは SAS データベース内処理用に適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、次を参照してください。

ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。グローバルステートメントを使用することもできます。リストについては、“Global Statements” (*SAS Statements: Reference*)を参照してください。

PROC TRANSPOSE <DATA=*input-data-set*> <DELIMITER=*delimiter*> <LABEL=*label*> <LET> <NAME=*name*> <OUT=*output-data-set*> <PREFIX=*prefix*> <SUFFIX=*suffix*>;

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2 …*>
<NOTSORTED>;

COPY *variable(s)*;

ID *variable*;

IDLABEL *variable*;

VAR *variable(s)*;

ステートメント	タスク	例
“PROC TRANSPOSE ステートメント”	SAS データセットの値を再構築し、選択した変数をオブザベーションに転置して、出力データセットを作成します。	Ex. 1, Ex. 2, Ex. 3, Ex. 5
“BY ステートメント”	各 BY グループを転置します。	Ex. 4

ステートメント	タスク	例
“COPY ステートメント”	変数を転置せずに直接コピーします。	Ex. 6
“ID ステートメント”	転置された変数の命名元の値を含む変数を指定します。	Ex. 2
“IDLABEL ステートメント”	転置された変数のラベルを作成します。	Ex. 3
“VAR ステートメント”	転置する変数をリスト表示します。	Ex. 4, Ex. 6

PROC TRANSPOSE ステートメント

SAS データセットの値を再構築し、選択した変数をオブザベーションに転置して、出力データセットを作成します。

ヒント: DATA=および OUT=オプションではデータセットオプションを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (67 ページ)を参照してください。グローバルステートメントを使用することもできます。リストについては、“Global Statements” (*SAS Statements: Reference*)を参照してください。

例:
“例 1: 単純な転置の実行” (2051 ページ)
“例 2: 転置された変数の名前の指定” (2053 ページ)
“例 3: 転置された変数のラベル作成” (2054 ページ)
“例 4: BY グループの転置” (2056 ページ)
“例 5: ID 変数の値が重複する場合の転置された変数名の指定” (2058 ページ)
“例 6: 統計分析用データの転置” (2060 ページ)

構文

PROC TRANSPOSE <DATA=*input-data-set*> <DELIMITER=*delimiter*<INDB=YES|NO>> > <LABEL=*label*> <LET> <NAME=*name*> <OUT=*output-data-set* > <PREFIX=*prefix*> <SUFFIX=*suffix*>;

オプション引数

DATA= *input-data-set*
転置する SAS データセットの名前を指定します。

デフォルト 直前に作成された SAS データセット

DELIMITER= *delimiter*
出力データセットの転置された変数の名前の作成に使用するデリミタを指定します。指定した場合、ID ステートメントで変数を複数指定すると、変数値の間にデリミタが挿入されます。

別名 DELIM=

ヒント DELIMITER の値には名前リテラル(n-literals)を使用できます。名前リテラルは、特に VALIDVARNAME=ANY の場合、印刷用文字や外国語の文字を指定するときに役立ちます。

参照項目 “ID ステートメント” (2047 ページ)

INDB=YES|NO
データベース内処理がアクティブであるかどうかを指定します。YES は、INDB がアクティブであることを示します。YES がデフォルトです。NO は、INDB がアクティブではないことを示します。

LABEL= ***label***
現在のオブザベーションを作成するために転置されている変数のラベルを含む、出力データセットの変数の名前を指定します。

デフォルト _LABEL_

ヒント LABEL の値には名前リテラル(n-literals)を使用できます。名前リテラルは、特に VALIDVARNAME=ANY の場合、印刷用文字や外国語の文字を指定するときに役立ちます。

LET
ID 変数の重複する値を許可します。PROC TRANSPOSE では、データセットまたは BY グループ内で最後に発生した特定の ID 値を含むオブザベーションが転置されます。

参照項目 “例 5: ID 変数の値が重複する場合の転置された変数名の指定” (2058 ページ)

NAME= ***name***
現在のオブザベーションを作成するために転置されている変数の名前を含む、出力データセットの変数の名前を指定します。

デフォルト _NAME_

参照項目 “例 2: 転置された変数の名前の指定” (2053 ページ)

OUT= ***output-data-set***
出力データセットを指定します。*output-data-set* が存在しない場合は、PROC TRANSPOSE で、DATA*n* 命名規則を使用して作成されます。

デフォルト DATA*n*

参照項目 “例 1: 単純な転置の実行” (2051 ページ)

PREFIX= ***prefix***
出力データセットの転置された変数の名前の作成に使用する接頭辞を指定します。たとえば、PREFIX=VAR の場合、変数の名前は VAR1、VAR2、…、VAR*n* となります。

操作 PREFIX=を ID ステートメントとともに使用すると、変数名は接頭辞の値で始まり、その後に ID 値が続きます。

ヒント PREFIX の値には名前リテラル(n-literals)を使用できます。名前リテラルは、特に VALIDVARNAME=ANY の場合、印刷用文字や外国語の文字を指定するときに役立ちます。

参照項目 “例 2: 転置された変数の名前の指定” (2053 ページ)

SUFFIX= ***suffix***
出力データセットの転置された変数の名前の作成に使用する接尾辞を指定します。

操作 SUFFIX=を ID ステートメントとともに使用すると、値が ID 値に追加されます。

ヒント SUFFIX の値には名前リテラル(n-literals)を使用できます。名前リテラルは、特に VALIDVARNAME=ANY の場合、印刷用文字や外国語の文字を指定するときに役立ちます。

BY ステートメント

BY グループを定義します。

制限事項: 別のユーザーが同時にデータセットを更新している場合、PROC TRANSPOSE を BY ステートメントや ID ステートメントとともに使用しないでください。

例: “例 4: BY グループの転置” (2056 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-1* …>
<NOTSORTED>;

必須引数

variable
PROC TRANSPOSE が BY グループの形成に使用する変数を指定します。複数の変数を指定できます。BY ステートメントで NOTSORTED オプションを使用しない場合、オブザベーションは指定するすべての変数別に並べ替えるか、適切にインデックス付けする必要があります。BY ステートメントの変数は *BY 変数*といいます。

オプション引数

DESCENDING
データセットが BY ステートメントで単語 DESCENDING の直後に続く変数を基準にして降順で並べ替えられるように指定します。

NOTSORTED
オブザベーションが必ずしもアルファベット順または数字順で並べ替えられないように指定します。データは、時系列などの別の方法でグループ化されます。

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。プロシジャは、NOTSORTED を指定する場合はインデックスを使用しません。プロシ

ジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

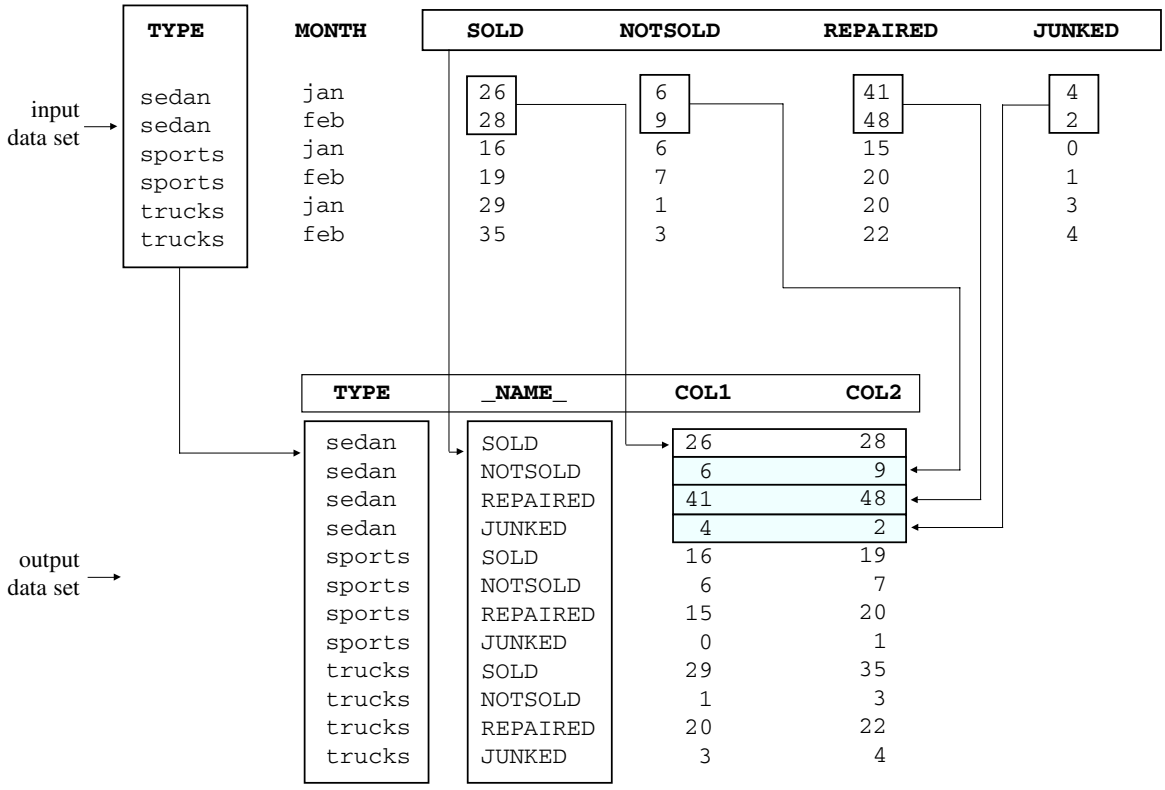
NOBYSORTED システムオプションは、システム全体にわたるオブザベーション順序チェックを無効化し、すべてのプロシジャと BY ステートメントに適用されます。“BYSORTED System Option” (*SAS System Options: Reference*)を参照してください。

詳細

PROC TRANSPOSE では、BY グループは転置されません。その代わりに、各 BY グループに対して、転置する変数ごとに 1 つずつオブザベーションが作成されます。

次の図は、BY グループを使用してデータセットを転置した結果を示しています。TYPE が BY 変数で、SOLD、NOTSOLD、REPAIRED、JUNKED が転置する変数です。

図 66.1 *BY グループによる転置*



- 出力データセットのオブザベーション数(12)は、BY グループ数(3)に転置する変数の数(4)を乗算した結果です。
- BY 変数は転置されません。
- _NAME_には、出力データセットの現在のオブザベーションを作成するために転置された入力データセットの変数の名前が含まれます。_NAME_変数に別の名前を指定するには、NAME=オプションを使用します。
- 入力データセットのどの BY グループでもオブザベーションの最大数は 2 です。したがって、出力データセットには、COL1 と COL2 という 2 つの変数が含まれます。

COL1 と COL2 には、SOLD、NOTSOLD、REPAIRED、JUNKED の値が含まれます。

注: 入力データセットに他の BY グループよりもオブザベーションが多い BY グループがある場合、PROC TRANSPOSE は、対応する入力オブザベーションのない変数に、出力データセットで欠損値を割り当てます。

COPY ステートメント

変数を転置せずに、入力データセットから出力データセットに直接コピーします。

例: "例 6: 統計分析用データの転置" (2060 ページ)

構文

COPY *variable(s)*;

必須引数

variable(s)
COPY ステートメントが転置せずに入力データセットから出力データセットに直接コピーする変数の名前を 1 つ以上指定します。

詳細

COPY ステートメントでは、変数が直接出力データセットにコピーされるので、出力データセットのオブザベーション数と入力データセットのオブザベーション数が等しくなります。

このプロシジャでは、入力データセットのオブザベーションの数と、転置される変数の数が等しくない場合、出力データセットに欠損値が埋め込まれます。

ID ステートメント

出力データセットの転置された変数の名前になるフォーマットされた非欠損値を含む、入力データセットの変数を 1 つ以上指定します。転置後の(出力)データセットで変数名が生成されるときに、リストされたすべての ID 変数のフォーマットされた値が、ID ステートメントの変数リストと同じ順序で連結されます。PREFIX=、DELIMITER=および SUFFIX=オプションを使用すると、生成された変数名を変更できます。PREFIX=オプションでは、生成された変数名の先頭にくる共通の文字または文字列を指定します。DELIMITER=オプションでは、ID 変数の値間に挿入される共通の文字または文字列を指定します。SUFFIX=オプションでは、生成された各変数名の末尾に追加される共通の文字または文字列を指定します。

制限事項: 別のユーザーが同時にデータセットを更新している場合、同時アクセス権をサポートするエンジンで PROC TRANSPOSE を ID ステートメントや BY ステートメントとともに使用することはできません。

ヒント: いずれかの ID 変数の値が欠損している場合、PROC TRANSPOSE により、警告メッセージがログに書き込まれます。このプロシジャでは、いずれかの ID 変数の値が欠損しているオブザベーションは転置されません。

例: "例 2: 転置された変数の名前の指定" (2053 ページ)

構文

ID *variable(s)*;

必須引数

variable(s)
出力データセットの変数の名前の生成に使用するフォーマットされた値を含む変数の名前を 1 つ以上指定します。

詳細

重複する出力データセット変数名

入力データセットの ID 変数値から生成される変数名(PREFIX、DELIMITER および SUFFIX オプション値の連結)は、出力データセット内で一意にしてください。出力データセット変数名が 2 回以上出現する場合は、入力データセットの 2 つ以上のオブザベーションが出力データセットの 1 つの変数に転置され、その結果、データ損失が生じたことを示します。ID 変数が 1 つしかない場合に、入力データセット内または(BY ステートメントを使用する場合)BY グループ内に重複するフォーマットされた値があると、このような状況が生じます。同様に、ID 変数が複数ある場合、ID 変数のフォーマットされた値の組み合わせが、入力データセット内または BY グループ内で 2 回以上発生すると、このような状況が発生します。PROC TRANSPOSE では、重複する出力データセット変数名が生成された場合、データ損失を防ぐために、重複する ID 値について警告メッセージが出力され、処理が停止されます。ただし、PROC TRANSPOSE ステートメントで LET オプションを指定した場合は、プロシジャで警告メッセージが出力され、処理が続行されて、重複するフォーマット変数値のうち最後に発生した値を含むオブザベーションが転置されます。

注: ID 変数または ID 変数の組み合わせのフォーマットされた値が入力データセット内で一意である場合、ID 変数を PREFIX、DELIMITER および SUFFIX オプション値と結合して変数名を生成するために必要な文字の置き換えや切り捨てによって、出力データセットの変数名が重複する可能性があります。

数値からの変数名の作成

数値変数を ID 変数として使用すると、PROC TRANSPOSE により、フォーマットされた ID 値が有効な SAS 名に変更されます。

VALIDVARNAME=ANY を設定しない限り、SAS 変数名の先頭は数字にすることはできません。フォーマットされた値の最初の文字が数値の場合、プロシジャによりその値にアンダースコアの接頭辞が付けられます。このアクションによって、32 文字値の最後の文字が切り捨てられます。その他の無効な文字はアンダースコアに置き換えられます。32 文字よりも長い ID 値を使用して転置される変数に名前を付けると、その ID 値はすべて 32 文字まで切り捨てられます。

フォーマットされた値が数値定数のように見える場合、文字+、−、.が、それぞれ P、N、D に変更されます。フォーマットされた値に数値ではない文字が含まれている場合、文字+、−、.がアンダースコアに変更されます。

注: VALIDVARNAME システムオプションの値が V6 の場合、転置された変数名が 8 文字まで切り捨てられます。

複数の ID 変数からの変数名の作成

ID 変数を 1 つだけ指定した場合は、出力データセット変数名を生成するときに、変数のフォーマットされた値は VALIDVARNAME オプションによって課された SAS 変数命名規則に従って作成されます。また、ID 変数値を PREFIX および SUFFIX オプションと結合して生成する名前も、SAS 変数命名規則に従って作成されます。フォーマットさ

れた ID 変数値でも、その変数値と PREFIX および SUFFIX オプションとの組み合わせでも、無効な文字はアンダースコアに置き換えられます。または名前が数値定数であるように見える場合は、アンダースコアが接頭辞として使用され、文字+、-、.が P、N、D に変更されます。その結果生成された名前は、VALIDVARNAME オプション設定で許可される名前の最大長に合わせて切り捨てられます。複数の ID 変数を指定する場合、各 ID 変数のフォーマットされた値を使用する変数名のコンポーネントも、ID 変数値と PREFIX、DELIMITER および SUFFIX オプションからなる名前も、SAS 変数の命名規則に従う必要があります。その結果生成された名前は、VALIDVARNAME オプション設定に適した長さに切り捨てられます。

IDLABEL ステートメント

転置された変数のラベルを作成します。

制限事項: ID ステートメントの後に記述する必要があります。

例: “例 3: 転置された変数のラベル作成” (2054 ページ)

構文

IDLABEL *variable*;

必須引数

variable
ID ステートメントで名前を指定された変数にラベルを付けるためにプロシジャで使用される値を含む変数の名前を指定します。*variable* には文字または数値を指定できます。

注: IDLABEL ステートメントの結果を確認するには、PRINT プロシジャで LABEL オプションを使用して出力データセットを出力します。また、DATASETS プロシジャで CONTENTS ステートメントを使用して、出力データセットの内容を印刷することもできます。

VAR ステートメント

転置する変数をリストします。

例: “例 4: BY グループの転置” (2056 ページ)
“例 6: 統計分析用データの転置” (2060 ページ)

構文

VAR *variable(s)*;

必須引数

variable(s)
転置する変数の名前を 1 つ以上指定します。

詳細

- VAR ステートメントを省略すると、TRANSPOSE プロシジャでは、別のステートメントにリストされていない入力データセットのすべての数値変数が転置されます。
- 文字変数を転置する場合は、VAR ステートメントでそれらの文字変数をリストする必要があります。

  注: プロシジャで文字変数を転置すると、転置された変数はすべて文字変数になります。

結果:TRANSPOSE プロシジャ

出力データセット

TRANSPOSE プロシジャでは、PROC TRANSPOSE ステートメントで OUT=オプションを指定するかどうかに関係なく、常に出力データセットが生成されます。PROC TRANSPOSE では、出力データセットは印刷されません。PROC PRINT、PROC REPORT または別の SAS レポートツールを使用して、出力データセットを印刷します。

出力データセット変数

出力データセットには、次の変数が含まれています。

- 各変数の値をオブザベーションに転置した結果として生じる変数。
- PROC TRANSPOSE で、出力データセットの各オブザベーションの値のソースを識別するために作成される変数。この変数は文字変数で、その値は入力データセットから転置された変数の名前です。PROC TRANSPOSE では、この変数の名前はデフォルトで_NAME_になります。デフォルト名を無効にするには、NAME=オプションを使用します。_NAME_変数のラベルは **NAME OF FORMER VARIABLE** です。
- BY または COPY ステートメントの使用時に、PROC TRANSPOSE で入力データセットからコピーされる変数。これらの変数の名前と値は、入力データセットと同じです。これらの変数は、属性も同じです(例:種類、長さ、ラベル、入力形式、出力形式)。
- (プロシジャで転置される変数のいずれかにラベルが含まれる場合に)転置される変数の変数ラベルを値として含む文字変数。LABEL=オプションを使用して変数名を指定します。デフォルトは **_LABEL_** です。

  注: LABEL=オプションまたは NAME=オプションの値が、BY または COPY ステートメントに記述された変数と同じ場合、出力データセットには、転置された変数の名前またはラベルが値となる変数は含まれません。

転置された変数の属性

- 転置された変数はすべて、種類と長さが同じです。
- プロシジャで転置されている変数がすべて数値の場合、転置された変数は数値になります。したがって、数値変数にフォーマットされた値として文字列が含まれていると、未フォーマットの数値が転置されます。

- プロシジャで転置されている変数が文字の場合、転置された変数はすべて文字になります。フォーマットされた値として文字列が含まれている数値変数を転置すると、フォーマットされた値が転置されます。
- 転置された変数の長さは、転置する最長の変数の長さと等しくなります。

転置された変数の名前

PROC TRANSPOSE では、転置された変数の命名に次のルールが使用されます。

1. ID ステートメントで入力データセットの変数を 1 つまたは複数指定すると、そのフォーマットされた値が転置された変数の名前になります。複数の ID 変数を指定した場合、転置された変数の名前は、ID 変数の値の連結になります。DELMITER=オプションを指定した場合は、転置された変数の名前を生成するときに、ID 変数のフォーマットされた値の間にその値が挿入されます。
2. PREFIX=オプションで、転置された変数の名前の作成に使用する接頭辞を指定します。また、SUFFIX=オプションで、転置された変数の名前に追加する接尾辞も指定します。
3. ID ステートメント、PREFIX=オプションまたは SUFFIX=オプションを使用しない場合、PROC TRANSPOSE では、_NAME_という名前の入力変数が検索され、転置された変数の名前が取得されます。
4. ID ステートメントも PREFIX=オプションも使用せず、入力データセットに_NAME_という変数が含まれていない場合、PROC TRANSPOSE では、転置された変数に名前 COL1、COL2、…、COL*n* が割り当てられます。

例: TRANSPOSE プロシジャ

例 1: 単純な転置の実行

要素: PROC TRANSPOSE ステートメントオプション
OUT=

この例では、デフォルトの転置を実行し、従属ステートメントは使用しません。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=40;

data score;
   input Student $9. +1 StudentID $ Section $ Test1 Test2 Final;
   datalines;
Capalleti 0545 1  94 91 87
Dubose    1252 2  51 65 91
Engles    1167 1  95 97 97
Grant     1230 2  63 75 80
Krupski   2527 2  80 76 71
Lundsford 4860 1  92 40 86
McBane    0674 1  75 78 72
```

```
;

proc transpose data=score out=score_transposed;
run;

proc print data=score_transposed noobs;
    title 'Student Test Scores in Variables';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

SCORE データセットを作成します。 データセット SCORE には、学生の名前、ID 番号、および 2 つのテストと期末試験の成績が含まれます。

```
data score;
   input Student $9. +1 StudentID $ Section $ Test1 Test2 Final;
   datalines;
Capalleti 0545 1  94 91 87
Dubose    1252 2  51 65 91
Engles    1167 1  95 97 97
Grant     1230 2  63 75 80
Krupski   2527 2  80 76 71
Lundsford 4860 1  92 40 86
McBane    0674 1  75 78 72
;
```

データセットを転置します。 PROC TRANSPOSE では、数値変数 Test1、Test2 および Final のみが転置されます。これは、VAR ステートメントが記述されておらず、別のステートメントにも数値変数が記述されていないためです。OUT=では、転置結果がデータセット SCORE_TRANSPOSED に入力されます。

```
proc transpose data=score out=score_transposed;
run;
```

SCORE_TRANSPOSED データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=score_transposed noobs;
    title 'Student Test Scores in Variables';
run;
```

出力

出力データセット SCORE_TRANSPOSED の変数 COL1～COL7 に、学生の個々のスコアが含まれます。各オブザベーションに、1 つのテストのスコアがすべて含まれます。変数_NAME_には、転置された入力データセットの変数名が含まれます。

アウトプット 66.3　変数に含まれる学生のテストのスコア

```
Student Test Scores in Variables                        1 _NAME_    COL1    COL2    COL3    COL4
COL5    COL6    COL7 Test1      94      51      95      63      80      92      75 Test2
91      65      97      75      76      40      78 Final      87      91      97      80
71      86      72
```

例 2: 転置された変数の名前の指定

要素: PROC TRANSPOSE ステートメントオプション
NAME=
PREFIX=
ID ステートメント

データセット: SCORE

この例では、変数値とユーザー指定の値を使用して、転置された変数に名前を付けます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=40;

proc transpose data=score out=idnumber name=Test
     prefix=sn;
    id studentid;
run;

proc print data=idnumber noobs;
   title 'Student Test Scores';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

データセットを転置します。 PROC TRANSPOSE では、VAR ステートメントが記述されていないので、数値変数 Test1、Test2 および Final のみが転置されます。OUT=では、転置結果が IDNUMBER データセットに入力されます。NAME=では、プロシジャが転置する入力データセットの変数名を含む変数の名前として Test が指定されます。プロシジャでは、PREFIX=の値 sn と ID 変数 StudentID の値を使用して、転置された変数に名前が付けられます。

```
proc transpose data=score out=idnumber name=Test
     prefix=sn;
    id studentid;
run;
```

IDNUMBER データセットを出力します。 NOOBS オプションはオブザベーション番号の印刷を行いません。

```
proc print data=idnumber noobs;
   title 'Student Test Scores';
run;
```

出力

次のデータセットは、出力データセット IDNUMBER です。

アウトプット 66.4 *学生のテストのスコア*

```
Student Test Scores                                1 Test     sn0545    sn1252    sn1167    sn1230
sn2527    sn4860    sn0674 Test1      94        51        95        63        80        92        75
Test2      91        65        97        75        76        40        78 Final      87
91        97        80        71        86        72
```

例 3: 転置された変数のラベル作成

要素: PROC TRANSPOSE ステートメントオプション
PREFIX=
IDLABEL ステートメント

データセット: SCORE

この例では、IDLABEL ステートメントの変数値を使用して、転置された変数にラベルを付けます。

プログラム 1

```
options nodate pageno=1 linesize=80 pagesize=40;

proc transpose data=score out=idlabel name=Test
     prefix=sn;
   id studentid;

     idlabel student;
run;

  proc print data=idlabel label noobs;
     title 'Student Test Scores';
  run;

proc contents data=idlabel;
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

データセットを転置します。 PROC TRANSPOSE では、VAR ステートメントが記述されていないので、数値変数 Test1、Test2 および Final のみが転置されます。OUT=では、転

置結果が IDLABEL データセットに入力されます。NAME=では、プロシジャが転置する入力データセットの変数名を含む変数の名前として Test が指定されます。プロシジャでは、PREFIX=の値 sn と ID 変数 StudentID の値を使用して、転置された変数に名前が付けられます。

```
proc transpose data=score out=idlabel name=Test
     prefix=sn;
   id studentid;
```

出力変数にラベルを割り当てます。 PROC TRANSPOSE では、変数 Student の値を使用して、転置された変数にラベルが付けられます。プロシジャでは、_NAME_変数のラベルとして NAME OF FORMER VARIABLE が提供されます。

```
     idlabel student;
run;
```

IDLABEL データセットを出力します。 LABEL オプションを使用すると、PROC PRINT で、列ヘッダーの変数ラベルが印刷されます。NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
  proc print data=idlabel label noobs;
     title 'Student Test Scores';
  run;
```

IDLABEL 変数の名前とラベルを表示します。 PROC CONTENTS では、変数の名前とラベルが表示されます。

```
proc contents data=idlabel;
run;
```

出力 1

このデータセットは、出力データセット IDLABEL です。

アウトプット 66.5 *学生のテストのスコア(IDLABEL)*

```
Student Test Scores                              1 NAME OF FORMER VARIABLE Capalleti Dubose Engles
Grant Krupski Lundsford McBane Test1         94        51       95       63       80
92        75 Test2         91        65       97       75       76         40        78
Final         87        91       97       80       71         86        72
```

プログラム 2

```
proc contents data=idlabel;
run;
```

プログラムの説明

変数とラベルの名前を表示します。 PROC CONTENTS では、最初のプログラムで使用した変数名とラベルが表示されます。

```
proc contents data=idlabel;
run;
```

出力 2

次の出力では、PROC CONTENTS によって変数とラベルが表示されます。

アウトプット 66.6 CONTENTS プロシジャ

```
Student Test Scores                                    2 CONTENTS プロシジャ 変数と属性のアルファベット順
リスト # Variable Type Len Label 1 Test Char 8 NAME OF FORMER VARIABLE 2 sn0545 Num 8 Capalleti 8
sn0674 Num 8 McBane 4 sn1167 Num 8 Engles 5 sn1230 Num 8 Grant 3 sn1252 Num 8 Dubose 6 sn2527 Num 8
Krupski 7 sn4860 Num 8 Lundsford
```

例 4: BY グループの転置

要素: BY ステートメント
VAR ステートメント

他の要素: データセットオプション
RENAME=

この例では、BY グループの転置と転置する変数の選択方法を示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=40;

data fishdata;
   infile datalines missover;
   input Location & $10. Date date7.
         Length1 Weight1 Length2 Weight2 Length3 Weight3
         Length4 Weight4;
   format date date7.;
   datalines;
Cole Pond   2JUN95 31 .25 32 .3  32 .25 33 .3
Cole Pond   3JUL95 33 .32 34 .41 37 .48 32 .28
Cole Pond   4AUG95 29 .23 30 .25 34 .47 32 .3
Eagle Lake  2JUN95 32 .35 32 .25 33 .30
Eagle Lake  3JUL95 30 .20 36 .45
Eagle Lake  4AUG95 33 .30 33 .28 34 .42
;

proc transpose data=fishdata
     out=fishlength(rename=(col1=Measurement));

   var length1-length4;

   by location date;
run;

proc print data=fishlength noobs;
   title 'Fish Length Data for Each Location and Date';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

FISHDATA データセットを作成します。 FISHDATA のデータは、3 日間に 2 つの池で釣れた魚の体長と重量の測定値を日ごとに表したものです。生データが Location および Date 順に並べ替えられます。

```
data fishdata;
   infile datalines missover;
   input Location & $10. Date date7.
         Length1 Weight1 Length2 Weight2 Length3 Weight3
         Length4 Weight4;
   format date date7.;
   datalines;
Cole Pond   2JUN95 31 .25 32 .3  32 .25 33 .3
Cole Pond   3JUL95 33 .32 34 .41 37 .48 32 .28
Cole Pond   4AUG95 29 .23 30 .25 34 .47 32 .3
Eagle Lake  2JUN95 32 .35 32 .25 33 .30
Eagle Lake  3JUL95 30 .20 36 .45
Eagle Lake  4AUG95 33 .30 33 .28 34 .42
;
```

データセットを転置します。 OUT=では、転置結果が FISHLENGTH データセットに入力されます。RENAME=では、出力データセットの名前 COL1 が Measurement に変更されます。

```
proc transpose data=fishdata
     out=fishlength(rename=(col1=Measurement));
```

転置する変数を指定します。 VAR ステートメントでは、PROC TRANSPOSE で転置される変数が制限されます。

```
   var length1-length4;
```

出力データセットを BY グループに編成します。 BY ステートメントでは、Location と Date の値の一意な組み合わせごとに BY グループが作成されます。プロシジャでは、BY 変数は転置されません。

```
   by location date;
run;
```

FISHLENGTH データセットを出力します。 NOOBS オプションはオブザベーション番号の印刷を行いません。

```
proc print data=fishlength noobs;
   title 'Fish Length Data for Each Location and Date';
run;
```

出力

次のデータセットは、出力データセット FISHLENGTH です。PROC TRANSPOSE では、元のデータセットの各 BY グループに対して、転置する変数ごとに 1 つずつ、4 つのオブザベーションが作成されます。その BY グループに対して転置する変数の値が入力データセットにない場合、変数 Measurement(COL1 から名前を変更)に欠損値が表示されます。複数のオブザベーションで、Measurement に欠損値があります。たとえば、最後のオブザベーションでは、入力データに Eagle Lake での 04AUG95 の Length4 に対する値が含まれていないため、欠損値が表示されます。

アウトプット 66.7 魚の体長のデータ

```
Fish Length Data for Each Location and Date                  1 Location        Date    _NAME_
Measurement Cole Pond     02JUN95    Length1         31 Cole Pond     02JUN95    Length2         32
Cole Pond     02JUN95    Length3         32 Cole Pond     02JUN95    Length4         33 Cole
Pond     03JUL95    Length1         33 Cole Pond     03JUL95    Length2         34 Cole Pond
03JUL95    Length3         37 Cole Pond     03JUL95    Length4         32 Cole Pond     04AUG95
Length1         29 Cole Pond     04AUG95    Length2         30 Cole Pond     04AUG95
Length3         34 Cole Pond     04AUG95    Length4         32 Eagle Lake    02JUN95
Length1         32 Eagle Lake    02JUN95    Length2         32 Eagle Lake    02JUN95
Length3         33 Eagle Lake    02JUN95    Length4          .Eagle Lake    03JUL95
Length1         30 Eagle Lake    03JUL95    Length2         36 Eagle Lake    03JUL95
Length3          .Eagle Lake    03JUL95    Length4          .Eagle Lake    04AUG95
Length1         33 Eagle Lake    04AUG95    Length2         33 Eagle Lake    04AUG95
Length3         34 Eagle Lake    04AUG95    Length4          .
```

例 5: ID 変数の値が重複する場合の転置された変数名の指定

要素: PROC TRANSPOSE ステートメントオプション
LET

この例では、変数(ID)の値を使用して、ID 変数の値が重複していても、転置された変数の名前を指定する方法を示します。

プログラム

```
options nodate pageno=1 linesize=64 pagesize=40;

data stocks;
    input Company $14. Date $ Time $ Price;
    datalines;
Horizon Kites jun11 opening 29
Horizon Kites jun11 noon    27
Horizon Kites jun11 closing 27
Horizon Kites jun12 opening 27
Horizon Kites jun12 noon    28
Horizon Kites jun12 closing 30
SkyHi Kites   jun11 opening 43
SkyHi Kites   jun11 noon    43
SkyHi Kites   jun11 closing 44
SkyHi Kites   jun12 opening 44
SkyHi Kites   jun12 noon    45
SkyHi Kites   jun12 closing 45
;

proc transpose data=stocks out=close let;

   by company;

   id date;
 run;

proc print data=close noobs;
   title 'Closing Prices for Horizon Kites and SkyHi Kites';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=64 pagesize=40;
```

STOCKS データセットを作成します。 STOCKS には、競合する 2 つの凧製造会社の株価が含まれます。価格は、2 日間にわたり 1 日 3 回(開始時、正午、終了時)記録されます。入力データセットでは、Date 変数に重複する値が含まれることに注意してください。

```
data stocks;
    input Company $14. Date $ Time $ Price;
    datalines;
Horizon Kites jun11 opening 29
Horizon Kites jun11 noon    27
Horizon Kites jun11 closing 27
Horizon Kites jun12 opening 27
Horizon Kites jun12 noon    28
Horizon Kites jun12 closing 30
SkyHi Kites   jun11 opening 43
SkyHi Kites   jun11 noon    43
SkyHi Kites   jun11 closing 44
SkyHi Kites   jun12 opening 44
SkyHi Kites   jun12 noon    45
SkyHi Kites   jun12 closing 45
;
```

データセットを転置します。 LET では、各 BY グループの最後のオブザベーションのみが転置されます。PROC TRANSPOSE では、Price 変数のみが転置されます。OUT=では、転置結果が CLOSE データセットに入力されます。

```
proc transpose data=stocks out=close let;
```

出力データセットを BY グループに編成します。 BY ステートメントでは、会社ごとに 1 つずつ、2 つの BY グループが作成されます。

```
   by company;
```

転置された変数の名前を指定します。 Date の値が、転置された変数の名前として使用されます。

```
   id date;
 run;
```

CLOSE データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=close noobs;
   title 'Closing Prices for Horizon Kites and SkyHi Kites';
run;
```

出力

次のデータセットは、出力データセット CLOSE です。

アウトプット 66.8 終値

```
Closing Prices for Horizon Kites and SkyHi Kites       1 Company
_NAME_    jun11    jun12 Horizon Kites    Price       27       30 SkyHi
Kites      Price       44       45
```

例 6: 統計分析用データの転置

要素: COPY ステートメント
VAR ステートメント

この例では、多変量または単変量の反復測定分析に適したものになるようにデータを配置します。

データ元は、SAS System for Linear Models, Third Edition の第 8 章“Repeated-Measures Analysis of Variance”です。

プログラム 1

```
options nodate pageno=1 linesize=80 pagesize=40;

data weights;
   input Program $ s1-s7;
   datalines;
CONT  85 85 86 85 87 86 87
CONT  80 79 79 78 78 79 78
CONT  78 77 77 77 76 76 77
CONT  84 84 85 84 83 84 85
CONT  80 81 80 80 79 79 80
RI    79 79 79 80 80 78 80
RI    83 83 85 85 86 87 87
RI    81 83 82 82 83 83 82
RI    81 81 81 82 82 83 81
RI    80 81 82 82 82 84 86
WI    84 85 84 83 83 83 84
WI    74 75 75 76 75 76 76
WI    83 84 82 81 83 83 82
WI    86 87 87 87 87 87 86
WI    82 83 84 85 84 85 86
;

data split;
   set weights;
   array s{7} s1-s7;
   Subject + 1;
   do Time=1 to 7;
      Strength=s{time};
      output;
   end;
   drop s1-s7;
run;

proc print data=split(obs=15) noobs;
   title 'SPLIT Data Set';
```

```
   title2 'First 15 Observations Only';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

WEIGHTS データセットを作成します。 WEIGHTS のデータは、3 つの重量挙げプログラムの運動療法研究の結果を表しています。ここでは、CONT は対照群、RI は反復回数を増やすプログラム、WI は重量を増やすプログラムです。

```
data weights;
   input Program $ s1-s7;
   datalines;
CONT  85 85 86 85 87 86 87
CONT  80 79 79 78 78 79 78
CONT  78 77 77 77 76 76 77
CONT  84 84 85 84 83 84 85
CONT  80 81 80 80 79 79 80
RI    79 79 79 80 80 78 80
RI    83 83 85 85 86 87 87
RI    81 83 82 82 83 83 82
RI    81 81 81 82 82 83 81
RI    80 81 82 82 82 84 86
WI    84 85 84 83 83 83 84
WI    74 75 75 76 75 76 76
WI    83 84 82 81 83 83 82
WI    86 87 87 87 87 87 86
WI    82 83 84 85 84 85 86
;
```

SPLIT データセットを作成します。 この DATA ステップでは、WEIGHT が再配置され、データセット SPLIT が作成されます。DATA ステップでは、強さの値が転置され、2 つの新しい変数 Time と Subject が作成されます。SPLIT には、反復測定値ごとに 1 つずつオブザベーションが含まれます。PROC GLM ステップで SPLIT を使用すると、単変量反復測定分析を行うことができます。

```
data split;
   set weights;
   array s{7} s1-s7;
   Subject + 1;
   do Time=1 to 7;
      Strength=s{time};
      output;
   end;
   drop s1-s7;
run;
```

SPLIT データセットを出力します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。OBS=データセットオプションは、出力を最初の 15 のオブザベーションのみに制限します。SPLIT には 105 のオブザベーションがあります。

```
proc print data=split(obs=15) noobs;
```

```
   title 'SPLIT Data Set';
   title2 'First 15 Observations Only';
run;
```

出力 1

アウトプット 66.9 *SPLIT データセット*

SPLIT Data Set					1 First 15 Observations Only		
Program	Subject	Time	Strength	CONT	1	1	85
CONT	1	2	85	CONT	1	3	86
CONT	1	4	85	CONT	1	5	87
CONT	1	6	86	CONT	1	7	87
CONT	2	1	80	CONT	2	2	79
CONT	2	3	79	CONT	2	4	78
CONT	2	5	78	CONT	2	6	79
CONT	2	7	78	CONT	3	1	78

プログラム 2

```
options nodate pageno=1 linesize=80 pagesize=40;

proc transpose data=split out=totsplit prefix=Str;

   by program subject;
   copy time strength;

   var strength;
run;

proc print data=totsplit(obs=15) noobs;
   title  'TOTSPLIT Data Set';
   title2 'First 15 Observations Only';
run;
```

プログラムの説明

SAS システムオプションを設定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

SPLIT データセットを転置します。 PROC TRANSPOSE では、SPLIT を転置して TOTSPLIT を作成します。TOTSPLIT データセットには、SPLIT と同じ変数に加えて、強さの測定値それぞれに対する変数(Str1-Str7)が含まれます。TOTSPLIT は、多変量反復測定分析または単変量反復測定分析に使用できます。

```
proc transpose data=split out=totsplit prefix=Str;
```

出力データセットを BY グループに編成し、各 BY グループに未転置の値を入力します。 BY および COPY ステートメントの変数は転置されません。TOTSPLIT には、SPLIT と同じ値の変数 Program、Subject、Time、Strength が含まれます。BY ステートメントでは、各 BY グループの最初のオブザベーションが作成されます。これには転置された Strength の値が含まれます。COPY ステートメントでは、転置はせずに Time と Strength の値がコピーされ、各 BY グループのその他のオブザベーションが作成されます。

```
   by program subject;
   copy time strength;
```

転置する変数を指定します。 VAR ステートメントでは、転置する唯一の変数として Strength 変数が指定されます。

```
   var strength;
run;
```

TOTSPLIT データセットを出力します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。OBS=データセットオプションは、出力を最初の 15 のオブザベーションのみに制限します。SPLIT には 105 のオブザベーションがあります。

```
proc print data=totsplit(obs=15) noobs;
   title  'TOTSPLIT Data Set';
   title2 'First 15 Observations Only';
run;
```

出力 2

次の出力では、値の欠損している TOTSPLIT の変数は、多変量反復測定分析でのみ使用されます。PROC GLM の MODEL ステートメントでは欠損値のあるオブザベーションは無視されるので、欠損値があることでこのデータセットが反復測定分析で使用されなくなることはありません。

アウトプット 66.10 *TOTSPLIT データセット*

```
TOTSPLIT Data Set                              1 First 15 Observations Only Program Subject Time
Strength  _NAME_  Str1 Str2 Str3 Str4 Str5 Str6 Str7 CONT      1      1     85    Strength  85
85   86   85   87   86   87 CONT      1      2
85               ..    .    .    .    .    .    .CONT      1      3
86               .    .    .    .    .    .    .CONT      1      4
85               .    .    .    .    .    .    .CONT      1      5
87               .    .    .    .    .    .    .CONT      1      6
86               .    .    .    .    .    .    .CONT      1      7
87               .    .    .    .    .    .    .CONT      2      1     80    Strength  80   79
79   78   78   79   78 CONT      2      2
79               ..    .    .    .    .    .    .CONT      2      3
79               .    .    .    .    .    .    .CONT      2      4
78               .    .    .    .    .    .    .CONT      2      5
78               .    .    .    .    .    .    .CONT      2      6
79               .    .    .    .    .    .    .CONT      2      7
78               .    .    .    .    .    .    .CONT      3      1     78    Strength  78   77
77   77   76   76   77
```

67 章

XSL プロシジャ

概要: XSL プロシジャ

Extensible Style Sheet Language (XSL)プロシジャの動作について

XSL プロシジャは、XML ドキュメントを、HTML、テキスト、または別の種類の XML ドキュメントなどの別の出力形式に変換します。PROC XSL は入力 XML ドキュメントを読み込み、XSL スタイルシートを使用して変換して、出力ファイルを書き出します。

PROC XSL は、XML ドキュメントを変換するために、Saxonica の Saxon-EE version 9.3 ソフトウェアアプリケーションを使用します。このアプリケーションは、XML ドキュメント処理用ツールを集めたものです。XSLT プロセッサには、XSLT 2.0 基準が導入されています。Saxon の詳細については、ウェブサイト About Saxon を参照してください。

XSL について

XSL は、XML でエンコードされるファイルの変換方法の説明を可能にする一群の変換言語です。言語には、次が含まれます。

- XML ドキュメントを変換するための XSL Transformations (XSLT)
- XSLT によって使用される、XML ドキュメントの一部を選択するための XML Path Language (XPath)

XSLT 基準の詳細については、ウェブサイト XSL Transformations (XSLT) Version 2.0 を参照してください。

構文: XSL プロシジャ

PROC XSL IN=*fileref* | '*external-file*' OUT=*fileref* | '*external-file*' XSL=*fileref* | '*external-file*';
PARAMETER '*parameter*'='*value*';

ステートメント	タスク	例
"PROC XSL ステートメント"	XML ドキュメントを変換する	Ex. 1, Ex. 2, Ex. 3
"PARAMETER ステートメント"	パラメータを XSL スタイルシートに渡して値を設定する	Ex. 2, Ex. 3

PROC XSL ステートメント

XML ドキュメントを変換します。

例: "例 1: XML ドキュメントの別の XML ドキュメントへの変換" (2067 ページ)

構文

PROC XSL IN=*fileref* | '*external-file*' OUT=*fileref* | '*external-file*' XSL=*fileref* | '*external-file*';

必須引数

IN=*fileref* | '*external-file*'
入力ファイルを指定します。ファイルは、正しい形式の XML ドキュメントである必要があります。

fileref
入力 XML ドキュメントに割り当てられている SAS ファイル参照名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'*external-file*'
入力 XML ドキュメントの物理的な場所です。完全パス名とファイル名を含みます。物理名を一重引用符または二重引用符で囲みます。最大長は、200 文字です。

例 "例 1: XML ドキュメントの別の XML ドキュメントへの変換" (2067 ページ)

OUT=*fileref* | '*external-file*'
出力ファイルを指定します。

fileref
出力ファイルに割り当てられる SAS ファイル参照名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'*external-file*'
出力ファイルの物理的な場所です。完全パス名とファイル名を含みます。物理名を一重引用符または二重引用符で囲みます。最大長は、200 文字です。

例 “例 1: XML ドキュメントの別の XML ドキュメントへの変換” (2067 ページ)

XSL=*fileref* | '*external-file*'
XML ドキュメントを変換する XSL スタイルシートを指定します。XSL スタイルシートは、XSLT 言語を使用した XML ドキュメントの変換方法を説明するファイルです。正しい形式の XML ドキュメントである必要があります。

fileref
XSL スタイルシートに割り当てられる SAS ファイル参照名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'*external-file*'
XSL スタイルシートの物理的な場所です。完全パス名とファイル名を含みます。物理名を一重引用符または二重引用符で囲みます。最大長は、200 文字です。

別名 XSLT

例 “例 1: XML ドキュメントの別の XML ドキュメントへの変換” (2067 ページ)

PARAMETER ステートメント

XSL スタイルのパラメータのインスタンスを指定した値に変更します。

例: “例 2: 文字列パラメータ値の XSL スタイルシートへの受け渡し” (2069 ページ)
“例 3: 数値パラメータ値の XSL スタイルシートへの受け渡し” (2072 ページ)

構文

PARAMETER '*parameter*’=’*value*';

必須引数

'*parameter*'='*value*'
XSL スタイルシートに渡されるパラメータ名と値を指定します。PROC XSL は、スタイルシートのパラメータのインスタンスを、指定した値に変更するものです。指定したパラメータが XSL スタイルシート内に存在する必要があります。パラメータ名を一重引用符または二重引用符で囲みます。指定した値は文字列または数値になります。値が文字列である場合は、値を一重引用符か二重引用符で囲みます。

例: XSL プロシジャ

例 1: XML ドキュメントの別の XML ドキュメントへの変換

要素: PROC XSL ステートメント

詳細

次の PROC XSL 例では、XML ドキュメントを別の XML ドキュメントに変換しています。

これは、車に関するデータを含む XMLInput.xml という名前の入力 XML ドキュメントです。第 2 レベルの繰り返し要素は、それぞれ特定の車を、そのモデルと年度に関する情報を含むネスト要素を使用して記述しています。製造メーカー情報は、第 2 レベルの繰り返し要素の属性です。

```
<?xml version="1.0" ?>
<vehicles>
  <car make="Ford">
    <model>Mustang</model>
    <year>1965</year>
  </car>
  <car make="Chevrolet">
    <model>Nova</model>
    <year>1967</year>
  </car>
</vehicles>
```

これは、XML の変換方法を説明する XSLTransform.xsl という名前の XSL スタイルシートです。変換は<root>をルートのエンクロージング要素として、<model>を第 2 レベルの繰り返し要素として作成します。出力 XML ドキュメントの各<model> 要素は、<car>要素からの値と、入力 XML ドキュメントからの make=属性を含みます。

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/vehicles">
       <root> <xsl:apply-templates select="car"/> </root>
</xsl:template>

<xsl:template match="car">
    <model make="{@make}">
       <xsl:value-of select="model" />
    </model>
</xsl:template>

</xsl:stylesheet>
```

プログラム

```
proc xsl
   in='C:\XmlInput.xml'
   xsl='C:\XslTransform.xsl'
   out='C:\XmlOutput.xml';
run;
```

プログラムの説明

PROC XSL ステートメントを抽出します。 PROC XSL ステートメントは、入力 XML ドキュメント、XSL スタイルシート、出力 XML ドキュメントを指定します。

```
proc xsl
   in='C:\XmlInput.xml'
   xsl='C:\XslTransform.xsl'
   out='C:\XmlOutput.xml';
run;
```

出力:XML ドキュメントの別の XML ドキュメントへの変換

アウトプット 67.1 *変換された XML ドキュメントの PROC XSL 出力*

```
<?xml version="1.0" encoding="UTF-8"?> <root> <model make="Ford">Mustang</
model> <model make="Chevrolet">Nova</model> </root>
```

例 2: 文字列パラメータ値の XSL スタイルシートへの受け渡し

要素: PROC XSL ステートメント
PARAMETER statement

詳細

この例では、PROC XSL を使用して、文字列値を XSL スタイルシートのパラメータに渡す方法を示します。パラメータは、値を設定できるスタイルシート内にある名前がついている変数であり、生成した出力を簡単にカスタマイズできる方法です。

これは Format.xsl という名前の XSL スタイルシートです。XSL スタイルシートは、入力 XML ドキュメントから要素と属性を抽出して HTML 出力を生成します。このスタイルシートには、宣言パラメータ DateVar が含まれます。

```
<?xml version="1.0" encoding="UTF-8"?>
  <xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:param name="DateVar"/>
    <xsl:template match="TABLE">
     <html>
     <head/>
     <body>
     <h2><xsl:value-of select="$DateVar"/></h2>
     <table border="1" rules="all">
     <tr>
        <td>Name</td>
        <td>Sex</td>
        <td>Age</td>
        <td>Height</td>
        <td>Weight</td>
     </tr>
     <xsl:apply-templates/>
```

```
    </table>
    </body>
   </html>
   </xsl:template>
   <xsl:template match="CLASS">
    <tr>
        <td><xsl:value-of select="Name"/></td>
        <td><xsl:value-of select="Sex"/></td>
        <td><xsl:value-of select="Age"/></td>
        <td><xsl:value-of select="Height"/></td>
        <td><xsl:value-of select="Weight"/></td>
    </tr>
   </xsl:template>
  </xsl:stylesheet>
```

Class.xml という名前の入力 XML ドキュメントには、クラスルームデータが含まれます。

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
   <CLASS>
      <Name> Alfred </Name>
      <Sex> M </Sex>
      <Age> 14 </Age>
      <Height> 69 </Height>
      <Weight> 112.5 </Weight>
   </CLASS>
   <CLASS>
      <Name> Alice </Name>
      <Sex> F </Sex>
      <Age> 13 </Age>
      <Height> 56.5 </Height>
      <Weight> 84 </Weight>
   </CLASS>
   <CLASS>
      <Name> Barbara </Name>
      <Sex> F </Sex>
      <Age> 13 </Age>
      <Height> 65.3 </Height>
      <Weight> 98 </Weight>
   </CLASS>
   .
   .
   .
   <CLASS>
      <Name> Thomas </Name>
      <Sex> M </Sex>
      <Age> 11 </Age>
      <Height> 57.5 </Height>
      <Weight> 85 </Weight>
   </CLASS>
   <CLASS>
      <Name> William </Name>
      <Sex> M </Sex>
      <Age> 15 </Age>
      <Height> 66.5 </Height>
      <Weight> 112 </Weight>
```

```
   </CLASS>
</TABLE>
```

プログラム

```
proc xsl
   in='C:\XSL\Class.xml'
   xsl='C:\XSL\Format.xsl'
   out='C:\XSL\Class.html';

   parameter 'DateVar'="Report Date: &sysdate";
run;
```

プログラムの説明

PROC XSL ステートメントを抽出します。 PROC XSL ステートメントは、入力 XML ドキュメント、XSL スタイルシート、出力 HTML ファイルを指定します。

```
proc xsl
   in='C:\XSL\Class.xml'
   xsl='C:\XSL\Format.xsl'
   out='C:\XSL\Class.html';
```

パラメータ値を XSL スタイルシートに渡します。 PARAMETER ステートメントは、パラメータ DateVar の文字列値を XSL スタイルシートに渡します。この値はマクロ変数参照を含んでいるため、必ず二重引用符で囲んでください。

```
   parameter 'DateVar'="Report Date: &sysdate";
run;
```

出力:文字列パラメータ値の XSL スタイルシートへの受け渡し

アウトプット 67.2 *PROC XSL Output Class.html*

Report Date: 15NOV12

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

例 3: 数値パラメータ値の XSL スタイルシートへの受け渡し

要素: PROC XSL ステートメント
PARAMETER statement

詳細

この例では、PROC XSL を使用して、数値を XSL スタイルシートのパラメータに渡す方法を示します。

これは、Discount.xsl という名前の XSL スタイルシートです。XSL スタイルシートは、入力 XML ドキュメントから要素と属性を抽出して XML 出力を生成します。このスタイ

ルシートには、宣言パラメータ DiscountPct が含まれます。PROC XSL は数値の状態でパラメータへの受け渡しを行い、割引額が算出されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes" />
  <xsl:param name="DiscountPct" />

  <xsl:template match="table">
    <xsl:copy>
      <xsl:apply-templates select="product" />
    </xsl:copy>
  </xsl:template>

  <xsl:template match="product">
    <xsl:copy>
      <xsl:copy-of select="category|type|size|gender|price" />
      <discount>
        <xsl:value-of select="$DiscountPct" />
      </discount>
      <discountAmount>
        <xsl:value-of select="(price * $DiscountPct)" />
      </discountAmount>
      <xsl:copy-of select="in_stock|ship_type" />
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

Product.xml という名前の入力 XML ドキュメントには、製品データが含まれます。

```
<?xml version="1.0"?>
<table>
 <product>
   <category>shoe</category>
   <type>Nike Air</type>
   <size>12</size>
   <gender>male</gender>
   <price>99</price>
   <in_stock>yes</in_stock>
   <ship_type>overnight</ship_type>
 </product>
</table>
```

プログラム

```
proc xsl
   in='C:\XSL\Product.xml'
   xsl='C:\XSL\Discount.xsl'
   out='C:\XSL\Calculated.xml';

   parameter 'DiscountPct'=.20;
run;
```

プログラムの説明

PROC XSL ステートメントを抽出します。 PROC XSL ステートメントは、入力 XML ドキュメント、XSL スタイルシート、出力 XML ドキュメントを指定します。

```
proc xsl
   in='C:\XSL\Product.xml'
   xsl='C:\XSL\Discount.xsl'
   out='C:\XSL\Calculated.xml';
```

パラメータ値を XSL スタイルシートに渡します。 PARAMETER ステートメントは、パラメータ DiscountPct の数値を XSL スタイルシートに渡します。出力 XML ドキュメントに、割引値と算出された割引額が含まれるようになります。

```
   parameter 'DiscountPct'=.20;
run;
```

出力:数値パラメータ値の XSL スタイルシートへの受け渡し

アウトプット 67.3 *PROC XSL Output Calculated.xml*

```
<?xml version="1.0" encoding="UTF-8"?> <table> <product> <category>shoe</
category> <type>Nike Air</type> <size>12</size> <gender>male</gender>
<price>99</price> <discount>0.2</discount> <discountAmount>19.8</
discountAmount> <in_stock>yes</in_stock> <ship_type>overnight</ship_type> </
product> </table>
```

3 部

付録

付録 1

基本的な SAS 統計プロシジャ

基本的な SAS 統計プロシジャの概要

この付録では、基本的な統計に対する Base SAS プロシジャの出力の解釈に必要な一部の統計的概念について簡単に説明しています。また、この付録には統計表記、式、Base SAS プロシジャで一般的な統計に使用される標準キーワードがリストされています。簡単な例で、統計的概念を示します。

表 A1.1 (2079 ページ) に、最も一般的な統計量と、それらを計算するプロシジャを表示します。

キーワードと式

単純統計量

Base SAS プロシジャでは、標準化された一連のキーワードを使用して統計量を表します。これらのキーワードを SAS ステートメントで指定して、統計量が表示されるように、または出力データセットに保存されるように要求します。

次の表記では、合計が分析済変数の非欠損値を含むオブザベーションを超え、表示されている部分以外では、非欠損の重みと度数を超えています。

x_i
は、オブザベーション i に対して分析された変数の非欠損値です。

f_i
は、FREQ ステートメントを使用する場合、x_iに関連付けられる度数です。FREQ ステートメントを省略する場合、$f_i = 1$ (すべての i に対し)となります。

w_i
は、WEIGHT ステートメントを使用する場合、x_iに関連付けられている重みです。基本プロシジャは欠損した重みを含むx_iの値を分析から自動的に除外します。

デフォルトで、基本プロシジャは負の重みをゼロとして処理します。ただし、PROC ステートメントで EXCLNPWGT オプションを使用する場合、プロシジャは非正の重みを含むx_iの値も除外します。PROC TTEST や PROC GLM などの SAS/STAT プロシジャのほとんどが、デフォルトで非正の重みを持つ値を除外します。

WEIGHT ステートメントを省略すると、$w_i = 1$ (すべての i に対し)になります。

n
は、x_i、Σf_iの非欠損値の数です。EXCLNPWGT オプションと WEIGHT ステートメントを使用する場合、nは、正の重みを含む非欠損値の数です。

$\overline{x}$
平均です

$$\Sigma w_i x_i / \Sigma w_i$$

s^2
分散です

$$\frac{1}{d} \Sigma w_i (x_i - \overline{x})^2$$

dは、PROC ステートメントで指定する分散の計算のための分母(VARDEF=オプション)です。有効な値は次のとおりです。

When VARDEF=	d equals
N	n
DF	$n - 1$

WEIGHT	Σw_i
WDF	$\Sigma w_i - 1$

デフォルトは DF です。

z_i

は標準化された変数です

$$(x_i - \bar{x})/s$$

統計量ごとの標準キーワードおよび式が後に続きます。一部の式ではキーワードを使用して、対応する統計量を指定します。

表 A1.1 *最も一般的な基本統計量*

統計量	PROC MEANS および SUMMARY	PROC UNIVARIATE	PROC TABULATE	PROC REPORT	PROC CORR	PROC SQL
欠損値の数	X	X	X	X		X
非欠損値の数	X	X	X	X	X	X
オブザベーション数	X	X				X
重みの合計	X	X	X	X	X	X
平均値	X	X	X	X	X	X
合計	X	X	X	X	X	X
極値	X	X				
最小	X	X	X	X	X	X
最大	X	X	X	X	X	X
範囲	X	X	X	X		X
未修正平方和	X	X	X	X	X	X
修正済み平方和	X	X	X	X	X	X
分散	X	X	X	X	X	X
共分散					X	
標準偏差	X	X	X	X	X	X
平均の標準誤差	X	X	X	X		X
変動係数	X	X	X	X		X
歪度	X	X	X			

統計量	PROC MEANS および SUMMARY	PROC UNIVARIATE	PROC TABULATE	PROC REPORT	PROC CORR	PROC SQL
尖度	X	X	X			
信頼限界						
(平均)	X	X	X			
(分散)		X				
(分位点)		X				
中央値	X	X	X	X	X	
モード	X	X	X	X		
パーセント点/十分位点/分位点	X	X	X	X		
t 検定						
(平均=0)	X	X	X	X		X
(平均=) μ_0		X				
位置のノンパラメトリック検定		X				
正規性の検定		X				
相関係数					X	
Cronbach のアルファ統計量					X	

記述統計量

記述統計量のキーワードは、次のとおりです。

CSS
修正済平方和です。次のように計算されます

$$\Sigma\, w_i\left(x_i - \bar{x}\right)^2$$

CV
は単位がパーセントの変動係数です。次のように計算されます。

$$(100s)/\bar{x}$$

KURTOSIS | KURT
は尖度、度数の重みの尺度です。VARDEF=DF の場合、尖度は次のように計算されます。

$$c_{4_n}\,\Sigma\, z_i^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

c_{4_n} は、$\frac{n(n+1)}{(n-1)(n-2)(n-3)}$です。重み付けられた歪度は、次のように計算されます

$$= c_{4_n} \Sigma \left((x_i - \overline{x}) / \widehat{\sigma_i} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

$$= c_{4_n} \Sigma w_i^2 \left((x_i - \overline{x}) / \widehat{\sigma} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

VARDEF=N の場合、尖度は次のように計算されます。

$$= \frac{1}{n} \Sigma z_i^4 - 3$$

また、重みの付いた尖度は、次のように計算されます

$$= \frac{1}{n} \Sigma \left((x_i - \overline{x}) / \widehat{\sigma_i} \right)^4 - 3$$

$$= \frac{1}{n} \Sigma w_i^2 \left((x_i - \overline{x}) / \widehat{\sigma} \right)^4 - 3$$

σ_i^2は、σ^2 / w_iです。式は、変換$w_i^* = z w_i$, $z > 0$で変わりません。VARDEF=WDF または VARDEF=WEIGHT を使用する場合、歪度は欠損に設定されます。

注: PROC MEANS および PROC TABULATE は、重みの付いた尖度を計算しません。

MAX
は、x_iの最大値です。

MEAN
は、算術平均$\overline{x}$です。

MIN
は、x_iの最小値です。

MODE
は、x_iの最頻値です。

注: QMETHOD=P2 の場合、PROC REPORT、PROC MEANS および PROC TABULATE は MODE を計算しません。

N
は、欠損していないx_i値の数です。1 未満のf_i、欠損値に等しいw_i、または$w_i \leq 0$ (EXCLNPWGT オプションの使用時)のオブザベーションは分析から除外され、N の計算に含まれません。

NMISS
は、欠損しているx_i値の数です。1 未満のf_i、欠損値に等しいw_i、または$w_i \leq 0$ (EXCLNPWGT オプションの使用時)のオブザベーションは分析から除外され、NMISS の計算に含まれません。

NOBS
オブザベーションの総計で、N と NMISS の合計として計算されます。ただし、WEIGHT ステートメントを使用する場合、NOBS は N、NMISS、および欠損している重みまたは非正の重みのために除外されたオブザベーションの数の合計として計算されます。

RANGE
範囲で、最大値と最小値の間の差として計算されます。

SKEWNESS | SKEW
歪度です。1 つの方向でより大きくなる偏差の傾向を測定します。VARDEF=DF の場合、歪度は次のように計算されます

$$c_{3_n} \Sigma z_i^3$$

c_{3_n} は、$\frac{n}{(n-1)(n-2)}$ です。重み付けられた歪度は、次のように計算されます。

$$= c_{3_n} \Sigma \left((x_i - \overline{x}) / \widehat{\sigma_j} \right)^3$$
$$= c_{3_n} \Sigma w_i^{3/2} \left((x_i - \overline{x}) / \widehat{\sigma} \right)^3$$

VARDEF=N の場合、歪度は次のように計算されます

$$= \frac{1}{n} \Sigma z_i^3$$

また、重み付けられた歪度は、次のように計算されます。

$$= \frac{1}{n} \Sigma \left((x_i - \overline{x}) / \widehat{\sigma_j} \right)^3$$
$$= \frac{1}{n} \Sigma w_i^{3/2} \left((x_i - \overline{x}) / \widehat{\sigma} \right)^3$$

式は、変換$w_i^* = zw_i,\ z > 0$で変わりません。VARDEF=WDF または VARDEF=WEIGHT を使用する場合、歪度は欠損に設定されます。

注: PROC MEANS および PROC TABULATE は、重み付けられた歪度を計算しません。

STDDEV|STD
は、標準偏差 s です。分散の平方根、s^2として計算されます。

STDERR | STDMEAN
は平均の標準誤差です。VARDEF=DF (デフォルト)の場合、次のように計算されます。

$$s / \sqrt{\Sigma w_i}$$

デフォルトは VARDEF=DF です。その他の場合、STDERR は欠損に設定されます。

SUM
合計です。次のように計算されます。

$$\Sigma w_i x_i$$

SUMWGT
は、重みの合計です。W次のように計算されます。

$$\Sigma w_i$$

USS
無修正平方和です。次のように計算されます

$$\Sigma w_i x_i^2$$

VAR
は、分散s^2です。

分位数と関連統計量

分位点のキーワードおよび関連する統計量は、次のとおりです。

MEDIAN
中央値です。

P1
最初(1st)のパーセント点です。

P5
5 番目(5th)のパーセント点です。

P10
10 番目(5th)のパーセント点です。

P90
90 番目(5th)のパーセント点です。

P95
95 番目(5th)のパーセント点です。

P99
99 番目(5th)のパーセント点です。

Q1
下位の四分位(25 番目(25th)のパーセント点)です。

Q3
上位の四分位(75 番目(75th)のパーセント点)です。

QRANGE
四分位範囲です。次のように計算されます。

$$Q_3 - Q_1$$

QNTLDEF=オプション(PROC UNIVARIATE の PCTLDEF=)を使用して、プロシジャがパーセント点の計算に使用する方法を指定します。nを変数の非欠損値数にし、$x_1, x_2, \ldots, x_n$で、x_1が最小値、x_2が次の最小値、x_nが最大値になるように、変数の順序付けされた値を表します。t 番目のパーセント点(0 から 1 まで)の場合、$p = t/100$にします。次に、jをnpの整数部分として、gをnpまたは$(n+1)p$の小数部分として、次のようになるように定義します。

$$\begin{aligned} np &= j + g && \text{when QNTLDEF} = 1, 2, 3, \text{or } 5 \\ (n+1)p &= j + g && \text{when QNTLDEF} = 4 \end{aligned}$$

ここで QNTLDEF=は、後に続く表のように、プロシジャが t 番目のパーセント点の計算に使用する方法を指定します。

WEIGHT ステートメントを使用する場合、t 番目のパーセント点が次のように計算されます。

$$y = \begin{cases} \frac{1}{2}\left(x_i + x_{i+1}\right) & \text{if } \sum_{j=1}^{i} w_j = pW \\ x_{i+1} & \text{if } \sum_{j=1}^{i} w_j < pW < \sum_{j=1}^{i+1} w_j \end{cases}$$

w_jはx_iに関連付けられた重みであり、$W = \sum_{i=1}^{n} w_i$は重みの合計です。オブザベーションの重みが同じ場合、重み付けられているパーセント点は、QNTLDEF=5 の重み付けされていないパーセント点と同じです。

表 A1.2 *四分位範囲統計量の計算方法*

QNTLDEF=	説明	式	
1	次の重み付けられた平均 x_{np}	$y = (1-g)x_j + gx_{j+1}$	
		x_oは次のように解釈されます。x_1	
2	次に最も近い番号が付けられたオブザベーション np	$y = x_i$	の場合 $g \neq \frac{1}{2}$
		$y = x_j$	$g = \frac{1}{2}$で、jが偶数の場合
		$y = x_{j+1}$	$g = \frac{1}{2}$で、jが奇数の場合
		i は、次の小数部分です。$np + \frac{1}{2}$	
3	経験分布関数	$y = x_j$	の場合 $g = 0$
		$y = x_{j+1}$	の場合 $g > 0$
4	次を目的とした重み付けられた平均 $x_{(n+1)p}$	$y = (1-g)x_j + gx_{j+1}$	
		x_{n+1}は次のように解釈されます。x_n	
5	平均化を含む経験分布関数	$y = \frac{1}{2}\left(x_j + x_{j+1}\right)$	の場合 $g = 0$
		$y = x_{j+1}$	の場合 $g > 0$

仮説検定統計量

仮説検定統計量のキーワードは、次のとおりです。

T
スチューデントの t 統計量です。母平均がμ_0と等しい帰無仮説を検定します。

$$\frac{\overline{x} - \mu_0}{s/\sqrt{\sum w_i}}$$

デフォルトで、μ_0はゼロと等しくなります。PROC UNIVARIATE ステートメントで MU0=オプションを使用して、μ_0を指定できます。デフォルトの分散の計算のための分母となる VARDEF=DF を使用する必要があります。その他の場合、T は欠損に設定されます。

デフォルトで、WEIGHT ステートメントの使用時は、プロシジャは自由度に非正の重みを含むx_i値をカウントします。PROC ステートメントで EXCLNPWGT オプションを使用して、非正の重みを含む値を除外します。PROC TTEST、PROC GLM などの SAS/STAT プロシジャのほとんどが非正の重みを含む値を自動的に除外します。

PROBT | PRT
両側 p 値(スチューデントの t 統計量 T に対する)であり、$n-1$自由度が含まれます。この値は、このサンプルで観測されるよりも多くの極値 T を取得する帰無仮説での確率です。

平均の信頼限界

信頼限界のキーワードは、次のとおりです。

CLM
平均値の両側信頼限界です。平均の両側$100(1-\alpha)$パーセント信頼区間には、上限と下限があります。

$$\bar{x} \pm t_{(1-\alpha/2;n-1)}\frac{s}{\sqrt{\Sigma w_i}}$$

sは$\sqrt{\frac{1}{n-1}\Sigma\left(x_i-\bar{x}\right)^2}$、$t_{(1-\alpha/2;n-1)}$は$(1-\alpha/2)$スチューデントの t 統計量の臨界値(自由度が$n-1$)で、αは ALPHA=オプションの値で、デフォルトで 0.05 になります。デフォルトの分散の計算のための分母となる VARDEF=DF を使用する場合を除いて、CLM は欠損に設定されます。

LCLM
平均値より下の片側信頼限界です。平均の片側$100(1-\alpha)$パーセント信頼区間には、下限があります。

$$\bar{x} - t_{(1-\alpha;n-1)}\frac{s}{\sqrt{\Sigma w_i}}$$

デフォルトの分散の計算のための分母となる VARDEF=DF を使用する場合を除いて、LCLM は欠損に設定されます。

UCLM
平均より上の片側信頼限界です。平均の片側$100(1-\alpha)$パーセント信頼区間には、上限があります。

$$\bar{x} + t_{(1-\alpha;n-1)}\frac{s}{\sqrt{\Sigma w_i}}$$

デフォルトの分散の計算のための分母となる VARDEF=DF を使用する場合を除いて、UCLM は欠損に設定されます。

重みの使用

重みの使用と例については、“WEIGHT” (74 ページ)を参照してください。

要約プロシジャのデータの必要条件

次に、重みなしの統計量を計算するための最小データ要件を示します。推奨されるサンプルサイズは記載されていません。統計量は、VARDEF=DF (デフォルト)で、次の要件が満たされていない場合は欠損としてレポートされます。

- N および NMISS は、欠損オブザベーション数、または非欠損オブザベーション数に関係なく計算されます。
- SUM、MEAN、MAX、MIN、RANGE、USS、CSS には、非欠損オブザベーションが少なくとも 1 つ必要です。
- VAR、STD、STDERR、CV、T、PRT、PROBT には、非欠損オブザベーションが少なくとも 2 つ必要です。
- SKEWNESS には、非欠損オブザベーションが少なくとも 3 つ必要です。
- KURTOSIS には、非欠損オブザベーションが少なくとも 4 つ必要です。
- SKEWNESS、KURTOSIS、T、PROBT、PRT では、STD がゼロより大きい必要があります。
- CV では、MEAN がゼロに等しくない必要があります。
- CLM、LCLM、UCLM、STDERR、T、PRT、PROBT では、VARDEF=DF である必要があります。

統計的手法の知識

母集団とパラメータ

通常、対象となる要素は明確に定義されています。この一連の要素は*ユニバース*と呼ばれ、これらの要素と関連する一連の値は値の*母集団*と呼ばれます。統計用語の*母集団*は、人々とは関係ありません。統計的な母集団は、人々の集まりではなく、値の集まりです。たとえば、ユニバースは特定の学校の全生徒で、対象となる 2 つの母集団、1 つは身長の値、もう 1 つは体重の値が存在する可能性があります。または、ユニバースは特定の会社で製造された全ウィジェットで、値の母集団は各ウィジェットが故障するまで使用される時間の長さである可能性があります。

値の母集団は、その*累積経験分布関数*の観点から表すことができます。可能な値にそれぞれ満たない、または等しい母集団の割合が示されます。個別の母集団は、*確率関数*によっても表すことができます。可能な値にそれぞれ等しい母集団の割合が示されます。連続した母集団は、*密度関数*によって表すことができます。これは累積経験分布関数の導関数です。密度関数は、それぞれの一連の値内の母集団の割合を示すヒストグラムによって近似できます。確率密度関数は、無数の無限に小さな間隔を含むヒストグラムに似ています。

技術文献では、用語*分布*が条件なしで使用される場合、通常、累積経験分布関数を指します。一般的な文章では、*分布*は密度関数を表すことがあります。用語*分布*は、多くの場合、具体的な母集団ではなく、値の抽象的な母集団を指すために使用されます。したがって、統計的な文献では、正規分布、指数分布、Cauchy 分布など、多種の抽象的な分布を指します。*正規分布*などの言葉が使用される場合、累積経験分布関数または密度関数を指しているのどうかかは多くの場合関係ありません。

分布の関連機能を要約する指標の観点から母集団を説明することは適切です。母集団値から計算されるそのような指標は、*パラメータ*と呼ばれます。多くの異なるパラメータを定義して、分布のさまざまな側面を測定できます。

最も一般的に使用されているパラメータは、(算術) *平均*です。母集団に有限数の値が含まれている場合、母平均は母集団の要素数によって除算される母集団のすべての値の合計として計算されます。無限の母集団の場合、平均の概念は似ていますが、より複雑な計算が必要になります。

$\mathrm{E}(x)$は、x で表される、高さなどの値の母平均を示します。この場合、E は*期待値*を表します。元の値の導関数の期待値を考慮することもできます。たとえば、x が高さを表す場合、$\mathrm{E}(x^2)$は、高さの期待値の 2 乗、つまり、高さの母集団のすべての値を 2 乗することによって得られる母平均値です。

サンプルと統計量

母集団のすべての値を測定することはほとんど不可能です。測定された値の集合は、*サンプル*と呼ばれます。値のサンプルの数学関数は、*統計量*と呼ばれます。パラメータが母集団に対応しているように、統計量はサンプルに対応しています。従来、統計量はローマ文字、パラメータはギリシャ文字で表します。たとえば、多くの場合、母平均は μ、サンプル平均は$\bar{x}$と表されます。*統計量*の分野は、サンプル統計量の動作の調査と大きく関連しています。

サンプルは、さまざまな方法で選択可能です。ほとんどの SAS プロシジャは、データが*単純無作為抽出*を構成しているとみなします。つまり、サンプルはすべての可能なサンプルが均等に選択されるような方法で選択されました。

サンプルからの統計量は、母集団のパラメータに関して推定または合理的な推測を行うために使用できます。たとえば、高校から 30 名の生徒を無作為抽出する場合、これらの 30 名の生徒の平均身長が高校の全生徒の平均身長の合理的な推定、または*推測*となります。標準誤差などのその他の統計量では、推定がどれだけ正確になるかに関する情報を提供可能です。

母集団のパラメータの場合、複数の統計量で推定可能です。ただし、多くの場合、指定パラメータの推定に通常使用される特定の統計量が 1 つあります。たとえば、サンプル平均は母平均の通常の推定量です。平均の場合、パラメータの式と統計量の式は同じです。その他の場合、パラメータの式は最も一般的に使用される推定量の式とは異なることがあります。最も一般的に使用される推定量は、必ずしもすべてのアプリケーションでの最適な推定量ではありません。

位置の統計量

位置の統計量の概要

位置の統計量には、平均、中央値、モードが含まれます。これらの統計量は、分布の中央を表します。後に続く定義では、一定量を各オブザベーションに追加することによりサンプル全体が変わる場合、これらの位置の統計量は同じ一定量によって変わります。

平均値

母平均$\mu = \mathrm{E}(x)$は、通常、サンプル平均$\bar{x}$によって推定されます。

中央値

母集団の中央値は、母集団値の半分の上下にある中央値です。サンプル中央値は、データが昇順または降順に調整されるときの中間値です。偶数のオブザベーションの場合、2 つの中間値の中間点は通常、中央値としてレポートされます。

モード

モードは、母集団の密度が最大の値です。一部の密度には複数のローカル最大値(ピーク)があり、*多峰分布*と呼ばれます。サンプルモードは、サンプルで最も頻繁に発生する値です。デフォルトで、最も頻繁に発生するサンプル値のタイがある場合、PROC UNIVARIATE はそのような最低値をレポートします。PROC ステートメントで MODES オプションを指定する場合、PROC UNIVARIATE はすべての可能なモードをリストします。母集団が連続している場合、すべてのサンプル値が一度に発生し、サンプルモードはほとんど使用されません。

パーセント点

パーセント点(分位点、四分位点、中央値を含む)は、分布の詳細調査に使用されます。統計量はその大きさで並べられます。*p*th パーセント点には、その点の下に *p* パーセント、その点より上に(100−*p*)パーセントの数の統計量が存在します。中央値は 50 パーセント点です。希望どおりのパーセント点が取得できるようにデータを分割できないことがあるため、UNIVARIATE プロシジャはより正確な定義を使用します。

分布の上位の四分位は、統計量の 75% (75 番目のパーセント点)が下回る値です。統計量の 25 パーセントが下位の四分位値を下回ります。

分位数

次の例では、SAS で乱数関数を含むデータを作為的に生成します。UNIVARIATE プロシジャはさまざまな分位点と位置の統計量を計算し、値を SAS データセットに出力します。次に DATA ステップは SYMPUT ルーチンを使用して、統計量の値をマクロ変数に割り当てます。マクロ%FORMGEN はこれらのマクロ変数を使用して、FORMAT プロシジャに対し値ラベルを作成します。PROC CHART は結果の出力形式を使用して、統計量の値をヒストグラムに表示します。

```
options nodate pageno=1 linesize=80 pagesize=52;

title 'Example of Quantiles and Measures of Location';

data random;
   drop n;
   do n=1 to 1000;
      X=floor(exp(rannor(314159)*.8+1.8));
      output;
   end;
run;

proc univariate data=random nextrobs=0;
   var x;
   output out=location
          mean=Mean mode=Mode median=Median
          q1=Q1 q3=Q3 p5=P5 p10=P10 p90=P90 p95=P95
          max=Max;
run;

proc print data=location noobs;
run;

data _null_;
   set location;
   call symput('MEAN',round(mean,1));
   call symput('MODE',mode);
   call symput('MEDIAN',round(median,1));
```

```
   call symput('Q1',round(q1,1));
   call symput('Q3',round(q3,1));
   call symput('P5',round(p5,1));
   call symput('P10',round(p10,1));
   call symput('P90',round(p90,1));
   call symput('P95',round(p95,1));
   call symput('MAX',min(50,max));
run;

%macro formgen;
%do i=1 %to &max;
   %let value=&i;
   %if &i=&p5     %then %let value=&value  P5;
   %if &i=&p10    %then %let value=&value  P10;
   %if &i=&q1     %then %let value=&value  Q1;
   %if &i=&mode   %then %let value=&value  Mode;
   %if &i=&median %then %let value=&value  Median;
   %if &i=&mean   %then %let value=&value  Mean;
   %if &i=&q3     %then %let value=&value  Q3;
   %if &i=&p90    %then %let value=&value  P90;
   %if &i=&p95    %then %let value=&value  P95;
   %if &i=&max    %then %let value=>=&value;
   &i="&value"
%end;
%mend;

proc format print;
   value stat %formgen;
run;
options pagesize=42 linesize=80;

proc chart data=random;
   vbar x / midpoints=1 to &max by 1;
   format x stat.;
   footnote  'P5  =  5TH PERCENTILE';
   footnote2 'P10 = 10TH PERCENTILE';
   footnote3 'P90 = 90TH PERCENTILE';
   footnote4 'P95 = 95TH PERCENTILE';
   footnote5 'Q1  =  1ST QUARTILE  ';
   footnote6 'Q3  =  3RD QUARTILE  ';
run;
```

Example of Quantiles and Measures of Location

The UNIVARIATE Procedure
Variable: X

Moments			
N	1000	Sum Weights	1000
Mean	7.605	Sum Observations	7605
Std Deviation	7.38169794	Variance	54.4894645
Skewness	2.73038523	Kurtosis	11.1870588
Uncorrected SS	112271	Corrected SS	54434.975
Coeff Variation	97.0637467	Std Error Mean	0.23342978

Basic Statistical Measures			
Location		Variability	
Mean	7.605000	Std Deviation	7.38170
Median	5.000000	Variance	54.48946
Mode	3.000000	Range	62.00000
		Interquartile Range	6.00000

Tests for Location: Mu0=0				
Test	Statistic		p Value	
Student's t	t	32.57939	Pr > |t|	<.0001
Sign	M	494.5	Pr >= |M|	<.0001
Signed Rank	S	244777.5	Pr >= |S|	<.0001

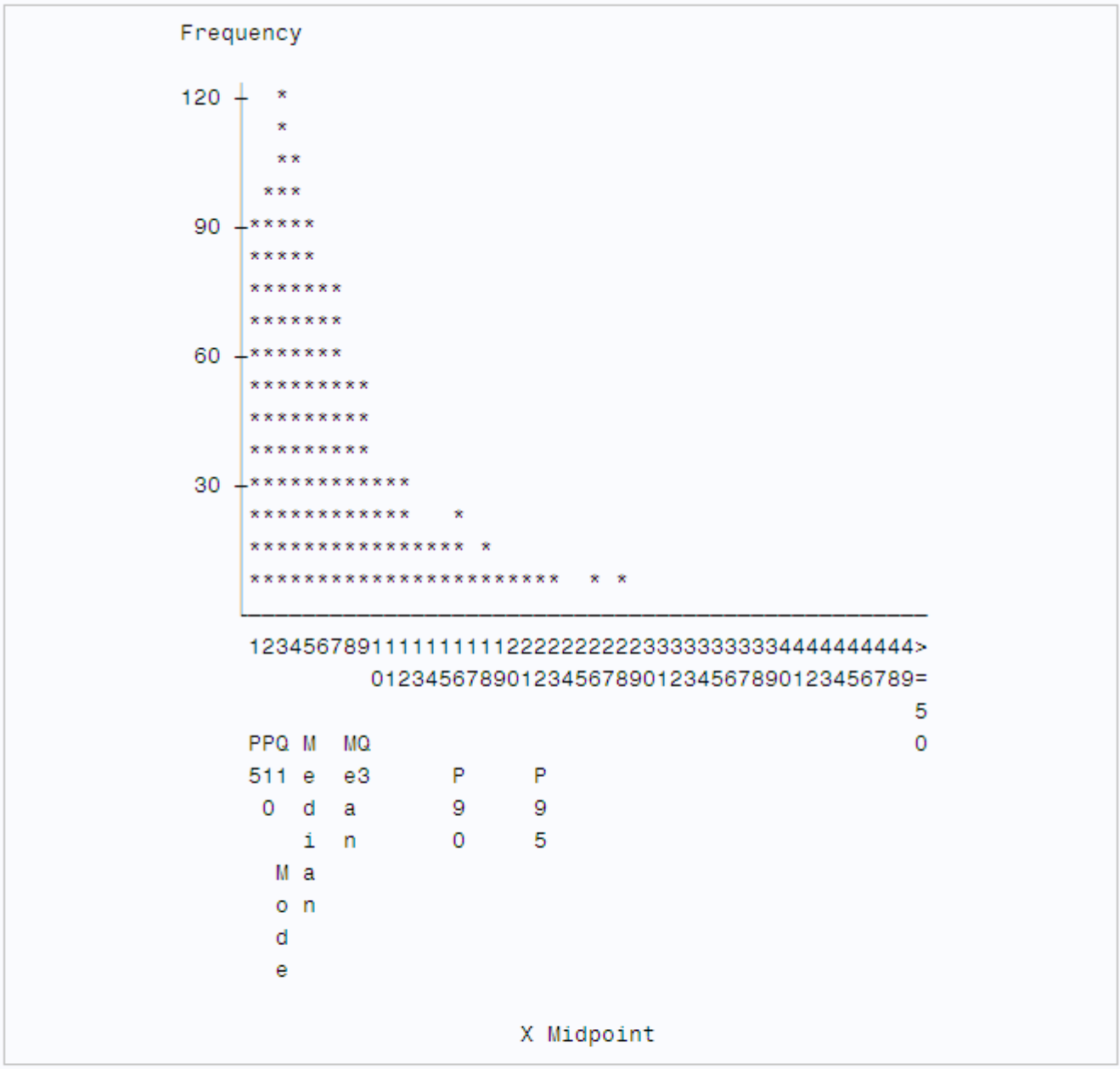
Tests for Location: Mu0=0				
Test	Statistic		p Value	
Student's t	t	32.57939	Pr > |t|	<.0001
Sign	M	494.5	Pr >= |M|	<.0001
Signed Rank	S	244777.5	Pr >= |S|	<.0001

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	62.0
99%	37.5
95%	21.5
90%	16.0
75% Q3	9.0
50% Median	5.0
25% Q1	3.0
10%	2.0
5%	1.0
1%	0.0
0% Min	0.0

Example of Quantiles and Measures of Location

Mean	Max	P95	P90	Q3	Median	Q1	P10	P5	Mode
7.605	62	21.5	16	9	5	3	2	1	3

Example of Quantiles and Measures of Location



P5 = 5TH PERCENTILE
P10 = 10TH PERCENTILE
P90 = 90TH PERCENTILE
P95 = 95TH PERCENTILE
Q1 = 1ST QUARTILE
Q3 = 3RD QUARTILE

ばらつきの統計量

ばらつきの統計量の概要

別の統計量グループが母集団の分布の調査で重要です。これらの統計量は、*拡がり*とも呼ばれる、値のばらつきを測定します。後に続くセクションで指定される定義では、サンプル全体が各オブザベーションへの一定量の追加によって変更される場合、これらの統計量の値は変わらないことに注意してください。ただし、サンプルの各オブザベーションに定数が乗算されると、これらの統計量の値は適切に調整されます。

範囲

サンプル範囲は、サンプルの最大値と最小値の間の差です。多くの母集団の場合、少なくとも統計的な理論では、範囲は無限です。このため、サンプル範囲には母集団が示されないことがあります。サンプル範囲は、サンプルサイズが大きくなるごとに増大する傾向があります。すべてのサンプル値に定数が乗算されると、サンプル範囲に同じ定数が乗算されます。

四分位範囲

四分位範囲は、上位の四分位と下位の四分位の間の差です。すべてのサンプル値に定数が乗算されると、サンブルの四分位範囲に同じ定数が乗算されます。

分散

通常、σ^2で表される母分散は、母平均からの値の 2 乗された差の期待値です。

$$\sigma^2 = \mathrm{E}(x - \mu)^2$$

サンプル分散は、s^2で表されます。値と平均の間の差は、*平均からの偏差*と呼ばれます。そのため、分散により 2 乗された偏差の平均が近似されます。

すべての値が平均に近い場合、分散は小さくなりますが、ゼロ未満にはなりません。値がさらに散在すると、分散はより大きくなります。すべてのサンプル値に定数が乗算されると、サンプル分散に定数の 2 乗が乗算されます。

$n - 1$以外の値が分母で使用される場合があります。VARDEF=オプションは、プロシジャが使用する除数を制御します。

標準偏差

標準偏差は母集団またはサンプルのいずれかで分散の平方根、または平均の標準偏差となります。通常、母集団には σ、サンプルには s のシンボルがそれぞれ使用されます。標準分散は、2 乗単位ではなく、オブザベーションと同じ単位で表されます。すべてのサンプル値に定数が乗算されると、サンプル標準分散に同じ定数が乗算されます。

変動係数

変動係数は、相対的なばらつきの単位を持たない指標です。これは、標準偏差と平均の比をパーセントで表示したものです。変動係数は、変数が比尺度で測定された場合にのみ意味を持ちます。すべてのサンプル値に定数が乗算されると、サンプルの変動係数は変わりません。

形状の統計量

歪度

分散は、平均からの偏差の全体サイズの指標です。分散の式では偏差を 2 乗するため、正と負の偏差は同じように分散に影響します。分布の多くでは、正の偏差は負の偏差より大きさが大きい、またはその逆となる傾向があります。*歪度*は、1 つの方向でより大きくなる偏差の傾向の指標です。たとえば、最後の例のデータは、右に偏っています。

母集団の歪度は、次のように定義されます。

$$\mathrm{E}(x-\mu)^3/\sigma^3$$

偏差は 2 乗ではなく 3 乗されるため、偏差の符号は保持されます。偏差の 3 乗により、大きな偏差の影響が強調されます。式にはスケールの影響を取り除くためのσ^3の除数が含まれているため、すべての値に定数を乗算しても、歪度は変わりません。そのため歪度は、母集団の裾が片方よりも重い傾向として解釈されます。歪度は、正または負である可能性があり、無制限です。

尖度

分布の裾の重さは、多くの統計量の動作に影響します。そのため、裾の重さの指標を使用すると便利です。そのような指標の 1 つは*尖度*です。母集団の尖度は、通常、次のように定義されます。

$$\frac{\mathrm{E}(x-\mu)^4}{\sigma^4}-3$$

注: 一部の統計量では、3 の減算が省略されます。

偏差が 4 累乗されるため、正と負の偏差の分布は同じで、大きな偏差は強く強調されます。除数によりσ^4、各値に定数を除算しても、尖度には影響しません。

母集団の尖度の範囲は、-2と$+\infty$の間です(両端の値を含める)。M_3が母集団の歪度を表し、M_4が母集団の尖度を表す場合、次のようになります。

$$M_4 > (M_3)^2 - 2$$

統計の文献では、尖度で密度の*集中具合*が測定されることが報告される場合があります。ただし、重い裾は、平均に近い分布の形状よりも尖度に大きく影響します (Kaplansky 1945; Ali 1974; Johnson, et al.1980)。

サンプルの歪度および尖度は、小さなサンプルの対応するパラメータの信頼性の低い推定量です。サンプルが非常に大きい場合は、より信頼できる推定量となります。ただし、歪度または尖度の値が大きい場合は、小さいサンプルででも注目に値することがあります。そのような値は、正規性の仮定に基づく統計手法が不適切である場合があることを示すためです。

正規分布

特に重要な理論分布は、*正規*分布または *Gaussian* 分布です。正規分布は滑らかな対称機能で、"つり鐘型"とも言います。歪度と尖度はゼロです。正規分布は、2 つのパラメータ、平均と標準偏差によってのみ完全に指定できます。正規母集団の値の約 68%が母平均の 1 つの標準偏差内、値の約 95%が平均の 2 つの標準偏差内、約 99.7%が 3 つの標準偏差内にあります。特定の分布の種類を表す用語*正規*を使用し

ても、その他の分布の種類が必ずしも正規ではない、または異常であるということではありません。

多くの統計手法は、抽出中の母集団が正規に分布されるという仮定の下で設計されます。それにも関らず、実際の母集団の多くには、正規分布は含まれていません。正規性の仮定に基づいて統計手法を使用する前に統計に関する文献を参照し、その手法が非正規性に対しどれだけ敏感かを確認し、必要に応じてそのサンプルを非正規性の証拠に関してチェックする必要があります。

次の例では、平均が 50、標準偏差が 10 の正規分布からのサンプルを生成します。UNIVARIATE プロシジャは位置と正規性について検定を実行します。データは正規分布からのものであるため、正規性の検定からの p 値はすべて 0.15 よりも大きくなります。CHART プロシジャは、オブザベーションのヒストグラムを表示します。ヒストグラムの形状は、つり鐘型の正規密度です。

```
options nodate pageno=1 linesize=80 pagesize=52;

title '10000 Obs Sample from a Normal Distribution';
title2 'with Mean=50 and Standard Deviation=10';

data normaldat;
   drop n;
   do n=1 to 10000;
      X=10*rannor(53124)+50;
      output;
   end;
run;

proc univariate data=normaldat nextrobs=0 normal
                           mu0=50 loccount;
   var x;
run;

proc format;
   picture msd
      20='20 3*Std' (noedit)
      30='30 2*Std' (noedit)
      40='40 1*Std' (noedit)
      50='50 Mean ' (noedit)
      60='60 1*Std' (noedit)
      70='70 2*Std' (noedit)
      80='80 3*Std' (noedit)
   other=' ';
run;
options linesize=80 pagesize=42;

proc chart;
   vbar x / midpoints=20 to 80 by 2;
   format x msd.;
run;
```

10000 Obs Sample from a Normal Distribution with Mean=50 and Standard Deviation=10

The UNIVARIATE Procedure
Variable: X

Moments			
N	10000	Sum Weights	10000
Mean	50.0323744	Sum Observations	500323.744
Std Deviation	9.92013874	Variance	98.4091525
Skewness	-0.019929	Kurtosis	-0.0163755
Uncorrected SS	26016378	Corrected SS	983993.116
Coeff Variation	19.8274395	Std Error Mean	0.09920139

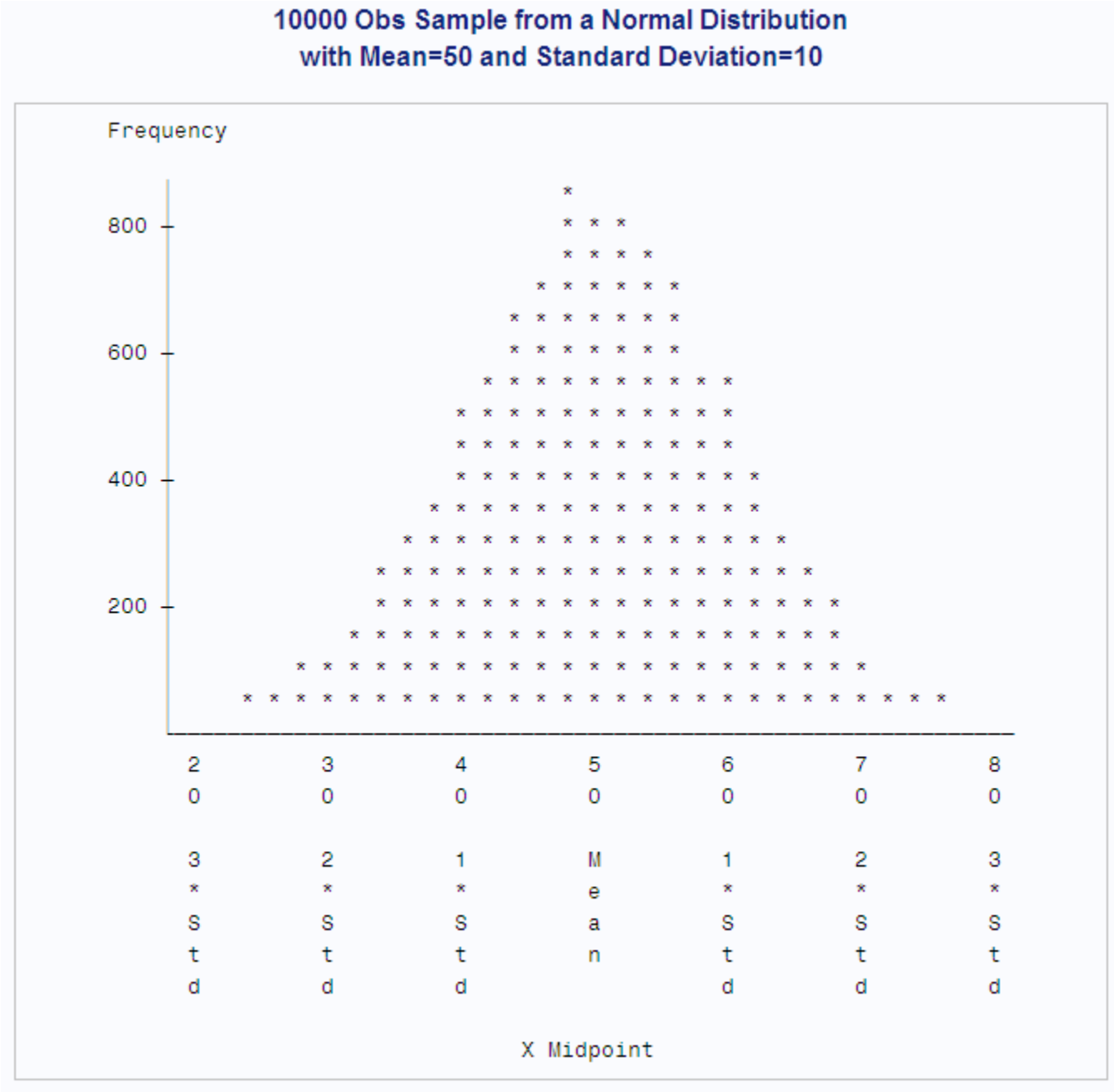
Basic Statistical Measures			
Location		Variability	
Mean	50.03237	Std Deviation	9.92014
Median	50.06492	Variance	98.40915
Mode	.	Range	76.51343
		Interquartile Range	13.28179

Tests for Location: Mu0=50				
Test	Statistic		p Value	
Student's t	t	0.32635	Pr > |t|	0.7442
Sign	M	26	Pr >= |M|	0.6101
Signed Rank	S	174063	Pr >= |S|	0.5466

Location Counts: Mu0=50.00	
Count	Value
Num Obs > Mu0	5026
Num Obs ^= Mu0	10000
Num Obs < Mu0	4974

Tests for Normality				
Test	Statistic		p Value	
Kolmogorov-Smirnov	D	0.006595	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.049963	Pr > W-Sq	>0.2500
Anderson-Darling	A-Sq	0.371151	Pr > A-Sq	>0.2500

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	90.2105
99%	72.6780
95%	66.2221
90%	62.6678
75% Q3	56.7280
50% Median	50.0649
25% Q1	43.4462
10%	37.1139
5%	33.5454
1%	26.9189
0% Min	13.6971



平均の標本分布

母集団からのサイズ n のサンプルを繰り返し取り出して、各サンプルの平均を計算する場合、サンプルはそれ自体に分布が含まれていることを意味します。元の母集団から取り出される可能性のあるすべてのサンプルの平均で構成される新しい母集団について考えます。この新しい母集団の分布は、*標本分布*と呼ばれます。

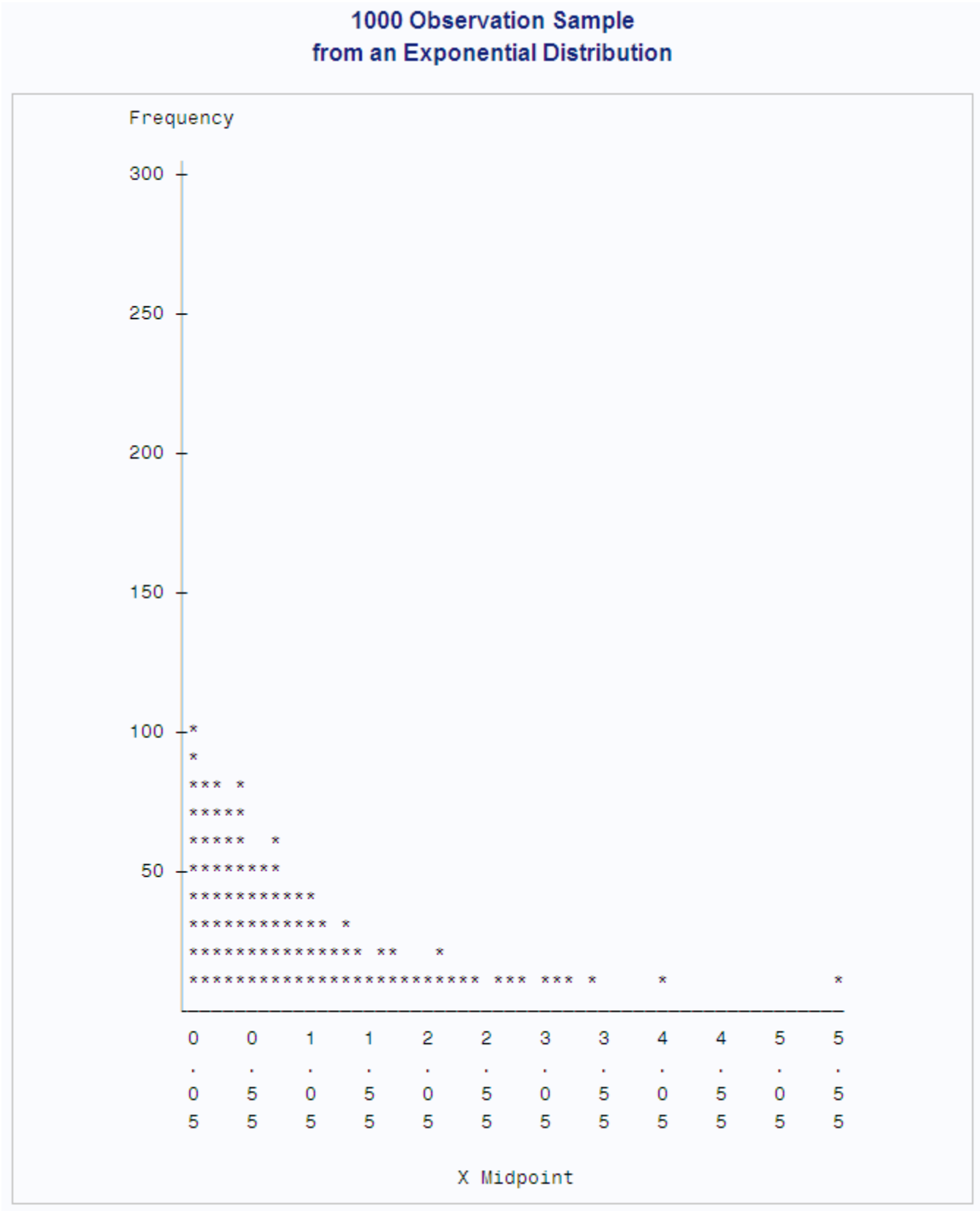
元の母集団に平均 μ、標準偏差 σ が含まれている場合、平均の標本分布にも平均 μ が含まれていますが、その標準偏差は$\sigma/\sqrt{n}$ということが数学的に実証可能です。平均の標本分布の標準偏差は、*平均値の標準誤差*と呼ばれます。平均の標準誤差により、母平均の推定量としてのサンプル平均の正確性が示されます。

元の母集団に正規分布が含まれている場合、平均の標本分布も正規です。元の分布が正規でなく、裾が過剰に長くない場合、平均の標本分布は大きなサンプルサイズの正規分布によって近似可能です。

次の例は、平均の標本分布がサンプルサイズが大きくなるにつれて正規分布によってどのように近似可能であるかを示す、3 つの別個のプログラムから構成されています。最初の DATA ステップでは RANEXP 関数を使用して、指数分布からの 1000 オブザベーションのサンプルを作成します。理論母平均は 1.00、サンプル平均は 1.01 (小数点以下 2 桁まで)です。母集団の標準偏差は 1.00 です。 サンプル標準差異は 1.04 です。

次の例は、非正規分布の例です。母集団の歪度は 2.00 で、これはサンプル歪度 1.97 に近いです。母集団の尖度は 6.00 ですが、サンプル尖度は 4.80 です。

```
options nodate pageno=1 linesize=80 pagesize=42;

title '1000 Observation Sample';
title2 'from an Exponential Distribution';

data expodat;
   drop n;
   do n=1 to 1000;
      X=ranexp(18746363);
      output;
   end;
run;
proc format;
    value axisfmt
      .05='0.05'
      .55='0.55'
     1.05='1.05'
     1.55='1.55'
     2.05='2.05'
     2.55='2.55'
     3.05='3.05'
     3.55='3.55'
     4.05='4.05'
     4.55='4.55'
     5.05='5.05'
     5.55='5.55'
     other=' ';
run;

proc chart data=expodat ;
   vbar x / axis=300
            midpoints=0.05 to 5.55 by .1;
   format x axisfmt.;
run;

options pagesize=64;

proc univariate data=expodat noextrobs=0 normal
                mu0=1;
   var x;
run;
```


1000 Observation Sample
from an Exponential Distribution
Frequency
300
250
200
150
100
50
0 0 1 1 2 2 3 3 4 4 5 5
. . . .
0 5 0 5 0 5 0 5 0 5 0 5
5 5 5 5 5 5 5 5 5 5 5 5
X Midpoint

1000 Observation Sample from an Exponential Distribution

The UNIVARIATE Procedure
Variable: X

Moments			
N	1000	Sum Weights	1000
Mean	1.01176214	Sum Observations	1011.76214
Std Deviation	1.04371187	Variance	1.08933447
Skewness	1.96963112	Kurtosis	4.80150594
Uncorrected SS	2111.90777	Corrected SS	1088.24514
Coeff Variation	103.15783	Std Error Mean	0.03300507

Basic Statistical Measures			
Location		Variability	
Mean	1.011762	Std Deviation	1.04371
Median	0.689502	Variance	1.08933
Mode	.	Range	6.63851
		Interquartile Range	1.06252

Tests for Location: Mu0=1				
Test	Statistic		p Value	
Student's t	t	0.356374	Pr > |t|	0.7216
Sign	M	-140	Pr >= |M|	<.0001
Signed Rank	S	-50781	Pr >= |S|	<.0001

Tests for Normality				
Test	Statistic		p Value	
Shapiro-Wilk	W	0.801498	Pr < W	<0.0001
Kolmogorov-Smirnov	D	0.166308	Pr > D	<0.0100
Cramer-von Mises	W-Sq	9.507975	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	54.5478	Pr > A-Sq	<0.0050

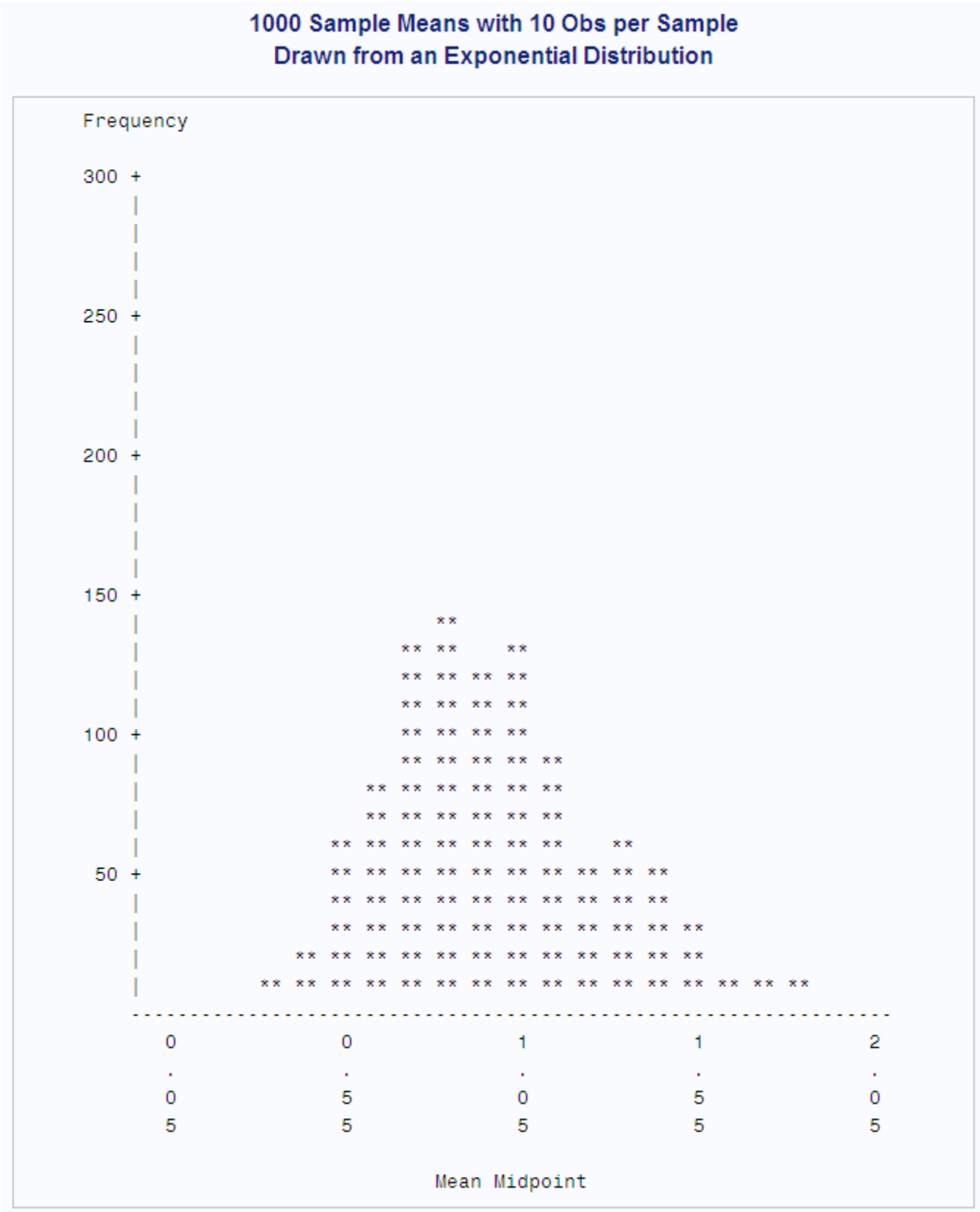
Quantiles (Definition 5)	
Quantile	Estimate
100% Max	6.63906758
99%	5.04491651
95%	3.13482318
90%	2.37803632
75% Q3	1.35733401
50% Median	0.68950221
25% Q1	0.29481436
10%	0.10219011
5%	0.05192799
1%	0.01195590
0% Min	0.00055441

次の DATA ステップは、同じ指数分布から 1000 の異なるサンプルを生成します。サンプルにはそれぞれ 10 のオブザベーションが含まれています。MEANS プロシジャは、各サンプルの平均を計算します。PROC MEANS によって作成されるデータセットで、オブザベーションはそれぞれ指数分布からの 10 のオブザベーションのサンプルの平均を表します。そのため、データセットは、指数母集団に対する平均の標本分布からのサンプルです。

PROC UNIVARIATE は、この平均値のサンプルに対する統計量を表示します。平均のサンプルの平均は.99 で、元の母平均とほぼ同じです。理論的に、標本分布の標準偏差は$\sigma/\sqrt{n} = 1.00/\sqrt{10} = .32$です。標本分布からのこのサンプルの標準偏差は.30です。歪度(.55)と尖度(-.006)は、指数分布からの元のサンプルのものより標本分布からのサンプルのものの方がゼロに近くなります。これは、標本分布が元の指数分布よりも正規分布に近いためです。CHART プロシジャは、1000 のサンプル平均のヒストグラムを表示します。ヒストグラムの形状はつり鐘型に近い正規密度ですが、極めて不均等です。

```
options nodate pageno=1 linesize=80 pagesize=48;

title '1000 Sample Means with 10 Obs per Sample';
title2 'Drawn from an Exponential Distribution';

data samp10;
   drop n;
   do Sample=1 to 1000;
      do n=1 to 10;
         X=ranexp(433879);
         output;
      end;
   end;

proc means data=samp10 noprint;
   output out=mean10 mean=Mean;
   var x;
```

```
   by sample;
run;

proc format;
     value axisfmt
       .05='0.05'
       .55='0.55'
      1.05='1.05'
      1.55='1.55'
      2.05='2.05'
      other=' ';
 run;

proc chart data=mean10;
   vbar mean/axis=300
             midpoints=0.05 to 2.05 by .1;
   format mean axisfmt.;
run;

options pagesize=64;
proc univariate data=mean10 nextrobs=0 normal
                mu0=1;
   var mean;
run;
```


1000 Sample Means with 10 Obs per Sample
Drawn from an Exponential Distribution
Frequency
300
250
200
150
100
50
0
0
1
1
2
0
5
0
5
0
5
5
5
5
Mean Midpoint

1000 Sample Means with 10 Obs per Sample Drawn from an Exponential Distribution

The UNIVARIATE Procedure
Variable: Mean

Moments			
N	1000	Sum Weights	1000
Mean	0.9906857	Sum Observations	990.685697
Std Deviation	0.30732649	Variance	0.09444957
Skewness	0.54575615	Kurtosis	-0.0060892
Uncorrected SS	1075.81327	Corrected SS	94.3551193
Coeff Variation	31.0215931	Std Error Mean	0.00971852

Basic Statistical Measures			
Location		Variability	
Mean	0.990686	Std Deviation	0.30733
Median	0.956152	Variance	0.09445
Mode	.	Range	1.79783
		Interquartile Range	0.41703

Tests for Location: Mu0=1				
Test	Statistic		p Value	
Student's t	t	-0.95841	Pr > |t|	0.3381
Sign	M	-53	Pr >= |M|	0.0009
Signed Rank	S	-22687	Pr >= |S|	0.0129

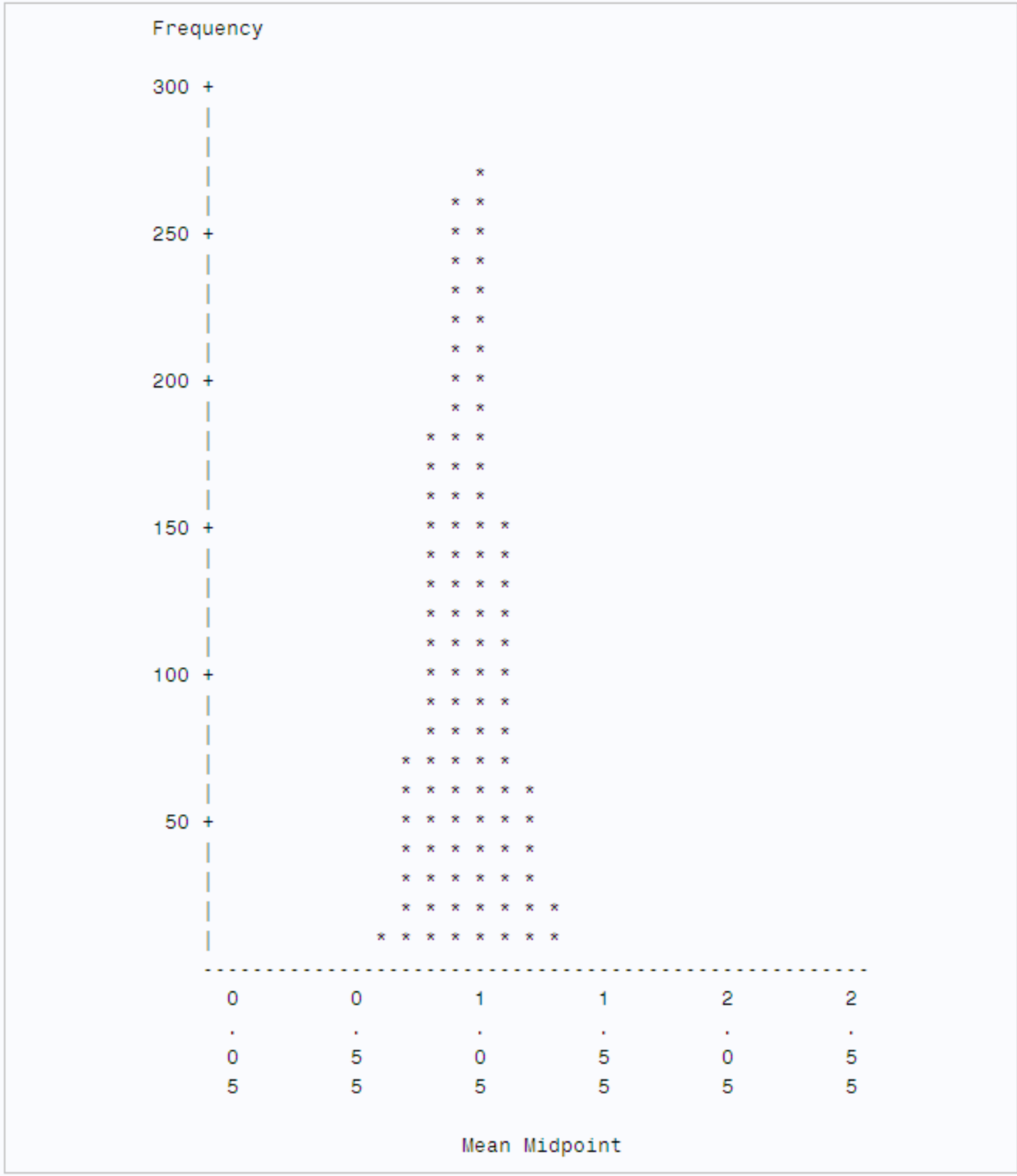
Tests for Normality				
Test	Statistic		p Value	
Shapiro-Wilk	W	0.9779	Pr < W	<0.0001
Kolmogorov-Smirnov	D	0.055498	Pr > D	<0.0100
Cramer-von Mises	W-Sq	0.953926	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	5.945023	Pr > A-Sq	<0.0050

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	2.053899
99%	1.827503
95%	1.557175
90%	1.416611
75% Q3	1.181006
50% Median	0.956152
25% Q1	0.763973
10%	0.621787
5%	0.553568
1%	0.433820
0% Min	0.256069

次の DATA ステップでは、指数分布からの各サンプルサイズが 50 に増えます。標本分布の標準偏差は前の例よりも小さくなります。これは、各サンプルサイズがより大きいためです。また、ヒストグラムと歪度から見て、標本分布は正規分布に近くなります。

```
options nodate pageno=1 linesize=80 pagesize=48;

title '1000 Sample Means with 50 Obs per Sample';
title2 'Drawn from an Exponential Distribution';

data samp50;
   drop n;
   do sample=1 to 1000;
      do n=1 to 50;
         X=ranexp(72437213);
         output;
      end;
   end;

proc means data=samp50 noprint;
   output out=mean50 mean=Mean;
   var x;
   by sample;
run;

proc format;
   value axisfmt
        .05='0.05'
        .55='0.55'
      1.05='1.05'
      1.55='1.55'
      2.05='2.05'
      2.55='2.55'
```

```
      other=' ';
run;

proc chart data=mean50;
   vbar mean / axis=300
               midpoints=0.05 to 2.55 by .1;
   format mean axisfmt.;
run;

options pagesize=64;

proc univariate data=mean50 nextrobs=0 normal
                mu0=1;
   var mean;
run;
```


1000 Sample Means with 50 Obs per Sample
Drawn from an Exponential Distribution
Frequency
300
250
200
150
100
50
0.05
0.55
1.05
1.55
2.05
2.55
Mean Midpoint

1000 Sample Means with 50 Obs per Sample Drawn from an Exponential Distribution

The UNIVARIATE Procedure
Variable: Mean

Moments			
N	1000	Sum Weights	1000
Mean	0.99679697	Sum Observations	996.796973
Std Deviation	0.13815404	Variance	0.01908654
Skewness	0.19062633	Kurtosis	-0.1438604
Uncorrected SS	1012.67166	Corrected SS	19.067451
Coeff Variation	13.8597969	Std Error Mean	0.00436881

Basic Statistical Measures			
Location		Variability	
Mean	0.996797	Std Deviation	0.13815
Median	0.996023	Variance	0.01909
Mode	.	Range	0.87040
		Interquartile Range	0.18956

Tests for Location: Mu0=1				
Test	Statistic		p Value	
Student's t	t	-0.73316	Pr > |t|	0.4636
Sign	M	-13	Pr >= |M|	0.4292
Signed Rank	S	-10767	Pr >= |S|	0.2388

Tests for Normality				
Test	Statistic		p Value	
Shapiro-Wilk	W	0.996493	Pr < W	0.0247
Kolmogorov-Smirnov	D	0.023687	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.084468	Pr > W-Sq	0.1882
Anderson-Darling	A-Sq	0.66039	Pr > A-Sq	0.0877

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	1.454957
99%	1.337016
95%	1.231508
90%	1.179223
75% Q3	1.086515
50% Median	0.996023
25% Q1	0.896953
10%	0.814906
5%	0.780783
1%	0.706588
0% Min	0.584558

仮説検定

仮説の定義

これまで述べてきた統計手法の目的は、標本統計量によって母集団のパラメータを推定することです。別のクラスの統計手法が、母集団パラメータに関する仮説の検定、または仮説に対する証拠量の測定に使用されます。

大学の生徒のユニバースについて考えます。変数 X を、生徒の体重が同じ性別、身長、体型の人の理想体重から外れるポンド数とします。生徒の母集団が平均、低体重、または過体重であるかどうかを見つけます。そのために、9 人の生徒から無作為抽出の X 値を取り出します。結果は次の DATA ステップにあるとおりです。

```
title 'Deviations from Normal Weight';

data x;
   input X @@;
   datalines;
-7 -2 1 3 6 10 15 21 30
;
```

関心のある複数の仮説を定義できます。1 つの仮説は、生徒は平均でまさに理想の体重というものです。μ が X 値の母平均を表す場合、*帰無*仮説と呼ばれるこの仮説を H_0: $\mu = 0$と表すことができます。*対立*仮説と呼ばれるその他 2 つの仮説は、生徒が平均で低体重であり、H_1: $\mu < 0$、学生が平均で過体重である、H_2: $\mu > 0$。

帰無仮説がそう呼ばれるのは、多くの場合、“影響なし”または“相違なし”という仮定に対応しているためです。ただし、この解釈はすべての検定問題に適しているわけではありません。帰無仮説は、統計的な証拠によって倒される可能性があるかかしに似ています。かかしが倒れる方法に従って、対立仮説のどちらかを決めます。

この問題にアプローチするためのネイティブな方法は、サンプル平均$\bar{x}$を見て、次のルールに従って 3 つの仮説から決めることです。

- $\bar{x} < 0$の場合、H_1: $\mu < 0$。
- $\bar{x} = 0$の場合、H_0: $\mu = 0$。
- $\bar{x} > 0$の場合、H_2: $\mu > 0$。

このアプローチに関する問題は、正しくない決定を行う可能性が高いことです。H_0 が true の場合、ほぼ確実に誤った決定を行います。これは、$\bar{x}$がゼロになる可能性がほとんどないためです。μ がわずかにゼロ未満で、H_1 が true の場合、$\bar{x}$が繰り返されるサンプリングでゼロより大きい可能性は約 50%です。そのため H_2 を誤って選択する可能性も約 50%になります。したがって、$\bar{x}$がゼロに近い場合は、選択を誤る可能性が高くなります。そのような場合は自信のある決定を行うために十分な証拠がないため、最善の対応としてはさらに多くの証拠が得られるまで判断を留保することです。

問題は、自信を持って決定できるように、$\bar{x}$がゼロからどのくらい離れているかということです。答えは、$\bar{x}$の標本分布について考えるとわかります。X の分布がほぼ正規分布である場合、$\bar{x}$の分布はほとんど正規標本分布です。$\bar{x}$の標本分布の平均は μ です。X の標準偏差が 12 であるとします。9 つのオブザベーションのサンプルに対する$\bar{x}$の標準誤差は、$\sigma/\sqrt{n} = 12/\sqrt{9} = 4$です。

正規分布からの値の約 95%が平均の 2 つの標準偏差内にあることがわかっています。このため、9 つの X 値の可能なサンプルの約 95%に$0 - 2(4)$から$0 + 2(4)$まで、または-8 から 8 までのサンプル平均$\bar{x}$が含まれています。次の決定ルールで間違いをする可能性について考えます。

- $\bar{x} < -8$の場合、H_1: $\mu < 0$。
- $-8 \le \bar{x} \le 8$の場合、判断を留保します。
- $\bar{x} > 8$の場合、H_2: $\mu > 0$。

H_0 が true の場合、可能なサンプルの約 95%は$\bar{x}$が*臨界値*-8から 8 までになるため、判断を留保します。この場合、統計的証拠は、かかしを倒すほど強くありません。サンプルの別の 5%で間違いをします。サンプルの 2.5%で誤って H_1 を選択し、2.5%で誤って H_2 を選択します。

間違いをする可能性を管理するために支払う代償は、帰無仮説の却下に十分な統計的証拠がない場合に判断を留保する必要性です。

有意性と検定力

帰無仮説を却下する可能性は、true の場合は統計的検定の*第 1 種の過誤率*と呼ばれ、通常はαと表されます。この例では、-8未満、または 8 より大きい$\bar{x}$値は、5%水準で*統計的に有意*と呼ばれます。異なる臨界値を選択して、第 1 種の過誤率をニーズに従って調整できます。たとえば、臨界値が-4 および 4 の場合は有意水準は約 32%となり、-12 および 12 の場合は、第 1 種の過誤率約 0.3%となります。

決定ルールは、*両側検定*です。これは、代替仮説では帰無仮説で指定された値よりも小さい、または大きい母平均が可能なためです。生徒が平均で過体重である可能性については、片側検定を使用できます。

- $\bar{x} \le 8$の場合、判断を留保します。
- $\bar{x} > 8$の場合、H_2: $\mu > 0$。

この片側検定では、第 1 種の過誤率は 2.5%で、両側検定の半分です。

帰無仮説を却下する可能性は、false の場合は統計的検定の*累乗*と呼ばれ、通常は $1 - \beta$と表されます。βは*第 2 種の過誤率*と呼ばれ、false の帰無仮説を却下しない可能性です。累乗は、パラメータの true 値によって異なります。例では、母平均を 4 とします。H_2 を検出するための累乗は、8 よりも大きいサンプル平均を取得する確率です。臨界値 8 は、母平均 4 よりも高い 1 つの標準誤差です。正規分布からの平均より大きい標準偏差を少なくとも 1 つ取得する可能性は、約 16%です。そのため、代替仮説 H_2 を検出するための累乗は約 16%です。母平均が 8 の場合、H_2 の累乗が 50%になります。母平均 12 により累乗が約 84%になります。

第 1 種の過誤率が低くなると、誤った決定を行う可能性が低くなりますが、判断を留保する必要がある可能性は高くなります。第 1 種の過誤率を選択する場合、対象となるさまざまな代替に対する結果の累乗を考慮する必要があります。

スチューデントの t 分布

実際には、σ に基づいた臨界値を使用する決定ルールを通常は使用できません。σ の値が不明なためです。ただし、*s* を σ の推定として使用できます。次の統計量について考えます。

$$t = \frac{\overline{x} - \mu_0}{s/\sqrt{n}}$$

この *t* 統計量は、サンプル平均と、平均の推定標準誤差で割られる仮説平均の差μ_0です。

帰無仮説が true で、母集団が正規分布されている場合、*t* 統計量には*スチューデントの t 分布*($n - 1$自由度を含む)と呼ばれるものが存在します。この分布は正規分布と非常に似ていますが、スチューデントの *t* 分布の裾はさらに重くなっています。サンプルサイズが大きくなるにつれて、サンプルの標準偏差は母集団の標準偏差のより適した推定量となり、*t* 分布は正規分布に近づきます。

決定ルールは *t* 統計量を基準にすることができます。

- $t < -2.3$の場合、H_1: $\mu < 0$。
- $-2.3 \leq t \leq 2.3$の場合、判断を留保します。
- $t > 2.3$の場合、H_0: $\mu > 0$。

値 2.3 は、スチューデントの *t* 分布の表から取得され、8 (つまり$9 - 1 = 8$)自由度に対し第 1 種の過誤率は 5%となります。一般的な統計テキストのほとんどには、スチューデントの *t* 分布の表が含まれています。統計テキストがない場合は、DATA ステップと TINV 関数を使用して、*t* 分布からの値を印刷できます。

デフォルトで、PROC UNIVARIATE は *t* 統計量を$\mu_0 = 0$という帰無仮説に対し関連する統計量とともに計算します。PROC ステートメントで MU0=オプションを使用して、帰無仮説に対し別の値を指定します。

この例では、9 つのオブザベーションから構成される標準体重からの偏差のデータが使用されます。まず、PROC MEANS は *t* 統計量を$\mu = 0$という帰無仮説に対し計算します。次に、DATA ステップの TINV 関数は両側検定(有意性水準 5%、自由度 8)に対するスチューデントの *t* 分布の値を計算します。

```
data devnorm;
   title 'Deviations from Normal Weight';
   input X @@;
   datalines;
-7 -2 1 3 6 10 15 21 30
;
```

```
proc means data=devnorm maxdec=3 n mean
           std stderr t probt;
run;

title 'Student''s t Critical Value';

data _null_;
   file print;
   t=tinv(.975,8);
   put t 5.3;
run;
```

Deviations from Normal Weight

The MEANS Procedure

Analysis Variable : X					
N	Mean	Std Dev	Std Error	t Value	Pr > |t|
9	8.556	11.759	3.920	2.18	0.0606

Student's t Critical Value

2.306

```
Deviations from Normal Weight                1 The MEANS Procedure Analysis Variable :X N
Mean       Std Dev     Std Error  t Value  Pr > |t|
------------------------------------------------------------ 9         8.556        11.759
3.920     2.18     0.0606 ------------------------------------------------------------------
```

```
Student's t Critical Value                  2 2.306
```

現在の例では、*t* 統計量の値は 2.18 です、これは、臨界 *t* 値 2.3 (有意水準 5%、自由度 8)よりも小さいです。そのため、有意水準 5%では、判断を留保する必要があります。有意水準 10%の使用を選択した場合、*t* 分布の臨界値は 1.86 となり、帰無仮説を却下できます。ただし、サンプルサイズが非常に小さいため、結論の妥当性は、母集団の分布がどれだけ正規分布に近いかということによって大きく異なります。

p 値

統計的検定の結果を報告する別の方法としては、*確率値*または *p 値*の計算があります。*p* 値は、繰り返されるサンプリングで、代替仮説によって指定される方向の実際に観測される値まで統計量を取得する確率を示します。*t* 統計量に対する両側の *p* 値は、絶対 *t* 値(観測された絶対 *t* 値よりも大きい)を取得する確率です。*t* 統計量に対する片側 *p* 値(代替仮説$\mu > \mu_0$)は、観測された *t* 値よりも大きい *t* 値を取得する確率です。*p* 値が計算されると、*p* 値と必要な有意水準を比較して、仮説検定を実行できます。*p* 値が検定の第 1 種の過誤率に満たない、または等しい場合、帰無仮説は却下できます。両側 *p* 値(PROC MEANS 出力で `Pr > |t|`とラベル付け)が.0606 であるため、帰無仮説は有意水準が 10%の場合は却下できますが、有意水準が 5%の場合は却下できません。

p 値は、帰無仮説に対する証拠の強度の指標です。*p* 値が小さくなると、帰無仮説を却下するための証拠の強度が増します。

注: 詳細については、統計の入門書(Mendenhall and Beaver (1998)、Ott and Mendenhall (1994)、Snedecor and Cochran (1989)など)を参照してください。

リファレンス

Ali, M. M. 1974. “Stochastic Ordering and Kurtosis Measure.” *Journal of the American Statistical Association* 69: 543–545.

Johnson, M. E., G. L. Tietjen, and R. J. Beckman. 1980. “A New Family of Probability Distributions with Applications to Monte Carlo Studies.” *Journal of the American Statistical Association* 75: 276-279.

Kaplansky, I. 1945. “A Common Error Concerning Kurtosis.” *Journal of the American Statistical Association,* 40: 259-263.

Mendenhall, W. and R. Beaver. 1998. *Introduction to Probability and Statistics*. 10th 版 Belmont, CA: Wadsworth Publishing Company.

Ott, R. and W. Mendenhall. 1994. *Understanding Statistics*. 6th 版 North Scituate, MA: Duxbury Press.

Schlotzhauer, S. D. and R. C. Littell. 1997. *SAS System for Elementary Statistical Analysis*. Second 版 Cary, NC: SAS Institute Inc.

Snedecor, G. W. and W. C. Cochran. 1989. *Statistical Methods*. 8th 版 Ames, IA: Iowa State University Press.

付録 2

動作環境固有のプロシジャ

次の表は、動作環境固有のプロシジャについての簡単な説明と、該当するリリースを示したものです。これらすべてのプロシジャの詳しい説明は、動作環境のドキュメントを参照してください。

表 A2.1 *ホスト固有のプロシジャ*

プロシジャ	説明	リリース
CONVERT	BMDP、OSIRIS および SPSS システムファイルを SAS データセットに変換します。	すべて
C16PORT	リリース 6.08 で作成された 16 ビット SAS ライブラリまたはカタログを移送ファイルに変換します。この移送ファイルは、現在のリリースの SAS で CIMPORT プロシジャを使用して 32 ビット形式に変換できます。	6.10 - 6.12
FSDEVICE	カタログのデバイス説明を作成、コピー、編集、削除、名前変更します。	すべて
PDS	区分データセットのメンバをリスト表示、削除および名前変更します。	6.09E
PDSCOPY	区分データセットをディクスからディスク、ディスクからテープ、テープからテープ、テープからディスクにコピーします。	6.09E
RELEASE	ディスクのデータセットの最後で未使用スペースを開放します。	6.09E
ソース	ソースライブラリデータセットを簡単にバックアップして処理できます。	6.09E
TAPECOPY	テープボリューム全体または 1 つ以上のテープボリュームからのファイルを、1 つの出力テープボリュームにコピーします。	6.09E
TAPELABEL	IBM 標準ラベル付きのテープボリュームのラベル情報を SAS プロシジャ出力ファイルに書き込みます。	6.09E

付録 3

Base SAS プロシジャの生データと DATA ステップ

Base SAS プロシジャの生データと DATA ステップの概要

次の生データ例と DATA ステップ例は、Base SAS プロシジャでの使用を示しています。

このドキュメント内のプログラム例では、通常、使用するデータセットの作成方法が説明されています。一部のデータのみが示されている例もあります。これらの例については、付録に完全なデータが示されています。

CARSURVEY

```
data carsurvey;
   input Rater Age Progressa Remark Jupiter Dynamo;
   datalines;
1   38  94  98  84  80
```

2	49	96	84	80	77
3	16	64	78	76	73
4	27	89	73	90	92
5	50	93	79	84	34
6	57	92	89	75	89
7	21	88	90	89	91
8	39	88	87	76	64
9	26	77	94	93	47
10	17	68	72	85	79
11	38	94	93	84	70
12	29	78	97	74	33
13	41	89	83	75	82
14	37	54	98	70	83
15	52	92	85	88	78
16	23	85	89	89	95
17	61	92	88	77	85
18	24	87	88	88	87
19	18	54	50	62	74
20	62	90	91	90	86
21	49	94	98	84	80
22	16	96	84	80	77
23	27	64	78	76	73
24	50	89	73	90	92
25	57	93	79	84	34
26	21	92	86	75	93
27	39	88	97	89	91
28	26	88	87	76	64
29	17	77	94	93	47
30	38	68	72	85	79
31	29	94	93	84	70
32	41	78	97	74	33
33	37	89	83	75	82
34	52	54	98	70	83
35	23	92	85	88	78
36	61	85	93	89	66
37	24	92	88	77	85
38	18	87	88	88	87
39	62	54	50	62	74
40	38	90	91	90	86
41	57	94	98	84	80
42	16	96	84	80	77
43	19	64	78	76	73
44	59	89	73	90	92
45	57	93	79	84	34
46	21	92	86	75	90
47	39	88	97	89	91
48	26	88	87	76	64
49	17	77	94	93	47
50	56	68	72	85	79
51	29	94	93	84	70
52	41	78	97	74	33
53	37	89	83	75	82
54	52	54	98	70	83
55	17	92	85	88	78
56	61	85	93	89	66
57	24	92	85	77	85

```
58  18  87  88  88  87
59  62  54  50  62  74
60  38  90  91  90  86
61  38  94  98  84  80
62  49  96  84  80  77
63  16  64  78  76  73
64  27  89  73  90  92
65  50  93  79  84  34
66  57  92  89  75  89
67  21  88  93  89  91
68  39  88  87  76  64
69  26  77  94  93  47
70  17  68  72  85  79
71  38  94  93  84  70
72  29  78  97  74  33
73  41  89  83  75  82
74  37  54  98  70  83
75  52  92  85  88  78
76  23  85  93  89  88
77  61  92  88  77  85
78  24  87  88  88  91
79  18  54  50  62  74
80  62  90  91  90  86
;
```

CENSUS

```
data census;
   input Density CrimeRate State $ 14-27 PostalCode $ 29-30;
   datalines;
263.3 4575.3 Ohio           OH
62.1 7017.1  Washington     WA
103.4 5161.9 South Carolina SC
53.4 3438.6  Mississippi    MS
180.0 8503.2 Florida        FL
80.8 2190.7  West Virginia  WV
428.7 5477.6 Maryland       MD
71.2 4707.5  Missouri       MO
43.9 4245.2  Arkansas       AR
7.3 6371.4   Nevada         NV
264.3 3163.2 Pennsylvania   PA
11.5 4156.3  Idaho          ID
44.1 6025.6  Oklahoma       OK
51.2 4615.8  Minnesota      MN
55.2 4271.2  Vermont        VT
27.4 6969.9  Oregon         OR
205.3 5416.5 Illinois       IL
94.1 5792.0  Georgia        GA
9.1 2678.0   South Dakota   SD
9.4 2833.0   North Dakota   ND
102.4 3371.7 New Hampshire  NH
54.3 7722.4  Texas          TX
76.6 4451.4  Alabama        AL
```

```
307.6 4938.8 Delaware       DE
151.4 6506.4 California     CA
111.6 4665.6 Tennessee      TN
120.4 4649.9 North Carolina NC
;
```

CHARITY

```
data Charity;
   input School $ 1-7 Year 9-12 Name $ 14-20 MoneyRaised 22-26
         HoursVolunteered 28-29;
   datalines;
Monroe  2007 Allison 31.65 19
Monroe  2007 Barry   23.76 16
Monroe  2007 Candace 21.11  5
Monroe  2007 Danny    6.89 23
Monroe  2007 Edward  53.76 31
Monroe  2007 Fiona   48.55 13
Monroe  2007 Gert    24.00 16
Monroe  2007 Harold  27.55 17
Monroe  2007 Ima     15.98  9
Monroe  2007 Jack    20.00 23
Monroe  2007 Katie   22.11  2
Monroe  2007 Lisa    18.34 17
Monroe  2007 Tonya   55.16 40
Monroe  2007 Max     26.77 34
Monroe  2007 Ned     28.43 22
Monroe  2007 Opal    32.66 14
Monroe  2008 Patsy   18.33 18
Monroe  2008 Quentin 16.89 15
Monroe  2008 Randall 12.98 17
Monroe  2008 Sam     15.88  5
Monroe  2008 Tyra    21.88 23
Monroe  2008 Myrtle  47.33 26
Monroe  2008 Frank   41.11 22
Monroe  2008 Cameron 65.44 14
Monroe  2008 Vern    17.89 11
Monroe  2008 Wendell 23.00 10
Monroe  2008 Bob     26.88  6
Monroe  2008 Leah    28.99 23
Monroe  2009 Becky   30.33 26
Monroe  2009 Sally   35.75 27
Monroe  2009 Edgar   27.11 12
Monroe  2009 Dawson  17.24 16
Monroe  2009 Lou      5.12 16
Monroe  2009 Damien  18.74 17
Monroe  2009 Mona    27.43  7
Monroe  2009 Della   56.78 15
Monroe  2009 Monique 29.88 19
Monroe  2009 Carl    31.12 25
Monroe  2009 Reba    35.16 22
Monroe  2009 Dax     27.65 23
Monroe  2009 Gary    23.11 15
```

```
Monroe  2009 Suzie   26.65 11
Monroe  2009 Benito  47.44 18
Monroe  2009 Thomas  21.99 23
Monroe  2009 Annie   24.99 27
Monroe  2009 Paul    27.98 22
Monroe  2009 Alex    24.00 16
Monroe  2009 Lauren  15.00 17
Monroe  2009 Julia   12.98 15
Monroe  2009 Keith   11.89 19
Monroe  2009 Jackie  26.88 22
Monroe  2009 Pablo   13.98 28
Monroe  2009 L.T.    56.87 33
Monroe  2009 Willard 78.65 24
Monroe  2009 Kathy   32.88 11
Monroe  2009 Abby    35.88 10
Kennedy 2007 Arturo  34.98 14
Kennedy 2007 Grace   27.55 25
Kennedy 2007 Winston 23.88 22
Kennedy 2007 Vince   12.88 21
Kennedy 2007 Claude  15.62  5
Kennedy 2007 Mary    28.99 34
Kennedy 2007 Abner   25.89 22
Kennedy 2007 Jay     35.89 35
Kennedy 2007 Alicia  28.77 26
Kennedy 2007 Freddy  29.00 27
Kennedy 2007 Eloise  31.67 25
Kennedy 2007 Jenny   43.89 22
Kennedy 2007 Thelma  52.63 21
Kennedy 2007 Tina    19.67 21
Kennedy 2007 Eric    24.89 12
Kennedy 2007 Bubba   37.88 12
Kennedy 2007 G.L.    25.89 21
Kennedy 2007 Bert    28.89 21
Kennedy 2008 Clay    26.44 21
Kennedy 2008 Leeann  27.17 17
Kennedy 2008 Georgia 38.90 11
Kennedy 2008 Bill    42.23 25
Kennedy 2008 Holly   18.67 27
Kennedy 2008 Benny   19.09 25
Kennedy 2008 Cammie  28.77 28
Kennedy 2008 Amy     27.08 31
Kennedy 2008 Doris   22.22 24
Kennedy 2008 Robbie  19.80 24
Kennedy 2008 Ted     27.07 25
Kennedy 2008 Sarah   24.44 12
Kennedy 2008 Megan   28.89 11
Kennedy 2008 Jeff    31.11 12
Kennedy 2008 Taz     30.55 11
Kennedy 2008 George  27.56 11
Kennedy 2008 Heather 38.67 15
Kennedy 2009 Nancy   29.90 26
Kennedy 2009 Rusty   30.55 28
Kennedy 2009 Mimi    37.67 22
Kennedy 2009 J.C.    23.33 27
Kennedy 2009 Clark   27.90 25
Kennedy 2009 Rudy    27.78 23
```

```
Kennedy 2009 Samuel  34.44 18
Kennedy 2009 Forrest 28.89 26
Kennedy 2009 Luther  72.22 24
Kennedy 2009 Trey     6.78 18
Kennedy 2009 Albert  23.33 19
Kennedy 2009 Che-Min 26.66 33
Kennedy 2009 Preston 32.22 23
Kennedy 2009 Larry   40.00 26
Kennedy 2009 Anton   35.99 28
Kennedy 2009 Sid     27.45 25
Kennedy 2009 Will    28.88 21
Kennedy 2009 Morty   34.44 25
;
```

CONTROL ライブラリ

CONTROL ライブラリの内容

DATASETS プロシジャセクションで使用する CONTROL ライブラリの内容を次に示します。

Directory

Libref	CONTROL
Engine	V9
Physical Name	\myfiles\control
File Name	\myfiles\control

#	Name	Member Type	Obs, Entries or Indexes	Vars	Label	File Size	Last Modified
1	A1	CATALOG	23			62464	04Jan11:14:20:12
2	A2	CATALOG	1			17408	04Jan11:14:20:12
3	ALL	DATA	23	17		13312	04Jan11:14:20:12
4	BODYFAT	DATA	1	2		5120	04Jan11:14:20:12
5	CONFOUND	DATA	8	4		5120	04Jan11:14:20:12
6	CORONARY	DATA	39	4		5120	04Jan11:14:20:12
7	DRUG1	DATA	6	2	JAN11 Data	5120	04Jan11:14:20:12
8	DRUG2	DATA	13	2	MAY11 Data	5120	04Jan11:14:20:12
9	DRUG3	DATA	11	2	JUL10 Data	5120	04Jan11:14:20:12
10	DRUG4	DATA	7	2	JAN10 Data	5120	04Jan11:14:20:12
11	DRUG5	DATA	1	2	JUL10 Data	5120	04Jan11:14:20:12
12	ETEST1	CATALOG	1			17408	04Jan11:14:20:16
13	ETEST2	CATALOG	1			17408	04Jan11:14:20:16
14	ETEST3	CATALOG	1			17408	04Jan11:14:20:16
15	ETEST4	CATALOG	1			17408	04Jan11:14:20:16
16	ETEST5	CATALOG	1			17408	04Jan11:14:20:16
17	ETESTS	CATALOG	1			17408	04Jan11:14:20:16
18	FORMATS	CATALOG	6			17408	04Jan11:14:20:16
19	GROUP	DATA	148	11		25600	04Jan11:14:20:16
20	MLSCL	DATA	32	4	Multiple Sclerosis Data	5120	04Jan11:14:20:16

```
21  NAMES     DATA          7         4                      5120 04Jan11:14:20:16
22  OXYGEN    DATA         31         7                      9216 04Jan11:14:20:16
23  PERSONL   DATA        148        11                     25600 04Jan11:14:20:16
24  PHARM     DATA          6         3   Sugar Study        5120 04Jan11:14:20:16
25  POINTS    DATA          6         6                      5120 04Jan11:14:20:16
26  PRENAT    DATA        149         6                     17408 04Jan11:14:20:16
27  RESULTS   DATA         10         5                      5120 04Jan11:14:20:16
28  SLEEP     DATA        108         6                      9216 04Jan11:14:20:16
29  SYNDROME  DATA         46         8                      9216 04Jan11:14:20:16
30  TENSION   DATA          4         3                      5120 04Jan11:14:20:16
31  TEST2     DATA         15         5                      5120 04Jan11:14:20:16
32  TRAIN     DATA          7         2                      5120 04Jan11:14:20:16
33  VISION    DATA         16         3                      5120 04Jan11:14:20:16
34  WEIGHT    DATA         83        13   California        13312 04Jan11:14:20:16
                                          Results
35  WGHT      DATA         83        13   California        13312 04Jan11:14:20:16
                                          Results
16
```

CONTROL.ALL

CONTROL ライブラリのすべてのデータファイルの生データと DATA ステップを次に示します。

```
data control.all;
    input FMTNAME $8. START $9. END $8. LABEL $20. MIN best4. MAX best4.
         DEFAULT best4. LENGTH best4. FUZZ best8. PREFIX $2. MULT best8.
         FILL $1. NOEDIT best4.  TYPE $2. SEXCL $2. EEXCL $2. HLO $7. ;
    label FMTNAME='Format name'
          START='Starting value for format'
          END='Ending value for format'
          LABEL='Format value label'
          MIN='Minimum length'
          MAX='Maximum length'
          DEFAULT='Default length'
          LENGTH='Format length'
          FUZZ='Fuzz value'
          PREFIX='Prefix characters'
          MULT='Multiplier'
          FILL='Fill character'
          NOEDIT='Is picture string noedit?'
          TYPE='Type of format'
          SEXCL='Start exclusion'
          EEXCL='End exclusion'
          HLO='Additional information';
    datalines;
BENEFIT       LOW    7304         WORDDATE20.   1 40  20  20   1E-12      0.00   0 N N N     LF
BENEFIT      7305    HIGH  ** Not Eligible **   1 40  20  20   1E-12      0.00   0 N N N
DOLLARS       LOW    HIGH             000,000   1 40   7   7   1E-12 $    1.96   0 P N N     LH
NOZEROS       LOW    0.01                 999   1 40   5   5   1E-12 .  1000.00   0 P N Y     L
NOZEROS      0.01     0.1                  99   1 40   5   5   1E-12 .   100.00   0 P N Y
NOZEROS       0.1       1               0.000   1 40   5   5   1E-12 .  1000.00   0 P N Y
NOZEROS         1    HIGH               0.000   1 40   5   5   1E-12    1000.00   0 P N N     H
BRIT          BR1     BR1          Birmingham   1 40  14  14       0      0.00   0 C N N
BRIT          BR2     BR2            Plymouth   1 40  14  14       0      0.00   0 C N N
```

```
BRIT          BR3     BR3                York   1 40  14  14       0      0.00   0 C N N
BRIT     *OTHER****OTHER*      INCORRECT CODE   1 40  14  14       0      0.00   0 C N N     0
SKILL           A      D~              Test A   1 40   6   6       0      0.00   0 C N N
SKILL           E      M~              Test B   1 40   6   6       0      0.00   0 C N N
SKILL           N      Z~              Test C   1 40   6   6       0      0.00   0 C N N
SKILL           a      d~              Test A   1 40   6   6       0      0.00   0 C N N
SKILL           e      m~              Test B   1 40   6   6       0      0.00   0 C N N
SKILL           n      z~              Test C   1 40   6   6       0      0.00   0 C N N
EVAL            0       4              _SAME_   1 40   1   1       0      0.00   0 I N N     I
EVAL            C       C                   1   1 40   1   1       0      0.00   0 I N N
EVAL            E       E                   2   1 40   1   1       0      0.00   0 I N N
EVAL            N       N                   0   1 40   1   1       0      0.00   0 I N N
EVAL            O       O                   4   1 40   1   1       0      0.00   0 I N N
EVAL            S       S                   3   1 40   1   1       0      0.00   0 I N N
;
run;
```

CONTROL.BODYFAT

```
data control.bodyfat;
    input NAME $  AGE $;
    datalines;
jeff    44
;
run;
```

CONTROL.CONFOUND

```
data control.confound;
    input SMOKING $8.  STATUS $8.  CANCER $8.  WT;
    datalines;
Yes     Single  Yes     34
Yes     Single  No      120
Yes     Married Yes     7
Yes     Married No      30
Yes     Single  Yes     2
Yes     Single  No      30
Yes     Married Yes     6
Yes     Married No      145
;
run;
```

CONTROL.CORONARY

```
 ata control.coronary;
    input SEX  ECG AGE CA;
    datalines;
0       0       28      0
0       0       34      0
0       0       38      0
0       0       41      0
0       0       44      0
0       0       45      1
```

```
0       0       46      0
0       0       47      0
0       0       50      0
0       0       51      0
0       0       51      0
0       0       53      0
0       0       55      1
0       0       59      0
0       0       60      1
0       0       32      1
0       0       33      0
0       0       35      0
0       0       39      0
0       0       40      0
0       0       46      0
0       0       48      1
0       0       49      0
0       0       49      0
0       0       52      0
0       0       53      1
0       0       54      1
0       0       55      0
0       0       57      1
0       0       46      1
0       0       48      0
0       0       57      1
0       0       60      1
0       0       30      0
0       0       34      0
0       0       36      1
0       0       38      1
0       0       39      0
0       0       42      0;
run;
```

CONTROL.DRUG1

```
data control.drug1 (label='JAN2011 DATA');
    input CHAR $8.  NUM;
    datalines;
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
;
run;
```

CONTROL.DRUG2

```
 data control.drug2 (label='MAY2011 DATA');
    input CHAR $8.  NUM;
    datalines;
```

```
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
;
run;
```

CONTROL.DRUG3

```
data control.drug3 (label='JUL2010 DATA');
    input CHAR $8.  NUM;
    datalines;
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
;
run;
```

CONTROL.DRUG4

```
data control.drug4 (label='JAN2010 DATA');
    input CHAR $8.  NUM;
    datalines;
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
;
run;
```

CONTROL.DRUG5

```
data control.drug5 (label='JUL2010 DATA');
    input CHAR $8.  NUM;
```

```
    datalines;
junk    0;
run;
```

CONTROL.GROUP

```
data control.group;
    input IDNUM $ 1-4 LNAME $ 5-19 FNAME $ 20-34 CITY $ 35-49 STATE $
          50-52 SEX $ 53-54 JOBCODE $ 55-58 SALARY comma8. BIRTH
          HIRED date7. HPHONE $ 85-96;
    format salary comma8.;
    format hired date7.;
    informat hired date7.;
    datalines;
1919 ADAMS          GERALD          STAMFORD        CT M TA2   34,377    -4125 07JUN75 203/781-1255
1653 AHMAD          AZEEM           BRIDGEPORT      CT F ME2   35,109    -2631 12AUG78 203/675-7715
1400 ALVAREZ        GLORIA          NEW YORK        NY M ME1   29,770    -1515 19OCT78 212/586-0808
1350 ARTHUR         BARBARA         NEW YORK        NY F FA3   32,887    -2311 01AUG78 718/383-1549
1401 AVERY          JERRY           PATERSON        NJ M TA3   38,823    -7686 20NOV73 201/732-8787
1499 BAREFOOT       JOSEPH          PRINCETON       NJ M ME3   43,026    -6456 10JUN68 201/812-5665
1101 BASQUEZ        RICHARDO        NEW YORK        NY M SCP   18,724    -3493 04OCT78 212/586-8060
1333 BEAULIEU       ARMANDO         NEW YORK        NY M TA2   32,616    -3268 05DEC78 718/384-2849
1479 BOSTIC         MARIE           NEW YORK        NY F TA3   38,786    -1102 08OCT77 718/384-8816
1403 BOWDEN         EARL            BRIDGEPORT      CT M ME1   28,073    -1065 24DEC79 203/675-3434
1739 BOYCE          JONATHAN        NEW YORK        NY M PT1   66,518    -2560 30JAN79 212/587-1247
1658 BRADLEY        JEREMY          NEW YORK        NY M SCP   17,944    -1726 03MAR80 212/587-3622
1428 BRADY          CHRISTINE       STAMFORD        CT F PT1   68,768     -634 19NOV79 203/781-1212
1782 BROWN          JASON           STAMFORD        CT M ME2   35,346     -390 25FEB80 203/781-0019
1244 BRYANT         LEONARD         NEW YORK        NY M ME2   36,926    -3042 20JAN76 718/383-3334
1383 BURNETTE       THOMAS          NEW YORK        NY M BCK   25,824    -1434 23OCT80 718/384-3569
1574 CAHILL         MARSHALL        NEW YORK        NY M FA2   28,573    -4263 23DEC80 718/383-2338
1789 CANALES        VIVIANA         NEW YORK        NY M SCP   18,327    -5451 14APR66 212/587-9000
1404 CARTER         DONALD          NEW YORK        NY M PT2   91,377    -6882 04JAN68 718/384-2946
1437 CARTER         DOROTHY         BRIDGEPORT      CT F FA3   33,105    -4117 03SEP72 203/675-4117
1639 CARTER         KAREN           STAMFORD        CT F TA3   40,261    -5299 31JAN72 203/781-8839
1269 CASTON         FRANKLIN        STAMFORD        CT M NA1   41,691      126 01DEC80 203/781-3335
1065 CHAPMAN        NEIL            NEW YORK        NY M ME2   35,091   -10199 10JAN75 718/384-5618
1876 CHIN           JACK            NEW YORK        NY M TA3   39,676    -4971 30APR73 212/588-5634
1037 CHOW           JANE            STAMFORD        CT F TA1   28,559    -2819 16SEP80 203/781-8868
1129 COOK           BRENDA          NEW YORK        NY F ME2   34,930    -3673 20AUG79 718/383-2313
1988 COOPER         ANTHONY         NEW YORK        NY M FA3   32,218    -4412 21SEP72 212/587-1228
1405 DAVIDSON       JASON           PATERSON        NJ M SCP   18,057    -2125 29JAN80 201/732-2323
1430 DEAN           SANDRA          BRIDGEPORT      CT F TA2   32,926    -3591 30APR75 203/675-1647
1983 DEAN           SHARON          NEW YORK        NY F FA3   33,420    -3591 30APR75 718/384-1647
1134 DELGADO        MARIA           STAMFORD        CT F TA2   33,463    -1029 24DEC76 203/781-1528
1118 DENNIS         ROGER           NEW YORK        NY M PT3  111,380   -10209 21DEC68 718/383-1122
1438 DESAI          AAKASH          STAMFORD        CT F TA3   39,224    -2480 21NOV75 203/781-2229
1125 DUNLAP         DONNA           NEW YORK        NY F FA2   28,889    -1146 14DEC75 718/383-2094
1475 EATON          ALICIA          NEW YORK        NY F FA2   27,788    -3666 16JUL78 718/383-2828
1117 EDGERTON       JOSHUA          NEW YORK        NY M TA3   39,772    -3129 16AUG80 212/588-1239
1935 FERNANDEZ      KATRINA         BRIDGEPORT      CT F NA2   51,082    -6485 19OCT69 203/675-2962
1124 FIELDS         DIANA           WHITE PLAINS    NY F FA1   23,178    -4920 04OCT78 914/455-2998
1422 FLETCHER       MARIE           PRINCETON       NJ F FA1   22,455    -2764 09APR79 201/812-0902
1616 FLOWERS        ANNETTE         NEW YORK        NY F TA2   34,138     -668 07JUN81 718/384-3329
1406 FOSTER         GERALD          BRIDGEPORT      CT M ME2   35,186    -3948 20FEB75 203/675-6363
```

1120	GARCIA	JACK	NEW YORK	NY	M	ME1	28,620	257	10OCT81 718/384-4930
1094	GOMEZ	ALAN	BRIDGEPORT	CT	M	FA1	22,269	-636	20APR79 203/675-7181
1389	GORDON	LEVI	NEW YORK	NY	M	BCK	25,029	-4550	21AUG78 718/384-9326
1905	GRAHAM	ALVIN	NEW YORK	NY	M	PT1	65,112	109	01JUN80 212/586-8815
1407	GRANT	DANIEL	MT. VERNON	NY	M	PT1	68,097	-1011	21MAR78 914/468-1616
1114	GREEN	JANICE	NEW YORK	NY	F	TA2	32,929	-832	30JUN75 212/588-1092
1410	HARRIS	CHARLES	STAMFORD	CT	M	PT2	84,686	-1701	10NOV74 203/781-0937
1439	HARRISON	FELICIA	BRIDGEPORT	CT	F	PT1	70,737	-2854	13SEP78 203/675-4987
1409	HARTFORD	RAYMOND	STAMFORD	CT	M	ME3	41,552	-7924	25OCT69 203/781-9697
1408	HENDERSON	WILLIAM	PRINCETON	NJ	M	TA2	34,139	-4292	17OCT75 201/812-4789
1121	HERNANDEZ	MICHAEL	NEW YORK	NY	M	ME1	29,113	-94	10DEC79 718/384-3313
1991	HOLMES	GABRIEL	BRIDGEPORT	CT	F	TA1	27,646	130	15DEC80 203/675-0007
1102	HOLMES	SHANE	WHITE PLAINS	NY	M	TA2	34,543	-4472	18APR79 914/455-0976
1356	HOLMES	SHAWN	NEW YORK	NY	M	ME2	36,870	-5207	25FEB71 212/586-8411
1545	HUNTER	CLYDE	STAMFORD	CT	M	PT1	66,131	-4522	01JUN78 203/781-1119
1292	HUNTER	HELEN	BRIDGEPORT	CT	F	ME2	36,692	-2618	05JUL77 203/675-4830
1440	JACKSON	LAURA	STAMFORD	CT	F	ME2	35,758	-3380	12APR79 203/781-0088
1368	JEPSEN	RONALD	STAMFORD	CT	M	FA2	27,809	-3853	06NOV72 203/781-8413
1369	JOHNSON	ANTHONY	NEW YORK	NY	M	TA2	33,706	-3653	16MAR75 212/587-5385
1411	JOHNSON	JACKSON	PATERSON	NJ	M	FA2	27,266	-3868	04DEC77 201/732-3678
1113	JONES	LESLIE	NEW YORK	NY	F	FA1	22,368	-1444	20OCT79 718/383-3003
1704	JOSHI	ABHAY	NEW YORK	NY	M	BCK	25,466	-1947	01JUL75 718/384-0049
1900	KING	WILLIAM	NEW YORK	NY	M	ME2	35,106	-3505	30OCT75 718/383-3698
1126	KOSTECKA	NICHOLAS	NEW YORK	NY	F	TA3	40,900	-3137	24NOV68 212/586-1229
1677	KRAMER	JACKSON	BRIDGEPORT	CT	M	BCK	26,008	-2976	30MAR77 203/675-7432
1441	LAWRENCE	KATHY	PRINCETON	NJ	F	FA2	27,159	-770	26MAR79 201/812-3337
1421	LEE	RUSSELL	MT. VERNON	NY	M	TA2	33,156	-4738	03MAR78 914/468-9143
1119	LI	JEFF	NEW YORK	NY	M	TA1	26,925	-3479	09SEP76 212/586-2344
1834	LONG	RUSSELL	NEW YORK	NY	M	BCK	26,897	41	05JUL80 718/384-0040
1777	LUFKIN	ROY	NEW YORK	NY	M	PT3	109,631	-7402	24JUN69 718/383-4413
1663	MAHANNAHS	SHANTHA	NEW YORK	NY	M	BCK	26,453	-1813	14AUG79 212/587-7742
1106	MARSHBURN	JASPER	STAMFORD	CT	M	PT2	89,633	-5166	19AUG72 203/781-1457
1103	MCDANIEL	RONDA	NEW YORK	NY	F	FA1	23,739	-1412	26JUL80 212/586-0013
1477	MEYERS	PRESTON	BRIDGEPORT	CT	M	FA2	28,567	-2839	10MAR76 203/675-8125
1476	MONROE	JOYCE	STAMFORD	CT	F	TA2	34,804	-2039	20MAR75 203/781-2837
1379	MORGAN	ALFRED	STAMFORD	CT	M	ME3	42,265	-3795	13JUN72 203/781-2216
1104	MORGAN	CHRISTOPHER	NEW YORK	NY	M	SCP	17,947	-3170	13JUN79 718/383-9740
1009	MORGAN	GEORGE	NEW YORK	NY	M	TA1	28,881	-4685	29MAR80 212/586-7753
1412	MURPHEY	JOHN	PRINCETON	NJ	M	ME1	27,800	-5672	08DEC79 201/812-4414
1115	MURPHY	ALICE	NEW YORK	NY	F	FA3	32,700	-4146	03MAR68 718/384-1982
1128	NELSON	FELICIA	BRIDGEPORT	CT	F	TA2	32,778	-2411	23OCT78 203/675-1166
1442	NEWKIRK	SANDRA	PRINCETON	NJ	F	PT2	84,537	-1941	15APR76 201/812-3331
1417	NEWKIRK	WILLIAM	PATERSON	NJ	M	NA2	52,271	-2741	10MAR77 201/732-6611
1478	NEWTON	JAMES	NEW YORK	NY	M	PT2	84,204	-4525	27OCT78 212/587-5549
1673	NICHOLLS	HENRY	STAMFORD	CT	M	BCK	25,478	-670	18JUL79 203/781-7770
1839	NORRIS	DIANE	NEW YORK	NY	F	NA1	43,434	-395	06JUL81 718/384-1767
1347	O'NEAL	BRYAN	NEW YORK	NY	M	TA3	40,080	-1560	09SEP72 718/384-0230
1423	OSWALD	LESLIE	MT. VERNON	NY	F	ME2	35,774	-1324	22AUG78 914/468-9171
1200	OVERMAN	MICHELLE	STAMFORD	CT	F	ME1	27,817	-353	17AUG80 203/781-1835
1970	PAPI	PAOLO	NEW YORK	NY	F	FA1	22,616	-2651	15MAR79 718/383-3895
1521	PAPIA	ISMAEL	NEW YORK	NY	M	ME3	41,527	-3183	16JUL76 212/587-7603
1354	PAPIA	FRANCISCO	WHITE PLAINS	NY	F	SCP	18,336	-214	19JUN80 914/455-2337
1424	PATTERSON	RENEE	NEW YORK	NY	F	FA2	28,979	-877	14DEC77 212/587-8991
1132	PEARCE	CAROL	NEW YORK	NY	F	FA1	22,414	153	25OCT81 718/384-1986
1845	PEARSON	JAMES	NEW YORK	NY	M	BCK	25,997	-4422	25MAR68 718/384-2311
1556	PENNINGTON	MICHAEL	NEW YORK	NY	M	PT1	71,350	-2746	14DEC79 718/383-5681

```
1413 PETERS        RANDALL       PRINCETON      NJ M FA2   27,436    -2295 05JAN78 201/812-2478
1907 PHELPS        WILLIAM       STAMFORD       CT M TA2   33,330    -4061 09JUL75 203/781-1118
1436 PORTER        SUSAN         NEW YORK       NY F TA2   34,476    -2757 15MAR75 718/383-5777
1385 RAYNOR        MILTON        BRIDGEPORT     CT M ME3   43,901    -3634 04APR74 203/675-2846
1432 REED          MARILYN       MT. VERNON     NY F ME2   35,328    -3708 13FEB73 914/468-5454
1111 RHODES        JEREMY        PRINCETON      NJ M NA1   40,587      563 03NOV80 201/812-1837
1116 RICHARDS      CASEY         NEW YORK       NY F FA1   22,863     -822 24MAR79 212/587-1224
1352 RIVERS        SIMON         NEW YORK       NY M NA2   53,799    -4044 19OCT74 718/383-3345
1555 RODRIGUEZ     JULIA         BRIDGEPORT     CT F FA2   27,500    -1383 07JUL80 203/675-2401
1038 RODRIGUEZ     MARIA         BRIDGEPORT     CT F TA1   26,534     -780 26NOV79 203/675-2048
1420 ROUSE         JEREMY        PATERSON       NJ M ME3   43,072    -2504 25JUL75 201/732-9834
1561 SANDERS       RAYMOND       NEW YORK       NY M TA2   34,515    -2951 10OCT75 212/588-6615
1434 SANDERSON     EDITH         STAMFORD       CT F FA2   28,623    -3458 31OCT78 203/781-1333
1414 SARKAR        ABHEEK        BRIDGEPORT     CT M FA1   23,645       86 15APR80 203/675-1715
1112 SAYERS        RANDY         NEW YORK       NY M TA1   26,906    -2586 10DEC80 718/384-4895
1390 SMART         JONATHAN      NEW YORK       NY M FA2   27,762    -2504 26JUN79 718/383-1141
1332 STEPHENSON    ADAM          BRIDGEPORT     CT M NA1   42,179     -468 07JUN79 203/675-1497
1890 STEPHENSON    ROBERT        NEW YORK       NY M PT2   85,897    -7467 28NOV67 718/384-9874
1429 THOMPSON      ALICE         STAMFORD       CT F TA1   27,940    -4322 10AUG80 203/781-3857
1107 THOMPSON      WAYNE         NEW YORK       NY M PT2   89,978    -6412 13FEB67 718/384-3785
1908 TRENTON       MELISSA       NEW YORK       NY F TA2   32,996     -749 26APR78 212/586-6262
1830 TRIPP         KATHY         BRIDGEPORT     CT F PT2   84,472    -5329 01FEB71 203/675-2479
1882 TUCKER        ALAN          NEW YORK       NY M ME3   41,539    -5285 24NOV66 718/384-0216
1050 TUTTLE        THOMAS        WHITE PLAINS   NY M ME2   35,168    -3090 27AUG74 914/455-2119
1425 UNDERWOOD     JENNY         STAMFORD       CT F FA1   23,980       -1 03MAR81 203/781-0978
1928 UPCHURCH      LARRY         WHITE PLAINS   NY M PT2   89,859    -6313 16JUL78 914/455-5009
1480 UPDIKE        THERESA       NEW YORK       NY F TA3   39,584    -5230 28MAR69 212/587-8729
1100 VANDEUSEN     RICHARD       NEW YORK       NY M BCK   25,005    -4045 10MAY76 212/586-2531
1995 VARNER        ELIZABETH     NEW YORK       NY F ME1   28,811      604 22SEP81 718/384-7113
1135 VEGA          ANNA          NEW YORK       NY F FA2   27,322    -4117 03APR78 718/384-5913
1415 VEGA          FRANKLIN      NEW YORK       NY M FA2   28,279    -5043 15FEB76 718/384-2823
1076 VENTER        RANDALL       NEW YORK       NY M PT1   66,559      290 06OCT79 718/383-2321
1426 VICK          THERESA       PRINCETON      NJ F TA2   32,992    -1850 28JUN78 201/812-2424
1564 WALTERS       ANNE          NEW YORK       NY F SCP   18,834    -3548 04JUL80 212/587-3257
1221 WALTERS       DIANE         NEW YORK       NY F FA2   27,897    -1559 07OCT79 718/384-1918
1133 WANG          CHIN          NEW YORK       NY M TA1   27,702    -1995 15FEB80 212/587-1956
1435 WARD          ELAINE        NEW YORK       NY F TA3   38,809    -4614 11FEB68 718/383-4987
1418 WATSON        BERNARD       NEW YORK       NY M ME1   28,006    -5388 09JAN80 718/383-1298
1017 WELCH         DARIUS        NEW YORK       NY M TA3   40,859    -5114 19OCT69 212/586-5535
1443 WELLS         AGNES         STAMFORD       CT F NA1   42,275    -1137 01SEP79 203/781-5546
1131 WELLS         NADINE        NEW YORK       NY F TA2   32,576       -3 22APR79 718/383-1045
1427 WHALEY        CAROLYN       MT. VERNON     NY F TA2   34,047     -424 02FEB78 914/468-4528
1036 WONG          LESLIE        NEW YORK       NY F TA3   39,393    -2415 26OCT72 212/587-2570
1130 WOOD          DEBORAH       NEW YORK       NY F FA1   23,917     -227 08JUN80 212/587-0013
1127 WOOD          SANDRA        NEW YORK       NY F TA2   33,012    -2606 10DEC74 212/587-2881
1433 YANCEY        ROBIN         PRINCETON      NJ F FA3   32,983    -2000 20JAN75 201/812-1874
1431 YOUNG         DEBORAH       STAMFORD       CT F FA3   33,231    -2759 08APR76 203/781-2987
1122 YOUNG         JOANN         NEW YORK       NY F FA2   27,957    -3164 30NOV76 718/384-2021
1105 YOUNG         LAWRENCE      NEW YORK       NY M ME2   34,806    -3590 16AUG78 718/384-0008
;
run;
```

CONTROL.MLSCL

```
data control.mlscl (label='Multiple Sclerosis Data');
```

```
    input GROUP OBS1 OBS2 WT;
    datalines;
      1        4       4     10
      1        4       1      3
      1        4       2      7
      1        4       3      3
      1        1       4      1
      1        1       1     38
      1        1       2      5
      1        1       3      0
      1        2       4      0
      1        2       1     33
      1        2       2     11
      1        2       3      3
      1        3       4      6
      1        3       1     10
      1        3       2     14
      1        3       3      5
      2        4       1      1
      2        4       2      2
      2        4       3      4
      2        4       4     14
      2        3       1      2
      2        3       2     13
      2        3       3      3
      2        3       4      4
      2        2       1      3
      2        2       2     11
      2        2       3      4
      2        2       4      0
      2        1       1      5
      2        1       2      3
      2        1       3      0
      2        1       4      0
;
run;
```

CONTROL.NAMES

```
data control.names;
    input LABEL $ 1-16 START $ 17-24 FMTNAME $ 31-35 TYPE $ 41-41;
    datalines;
Capalleti, Jimmy    2355      bonus     C
Chen, Len           5889      bonus     C
Davis, Brad         3878      bonus     C
Leung, Brenda       4409      bonus     C
Patel, Mary         2398      bonus     C
Smith, Robert       5862      bonus     C
Zook, Carla         7385      bonus     C
;
run;
```

CONTROL.OXYGEN

```
data control.oxygen;
    input AGE WEIGHT RUNTIME RSTPULSE RUNPULSE MAXPULSE OXYGEN;
    datalines;
44     89.47     11.37        62          178         182      44.609
40     75.07     10.07        62          185         185      45.313
44     85.84      8.65        45          156         168      54.297
42     68.15      8.17        40          166         172      59.571
38     89.02      9.22        55          178         180      49.874
47     77.45     11.63        58          176         176      44.811
40     75.98     11.95        70          176         180      45.681
43     81.19     10.85        64          162         170      49.091
44     81.42     13.08        63          174         176      39.442
38     81.87      8.63        48          170         186      60.055
44     73.03     10.13        45          168         168      50.541
45     87.66     14.03        56          186         192      37.388
45     66.45     11.12        51          176         176      44.754
47     79.15     10.60        47          162         164      47.273
54     83.12     10.33        50          166         170      51.855
49     81.42      8.95        44          180         185      49.156
51     69.63     10.95        57          168         172      40.836
51     77.91     10.00        48          162         168      46.672
48     91.63     10.25        48          162         164      46.774
49     73.37     10.08        76          168         168      50.388
57     73.37     12.63        58          174         176      39.407
54     79.38     11.17        62          156         165      46.080
52     76.32      9.63        48          164         166      45.441
50     70.87      8.92        48          146         155      54.625
51     67.25     11.08        48          172         172      45.118
54     91.63     12.88        44          168         172      39.203
51     73.71     10.47        59          186         188      45.790
57     59.08      9.93        49          148         155      50.545
49     76.32      9.40        56          186         188      48.673
48     61.24     11.50        52          170         176      47.920
52     82.78     10.50        53          170         172      47.467
;
run;
```

CONTROL.PERSONL

```
data control.personl;
    input IDNUM $ 1-4 LNAME $ 5-19 FNAME $ 20-34 CITY $ 35-49 STATE $ 50-51
          SEX $ 53-53 JOBCODE $ 55-57 SALARY BIRTH date. @66
          HIRED date7. @74 HPHONE $ 84-95;
    format birth date7.;
    informat birth date.;
    format hired date7.;
    informat hired date.;
    datalines;
1919 ADAMS          GERALD          STAMFORD        CT M TA2  34376 15SEP70 07JUN05  203/781-1255
1653 AHMAD          AZEEM           BRIDGEPORT      CT F ME2  35108 18OCT72 12AUG98  203/675-7715
1400 ALVAREZ        GLORIA          NEW YORK        NY M ME1  29769 08NOV85 19OCT06  212/586-0808
```

1350	ARTHUR	BARBARA	NEW YORK	NY	F	FA3	32886	03SEP63	01AUG00	718/383-1549
1401	AVERY	JERRY	PATERSON	NJ	M	TA3	38822	16DEC68	20NOV93	201/732-8787
1499	BAREFOOT	JOSEPH	PRINCETON	NJ	M	ME3	43025	29APR62	10JUN95	201/812-5665
1101	BASQUEZ	RICHARDO	NEW YORK	NY	M	SCP	18723	09JUN80	04OCT98	212/586-8060
1333	BEAULIEU	ARMANDO	STAMFORD	CT	M	PT2	88606	02APR79	13FEB03	203/781-1777
1402	BLALOCK	RALPH	NEW YORK	NY	M	TA2	32615	20JAN71	05DEC98	718/384-2849
1479	BOSTIC	MARIE	NEW YORK	NY	F	TA3	38785	25DEC66	08OCT03	718/384-8816
1403	BOWDEN	EARL	BRIDGEPORT	CT	M	ME1	28072	31JAN79	24DEC99	203/675-3434
1739	BOYCE	JONATHAN	NEW YORK	NY	M	PT1	66517	28DEC82	30JAN00	212/587-1247
1658	BRADLEY	JEREMY	NEW YORK	NY	M	SCP	17943	11APR65	03MAR00	212/587-3622
1428	BRADY	CHRISTINE	STAMFORD	CT	F	PT1	68767	07APR80	19NOV02	203/781-1212
1428	BRADy	CHRISTINE	STAMFORD	CT	F	PT1	68767	07APR80	19NOV02	203/781-1212
1782	BROWN	JASON	STAMFORD	CT	M	ME2	35345	07DEC73	25FEB00	203/781-0019
1244	BRYANT	LEONARD	NEW YORK	NY	M	ME2	36925	03SEP71	20JAN96	718/383-3334
1383	BURNETTE	THOMAS	NEW YORK	NY	M	BCK	25824	28JAN76	23OCT00	718/384-3569
1574	CAHILL	MARSHALL	NEW YORK	NY	M	FA2	28572	30APR74	23DEC97	718/383-2338
1789	CANALES	VIVIANA	NEW YORK	NY	M	SCP	18326	28JAN85	14APR04	212/587-9000
1404	CARTER	DONALD	NEW YORK	NY	M	PT2	91376	27FEB71	04JAN98	718/384-2946
1437	CARTER	DOROTHY	BRIDGEPORT	CT	F	FA3	33104	23SEP68	03SEP92	203/675-4117
1639	CARTER	KAREN	STAMFORD	CT	F	TA3	40260	29JUN65	31JAN92	203/781-8839
1269	CASTON	FRANKLIN	STAMFORD	CT	M	NA1	41690	06MAY80	01DEC00	203/781-3335
1065	CHAPMAN	NEIL	NEW YORK	NY	M	ME2	35090	29JAN72	10JAN95	718/384-5618
1876	CHIN	JACK	NEW YORK	NY	M	TA3	39675	23MAY66	30APR96	212/588-5634
1037	CHOW	JANE	STAMFORD	CT	F	TA1	28558	13APR82	16SEP04	203/781-8868
1129	COOK	BRENDA	NEW YORK	NY	F	ME2	34929	11DEC79	20AUG03	718/383-2313
1988	COOPER	ANTHONY	NEW YORK	NY	M	FA3	32217	03DEC57	21SEP92	212/587-1228
1405	DAVIDSON	JASON	PATERSON	NJ	M	SCP	18056	08MAR54	29JAN00	201/732-2323
1430	DEAN	SANDRA	BRIDGEPORT	CT	F	TA2	32925	03MAR70	30APR05	203/675-1647
1983	DEAN	SHARON	NEW YORK	NY	F	FA3	33419	03MAR50	30APR85	718/384-1647
1134	DELGADO	MARIA	STAMFORD	CT	F	TA2	33462	08MAR77	24DEC04	203/781-1528
1118	DENNIS	ROGER	NEW YORK	NY	M	PT3	111379	19JAN57	21DEC88	718/383-1122
1438	DESAI	AAKASH	STAMFORD	CT	F	TA3	39223	18MAR63	21NOV03	203/781-2229
1125	DUNLAP	DONNA	NEW YORK	NY	F	FA2	28888	11NOV76	14DEC95	718/383-2094
1475	EATON	ALICIA	NEW YORK	NY	F	FA2	27787	18DEC71	16JUL98	718/383-2828
1117	EDGERTON	JOSHUA	NEW YORK	NY	M	TA3	39771	08JUN56	16AUG00	212/588-1239
1935	FERNANDEZ	KATRINA	BRIDGEPORT	CT	F	NA2	51081	31MAR72	19OCT01	203/675-2962
1124	FIELDS	DIANA	WHITE PLAINS	NY	F	FA1	23177	13JUL82	04OCT01	914/455-2998
1422	FLETCHER	MARIE	PRINCETON	NJ	F	FA1	22454	07JUN79	09APR99	201/812-0902
1616	FLOWERS	ANNETTE	NEW YORK	NY	F	TA2	34137	04MAR68	07JUN01	718/384-3329
1406	FOSTER	GERALD	BRIDGEPORT	CT	M	ME2	35185	11MAR69	20FEB95	203/675-6363
1120	GARCIA	JACK	NEW YORK	NY	M	ME1	28619	14SEP80	10OCT01	718/384-4930
1094	GOMEZ	ALAN	BRIDGEPORT	CT	M	FA1	22268	05APR78	20APR99	203/675-7181
1389	GORDON	LEVI	NEW YORK	NY	M	BCK	25028	18JUL67	21AUG03	718/384-9326
1905	GRAHAM	ALVIN	NEW YORK	NY	M	PT1	65111	19APR80	01JUN00	212/586-8815
1407	GRANT	DANIEL	MT. VERNON	NY	M	PT1	68096	26MAR77	21MAR98	914/468-1616
1114	GREEN	JANICE	NEW YORK	NY	F	TA2	32928	21SEP77	30JUN06	212/588-1092
1410	HARRIS	CHARLES	STAMFORD	CT	M	PT2	84685	06MAY75	10NOV99	203/781-0937
1439	HARRISON	FELICIA	BRIDGEPORT	CT	F	PT1	70736	09MAR72	13SEP98	203/675-4987
1409	HARTFORD	RAYMOND	STAMFORD	CT	M	ME3	41551	22APR68	25OCT99	203/781-9697
1408	HENDERSON	WILLIAM	PRINCETON	NJ	M	TA2	34138	01APR78	17OCT05	201/812-4789
1121	HERNANDEZ	MICHAEL	NEW YORK	NY	M	ME1	29112	29SEP79	10DEC99	718/384-3313
1991	HOLMES	GABRIEL	BRIDGEPORT	CT	F	TA1	27645	10MAY80	15DEC00	203/675-0007
1102	HOLMES	SHANE	WHITE PLAINS	NY	M	TA2	34542	04OCT77	18APR05	914/455-0976
1356	HOLMES	SHAWN	NEW YORK	NY	M	ME2	36869	29SEP75	25FEB01	212/586-8411
1545	HUNTER	CLYDE	STAMFORD	CT	M	PT1	66130	15AUG77	01JUN05	203/781-1119
1292	HUNTER	HELEN	BRIDGEPORT	CT	F	ME2	36691	31OCT72	05JUL97	203/675-4830

1440	JACKSON	LAURA	STAMFORD	CT	F	ME2	35757	30SEP70	12APR99	203/781-0088
1368	JEPSEN	RONALD	STAMFORD	CT	M	FA2	27808	14JUN69	06NOV02	203/781-8413
1369	JOHNSON	ANTHONY	NEW YORK	NY	M	TA2	33705	31DEC79	16MAR05	212/587-5385
1411	JOHNSON	JACKSON	PATERSON	NJ	M	FA2	27265	30MAY69	04DEC97	201/732-3678
1113	JONES	LESLIE	NEW YORK	NY	F	FA1	22367	18JAN76	20OCT99	718/383-3003
1704	JOSHI	ABHAY	NEW YORK	NY	M	BCK	25465	02SEP74	01JUL95	718/384-0049
1900	KING	WILLIAM	NEW YORK	NY	M	ME2	35105	28MAY70	30OCT95	718/383-3698
1126	kOSTECKA	NICHOLAS	NEW YORK	NY	F	TA3	40899	31MAY71	24NOV98	212/586-1229
1677	KRAMER	JACKSON	BRIDGEPORT	CT	M	BCK	26007	08NOV71	30MAR97	203/675-7432
1441	LAWRENCE	KATHY	PRINCETON	NJ	F	FA2	27158	22NOV77	26MAR99	201/812-3337
1421	LEE	RUSSELL	MT. VERNON	NY	M	TA2	33155	11JAN67	03MAR98	914/468-9143
1119	LI	JEFF	NEW YORK	NY	M	TA1	26924	23JUN70	09SEP96	212/586-2344
1834	LONG	RUSSELL	NEW YORK	NY	M	BCK	26896	11FEB80	05JUL00	718/384-0040
1777	LUFKIN	ROY	NEW YORK	NY	M	PT3	109630	26SEP59	24JUN89	718/383-4413
1663	MAHANNAHS	SHANTHA	NEW YORK	NY	M	BCK	26452	14JAN75	14AUG99	212/587-7742
1106	MARSHBURN	JASPER	STAMFORD	CT	M	PT2	89632	09NOV65	19AUG92	203/781-1457
1103	MCDANIEL	RONDA	NEW YORK	NY	F	FA1	23738	19FEB76	26JUL90	212/586-0013
1477	MEYERS	PRESTON	BRIDGEPORT	CT	M	FA2	28566	24MAR72	10MAR96	203/675-8125
1476	MONROE	JOYCE	STAMFORD	CT	F	TA2	34803	02JUN74	20MAR95	203/781-2837
1379	MORGAN	ALFRED	STAMFORD	CT	M	ME3	42264	11AUG69	13JUN02	203/781-2216
1104	MORGAN	CHRISTOPHER	NEW YORK	NY	M	SCP	17946	28APR71	13JUN99	718/383-9740
1009	MORGAN	GEORGE	NEW YORK	NY	M	TA1	28880	05MAR67	29MAR00	212/586-7753
1412	MURPHEY	JOHN	PRINCETON	NJ	M	ME1	27799	21JUN64	08DEC99	201/812-4414
1115	MURPHY	ALICE	NEW YORK	NY	F	FA3	32699	25AUG68	03MAR91	718/384-1982
1128	NELSON	FELICIA	BRIDGEPORT	CT	F	TA2	32777	26MAY73	23OCT98	203/675-1166
1442	NEWKIRK	SANDRA	PRINCETON	NJ	F	PT2	84536	08SEP74	15APR96	201/812-3331
1417	NEWKIRK	WILLIAM	PATERSON	NJ	M	NA2	52270	30JUN72	10MAR97	201/732-6611
1478	NEWTON	JAMES	NEW YORK	NY	M	PT2	84203	12AUG67	27OCT98	212/587-5549
1673	NICHOLLS	HENRY	STAMFORD	CT	M	BCK	25477	02MAR78	18JUL99	203/781-7770
1839	NORRIS	DIANE	NEW YORK	NY	F	NA1	44433	02DEC78	06JUL01	718/384-1767
1347	O'NEAL	BRYAN	NEW YORK	NY	M	TA3	40079	24SEP75	09SEP95	718/384-0230
1423	OSWALD	LESLIE	MT. VERNON	NY	F	ME2	35773	17MAY76	22AUG98	914/468-9171
1200	OVERMAN	MICHELLE	STAMFORD	CT	F	ME1	27816	13JAN79	17AUG02	203/781-1835
1970	PAPI	PAOLO	NEW YORK	NY	F	FA1	22615	28SEP72	15MAR99	718/383-3895
1521	PAPIA	ISMAEL	NEW YORK	NY	M	ME3	41526	15APR71	16JUL96	212/587-7603
1354	PAPIA	FRANCISCO	WHITE PLAINS	NY	F	SCP	18335	01JUN79	19JUN00	914/455-2337
1424	PATTERSON	RENEE	NEW YORK	NY	F	FA2	28978	07AUG77	14DEC97	212/587-8991
1132	PEARCE	CAROL	NEW YORK	NY	F	FA1	22413	02JUN80	25OCT01	718/384-1986
1845	PEARSON	JAMES	NEW YORK	NY	M	BCK	25996	23NOV67	25MAR95	718/384-2311
1556	PENNINGTON	MICHAEL	NEW YORK	NY	M	PT1	71349	25JUN72	14DEC99	718/383-5681
1413	PETERS	RANDALL	PRINCETON	NJ	M	FA2	27435	19SEP73	05JAN98	201/812-2478
1907	PHELPS	WILLIAM	STAMFORD	CT	M	TA2	33329	18NOV68	09JUL95	203/781-1118
1436	PORTER	SUSAN	NEW YORK	NY	F	TA2	34475	14JUN72	15MAR95	718/383-5777
1385	RAYNOR	MILTON	BRIDGEPORT	CT	M	ME3	43900	19JAN70	04APR94	203/675-2846
1432	REED	MARILYN	MT. VERNON	NY	F	ME2	35327	06NOV69	13FEB93	914/468-5454
1111	RHODES	JEREMY	PRINCETON	NJ	M	NA1	40586	17JUL81	03NOV02	201/812-1837
1116	RICHARDS	CASEY	NEW YORK	NY	F	FA1	22862	01OCT77	24MAR99	212/587-1224
1352	RIVERS	SIMON	NEW YORK	NY	M	NA2	53798	05DEC68	19OCT94	718/383-3345
1555	RODRIGUEZ	JULIA	BRIDGEPORT	CT	F	FA2	27499	19MAR76	07JUL00	203/675-2401
1038	RODRIGUEZ	MARIA	BRIDGEPORT	CT	F	TA1	26533	12NOV77	26NOV99	203/675-2048
1420	ROUSE	JEREMY	PATERSON	NJ	M	ME3	43071	22FEB73	25JUL95	201/732-9834
1561	SANDERS	RAYMOND	NEW YORK	NY	M	TA2	34514	03DEC71	10OCT95	212/588-6615
1434	SANDERSON	EDITH	STAMFORD	CT	F	FA2	28622	14JUL70	31OCT98	203/781-1333
1414	SARKAR	ABHEEK	BRIDGEPORT	CT	M	FA1	23644	27MAR80	15APR00	203/675-1715
1112	SAYERS	RANDY	NEW YORK	NY	M	TA1	26905	02DEC72	10DEC00	718/384-4895
1390	SMART	JONATHAN	NEW YORK	NY	M	FA2	27761	22FEB73	26JUN09	718/383-1141

```
1332 STEPHENSON    ADAM          BRIDGEPORT     CT M NA1  42178 20SEP78 07JUN99  203/675-1497
1890 STEPHENSON    ROBERT        NEW YORK       NY M PT2  85896 23JUL59 28NOV87  718/384-9874
1429 THOMPSON      ALICE         STAMFORD       CT F TA1  27939 02MAR68 10AUG00  203/781-3857
1107 THOMPSON      WAYNE         NEW YORK       NY M PT2  89977 12JUN62 13FEB87  718/384-3785
1908 TRENTON       MELISSA       NEW YORK       NY F TA2  32995 13DEC77 26APR98  212/586-6262
1830 TRIPP         KATHY         BRIDGEPORT     CT F PT2  84471 30MAY65 01FEB91  203/675-2479
1882 TUCKER        ALAN          NEW YORK       NY M ME3  41538 13JUL65 24NOV86  718/384-0216
1050 TUTTLE        THOMAS        WHITE PLAINS   NY M ME2  35167 17JUL71 27AUG94  914/455-2119
1425 UNDERWOOD     JENNY         STAMFORD       CT F FA1  23979 31DEC79 03MAR01  203/781-0978
1928 UPCHURCH      LARRY         WHITE PLAINS   NY M PT2  89858 19SEP62 16JUL98  914/455-5009
1480 UPDIKE        THERESA       NEW YORK       NY F TA3  39583 06SEP65 28MAR89  212/587-8729
1100 VANDEUSEN     RICHARD       NEW YORK       NY M BCK  25004 04DEC68 10MAY96  212/586-2531
1995 VARNER        ELIZABETH     NEW YORK       NY F ME1  28810 27AUG81 22SEP01  718/384-7113
1135 VEGA          ANNA          NEW YORK       NY F FA2  27321 23SEP68 03APR98  718/384-5913
1415 VEGA          FRANKLIN      NEW YORK       NY M FA2  28278 12MAR66 15FEB96  718/384-2823
1076 VENTER        RANDALL       NEW YORK       NY M PT1  66558 17OCT80 06OCT99  718/383-2321
1426 VICK          THERESA       PRINCETON      NJ F TA2  32991 08DEC74 28JUN98  201/812-2424
1564 WALTERS       ANNE          NEW YORK       NY F SCP  18833 15APR70 04JUL00  212/587-3257
1221 WALTERS       DIANE         NEW YORK       NY F FA2  27896 25SEP75 07OCT99  718/384-1918
1133 WANG          CHIN          NEW YORK       NY M TA1  27701 16JUL74 15FEB00  212/587-1956
1435 WARD          ELAINE        NEW YORK       NY F TA3  38808 15MAY67 11FEB88  718/383-4987
1418 WATSON        BERNARD       NEW YORK       NY M ME1  28005 01APR65 09JAN00  718/383-1298
1017 WELCH         DARIUS        NEW YORK       NY M TA3  40858 31DEC65 19OCT89  212/586-5535
1443 WELLS         AGNES         STAMFORD       CT F NA1  42274 20NOV66 01SEP99  203/781-5546
1131 WELLS         NADINE        NEW YORK       NY F TA2  32575 29DEC79 22APR99  718/383-1045
1427 WHALEY        CAROLYN       MT. VERNON     NY F TA2  34046 03NOV78 02FEB98  914/468-4528
1036 WONG          LESLIE        NEW YORK       NY F TA3  39392 22MAY73 26OCT92  212/587-2570
1130 WOOD          DEBORAH       NEW YORK       NY F FA1  23916 19MAY79 08JUN00  212/587-0013
1127 WOOD          SANDRA        NEW YORK       NY F TA2  33011 12NOV72 10DEC94  212/587-2881
1433 YANCEY        ROBIN         PRINCETON      NJ F FA3  32982 11JUL74 20JAN95  201/812-1874
1431 YOUNG         DEBORAH       STAMFORD       CT F FA3  33230 12JUN72 08APR96  203/781-2987
1122 YOUNG         JOANN         NEW YORK       NY F FA2  27956 04MAY71 30NOV96  718/384-2021
1105 YOUNG         LAWRENCE      NEW YORK       NY M ME2  34805 04MAR70 16AUG98  718/384-0008
;
run;
```

CONTROL.PHARM

```
data control.pharm (label='Sugar Study');
    input DRUG $8. RESPONSE $8. WT ;
    datalines;
      A cured   14
      A uncured 22
      B cured   24
      B uncured 19
      C cured   17
      C uncured 13
;
run;
```

CONTROL.POINTS

```
data control.points;
    input EMPID $8. Q1 Q2 Q3 Q4 TOTPTS;
```

```
    datalines;
2355           3       4       4       3       14
5889           2       2       2       2       8
3878           1       2       2       2       7
4409           0       1       1       1       3
2398           2       2       1       1       6
5862           1       1       1       2       5
;
run;
```

CONTROL.PRENAT

```
data control.prenat;
    input IDNUM $ 1-4 LNAME $ 6-20  FNAME $ 22-36 CITY $ 39-53
          STATE $ 55-56 HPHONE $ 58-69;
    datalines;
1919 ADAMS           GERALD          STAMFORD        CT 203/781-1255
1653 ALIBRANDI       MARIA           BRIDGEPORT      CT 203/675-7715
1400 ALHERTANI       ABDULLAH        NEW YORK        NY 212/586-0808
1350 ALVAREZ         MERCEDES        NEW YORK        NY 718/383-1549
1401 ALVAREZ         CARLOS          PATERSON        NJ 201/732-8787
1499 BAREFOOT        JOSEPH          PRINCETON       NJ 201/812-5665
1101 BAUCOM          WALTER          NEW YORK        NY 212/586-8060
1333 BANADYGA        JUSTIN          STAMFORD        CT 203/781-1777
1402 BLALOCK         RALPH           NEW YORK        NY 718/384-2849
1479 BALLETTI        MARIE           NEW YORK        NY 718/384-8816
1403 BOWDEN          EARL            BRIDGEPORT      CT 203/675-3434
1739 BRANCACCIO      JOSEPH          NEW YORK        NY 212/587-1247
1658 BREUHAUS        JEREMY          NEW YORK        NY 212/587-3622
1428 BRADY           CHRISTINE       STAMFORD        CT 203/781-1212
1782 BREWCZAK        JAKOB           STAMFORD        CT 203/781-0019
1244 BUCCI           ANTHONY         NEW YORK        NY 718/383-3334
1383 BURNETTE        THOMAS          NEW YORK        NY 718/384-3569
1574 CAHILL          MARSHALL        NEW YORK        NY 718/383-2338
1789 CARAWAY         DAVIS           NEW YORK        NY 212/587-9000
1404 COHEN           LEE             NEW YORK        NY 718/384-2946
1437 CARTER          DOROTHY         BRIDGEPORT      CT 203/675-4117
1639 CARTER-COHEN    KAREN           STAMFORD        CT 203/781-8839
1269 CASTON          FRANKLIN        STAMFORD        CT 203/781-3335
1065 COPAS           FREDERICO       NEW YORK        NY 718/384-5618
1876 CHIN            JACK            NEW YORK        NY 212/588-5634
1037 CHOW            JANE            STAMFORD        CT 203/781-8868
1129 COUNIHAN        BRENDA          NEW YORK        NY 718/383-2313
1988 COOPER          ANTHONY         NEW YORK        NY 212/587-1228
1405 DACKO           JASON           PATERSON        NJ 201/732-2323
1430 DABROWSKI       SANDRA          BRIDGEPORT      CT 203/675-1647
1983 DEAN            SHARON          NEW YORK        NY 718/384-1647
1134 DELGADO         MARIA           STAMFORD        CT 203/781-1528
1118 DENNIS          ROGER           NEW YORK        NY 718/383-1122
1438 DABBOUSSI       KAMILLA         STAMFORD        CT 203/781-2229
1125 DUNLAP          DONNA           NEW YORK        NY 718/383-2094
1475 ELGES           MARGARETE       NEW YORK        NY 718/383-2828
1117 EDGERTON        JOSHUA          NEW YORK        NY 212/588-1239
1935 FERNANDEZ       KATRINA         BRIDGEPORT      CT 203/675-2962
1124 FIELDS          DIANA           WHITE PLAINS    NY 914/455-2998
```

1422	FUJIHARA	KYOKO	PRINCETON	NJ 201/812-0902
1616	FUENTAS	CARLA	NEW YORK	NY 718/384-3329
1406	FOSTER	GERALD	BRIDGEPORT	CT 203/675-6363
1120	GARCIA	JACK	NEW YORK	NY 718/384-4930
1094	GOMEZ	ALAN	BRIDGEPORT	CT 203/675-7181
1389	GOLDSTEIN	LEVI	NEW YORK	NY 718/384-9326
1905	GRAHAM	ALVIN	NEW YORK	NY 212/586-8815
1407	GREGORSKI	DANIEL	MT. VERNON	NY 914/468-1616
1114	GREENWALD	JANICE	NEW YORK	NY 212/588-1092
1410	HARRIS	CHARLES	STAMFORD	CT 203/781-0937
1439	HASENHAUER	CHRISTINA	BRIDGEPORT	CT 203/675-4987
1409	HAVELKA	RAYMOND	STAMFORD	CT 203/781-9697
1408	HENDERSON	WILLIAM	PRINCETON	NJ 201/812-4789
1121	HERNANDEZ	ROBERTO	NEW YORK	NY 718/384-3313
1991	HOWARD	GRETCHEN	BRIDGEPORT	CT 203/675-0007
1102	HERMANN	JOACHIM	WHITE PLAINS	NY 914/455-0976
1356	HOWARD	MICHAEL	NEW YORK	NY 212/586-8411
1545	HERRERO	CLYDE	STAMFORD	CT 203/781-1119
1292	HUNTER	HELEN	BRIDGEPORT	CT 203/675-4830
1440	JACKSON	LAURA	STAMFORD	CT 203/781-0088
1368	JEPSEN	RONALD	STAMFORD	CT 203/781-8413
1369	JONSON	ANTHONY	NEW YORK	NY 212/587-5385
1411	JOHNSEN	JACK	PATERSON	NJ 201/732-3678
1113	JOHNSON	LESLIE	NEW YORK	NY 718/383-3003
1704	JONES	NATHAN	NEW YORK	NY 718/384-0049
1900	KING	WILLIAM	NEW YORK	NY 718/383-3698
1126	KIMANI	ANNE	NEW YORK	NY 212/586-1229
1677	KRAMER	JACKSON	BRIDGEPORT	CT 203/675-7432
1441	LAWRENCE	KATHY	PRINCETON	NJ 201/812-3337
1421	LEE	RUSSELL	MT. VERNON	NY 914/468-9143
1119	LI	JEFF	NEW YORK	NY 212/586-2344
1834	LEBLANC	RUSSELL	NEW YORK	NY 718/384-0040
1777	LUFKIN	ROY	NEW YORK	NY 718/383-4413
1663	MARKS	JOHN	NEW YORK	NY 212/587-7742
1106	MARSHBURN	JASPER	STAMFORD	CT 203/781-1457
1103	MCDANIEL	RONDA	NEW YORK	NY 212/586-0013
1477	MEYERS	PRESTON	BRIDGEPORT	CT 203/675-8125
1476	MONROE	JOYCE	STAMFORD	CT 203/781-2837
1379	MORGAN	ALFRED	STAMFORD	CT 203/781-2216
1104	MORGAN	CHRISTOPHER	NEW YORK	NY 718/383-9740
1009	MORGAN	GEORGE	NEW YORK	NY 212/586-7753
1412	MURPHEY	JOHN	PRINCETON	NJ 201/812-4414
1115	MURPHY	ALICE	NEW YORK	NY 718/384-1982
1128	NELSON	FELICIA	BRIDGEPORT	CT 203/675-1166
1442	NEWKIRK	SANDRA	PRINCETON	NJ 201/812-3331
1417	NEWKIRK	WILLIAM	PATERSON	NJ 201/732-6611
1478	NEWTON	JAMES	NEW YORK	NY 212/587-5549
1673	NICHOLLS	HENRY	STAMFORD	CT 203/781-7770
1839	NORRIS	DIANE	NEW YORK	NY 718/384-1767
1347	O'NEAL	BRYAN	NEW YORK	NY 718/384-0230
1423	OSWALD	LESLIE	MT. VERNON	NY 914/468-9171
1200	OVERMAN	MICHELLE	STAMFORD	CT 203/781-1835
1970	PARKER	ANNE	NEW YORK	NY 718/383-3895
1521	PARKER	JAY	NEW YORK	NY 212/587-7603
1354	PARKER	MARY	WHITE PLAINS	NY 914/455-2337
1424	PATTERSON	RENEE	NEW YORK	NY 212/587-8991

```
1132 PEARCE          CAROL           NEW YORK        NY 718/384-1986
1845 PEARSON         JAMES           NEW YORK        NY 718/384-2311
1556 PENNINGTON      MICHAEL         NEW YORK        NY 718/383-5681
1413 PETERS          RANDALL         PRINCETON       NJ 201/812-2478
1123 PETERSON        SUZANNE         NEW YORK        NY 718/383-0077
1907 PHELPS          WILLIAM         STAMFORD        CT 203/781-1118
1436 PORTER          SUSAN           NEW YORK        NY 718/383-5777
1385 RAYNOR          MILTON          BRIDGEPORT      CT 203/675-2846
1432 REED            MARILYN         MT. VERNON      NY 914/468-5454
1111 RHODES          JEREMY          PRINCETON       NJ 201/812-1837
1116 RICHARDS        CASEY           NEW YORK        NY 212/587-1224
1352 RIVERS          SIMON           NEW YORK        NY 718/383-3345
1555 RODRIGUEZ       JULIA           BRIDGEPORT      CT 203/675-2401
1038 RODRIGUEZ       MARIA           BRIDGEPORT      CT 203/675-2048
1420 ROUSE           JEREMY          PATERSON        NJ 201/732-9834
1561 SANDERS         RAYMOND         NEW YORK        NY 212/588-6615
1434 SANDERSON       EDITH           STAMFORD        CT 203/781-1333
1414 SANDERSON       NATHAN          BRIDGEPORT      CT 203/675-1715
1112 SANYERS         RANDY           NEW YORK        NY 718/384-4895
1390 SMART           JONATHAN        NEW YORK        NY 718/383-1141
1332 STEPHENSON      ADAM            BRIDGEPORT      CT 203/675-1497
1890 STEPHENSON      ROBERT          NEW YORK        NY 718/384-9874
1429 THOMPSON        ALICE           STAMFORD        CT 203/781-3857
1107 THOMPSON        WAYNE           NEW YORK        NY 718/384-3785
1908 TRENTON         MELISSA         NEW YORK        NY 212/586-6262
1830 TRIPP           KATHY           BRIDGEPORT      CT 203/675-2479
1882 TUCKER          ALAN            NEW YORK        NY 718/384-0216
1050 TUTTLE          THOMAS          WHITE PLAINS    NY 914/455-2119
1425 UNDERWOOD       JENNY           STAMFORD        CT 203/781-0978
1928 UPCHURCH        LARRY           WHITE PLAINS    NY 914/455-5009
1480 UPDIKE          THERESA         NEW YORK        NY 212/587-8729
1100 VANDEUSEN       RICHARD         NEW YORK        NY 212/586-2531
1995 VARNER          ELIZABETH       NEW YORK        NY 718/384-7113
1135 VEGA            ANNA            NEW YORK        NY 718/384-5913
1415 VEGA            FRANKLIN        NEW YORK        NY 718/384-2823
1076 VENTER          RANDALL         NEW YORK        NY 718/383-2321
1426 VICK            THERESA         PRINCETON       NJ 201/812-2424
1564 WALTERS         ANNE            NEW YORK        NY 212/587-3257
1221 WALTERS         DIANE           NEW YORK        NY 718/384-1918
1133 WANG            CHIN            NEW YORK        NY 212/587-1956
1435 WARD            ELAINE          NEW YORK        NY 718/383-4987
1418 WATSON          BERNARD         NEW YORK        NY 718/383-1298
1017 WELCH           DARIUS          NEW YORK        NY 212/586-5535
1443 WELLS           AGNES           STAMFORD        CT 203/781-5546
1131 WELLS           NADINE          NEW YORK        NY 718/383-1045
1427 WHALEY          CAROLYN         MT. VERNON      NY 914/468-4528
1036 WONG            LESLIE          NEW YORK        NY 212/587-2570
1130 WOOD            DEBORAH         NEW YORK        NY 212/587-0013
1127 WOOD            SANDRA          NEW YORK        NY 212/587-2881
1433 YANCEY          ROBIN           PRINCETON       NJ 201/812-1874
1431 YOUNG           DEBORAH         STAMFORD        CT 203/781-2987
1122 YOUNG           JOANN           NEW YORK        NY 718/384-2021
1105 YOUNG           LAWRENCE        NEW YORK        NY 718/384-0008

;
run;
```

CONTROL.RESULTS

```
data control.results;
    input ID    TREAT $8.   INITWT    WT3MOS    AGE;
    datalines;
       1   Other  166.28  146.98     35
       2   Other  214.42  210.22     54
       3   Other  172.46  159.42     33
       5   Other  175.41  160.66     37
       6   Other  173.13  169.40     20
       7   Other  181.25  170.94     30
      10   Other  239.83  214.48     48
      11   Other  175.32  162.66     51
      12   Other  227.01  211.06     29
      13   Other  274.82  251.82     31
;
run;
```

CONTROL.SLEEP

```
data control.sleep;
    input  GROUP TIME  SOL  WASO FNA TST;
    datalines;
    1.00    1.00   38.69   48.43     0      0
    2.36  424.50    0.00    0.00     0      0
    1.00    2.00   15.83    9.67     0      0
    2.16  500.30    0.00    0.00     0      0
    1.00    1.00   93.04   87.10     0      0
    1.86  302.40    0.00    0.00     0      0
    1.00    2.00   65.00   16.67     0      0
    1.50  305.00    0.00    0.00     0      0
    1.00    1.00   19.82   74.38     0      0
    2.00  359.20    0.00    0.00     0      0
    1.00    2.00   13.75   13.90     0      0
    1.40  378.13    0.00    0.00     0      0
    1.00    1.00   47.35   60.52     0      0
    2.89  248.10    0.00    0.00     0      0
    1.00    2.00   72.14   86.07     0      0
    4.14  308.50    0.00    0.00     0      0
    1.00    1.00   99.65  151.79     0      0
    3.86  263.93    0.00    0.00     0      0
    1.00    2.00   65.35  118.33     0      0
    2.71  374.17    0.00    0.00     0      0
    1.00    1.00   47.15  105.36     0      0
    2.50  254.60    0.00    0.00     0      0
    1.00    2.00   66.00   92.60     0      0
    2.20  297.40    0.00    0.00     0      0
    1.00    1.00   30.84   84.97     0      0
    3.24  242.90    0.00    0.00     0      0
    1.00    2.00    7.86   23.86     0      0
    1.28  340.70    0.00    0.00     0      0
    1.00    1.00  117.41   36.93     0      0
    1.00  204.20    0.00    0.00     0      0
```

1.00	2.00	50.42	24.86	0	0
1.85	231.00	0.00	0.00	0	0
1.00	1.00	6.50	41.15	0	0
2.67	308.00	0.00	0.00	0	0
1.00	2.00	2.00	0.00	0	0
0.00	454.40	0.00	0.00	0	0
2.00	1.00	54.29	84.93	0	0
3.43	394.70	0.00	0.00	0	0
2.00	2.00	13.57	48.00	0	0
1.00	405.00	0.00	0.00	0	0
2.00	1.00	4.43	58.43	0	0
2.89	375.70	0.00	0.00	0	0
2.00	2.00	5.00	49.28	0	0
3.57	423.60	0.00	0.00	0	0
2.00	1.00	32.50	38.43	0	0
4.57	357.20	0.00	0.00	0	0
2.00	2.00	17.14	33.14	0	0
3.57	315.70	0.00	0.00	0	0
2.00	1.00	33.57	83.43	0	0
4.36	282.50	0.00	0.00	0	0
2.00	2.00	22.85	37.86	0	0
2.42	288.57	0.00	0.00	0	0
2.00	1.00	53.21	120.00	0	0
3.50	366.80	0.00	0.00	0	0
2.00	2.00	29.00	97.40	0	0
2.80	388.00	0.00	0.00	0	0
2.00	1.00	56.79	67.73	0	0
3.19	326.00	0.00	0.00	0	0
2.00	2.00	52.50	64.16	0	0
4.00	45.80	0.00	0.00	0	0
2.00	1.00	23.50	35.19	0	0
6.60	416.80	0.00	0.00	0	0
2.00	2.00	25.71	38.43	0	0
8.50	446.30	0.00	0.00	0	0
2.00	1.00	43.00	95.59	0	0
1.75	288.90	0.00	0.00	0	0
2.00	2.00	45.00	85.71	0	0
1.43	272.10	0.00	0.00	0	0
3.00	1.00	24.70	67.17	0	0
3.90	399.90	0.00	0.00	0	0
3.00	2.00	3.86	22.50	0	0
3.00	425.14	0.00	0.00	0	0
3.00	1.00	70.72	80.43	0	0
3.00	286.30	0.00	0.00	0	0
3.00	2.00	22.50	139.33	0	0
3.50	283.50	0.00	0.00	0	0
3.00	1.00	48.23	40.57	0	0
1.65	407.20	0.00	0.00	0	0
3.00	2.00	30.00	38.00	0	0
1.57	372.90	0.00	0.00	0	0
3.00	1.00	43.93	34.57	0	0
1.29	393.40	0.00	0.00	0	0
3.00	2.00	30.71	58.43	0	0
2.28	348.60	0.00	0.00	0	0
3.00	1.00	95.00	35.22	0	0
2.06	409.90	0.00	0.00	0	0

```
    3.00    2.00   66.43   68.57     0      0
    2.14  345.60    0.00    0.00     0      0
    3.00    1.00   18.93   86.43     0      0
    5.36  349.50    0.00    0.00     0      0
    3.00    2.00   17.50  116.50     0      0
    5.00  337.70    0.00    0.00     0      0
    3.00    1.00  125.00   69.15     0      0
    2.00  242.50    0.00    0.00     0      0
    3.00    2.00   60.00   81.43     0      0
    1.57  304.30    0.00    0.00     0      0
    3.00    1.00   38.57   68.61     0      0
    3.64  394.50    0.00    0.00     0      0
    3.00    2.00   18.33   19.83     0      0
    2.00  448.83    0.00    0.00     0      0
    3.00    1.00   43.00  121.72     0      0
    2.63  247.80    0.00    0.00     0      0
    3.00    2.00   40.00   98.83     0      0
    1.86  199.67    0.00    0.00     0      0
    3.00    1.00   11.61   70.36     0      0
    2.86  348.40    0.00    0.00     0      0
    3.00    2.00   20.43   70.57     0      0
    3.14  335.40    0.00    0.00     0      0
 ;
run;
```

CONTROL.SYNDROME

```
data control.syndrome;
    input FLIGHT $ 1-3 @10 DATE DATE7.  @22 DEPART TIME5. ORIG $ 31-33
          DEST $ 39-41 MILES  BOARDED  CAPACITY;
    format date DATE7.;
    format depart TIME5.;
    informat date DATE7.;
    informat depart TIME5.;
    datalines;
114      01MAR11     7:10     LGA     LAX      2475      172         210
202      01MAR11    10:43     LGA     ORD       740      151         210
219      01MAR11     9:31     LGA     LON      3442      198         250
622      01MAR11    12:19     LGA     FRA      3857      207         250
132      01MAR11    15:35     LGA     YYZ       366      115         178
271      01MAR11    13:17     LGA     PAR      3635      138         250
302      01MAR11    20:22     LGA     WAS       229      105         180
114      02MAR11     7:10     LGA     LAX      2475      119         210
202      02MAR11    10:43     LGA     ORD       740      120         210
219      02MAR11     9:31     LGA     LON      3442      147         250
622      02MAR11    12:19     LGA     FRA      3857      176         250
132      02MAR11    15:35     LGA     YYZ       366      106         178
302      02MAR11    20:22     LGA     WAS       229       78         180
271      02MAR11    13:17     LGA     PAR      3635      104         250
114      03MAR11     7:10     LGA     LAX      2475      197         210
202      03MAR11    10:43     LGA     ORD       740      118         210
219      03MAR11     9:31     LGA     LON      3442      197         250
622      03MAR11    12:19     LGA     FRA      3857      180         250
132      03MAR11    15:35     LGA     YYZ       366       75         178
271      03MAR11    13:17     LGA     PAR      3635      147         250
```

```
302      03MAR11    20:22     LGA     WAS       229      123         180
114      04MAR11     7:10     LGA     LAX      2475      178         210
202      04MAR11    10:43     LGA     ORD       740      148         210
219      04MAR11     9:31     LGA     LON      3442      232         250
622      04MAR11    12:19     LGA     FRA      3857      137         250
132      04MAR11    15:35     LGA     YYZ       366      117         178
271      04MAR11    13:17     LGA     PAR      3635      146         250
302      04MAR11    20:22     LGA     WAS       229      115         180
114      05MAR11     7:10     LGA     LAX      2475      117         210
202      05MAR11    10:43     LGA     ORD       740      104         210
219      05MAR11     9:31     LGA     LON      3442      160         250
622      05MAR11    12:19     LGA     FRA      3857      185         250
132      05MAR11    15:35     LGA     YYZ       366      157         178
271      05MAR11    13:17     LGA     PAR      3635      177         250
114      06MAR11     7:10     LGA     LAX      2475      128         210
202      06MAR11    10:43     LGA     ORD       740      115         210
219      06MAR11     9:31     LGA     LON      3442      163         250
132      06MAR11    15:35     LGA     YYZ       366      150         178
302      06MAR11    20:22     LGA     WAS       229       66         180
114      07MAR11     7:10     LGA     LAX      2475      160         210
202      07MAR11    10:43     LGA     ORD       740      175         210
219      07MAR11     9:31     LGA     LON      3442      241         250
622      07MAR11    12:19     LGA     FRA      3857      210         250
132      07MAR11    15:35     LGA     YYZ       366      164         178
271      07MAR11    13:17     LGA     PAR      3635      155         250
302      07MAR11    20:22     LGA     WAS       229      135         180
;
run;
```

CONTROL.TENSION

```
data control.tension;
    input TENSION $8.  CHD $8.  COUNT ;
    datalines;
yes      yes       97
yes      no       307
no       yes      200
no       no      1409
;
run;
```

CONTROL.TEST2

```
data control.test2;
    input STD1 $ TEST1 $  STD2 $   TEST2 $    WT;
    datalines ;
neg      neg     neg      neg     509
neg      neg     neg      pos       4
neg      neg     pos      neg      17
neg      neg     pos      pos       3
neg      pos     neg      neg      13
neg      pos     neg      pos       8
neg      pos     pos      pos       8
```

```
pos      neg     neg      neg      14
pos      neg     neg      pos       1
pos      neg     pos      neg      17
pos      neg     pos      pos       9
pos      pos     neg      neg       7
pos      pos     neg      pos       4
pos      pos     pos      neg       9
pos      pos     pos      pos     170
;
run;
```

CONTROL.TRAIN

```
data control.train;
    input NAME $ 1-16 IDNUM $ 17-24;
    datalines;
Capalleti, Jimmy    2355
Chen, Len           5889
Davis, Brad         3878
Leung, Brenda       4409
Patel, Mary         2398
Smith, Robert       5862
Zook, Carla         7385
;
run;
```

CONTROL.VISION

```
data control.vision;
    input RIGHT LEFT COUNT;
    datalines;
      1        1    1520
      1        2     266
      1        3     124
      1        4      66
      2        1     234
      2        2    1512
      2        3     432
      2        4      78
      3        1     117
      3        2     362
      3        3    1772
      3        4     205
      4        1      36
      4        2      82
      4        3     179
      4        4     492
;
run;
```

CONTROL.WEIGHT

```
data control.weight (label='California Results');
    input ID TREAT $8.  IBW INITWT WT3MOS WT6MOS WT9MOS AGE MMPI1 MMPI2
          MMPI3 MMPI4 MMPI5;
    datalines;
  1  Other    149  166.28  146.98  138.26     .       35    62     68     67    55     67
  2  Other    137  214.42  210.22     .    213.87     54    57     56     59    57     47
  3  Other    138  172.46  159.42  146.01  143.84     33    54     69     63    87     34
  5  Other    122  175.41  160.66  154.30     .       37    56     67     64    71     32
  6  Other    134  173.13  169.40  176.12     .       20    42     51     63    71     45
  7  Other    160  181.25  170.94     .       .       30    72     58     70    71     80
  9  Other    152  212.83  179.93  169.74  164.47  49100    65     87     71    74      4
 10  Other    145  239.83  214.48  208.28     .       48    56     51     56    53     53
 11  Other    158  175.32  162.66  161.39     .       51    66     60     71    62     84
 12  Other    174  227.01  211.06  202.87  205.17     29    77     70     69    65     64
 13  Other    137  274.82  251.82  248.18     .       31    66     82     66    69     55
 15  Other    152  168.75  156.58  154.61  156.58     42    52     58     65    67     51
 16  Other    162  187.81  172.07     .       .       40    57     68     67    67     74
 17  Other    123  226.63     .    219.72     .       21    58     49     59    74     70
 18  Other    146  176.03  160.27  160.27     .       41    48     55     49    68     43
 19  Other    166  190.96  159.04     .       .       32    53     52     51    62     71
 21  Other    148  165.54     .    166.22     .       48    57     60     65    74     55
 22  Other    125  193.60  184.00     .       .       28    64     67     70    69     54
 24  Other    152  267.43  230.26  206.09     .       30    48     45     55    50     41
 25  Other    151  193.38  185.43     .       .       33    54     99     67    75     74
 26  Other    134  252.61  227.61  217.72  223.88     31    62     51     70    57     39
 27  Other    140  193.93  191.43  196.43     .       25    54     65     66    55     55
 29  Other    119  182.77     .       .       .       38    66     63     64    60     41
 30  Other    134  189.93  172.39  175.37     .       35    70     92     73    98     39
 31  Other    138  190.22  181.88  178.99  181.16     36    56     69     61    62     34
 32  Other    134  182.09  169.40  163.81  163.81     34    42     65     54    55     41
 34  Other    133  200.00  189.47     .       .       24    52     61     64    62     39
 35  Other    149  221.81  216.11     .       .       46    66     52     69    71     61
 36  Other    133  241.35  247.37     .       .       34    50     65     57    74     47
 37  Other    134  223.13  217.91     .       .       33    64     63     63    60     41
 38  Other    140  235.36  228.57  210.71     .       50    56     61     70    74     39
 39  Other    125  178.60  178.40     .       .       39    50     73     58    58     32
 40  Other    143  243.01  226.57  210.49     .       38    64     66     70    54     46
 41  Other    134  282.65  239.55     .       .       26    66     65     75    74     53
 44  Other    139  282.37  258.99  238.13  241.01     43    66     69     68    65     39
 45  Other    134  216.04  182.09     .       .       39    50     55     59    67     43
 46  Other    115  190.00  171.30     .       .       36    64     59     58    58     44
 47  Other    134  175.19  167.16     .       .       43    57     48     52    71     47
 48  Other    118  179.87     .       .       .       37    52     57     45    50     30
 50  Other    137  173.54  166.97  164.60     .       32    54     76     63    69     25
 51  Other    125  180.60  162.40  152.00  157.60     32    50     56     64    53     53
 52  Other    155  235.32  225.16  210.32  208.39     35    62     64     55    60     51
 53  Other    143  183.39  169.23     .       .       38    64     57     72    81     61
 54  Other    131  212.60  208.40  211.45     .       27    50     67     53    74     47
 63  Surgery  146  219.18  167.12  139.73  119.18     35    58     53     67    48     55
 65  Surgery  123  192.68  155.28  127.64  115.45     31    52     74     54    64     32
 66  Surgery  134  199.25  173.88  161.19  144.03     38    68     64     70    77     30
 71  Surgery  139  209.35  172.66  156.83  138.13     39    66     56     66    71     42
```

```
 77  Surgery  137  179.56  150.36  132.12  123.36     29    46     49     42    47     49
 80  Surgery  170  138.24  121.76  100.00     .       31    56     57     64    64     39
 81  Surgery  129  206.98  173.64  132.56  121.71     28    42     78     52    62     41
 84  Surgery  143  220.28  178.32     .       .      29    58     67     54    46     59
 85  Surgery  152  189.47  156.58  140.79  140.79     34    75     73     62    74     51
 86  Surgery  210  157.14  138.57  121.90  109.05     30    88     75     73    69     65
 92  Surgery  139  203.60  169.78  143.88     .       38    62     59     68    60     49
 93  Surgery  151  171.52  150.33  123.18  109.27     42    72     69     59    53     47
 95  Surgery  134  207.46  155.22     .       .      41    51     52     56    63     57
 96  Surgery  138  201.45  172.46  172.46  155.07     55    57     49     57    67     37
 97  Surgery  128  209.38  182.81  162.50  153.91     34    68     88     73    74      .
101  Surgery  166  185.54  146.39  139.76  132.53     42    82     80     89    79     51
102  Surgery  127  218.11  173.23  152.76  137.80     39    76     80     70    74     45
107  Surgery  128  227.34  192.97  184.38  170.31     49    50     51     59    50     55
110  Surgery  143  251.75  207.69  183.22  165.03     42    48     84     52    76     38
111  Surgery  152  197.37  164.47  148.68  132.89     56    90     70     86    76     70
112  Surgery  140  202.14  156.43  137.86     .       31    48     51     50    41     41
116  Surgery  139  273.38  235.25  194.24     .       31    58     55     66    81     45
117  Surgery  125  192.00  156.80  140.00  127.20     42    58     44     63    53     51
120  Surgery  119  277.31  231.93  208.40  192.44     31    60     69     59    95     34
122  Surgery  138  165.22  130.43  119.57     .       44    66     67     70    62     34
125  Surgery  152  257.89  200.00  182.89  134.21     39    47     84     53    88     74
126  Surgery  138  170.29  142.75     .       .      39    64     64     66    65     46
132  Surgery  149  208.05  173.83     .    126.85     34    82     57     75    65     45
133  Surgery  136  230.15  205.15  190.44  180.88     39    52     71     52    65     30
134  Surgery  168  177.98  140.48  119.64     .       45    70     77     62    67     69
137  Surgery  138  188.41  164.49  152.90  135.51     39    58     61     57    60     46
138  Surgery  141  268.09  220.57  185.82     .       32    52     59     66    74     53
158  Surgery  130  220.00  190.77  173.85     .       27    68     73     66    58     43
223  Surgery  157  220.38  185.35  152.23  138.85     43    78     76     70    57     53
266  Surgery  135  184.44  148.15  123.70     .       41    56     55     59    64     49
269  Surgery  155  201.29  151.61  140.65     .       57    82     84     86    77     73
293  Surgery  160  163.75  130.63     .       .      47    52     67     47    64     76
295  Surgery  128  211.72  172.66  151.56  137.50     45    93     65     80    60     41
298  Surgery  140  243.57  195.00     .    166.43     40    63     62     62    76     52
;
run;
```

CONTROL.WGHT

```
data control.wght (label='California Results');
    input ID TREAT $8.  IBW INITWT WT3MOS WT6MOS WT9MOS AGE MMPI1 MMPI2
          MMPI3 MMPI4 MMPI5;
    datalines;
  1  Other    149  166.28  146.98  138.26     .       35    62     68     67    55     67
  2  Other    137  214.42  210.22     .    213.87     54    57     56     59    57     47
  3  Other    138  172.46  159.42  146.01  143.84     33    54     69     63    87     34
  5  Other    122  175.41  160.66  154.30     .       37    56     67     64    71     32
  6  Other    134  173.13  169.40  176.12     .       20    42     51     63    71     45
  7  Other    160  181.25  170.94     .       .      30    72     58     70    71     80
  9  Other    152  212.83  179.93  169.74  164.47  49100    65     87     71    74      4
 10  Other    145  239.83  214.48  208.28     .       48    56     51     56    53     53
 11  Other    158  175.32  162.66  161.39     .       51    66     60     71    62     84
 12  Other    174  227.01  211.06  202.87  205.17     29    77     70     69    65     64
 13  Other    137  274.82  251.82  248.18     .       31    66     82     66    69     55
```

15	Other	152	168.75	156.58	154.61	156.58	42	52	58	65	67	51
16	Other	162	187.81	172.07	.	.	40	57	68	67	67	74
17	Other	123	226.63	.	219.72	.	21	58	49	59	74	70
18	Other	146	176.03	160.27	160.27	.	41	48	55	49	68	43
19	Other	166	190.96	159.04	.	.	32	53	52	51	62	71
21	Other	148	165.54	.	166.22	.	48	57	60	65	74	55
22	Other	125	193.60	184.00	.	.	28	64	67	70	69	54
24	Other	152	267.43	230.26	206.09	.	30	48	45	55	50	41
25	Other	151	193.38	185.43	.	.	33	54	99	67	75	74
26	Other	134	252.61	227.61	217.72	223.88	31	62	51	70	57	39
27	Other	140	193.93	191.43	196.43	.	25	54	65	66	55	55
29	Other	119	182.77	.	.	.	38	66	63	64	60	41
30	Other	134	189.93	172.39	175.37	.	35	70	92	73	98	39
31	Other	138	190.22	181.88	178.99	181.16	36	56	69	61	62	34
32	Other	134	182.09	169.40	163.81	163.81	34	42	65	54	55	41
34	Other	133	200.00	189.47	.	.	24	52	61	64	62	39
35	Other	149	221.81	216.11	.	.	46	66	52	69	71	61
36	Other	133	241.35	247.37	.	.	34	50	65	57	74	47
37	Other	134	223.13	217.91	.	.	33	64	63	63	60	41
38	Other	140	235.36	228.57	210.71	.	50	56	61	70	74	39
39	Other	125	178.60	178.40	.	.	39	50	73	58	58	32
40	Other	143	243.01	226.57	210.49	.	38	64	66	70	54	46
41	Other	134	282.65	239.55	.	.	26	66	65	75	74	53
44	Other	139	282.37	258.99	238.13	241.01	43	66	69	68	65	39
45	Other	134	216.04	182.09	.	.	39	50	55	59	67	43
46	Other	115	190.00	171.30	.	.	36	64	59	58	58	44
47	Other	134	175.19	167.16	.	.	43	57	48	52	71	47
48	Other	118	179.87	.	.	.	37	52	57	45	50	30
50	Other	137	173.54	166.97	164.60	.	32	54	76	63	69	25
51	Other	125	180.60	162.40	152.00	157.60	32	50	56	64	53	53
52	Other	155	235.32	225.16	210.32	208.39	35	62	64	55	60	51
53	Other	143	183.39	169.23	.	.	38	64	57	72	81	61
54	Other	131	212.60	208.40	211.45	.	27	50	67	53	74	47
63	Surgery	146	219.18	167.12	139.73	119.18	35	58	53	67	48	55
65	Surgery	123	192.68	155.28	127.64	115.45	31	52	74	54	64	32
66	Surgery	134	199.25	173.88	161.19	144.03	38	68	64	70	77	30
71	Surgery	139	209.35	172.66	156.83	138.13	39	66	56	66	71	42
77	Surgery	137	179.56	150.36	132.12	123.36	29	46	49	42	47	49
80	Surgery	170	138.24	121.76	100.00	.	31	56	57	64	64	39
81	Surgery	129	206.98	173.64	132.56	121.71	28	42	78	52	62	41
84	Surgery	143	220.28	178.32	.	.	29	58	67	54	46	59
85	Surgery	152	189.47	156.58	140.79	140.79	34	75	73	62	74	51
86	Surgery	210	157.14	138.57	121.90	109.05	30	88	75	73	69	65
92	Surgery	139	203.60	169.78	143.88	.	38	62	59	68	60	49
93	Surgery	151	171.52	150.33	123.18	109.27	42	72	69	59	53	47
95	Surgery	134	207.46	155.22	.	.	41	51	52	56	63	57
96	Surgery	138	201.45	172.46	172.46	155.07	55	57	49	57	67	37
97	Surgery	128	209.38	182.81	162.50	153.91	34	68	88	73	74	.
101	Surgery	166	185.54	146.39	139.76	132.53	42	82	80	89	79	51
102	Surgery	127	218.11	173.23	152.76	137.80	39	76	80	70	74	45
107	Surgery	128	227.34	192.97	184.38	170.31	49	50	51	59	50	55
110	Surgery	143	251.75	207.69	183.22	165.03	42	48	84	52	76	38
111	Surgery	152	197.37	164.47	148.68	132.89	56	90	70	86	76	70
112	Surgery	140	202.14	156.43	137.86	.	31	48	51	50	41	41
116	Surgery	139	273.38	235.25	194.24	.	31	58	55	66	81	45
117	Surgery	125	192.00	156.80	140.00	127.20	42	58	44	63	53	51

```
120  Surgery  119  277.31  231.93  208.40  192.44     31    60     69     59     95     34
122  Surgery  138  165.22  130.43  119.57     .       44    66     67     70     62     34
125  Surgery  152  257.89  200.00  182.89  134.21     39    47     84     53     88     74
126  Surgery  138  170.29  142.75     .       .       39    64     64     66     65     46
132  Surgery  149  208.05  173.83     .    126.85     34    82     57     75     65     45
133  Surgery  136  230.15  205.15  190.44  180.88     39    52     71     52     65     30
134  Surgery  168  177.98  140.48  119.64     .       45    70     77     62     67     69
137  Surgery  138  188.41  164.49  152.90  135.51     39    58     61     57     60     46
138  Surgery  141  268.09  220.57  185.82     .       32    52     59     66     74     53
158  Surgery  130  220.00  190.77  173.85     .       27    68     73     66     58     43
223  Surgery  157  220.38  185.35  152.23  138.85     43    78     76     70     57     53
266  Surgery  135  184.44  148.15  123.70     .       41    56     55     59     64     49
269  Surgery  155  201.29  151.61  140.65     .       57    82     84     86     77     73
293  Surgery  160  163.75  130.63     .       .       47    52     67     47     64     76
295  Surgery  128  211.72  172.66  151.56  137.50     45    93     65     80     60     41
298  Surgery  140  243.57  195.00     .    166.43     40    63     62     62     76     52
;
run;
```

CUSTOMER_RESPONSE

```
data customer_response;
   input Customer Factor1-Factor4 Source1-Source3
        Quality1-Quality3;
   datalines;
1 . . 1 1 1 1 . 1 . .
2 1 1 . 1 1 1 . 1 1 .
3 . . 1 1 1 1 . . . .
4 1 1 . 1 . 1 . . . 1
5 . 1 . 1 1 . . . . 1
6 . 1 . 1 1 . . . . .
7 . 1 . 1 1 . . 1 . .
8 1 . . 1 1 1 . 1 1 .
9 1 1 . 1 1 . . . . 1
10 1 . . 1 1 1 . 1 1 .
11 1 1 1 1 . 1 . 1 1 1
12 1 1 . 1 1 1 . . . .
13 1 1 . 1 . 1 . 1 1 .
14 1 1 . 1 1 1 . . . .
15 1 1 . 1 . 1 . 1 1 1
16 1 . . 1 1 . . 1 . .
17 1 1 . 1 1 1 . . 1 .
18 1 1 . 1 1 1 1 . . 1
19 . 1 . 1 1 1 1 . 1 .
20 1 . . 1 1 1 . 1 1 1
21 . . . 1 1 1 . 1 . .
22 . . . 1 1 1 . 1 1 .
23 1 . . 1 . . . . . 1
24 . 1 . 1 1 . . 1 . 1
25 1 1 . 1 1 . . . 1 1
26 1 1 . 1 1 . . 1 . .
27 1 . . 1 1 . . . 1 .
28 1 1 . 1 . . . 1 1 1
```

```
29 1 . . 1 1 1 . 1 . 1
30 1 . 1 1 1 . . 1 1 .
31 . . . 1 1 . . 1 1 .
32 1 1 1 1 1 . . 1 1 1
33 1 . . 1 1 . . 1 . 1
34 . . 1 1 . . . 1 1 .
35 1 1 1 1 1 . 1 1 . .
36 1 1 1 1 . 1 . 1 . .
37 1 1 . 1 . . . 1 . .
38 . . . 1 1 1 . 1 . .
39 1 1 . 1 1 . . 1 . 1
40 1 . . 1 . . 1 1 . 1
41 1 . . 1 1 1 1 1 . 1
42 1 1 1 1 . . 1 1 . .
43 1 . . 1 1 1 . 1 . .
44 1 . 1 1 . 1 . 1 . 1
45 . . . 1 . . 1 . . 1
46 . . . 1 1 . . . 1 .
47 1 1 . 1 . . 1 1 . .
48 1 . 1 1 1 . 1 1 . .
49 . . 1 1 1 1 . 1 . 1
50 . 1 . 1 1 . . 1 1 .
51 1 . 1 1 1 1 . . . .
52 1 1 1 1 1 1 . 1 . .
53 . 1 1 1 . 1 . 1 1 1
54 1 . . 1 1 . . 1 1 .
55 1 1 . 1 1 1 . 1 . .
56 1 . . 1 1 . . 1 1 .
57 1 1 . 1 1 . 1 . . 1
58 . 1 . 1 . 1 . . 1 1
59 1 1 1 1 . . 1 1 1 .
60 . 1 1 1 1 1 . . 1 1
61 1 1 1 1 1 1 . 1 . .
62 1 1 . 1 1 . . 1 1 .
63 . . . 1 . . . 1 1 1
64 1 . . 1 1 1 . 1 . .
65 1 . . 1 1 1 . 1 . .
66 1 . . 1 1 1 1 1 1 .
67 1 1 . 1 1 1 . 1 1 .
68 1 1 . 1 1 1 . 1 1 .
69 1 1 . 1 1 . 1 . . .
70 . . . 1 1 1 . 1 . .
71 1 . . 1 1 . 1 . . 1
72 1 . 1 1 1 1 . . 1 .
73 1 1 . 1 . 1 . 1 1 .
74 1 1 1 1 1 1 . 1 . .
75 . 1 . 1 1 1 . . 1 .
76 1 1 . 1 1 1 . 1 1 1
77 . . . 1 1 1 . . . .
78 1 1 1 1 1 1 . 1 1 .
79 1 . . 1 1 1 . 1 1 .
80 1 1 1 1 1 . 1 1 . 1
81 1 1 . 1 1 1 1 1 1 .
82 . . . 1 1 1 1 . . .
83 1 1 . 1 1 1 . 1 1 .
84 1 . . 1 1 . . 1 1 .
```

```
85 . . . 1 . 1 . 1 . .
86 1 . . 1 1 1 . 1 1 1
87 1 1 . 1 1 1 . 1 . .
88 . . . 1 . 1 . . . .
89 1 . . 1 . 1 . . 1 1
90 1 1 . 1 1 1 . 1 . 1
91 . . . 1 1 . . . 1 .
92 1 . . 1 1 1 . 1 1 .
93 1 . . 1 1 . . 1 1 .
94 1 . . 1 1 1 1 1 . .
95 1 . . 1 . 1 1 1 1 .
96 1 . 1 1 1 1 . . 1 .
97 1 1 . 1 1 . . . 1 .
98 1 . 1 1 1 1 1 1 . .
99 1 1 . 1 1 1 1 1 1 .
100 1 . 1 1 1 . . . 1 1
101 1 . 1 1 1 1 . . . .
102 1 . . 1 1 . 1 1 . .
103 1 1 . 1 1 1 . 1 . .
104 . . . 1 1 1 . 1 1 1
105 1 . 1 1 1 . . 1 . 1
106 1 1 1 1 1 1 1 1 1 1
107 1 1 1 1 . . . 1 . 1
108 1 . . 1 . 1 1 1 . .
109 . 1 . 1 1 . . 1 1 .
110 1 . . 1 . . . . . .
111 1 . . 1 1 1 . 1 1 .
112 1 1 . 1 1 1 . . . 1
113 1 1 . 1 1 . 1 1 1 .
114 1 1 . 1 1 . . . . .
115 1 1 . 1 1 . . 1 . .
116 . 1 . 1 1 1 1 1 . .
117 . 1 . 1 1 1 . . . .
118 . 1 1 1 1 . . 1 1 .
119 . . . 1 . . . 1 . .
120 1 1 . 1 . . . . 1 .
;
```

DJIA

```
data djia;
      input Year HighDate date7. High LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1968 03DEC68   985.21 21MAR68   825.13
1969 14MAY69   968.85 17DEC69   769.93
1970 29DEC70   842.00 06MAY70   631.16
1971 28APR71   950.82 23NOV71   797.97
1972 11DEC72  1036.27 26JAN72   889.15
1973 11JAN73  1051.70 05DEC73   788.31
1974 13MAR74   891.66 06DEC74   577.60
1975 15JUL75   881.81 02JAN75   632.04
1976 21SEP76  1014.79 02JAN76   858.71
```

```
1977 03JAN77   999.75 02NOV77   800.85
1978 08SEP78   907.74 28FEB78   742.12
1979 05OCT79   897.61 07NOV79   796.67
1980 20NOV80  1000.17 21APR80   759.13
1981 27APR81  1024.05 25SEP81   824.01
1982 27DEC82  1070.55 12AUG82   776.92
1983 29NOV83  1287.20 03JAN83  1027.04
1984 06JAN84  1286.64 24JUL84  1086.57
1985 16DEC85  1553.10 04JAN85  1184.96
1986 02DEC86  1955.57 22JAN86  1502.29
1987 25AUG87  2722.42 19OCT87  1738.74
1988 21OCT88  2183.50 20JAN88  1879.14
1989 09OCT89  2791.41 03JAN89  2144.64
1990 16JUL90  2999.75 11OCT90  2365.10
1991 31DEC91  3168.83 09JAN91  2470.30
1992 01JUN92  3413.21 09OCT92  3136.58
1993 29DEC93  3794.33 20JAN93  3241.95
1994 31JAN94  3978.36 04APR94  3593.35
1995 13DEC95  5216.47 30JAN95  3832.08
1996 27DEC96  6560.90 10JAN96  5032.94
1997 06AUG97  8259.30 11APR97  6391.69
1998 23NOV98  9374.27 31AUG98  7539.06
1999 31DEC99 11497.12 22JAN99  9120.67
2000 14JAN00 11722.98 07MAR00  9796.04
2001 21MAY01 11337.92 21SEP01  8235.81
2002 19MAR02 10635.25 09OCT02  7286.27
2003 31DEC03 10453.92 11MAR03  7524.06
2004 28DEC04 10854.54 25OCT04  9749.99
2005 04MAR05 10940.55 20APR05 10012.36
2006 27DEC06 12510.57 20JAN06 10667.39
2007 09OCT07 14164.53 05MAR07 12050.41
2008 02MAY08 13058.20 10OCT08  8451.19
;
```

EDUCATION

```
data education;
   input State $14. +1 Code $ DropoutRate Expenditures MathScore Region $;
   label dropoutrate='Dropout Percentage - 2008'
         expenditures='Expenditure Per Pupil - 2008'
           mathscore='8th Grade Math Exam - 2009';
   datalines;
Alabama        AL 22.3 3197 252 SE
Alaska         AK 35.8 7716 .   W
Arizona        AZ 31.2 3902 259 W
Arkansas       AR 11.5 3273 256 SE
California     CA 32.7 4121 256 W
Colorado       CO 24.7 4408 267 W
Connecticut    CT 16.8 6857 270 NE
Delaware       DE 28.5 5422 261 NE
Florida        FL 38.5 4563 255 SE
Georgia        GA 27.9 3852 258 SE
Hawaii         HI 18.3 4121 251 W
```

```
Idaho          ID 21.8 2838 272 W
Illinois       IL 21.5 4906 260 MW
Indiana        IN 13.8 4284 267 MW
Iowa           IA 13.6 4285 278 MW
Kansas         KS 17.9 4443 .   MW
Kentucky       KY 32.7 3347 256 SE
Louisiana      LA 43.1 3317 246 SE
Maine          ME 22.5 4744 .   NE
Maryland       MD 26.0 5758 260 NE
Massachusetts  MA 28.0 5979 .   NE
Michigan       MI 29.3 5116 264 MW
Minnesota      MN 11.4 4755 276 MW
Mississippi    MS 39.9 2874 .   SE
Missouri       MO 26.5 4263 .   MW
Montana        MT 15.0 4293 280 W
Nebraska       NE 13.9 4360 276 MW
Nevada         NV 28.1 3791 .   W
New Hampshire  NH 25.9 4807 273 NE
New Jersey     NE 20.4 7549 269 NE
New Mexico     NM 28.5 3473 256 W
New York       NY 35.0 .    261 NE
North Carolina NC 31.2 3874 250 SE
North Dakota   ND 12.1 3952 281 MW
Ohio           OH 24.4 4649 264 MW
;
```

EMPDATA

```
data empdata;
input IdNumber $ 1-4 LastName $ 8-18 FirstName $ 19-28
      City $ 29-41 State $ 42-43 /
      Gender $ 45 JobCode $ 49-51 Salary 55-60 @63 Birth date7.
      @73 Hired date7. HomePhone $ 83-95;
format birth hired date7.;
datalines;
1919   Adams      Gerald    Stamford     CT M   TA2   34376   15SEP70   07JUN05   203/781-1255
1653   Alexander  Susan     Bridgeport   CT F   ME2   35108   18OCT72   12AUG98   203/675-7715
1400   Apple      Troy      New York     NY M   ME1   29769   08NOV85   19OCT06   212/586-0808
1350   Arthur     Barbara   New York     NY F   FA3   32886   03SEP63   01AUG00   718/383-1549
1401   Avery      Jerry     Paterson     NJ M   TA3   38822   16DEC68   20NOV93   201/732-8787
1499   Barefoot   Joseph    Princeton    NJ M   ME3   43025   29APR62   10JUN95   201/812-5665
1101   Baucom     Walter    New York     NY M   SCP   18723   09JUN80   04OCT98   212/586-8060
1333   Blair      Justin    Stamford     CT M   PT2   88606   02APR79   13FEB03   203/781-1777
1402   Blalock    Ralph     New York     NY M   TA2   32615   20JAN71   05DEC98   718/384-2849
1479   Bostic     Marie     New York     NY F   TA3   38785   25DEC66   08OCT03   718/384-8816
1403   Bowden     Earl      Bridgeport   CT M   ME1   28072   31JAN79   24DEC99   203/675-3434
1739   Boyce      Jonathan  New York     NY M   PT1   66517   28DEC82   30JAN00   212/587-1247
1658   Bradley    Jeremy    New York     NY M   SCP   17943   11APR65   03MAR00   212/587-3622
1428   Brady      Christine Stamford     CT F   PT1   68767   07APR80   19NOV02   203/781-1212
1782   Brown      Jason     Stamford     CT M   ME2   35345   07DEC73   25FEB00   203/781-0019
1244   Bryant     Leonard   New York     NY M   ME2   36925   03SEP71   20JAN96   718/383-3334
1383   Burnette   Thomas    New York     NY M   BCK   25823   28JAN76   23OCT00   718/384-3569
1574   Cahill     Marshall  New York     NY M   FA2   28572   30APR74   23DEC97   718/383-2338
```

```
1789   Caraway    Davis     New York     NY M   SCP   18326   28JAN85   14APR04   212/587-9000
1404   Carter     Donald    New York     NY M   PT2   91376   27FEB71   04JAN98   718/384-2946
1437   Carter     Dorothy   Bridgeport   CT F   A3    33104   23SEP68   03SEP92   203/675-4117
1639   Carter     Karen     Stamford     CT F   A3    40260   29JUN65   31JAN92   203/781-8839
1269   Caston     Franklin  Stamford     CT M   NA1   41690   06MAY80   01DEC00   203/781-3335
1065   Chapman    Neil      New York     NY M   ME2   35090   29JAN72   10JAN95   718/384-5618
1876   Chin       Jack      New York     NY M   TA3   39675   23MAY66   30APR96   212/588-5634
1037   Chow       Jane      Stamford     CT F   TA1   28558   13APR82   16SEP04   203/781-8868
1129   Cook       Brenda    New York     NY F   ME2   34929   11DEC79   20AUG03   718/383-2313
1988   Cooper     Anthony   New York     NY M   FA3   32217   03DEC57   21SEP92   212/587-1228
1405   Davidson   Jason     Paterson     NJ M   SCP   18056   08MAR54   29JAN00   201/732-2323
1430   Dean       Sandra    Bridgeport   CT F   TA2   32925   03MAR70   30APR05   203/675-1647
1983   Dean       Sharon    New York     NY F   FA3   33419   03MAR50   30APR85   718/384-1647
1134   Delgado    Maria     Stamford     CT F   TA2   33462   08MAR77   24DEC04   203/781-1528
1118   Dennis     Roger     New York     NY M   PT3   111379  19JAN57   21DEC88   718/383-1122
1438   Donaldson  Karen     Stamford     CT F   TA3   39223   18MAR63   21NOV03   203/781-2229
1125   Dunlap     Donna     New York     NY F   FA2   28888   11NOV76   14DEC95   718/383-2094
1475   Eaton      Alicia    New York     NY F   FA2   27787   18DEC71   16JUL98   718/383-2828
1117   Edgerton   Joshua    New York     NY M   TA3   39771   08JUN56   16AUG00   212/588-1239
1935   Fernandez  Katrina   Bridgeport   CT F   NA2   51081   31MAR72   19OCT01   203/675-2962
1124   Fields     Diana     White Plains NY F   FA1   23177   13JUL82   04OCT01   914/455-2998
1422   Fletcher   Marie     Princeton    NJ F   FA1   22454   07JUN79   09APR99   201/812-0902
1616   Flowers    Annette   New York     NY F   TA2   34137   04MAR68   07JUN01   718/384-3329
1406   Foster     Gerald    Bridgeport   CT M   ME2   35185   11MAR69   20FEB95   203/675-6363
1120   Garcia     Jack      New York     NY M   ME1   28619   14SEP80   10OCT01   718/384-4930
1094   Gomez      Alan      Bridgeport   CT M   FA1   22268   05APR78   20APR99   203/675-7181
1389   Gordon     Levi      New York     NY M   BCK   25028   18JUL67   21AUG03   718/384-9326
1905   Graham     Alvin     New York     NY M   PT1   65111   19APR80   01JUN00   212/586-8815
1407   Grant      Daniel    Mt. Vernon   NY M   PT1   68096   26MAR77   21MAR98   914/468-1616
1114   Green      Janice    New York     NY F   TA2   32928   21SEP77   30JUN06   212/588-1092
;
```

ENERGY

```
data energy;
   length State $2;
   input Region Division state $ Type Expenditures;
   datalines;
1 1 ME 1 708
1 1 ME 2 379
1 1 NH 1 597
1 1 NH 2 301
1 1 VT 1 353
1 1 VT 2 188
1 1 MA 1 3264
1 1 MA 2 2498
1 1 RI 1 531
1 1 RI 2 358
1 1 CT 1 2024
1 1 CT 2 1405
1 2 NY 1 8786
1 2 NY 2 7825
1 2 NJ 1 4115
```

```
1 2 NJ 2 3558
1 2 PA 1 6478
1 2 PA 2 3695
4 3 MT 1 322
4 3 MT 2 232
4 3 ID 1 392
4 3 ID 2 298
4 3 WY 1 194
4 3 WY 2 184
4 3 CO 1 1215
4 3 CO 2 1173
4 3 NM 1 545
4 3 NM 2 578
4 3 AZ 1 1694
4 3 AZ 2 1448
4 3 UT 1 621
4 3 UT 2 438
4 3 NV 1 493
4 3 NV 2 378
4 4 WA 1 1680
4 4 WA 2 1122
4 4 OR 1 1014
4 4 OR 2 756
4 4 CA 1 10643
4 4 CA 2 10114
4 4 AK 1 349
4 4 AK 2 329
4 4 HI 1 273
4 4 HI 2 298
;
```

EXP ライブラリ

EXP.RESULTS

次のセクションでは、EXP ライブラリの生データと DATA ステップを示します。

```
options ps=40 ls=64 nodate pageno=1;

LIBNAME exp 'library-name';

data exp.results;
   set exp.wght(firstobs=1 obs=11 keep=id treat initwt wt3mos
       age);
   if age>100 then delete;
run;
proc print data=exp.results noobs;
    title 'The RESULTS Data Set';
run;
proc datasets library=exp;

data exp.results;
```

```
    input id  treat $  initwt  wt3mos age;
    datalines;
 1    Other      166.28    146.98     35
 2    Other      214.42    210.22     54
 3    Other      172.46    159.42     33
 5    Other      175.41    160.66     37
 6    Other      173.13    169.40     20
 7    Other      181.25    170.94     30
10    Other      239.83    214.48     48
11    Other      175.32    162.66     51
12    Other      227.01    211.06     29
13    Other      274.82    251.82     31
;
run;
```

EXP.SUR

```
data exp.sur;
    input id  treat $  initwt  wt3mos  wt6mos  age;
    datalines;
14    surgery    203.60    169.78    143.88     38
17    surgery    171.52    150.33    123.18     42
18    surgery    207.46    155.22       .       41
;
run;
```

EXPREV

```
ods html close;
data exprev;
input Country $ 1-24 Emp_ID $ 25-32 Order_Date $  Ship_Date $  Sale_Type $ 67-75 Quantity Price Cost;

datalines;
Antarctica              99999999     1/1/12      1/7/12           Internet    2     92.60      20.70
Puerto Rico             99999999     1/1/12      1/5/12           Catalog    14     51.20      12.10
Virgin Islands (U.S.)   99999999     1/1/12      1/4/12           In Store   25     31.10      15.65
Aruba                   99999999     1/1/12      1/4/12           Catalog    30    123.70      59.00
Bahamas                 99999999     1/1/12      1/4/12           Catalog     8    113.40      28.45
Bermuda                 99999999     1/1/12      1/4/12           Catalog     7     41.00       9.25
Belize                  120458       1/2/12      1/2/12           In Store    2    146.40      36.70
British Virgin Islands  99999999     1/2/12      1/5/12           Catalog    11     40.20      20.20
Canada                  99999999     1/2/12      1/5/12           Catalog   100     11.80       5.00
Cayman Islands          120454       1/2/12      1/2/12           In Store   20     71.00      32.30
Costa Rica              99999999     1/2/12      1/6/12           Internet   31     53.00      26.60
Cuba                    121044       1/2/12      1/2/12           Internet   12     42.40      19.35
Dominican Republic      121040       1/2/12      1/2/12           Internet   13     48.00      23.95
El Salvador             99999999     1/2/12      1/6/12           Catalog    21    266.40      66.70
Guatemala               120931       1/2/12      1/2/12           In Store   13    144.40      65.70
Haiti                   121059       1/2/12      1/2/12           Internet    5     47.90      23.45
Honduras                120455       1/2/12      1/2/12           Internet   20     66.40      30.25
Jamaica                 99999999     1/2/12      1/4/12           In Store   23    169.80      38.70
Mexico                  120127       1/2/12      1/2/12           In Store   30    211.80      33.65
```

```
Montserrat              120127      1/2/12      1/2/12          In Store   19    184.20     36.90
Nicaragua               120932      1/2/12      1/2/12          Internet   16    122.00     28.75
Panama                  99999999    1/2/12      1/6/12          Internet   20    88.20      38.40
Saint Kitts/Nevis       99999999    1/2/12      1/6/12          Internet   20    41.40      18.00
St. Helena              120360      1/2/12      1/2/12          Internet   19    94.70      47.45
St. Pierre/Miquelon     120842      1/2/12      1/16/12         Internet   16    103.80     47.25
Turks/Caicos Islands    120372      1/2/12      1/2/12          Internet   10    57.70      28.95
United States           120372      1/2/12      1/2/12          Internet   20    88.20      38.40
Anguilla                99999999    1/2/12      1/6/12          In Store   15    233.50     22.25
Antigua/Barbuda         120458      1/2/12      1/2/12          In Store   31    99.60      45.35
Argentina               99999999    1/2/12      1/6/12          In Store   42    408.80     87.15
Barbados                99999999    1/2/12      1/6/12          In Store   26    94.80      42.60
Bolivia                 120127      1/2/12      1/2/12          In Store   26    66.00      16.60
Brazil                  120127      1/2/12      1/2/12          Catalog    12    73.40      18.45
Chile                   120447      1/2/12      1/2/12          In Store   20    19.10      8.75
Colombia                121059      1/2/12      1/2/12          Internet   28    361.40     90.45
Dominica                121043      1/2/12      1/2/12          Internet   35    121.30     57.80
Ecuador                 121042      1/2/12      1/2/12          In Store   11    100.90     50.55
Falkland Islands        120932      1/2/12      1/2/12          In Store   15    61.40      30.80
French Guiana           120935      1/2/12      1/2/12          Catalog    15    96.40      43.85
Grenada                 120931      1/2/12      1/2/12          Catalog    19    56.30      25.05
Guadeloupe              120445      1/2/12      1/2/12          Internet   21    231.60     48.70
Guyana                  120455      1/2/12      1/2/12          In Store   25    132.80     30.25
Martinique              120841      1/2/12      1/3/12          In Store   16    56.30      31.05
Netherlands Antilles    99999999    1/2/12      1/6/12          In Store   31    41.80      19.45
Paraguay                120603      1/2/12      1/2/12          Catalog    17    117.60     58.90
Peru                    120845      1/2/12      1/2/12          Catalog    12    93.80      41.75
St. Lucia               120845      1/2/12      1/2/12          Internet   19    64.30      28.65
Suriname                120538      1/3/12      1/3/12          Internet   22    110.80     29.35
;
run;
```

GROC

```
data groc;
   input Region $9. Manager $ Department $ Sales;
   datalines;
Southeast    Hayes       Paper       250
Southeast    Hayes       Produce     100
Southeast    Hayes       Canned      120
Southeast    Hayes       Meat         80
Southeast    Michaels    Paper        40
Southeast    Michaels    Produce     300
Southeast    Michaels    Canned      220
Southeast    Michaels    Meat         70
Northwest    Jeffreys    Paper        60
Northwest    Jeffreys    Produce     600
Northwest    Jeffreys    Canned      420
Northwest    Jeffreys    Meat         30
Northwest    Duncan      Paper        45
Northwest    Duncan      Produce     250
Northwest    Duncan      Canned      230
Northwest    Duncan      Meat         73
```

```
Northwest    Aikmann     Paper        45
Northwest    Aikmann     Produce     205
Northwest    Aikmann     Canned      420
Northwest    Aikmann     Meat         76
Southwest    Royster     Paper        53
Southwest    Royster     Produce     130
Southwest    Royster     Canned      120
Southwest    Royster     Meat         50
Southwest    Patel       Paper        40
Southwest    Patel       Produce     350
Southwest    Patel       Canned      225
Southwest    Patel       Meat         80
Northeast    Rice        Paper        90
Northeast    Rice        Produce      90
Northeast    Rice        Canned      420
Northeast    Rice        Meat         86
Northeast    Fuller      Paper       200
Northeast    Fuller      Produce     300
Northeast    Fuller      Canned      420
Northeast    Fuller      Meat        125
;
```

MATCH_11

```
data match_11;
   input Pair Low Age Lwt Race Smoke Ptd Ht UI @@;
   select(race);
      when (1) do;
         race1=0;
         race2=0;
      end;
      when (2) do;
         race1=1;
         race2=0;
      end;
      when (3) do;
         race1=0;
         race2=1;
      end;
   end;
   datalines;
1  0 14 135 1 0 0 0 0     1  1 14 101 3 1 1 0 0
2  0 15  98 2 0 0 0 0     2  1 15 115 3 0 0 0 1
3  0 16  95 3 0 0 0 0     3  1 16 130 3 0 0 0 0
4  0 17 103 3 0 0 0 0     4  1 17 130 3 1 1 0 1
5  0 17 122 1 1 0 0 0     5  1 17 110 1 1 0 0 0
6  0 17 113 2 0 0 0 0     6  1 17 120 1 1 0 0 0
7  0 17 113 2 0 0 0 0     7  1 17 120 2 0 0 0 0
8  0 17 119 3 0 0 0 0     8  1 17 142 2 0 0 1 0
9  0 18 100 1 1 0 0 0     9  1 18 148 3 0 0 0 0
10 0 18  90 1 1 0 0 1     10 1 18 110 2 1 1 0 0
11 0 19 150 3 0 0 0 0     11 1 19  91 1 1 1 0 1
12 0 19 115 3 0 0 0 0     12 1 19 102 1 0 0 0 0
```

```
13 0 19 235 1 1 0 1 0     13 1 19 112 1 1 0 0 1
14 0 20 120 3 0 0 0 1     14 1 20 150 1 1 0 0 0
15 0 20 103 3 0 0 0 0     15 1 20 125 3 0 0 0 1
16 0 20 169 3 0 1 0 1     16 1 20 120 2 1 0 0 0
17 0 20 141 1 0 1 0 1     17 1 20  80 3 1 0 0 1
18 0 20 121 2 1 0 0 0     18 1 20 109 3 0 0 0 0
19 0 20 127 3 0 0 0 0     19 1 20 121 1 1 1 0 1
20 0 20 120 3 0 0 0 0     20 1 20 122 2 1 0 0 0
21 0 20 158 1 0 0 0 0     21 1 20 105 3 0 0 0 0
22 0 21 108 1 1 0 0 1     22 1 21 165 1 1 0 1 0
23 0 21 124 3 0 0 0 0     23 1 21 200 2 0 0 0 0
24 0 21 185 2 1 0 0 0     24 1 21 103 3 0 0 0 0
25 0 21 160 1 0 0 0 0     25 1 21 100 3 0 1 0 0
26 0 21 115 1 0 0 0 0     26 1 21 130 1 1 0 1 0
27 0 22  95 3 0 0 1 0     27 1 22 130 1 1 0 0 0
28 0 22 158 2 0 1 0 0     28 1 22 130 1 1 1 0 1
29 0 23 130 2 0 0 0 0     29 1 23  97 3 0 0 0 1
30 0 23 128 3 0 0 0 0     30 1 23 187 2 1 0 0 0
31 0 23 119 3 0 0 0 0     31 1 23 120 3 0 0 0 0
32 0 23 115 3 1 0 0 0     32 1 23 110 1 1 1 0 0
33 0 23 190 1 0 0 0 0     33 1 23  94 3 1 0 0 0
34 0 24  90 1 1 1 0 0     34 1 24 128 2 0 1 0 0
35 0 24 115 1 0 0 0 0     35 1 24 132 3 0 0 1 0
36 0 24 110 3 0 0 0 0     36 1 24 155 1 1 1 0 0
37 0 24 115 3 0 0 0 0     37 1 24 138 1 0 0 0 0
38 0 24 110 3 0 1 0 0     38 1 24 105 2 1 0 0 0
39 0 25 118 1 1 0 0 0     39 1 25 105 3 0 1 1 0
40 0 25 120 3 0 0 0 1     40 1 25  85 3 0 0 0 1
41 0 25 155 1 0 0 0 0     41 1 25 115 3 0 0 0 0
42 0 25 125 2 0 0 0 0     42 1 25  92 1 1 0 0 0
43 0 25 140 1 0 0 0 0     43 1 25  89 3 0 1 0 0
44 0 25 241 2 0 0 1 0     44 1 25 105 3 0 1 0 0
45 0 26 113 1 1 0 0 0     45 1 26 117 1 1 1 0 0
46 0 26 168 2 1 0 0 0     46 1 26  96 3 0 0 0 0
47 0 26 133 3 1 1 0 0     47 1 26 154 3 0 1 1 0
48 0 26 160 3 0 0 0 0     48 1 26 190 1 1 0 0 0
49 0 27 124 1 1 0 0 0     49 1 27 130 2 0 0 0 1
50 0 28 120 3 0 0 0 0     50 1 28 120 3 1 1 0 1
51 0 28 130 3 0 0 0 0     51 1 28  95 1 1 0 0 0
52 0 29 135 1 0 0 0 0     52 1 29 130 1 0 0 0 1
53 0 30  95 1 1 0 0 0     53 1 30 142 1 1 1 0 0
54 0 31 215 1 1 0 0 0     54 1 31 102 1 1 1 0 0
55 0 32 121 3 0 0 0 0     55 1 32 105 1 1 0 0 0
56 0 34 170 1 0 1 0 0     56 1 34 187 2 1 0 1 0
;
```

PROCLIB.DELAY

```
data proclib.delay;
   input flight $3. +5 date date7. +2 orig $3. +3 dest $3. +3
         delaycat $15. +2 destype $15. +8 delay;
   informat date date7.;
   format date date7.;
```

```
   datalines;
114     01MAR08  LGA   LAX   1-10 Minutes     Domestic              8
202     01MAR08  LGA   ORD   No Delay         Domestic             -5
219     01MAR08  LGA   LON   11+ Minutes      International        18
622     01MAR08  LGA   FRA   No Delay         International        -5
132     01MAR08  LGA   YYZ   11+ Minutes      International        14
271     01MAR08  LGA   PAR   1-10 Minutes     International         5
302     01MAR08  LGA   WAS   No Delay         Domestic             -2
114     02MAR08  LGA   LAX   No Delay         Domestic              0
202     02MAR08  LGA   ORD   1-10 Minutes     Domestic              5
219     02MAR08  LGA   LON   11+ Minutes      International        18
622     02MAR08  LGA   FRA   No Delay         International         0
132     02MAR08  LGA   YYZ   1-10 Minutes     International         5
271     02MAR08  LGA   PAR   1-10 Minutes     International         4
302     02MAR08  LGA   WAS   No Delay         Domestic              0
114     03MAR08  LGA   LAX   No Delay         Domestic             -1
202     03MAR08  LGA   ORD   No Delay         Domestic             -1
219     03MAR08  LGA   LON   1-10 Minutes     International         4
622     03MAR08  LGA   FRA   No Delay         International        -2
132     03MAR08  LGA   YYZ   1-10 Minutes     International         6
271     03MAR08  LGA   PAR   1-10 Minutes     International         2
302     03MAR08  LGA   WAS   1-10 Minutes     Domestic              5
114     04MAR08  LGA   LAX   11+ Minutes      Domestic             15
202     04MAR08  LGA   ORD   No Delay         Domestic             -5
219     04MAR08  LGA   LON   1-10 Minutes     International         3
622     04MAR08  LGA   FRA   11+ Minutes      International        30
132     04MAR08  LGA   YYZ   No Delay         International        -5
271     04MAR08  LGA   PAR   1-10 Minutes     International         5
302     04MAR08  LGA   WAS   1-10 Minutes     Domestic              7
114     05MAR08  LGA   LAX   No Delay         Domestic             -2
202     05MAR08  LGA   ORD   1-10 Minutes     Domestic              2
219     05MAR08  LGA   LON   1-10 Minutes     International         3
622     05MAR08  LGA   FRA   No Delay         International        -6
132     05MAR08  LGA   YYZ   1-10 Minutes     International         3
271     05MAR08  LGA   PAR   1-10 Minutes     International         5
114     06MAR08  LGA   LAX   No Delay         Domestic             -1
202     06MAR08  LGA   ORD   No Delay         Domestic             -3
219     06MAR08  LGA   LON   11+ Minutes      International        27
132     06MAR08  LGA   YYZ   1-10 Minutes     International         7
302     06MAR08  LGA   WAS   1-10 Minutes     Domestic              1
114     07MAR08  LGA   LAX   No Delay         Domestic             -1
202     07MAR08  LGA   ORD   No Delay         Domestic             -2
219     07MAR08  LGA   LON   11+ Minutes      International        15
622     07MAR08  LGA   FRA   11+ Minutes      International        21
132     07MAR08  LGA   YYZ   No Delay         International        -2
271     07MAR08  LGA   PAR   1-10 Minutes     International         4
302     07MAR08  LGA   WAS   No Delay         Domestic              0
;
```

PROCLIB.EMP95

```
data proclib.emp95;
   input #1 idnum $4. @6 name $15.
```

```
         #2 address $42.
         #3 salary 6.;
   datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
2776 Robert Jones
12988 Wellington Farms Ave. Cary NC 27512
29025
8699 Jerry Capalleti
222 West L St. Oxford NC 27587
39985
2100 Lanny Engles
293 Manning Pl. Raleigh NC 27606
30998
9857 Kathy Krupski
1000 Taft Ave. Morrisville NC 27508
38756
0987 Dolly Lunford
2344 Persimmons Branch  Apex NC 27505
44010
3286 Hoa Nguyen
2818 Long St. Cary NC 27513
87734
6579 Bryan Samosky
3887 Charles Ave. Garner NC 27508
50234
3888 Kim Siu
5662 Magnolia Blvd Southeast Cary NC 27513
77558
;
```

PROCLIB.EMP96

```
data proclib.emp96;
   input #1 idnum $4. @6 name $15.
         #2 address $42.
         #3 salary 6.;
   datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
2776 Robert Jones
12988 Wellington Farms Ave. Cary NC 27511
29025
8699 Jerry Capalleti
222 West L St. Oxford NC 27587
```

```
39985
3278 Mary Cravens
211 N. Cypress St. Cary NC 27512
35362
2100 Lanny Engles
293 Manning Pl. Raleigh NC 27606
30998
9857 Kathy Krupski
100 Taft Ave. Morrisville NC 27508
40456
0987 Dolly Lunford
2344 Persimmons Branch Trail Apex NC 27505
45110
3286 Hoa Nguyen
2818 Long St. Cary NC 27513
89834
6579 Bryan Samosky
3887 Charles Ave. Garner NC 27508
50234
3888 Kim Siu
5662 Magnolia Blvd Southwest Cary NC 27513
79958
6544 Roger Monday
3004 Crepe Myrtle Court Raleigh NC 27604
47007
;
```

PROCLIB.INTERNAT

```
data proclib.internat;
   input flight $3.  +5 date date7. +2 dest $3. +8 boarded;
   informat date date7.;
   format date date7.;
   datalines;
219     01MAR08  LON        198
622     01MAR08  FRA        207
132     01MAR08  YYZ        115
271     01MAR08  PAR        138
219     02MAR08  LON        147
622     02MAR08  FRA        176
132     02MAR08  YYZ        106
271     02MAR08  PAR        172
219     03MAR08  LON        197
622     03MAR08  FRA        180
132     03MAR08  YYZ         75
271     03MAR08  PAR        147
219     04MAR08  LON        232
622     04MAR08  FRA        137
132     04MAR08  YYZ        117
271     04MAR08  PAR        146
219     05MAR08  LON        160
622     05MAR08  FRA        185
132     05MAR08  YYZ        157
```

```
271     05MAR08  PAR        177
219     06MAR08  LON        163
132     06MAR08  YYZ        150
219     07MAR08  LON        241
622     07MAR08  FRA        210
132     07MAR08  YYZ        164
271     07MAR08  PAR        155
;
```

PROCLIB.LAKES

```
data proclib.lakes;
    input region $ 1-2 lake $ 5-13 pol_a1 pol_a2 pol_b1-pol_b4;
    datalines;
NE  Carr       0.24     0.99     0.95     0.36     0.44     0.67
NE  Duraleigh  0.34     0.01     0.48     0.58     0.12     0.56
NE  Charlie    0.40     0.48     0.29     0.56     0.52     0.95
NE  Farmer     0.60     0.65     0.25     0.20     0.30     0.64
NW  Canyon     0.63     0.44     0.20     0.98     0.19     0.01
NW  Morris     0.85     0.95     0.80     0.67     0.32     0.81
NW  Golf       0.69     0.37     0.08     0.72     0.71     0.32
NW  Falls      0.01     0.02     0.59     0.58     0.67     0.02
SE  Pleasant   0.16     0.96     0.71     0.35     0.35     0.48
SE  Juliette   0.82     0.35     0.09     0.03     0.59     0.90
SE  Massey     1.01     0.77     0.45     0.32     0.55     0.66
SE  Delta      0.84     1.05     0.90     0.09     0.64     0.03
SW  Alumni     0.45     0.32     0.45     0.44     0.55     0.12
SW  New Dam    0.80     0.70     0.31     0.98     1.00     0.22
SW  Border     0.51     0.04     0.55     0.35     0.45     0.78
SW  Red        0.22     0.09     0.02     0.10     0.32     0.01
;
```

PROCLIB.MARCH

```
data proclib.march;
   input flight $3. +5 date date7. +3 depart time5. +2 orig $3.
         +3 dest $3.  +7 miles +6 boarded +6 capacity;
   format date date7. depart time5.;
   informat date date7. depart time5.;
   datalines;
114     01MAR08    7:10  LGA   LAX       2475       172       210
202     01MAR08   10:43  LGA   ORD        740       151       210
219     01MAR08    9:31  LGA   LON       3442       198       250
622     01MAR08   12:19  LGA   FRA       3857       207       250
132     01MAR08   15:35  LGA   YYZ        366       115       178
271     01MAR08   13:17  LGA   PAR       3635       138       250
302     01MAR08   20:22  LGA   WAS        229       105       180
114     02MAR08    7:10  LGA   LAX       2475       119       210
202     02MAR08   10:43  LGA   ORD        740       120       210
219     02MAR08    9:31  LGA   LON       3442       147       250
```

```
622     02MAR08   12:19  LGA   FRA       3857       176       250
132     02MAR08   15:35  LGA   YYZ        366       106       178
302     02MAR08   20:22  LGA   WAS        229        78       180
271     02MAR08   13:17  LGA   PAR       3635       104       250
114     03MAR08    7:10  LGA   LAX       2475       197       210
202     03MAR08   10:43  LGA   ORD        740       118       210
219     03MAR08    9:31  LGA   LON       3442       197       250
622     03MAR08   12:19  LGA   FRA       3857       180       250
132     03MAR08   15:35  LGA   YYZ        366        75       178
271     03MAR08   13:17  LGA   PAR       3635       147       250
302     03MAR08   20:22  LGA   WAS        229       123       180
114     04MAR08    7:10  LGA   LAX       2475       178       210
202     04MAR08   10:43  LGA   ORD        740       148       210
219     04MAR08    9:31  LGA   LON       3442       232       250
622     04MAR08   12:19  LGA   FRA       3857       137       250
132     04MAR08   15:35  LGA   YYZ        366       117       178
271     04MAR08   13:17  LGA   PAR       3635       146       250
302     04MAR08   20:22  LGA   WAS        229       115       180
114     05MAR08    7:10  LGA   LAX       2475       117       210
202     05MAR08   10:43  LGA   ORD        740       104       210
219     05MAR08    9:31  LGA   LON       3442       160       250
622     05MAR08   12:19  LGA   FRA       3857       185       250
132     05MAR08   15:35  LGA   YYZ        366       157       178
271     05MAR08   13:17  LGA   PAR       3635       177       250
114     06MAR08    7:10  LGA   LAX       2475       128       210
202     06MAR08   10:43  LGA   ORD        740       115       210
219     06MAR08    9:31  LGA   LON       3442       163       250
132     06MAR08   15:35  LGA   YYZ        366       150       178
302     06MAR08   20:22  LGA   WAS        229        66       180
114     07MAR08    7:10  LGA   LAX       2475       160       210
202     07MAR08   10:43  LGA   ORD        740       175       210
219     07MAR08    9:31  LGA   LON       3442       241       250
622     07MAR08   12:19  LGA   FRA       3857       210       250
132     07MAR08   15:35  LGA   YYZ        366       164       178
271     07MAR08   13:17  LGA   PAR       3635       155       250
302     07MAR08   20:22  LGA   WAS        229       135       180
;
```

PROCLIB.PAYLIST2

```
proc sql;
   create table proclib.paylist2
       (IdNum char(4),
        Gender char(1),
        Jobcode char(3),
        Salary num,
        Birth num informat=date7.
                  format=date7.,
        Hired num informat=date7.
                  format=date7.);

insert into proclib.paylist2
values('1919','M','TA2',34376,'12SEP66'd,'04JUN87'd)
```

```
values('1653','F','ME2',31896,'15OCT64'd,'09AUG92'd)
values('1350','F','FA3',36886,'31AUG55'd,'29JUL91'd)
values('1401','M','TA3',38822,'13DEC55'd,'17NOV93'd)
values('1499','M','ME1',23025,'26APR74'd,'07JUN92'd);

title 'PROCLIB.PAYLIST2 Table';
select * from proclib.paylist2;
```

PROCLIB.PAYROLL

このデータセット(テーブル)は、“Updating Data in a PROC SQL Table” (*SAS SQL Procedure User's Guide*)で更新され、その更新データはその後の例で使用されます。

```
data proclib.payroll;
   input IdNumber $4. +3 Gender $1. +4 Jobcode $3. +9 Salary 5.
         +2 Birth date7. +2 Hired date7.;
   informat birth date7. hired date7.;
   format birth date7. hired date7.;
   datalines;
1919   M    TA2         34376  12SEP60  04JUN87
1653   F    ME2         35108  15OCT64  09AUG90
1400   M    ME1         29769  05NOV67  16OCT90
1350   F    FA3         32886  31AUG65  29JUL90
1401   M    TA3         38822  13DEC50  17NOV85
1499   M    ME3         43025  26APR54  07JUN80
1101   M    SCP         18723  06JUN62  01OCT90
1333   M    PT2         88606  30MAR61  10FEB81
1402   M    TA2         32615  17JAN63  02DEC90
1479   F    TA3         38785  22DEC68  05OCT89
1403   M    ME1         28072  28JAN69  21DEC91
1739   M    PT1         66517  25DEC64  27JAN91
1658   M    SCP         17943  08APR67  29FEB92
1428   F    PT1         68767  04APR60  16NOV91
1782   M    ME2         35345  04DEC70  22FEB92
1244   M    ME2         36925  31AUG63  17JAN88
1383   M    BCK         25823  25JAN68  20OCT92
1574   M    FA2         28572  27APR60  20DEC92
1789   M    SCP         18326  25JAN57  11APR78
1404   M    PT2         91376  24FEB53  01JAN80
1437   F    FA3         33104  20SEP60  31AUG84
1639   F    TA3         40260  26JUN57  28JAN84
1269   M    NA1         41690  03MAY72  28NOV92
1065   M    ME2         35090  26JAN44  07JAN87
1876   M    TA3         39675  20MAY58  27APR85
1037   F    TA1         28558  10APR64  13SEP92
1129   F    ME2         34929  08DEC61  17AUG91
1988   M    FA3         32217  30NOV59  18SEP84
1405   M    SCP         18056  05MAR66  26JAN92
1430   F    TA2         32925  28FEB62  27APR87
1983   F    FA3         33419  28FEB62  27APR87
1134   F    TA2         33462  05MAR69  21DEC88
1118   M    PT3        111379  16JAN44  18DEC80
1438   F    TA3         39223  15MAR65  18NOV87
```

1125	F	FA2	28888	08NOV68	11DEC87
1475	F	FA2	27787	15DEC61	13JUL90
1117	M	TA3	39771	05JUN63	13AUG92
1935	F	NA2	51081	28MAR54	16OCT81
1124	F	FA1	23177	10JUL58	01OCT90
1422	F	FA1	22454	04JUN64	06APR91
1616	F	TA2	34137	01MAR70	04JUN93
1406	M	ME2	35185	08MAR61	17FEB87
1120	M	ME1	28619	11SEP72	07OCT93
1094	M	FA1	22268	02APR70	17APR91
1389	M	BCK	25028	15JUL59	18AUG90
1905	M	PT1	65111	16APR72	29MAY92
1407	M	PT1	68096	23MAR69	18MAR90
1114	F	TA2	32928	18SEP69	27JUN87
1410	M	PT2	84685	03MAY67	07NOV86
1439	F	PT1	70736	06MAR64	10SEP90
1409	M	ME3	41551	19APR50	22OCT81
1408	M	TA2	34138	29MAR60	14OCT87
1121	M	ME1	29112	26SEP71	07DEC91
1991	F	TA1	27645	07MAY72	12DEC92
1102	M	TA2	34542	01OCT59	15APR91
1356	M	ME2	36869	26SEP57	22FEB83
1545	M	PT1	66130	12AUG59	29MAY90
1292	F	ME2	36691	28OCT64	02JUL89
1440	F	ME2	35757	27SEP62	09APR91
1368	M	FA2	27808	11JUN61	03NOV84
1369	M	TA2	33705	28DEC61	13MAR87
1411	M	FA2	27265	27MAY61	01DEC89
1113	F	FA1	22367	15JAN68	17OCT91
1704	M	BCK	25465	30AUG66	28JUN87
1900	M	ME2	35105	25MAY62	27OCT87
1126	F	TA3	40899	28MAY63	21NOV80
1677	M	BCK	26007	05NOV63	27MAR89
1441	F	FA2	27158	19NOV69	23MAR91
1421	M	TA2	33155	08JAN59	28FEB90
1119	M	TA1	26924	20JUN62	06SEP88
1834	M	BCK	26896	08FEB72	02JUL92
1777	M	PT3	109630	23SEP51	21JUN81
1663	M	BCK	26452	11JAN67	11AUG91
1106	M	PT2	89632	06NOV57	16AUG84
1103	F	FA1	23738	16FEB68	23JUL92
1477	M	FA2	28566	21MAR64	07MAR88
1476	F	TA2	34803	30MAY66	17MAR87
1379	M	ME3	42264	08AUG61	10JUN84
1104	M	SCP	17946	25APR63	10JUN91
1009	M	TA1	28880	02MAR59	26MAR92
1412	M	ME1	27799	18JUN56	05DEC91
1115	F	FA3	32699	22AUG60	29FEB80
1128	F	TA2	32777	23MAY65	20OCT90
1442	F	PT2	84536	05SEP66	12APR88
1417	M	NA2	52270	27JUN64	07MAR89
1478	M	PT2	84203	09AUG59	24OCT90
1673	M	BCK	25477	27FEB70	15JUL91
1839	F	NA1	43433	29NOV70	03JUL93
1347	M	TA3	40079	21SEP67	06SEP84
1423	F	ME2	35773	14MAY68	19AUG90

```
1200   F    ME1         27816  10JAN71  14AUG92
1970   F    FA1         22615  25SEP64  12MAR91
1521   M    ME3         41526  12APR63  13JUL88
1354   F    SCP         18335  29MAY71  16JUN92
1424   F    FA2         28978  04AUG69  11DEC89
1132   F    FA1         22413  30MAY72  22OCT93
1845   M    BCK         25996  20NOV59  22MAR80
1556   M    PT1         71349  22JUN64  11DEC91
1413   M    FA2         27435  16SEP65  02JAN90
1123   F    TA1         28407  31OCT72  05DEC92
1907   M    TA2         33329  15NOV60  06JUL87
1436   F    TA2         34475  11JUN64  12MAR87
1385   M    ME3         43900  16JAN62  01APR86
1432   F    ME2         35327  03NOV61  10FEB85
1111   M    NA1         40586  14JUL73  31OCT92
1116   F    FA1         22862  28SEP69  21MAR91
1352   M    NA2         53798  02DEC60  16OCT86
1555   F    FA2         27499  16MAR68  04JUL92
1038   F    TA1         26533  09NOV69  23NOV91
1420   M    ME3         43071  19FEB65  22JUL87
1561   M    TA2         34514  30NOV63  07OCT87
1434   F    FA2         28622  11JUL62  28OCT90
1414   M    FA1         23644  24MAR72  12APR92
1112   M    TA1         26905  29NOV64  07DEC92
1390   M    FA2         27761  19FEB65  23JUN91
1332   M    NA1         42178  17SEP70  04JUN91
1890   M    PT2         91908  20JUL51  25NOV79
1429   F    TA1         27939  28FEB60  07AUG92
1107   M    PT2         89977  09JUN54  10FEB79
1908   F    TA2         32995  10DEC69  23APR90
1830   F    PT2         84471  27MAY57  29JAN83
1882   M    ME3         41538  10JUL57  21NOV78
1050   M    ME2         35167  14JUL63  24AUG86
1425   F    FA1         23979  28DEC71  28FEB93
1928   M    PT2         89858  16SEP54  13JUL90
1480   F    TA3         39583  03SEP57  25MAR81
1100   M    BCK         25004  01DEC60  07MAY88
1995   F    ME1         28810  24AUG73  19SEP93
1135   F    FA2         27321  20SEP60  31MAR90
1415   M    FA2         28278  09MAR58  12FEB88
1076   M    PT1         66558  14OCT55  03OCT91
1426   F    TA2         32991  05DEC66  25JUN90
1564   F    SCP         18833  12APR62  01JUL92
1221   F    FA2         27896  22SEP67  04OCT91
1133   M    TA1         27701  13JUL66  12FEB92
1435   F    TA3         38808  12MAY59  08FEB80
1418   M    ME1         28005  29MAR57  06JAN92
1017   M    TA3         40858  28DEC57  16OCT81
1443   F    NA1         42274  17NOV68  29AUG91
1131   F    TA2         32575  26DEC71  19APR91
1427   F    TA2         34046  31OCT70  30JAN90
1036   F    TA3         39392  19MAY65  23OCT84
1130   F    FA1         23916  16MAY71  05JUN92
1127   F    TA2         33011  09NOV64  07DEC86
1433   F    FA3         32982  08JUL66  17JAN87
1431   F    FA3         33230  09JUN64  05APR88
```

```
1122   F    FA2         27956  01MAY63  27NOV88
1105   M    ME2         34805  01MAR62  13AUG90
;
```

PROCLIB.PAYROLL2

```
data proclib.payroll2;
   input idnum $4. +3 gender $1. +4 jobcode $3. +9 salary 5.
         +2 birth date7. +2 hired date7.;
   informat birth date7. hired date7.;
   format birth date7. hired date7.;
   datalines;
1639   F    TA3         42260  26JUN57  28JAN84
1065   M    ME3         38090  26JAN44  07JAN87
1561   M    TA3         36514  30NOV63  07OCT87
1221   F    FA3         29896  22SEP67  04OCT91
1447   F    FA1         22123  07AUG72  29OCT92
1998   M    SCP         23100  10SEP70  02NOV92
1036   F    TA3         42465  19MAY65  23OCT84
1106   M    PT3         94039  06NOV57  16AUG84
1129   F    ME3         36758  08DEC61  17AUG91
1350   F    FA3         36098  31AUG65  29JUL90
1369   M    TA3         36598  28DEC61  13MAR87
1076   M    PT1         69742  14OCT55  03OCT91
;
```

PROCLIB.SCHEDULE

```
data proclib.schedule;
   input flight $3. +5 date date7. +2 dest $3. +3 idnum $4.;
   format date date7.;
   informat date date7.;
   datalines;
132     01MAR08  YYZ   1739
132     01MAR08  YYZ   1478
132     01MAR08  YYZ   1130
132     01MAR08  YYZ   1390
132     01MAR08  YYZ   1983
132     01MAR08  YYZ   1111
219     01MAR08  LON   1407
219     01MAR08  LON   1777
219     01MAR08  LON   1103
219     01MAR08  LON   1125
219     01MAR08  LON   1350
219     01MAR08  LON   1332
271     01MAR08  PAR   1439
271     01MAR08  PAR   1442
271     01MAR08  PAR   1132
271     01MAR08  PAR   1411
271     01MAR08  PAR   1988
```

```
271     01MAR08  PAR   1443
622     01MAR08  FRA   1545
622     01MAR08  FRA   1890
622     01MAR08  FRA   1116
622     01MAR08  FRA   1221
622     01MAR08  FRA   1433
622     01MAR08  FRA   1352
132     02MAR08  YYZ   1556
132     02MAR08  YYZ   1478
132     02MAR08  YYZ   1113
132     02MAR08  YYZ   1411
132     02MAR08  YYZ   1574
132     02MAR08  YYZ   1111
219     02MAR08  LON   1407
219     02MAR08  LON   1118
219     02MAR08  LON   1132
219     02MAR08  LON   1135
219     02MAR08  LON   1441
219     02MAR08  LON   1332
271     02MAR08  PAR   1739
271     02MAR08  PAR   1442
271     02MAR08  PAR   1103
271     02MAR08  PAR   1413
271     02MAR08  PAR   1115
271     02MAR08  PAR   1443
622     02MAR08  FRA   1439
622     02MAR08  FRA   1890
622     02MAR08  FRA   1124
622     02MAR08  FRA   1368
622     02MAR08  FRA   1477
622     02MAR08  FRA   1352
132     03MAR08  YYZ   1739
132     03MAR08  YYZ   1928
132     03MAR08  YYZ   1425
132     03MAR08  YYZ   1135
132     03MAR08  YYZ   1437
132     03MAR08  YYZ   1111
219     03MAR08  LON   1428
219     03MAR08  LON   1442
219     03MAR08  LON   1130
219     03MAR08  LON   1411
219     03MAR08  LON   1115
219     03MAR08  LON   1332
271     03MAR08  PAR   1905
271     03MAR08  PAR   1118
271     03MAR08  PAR   1970
271     03MAR08  PAR   1125
271     03MAR08  PAR   1983
271     03MAR08  PAR   1443
622     03MAR08  FRA   1545
622     03MAR08  FRA   1830
622     03MAR08  FRA   1414
622     03MAR08  FRA   1368
622     03MAR08  FRA   1431
622     03MAR08  FRA   1352
132     04MAR08  YYZ   1428
```

```
132     04MAR08  YYZ   1118
132     04MAR08  YYZ   1103
132     04MAR08  YYZ   1390
132     04MAR08  YYZ   1350
132     04MAR08  YYZ   1111
219     04MAR08  LON   1739
219     04MAR08  LON   1478
219     04MAR08  LON   1130
219     04MAR08  LON   1125
219     04MAR08  LON   1983
219     04MAR08  LON   1332
271     04MAR08  PAR   1407
271     04MAR08  PAR   1410
271     04MAR08  PAR   1094
271     04MAR08  PAR   1411
271     04MAR08  PAR   1115
271     04MAR08  PAR   1443
622     04MAR08  FRA   1545
622     04MAR08  FRA   1890
622     04MAR08  FRA   1116
622     04MAR08  FRA   1221
622     04MAR08  FRA   1433
622     04MAR08  FRA   1352
132     05MAR08  YYZ   1556
132     05MAR08  YYZ   1890
132     05MAR08  YYZ   1113
132     05MAR08  YYZ   1475
132     05MAR08  YYZ   1431
132     05MAR08  YYZ   1111
219     05MAR08  LON   1428
219     05MAR08  LON   1442
219     05MAR08  LON   1422
219     05MAR08  LON   1413
219     05MAR08  LON   1574
219     05MAR08  LON   1332
271     05MAR08  PAR   1739
271     05MAR08  PAR   1928
271     05MAR08  PAR   1103
271     05MAR08  PAR   1477
271     05MAR08  PAR   1433
271     05MAR08  PAR   1443
622     05MAR08  FRA   1545
622     05MAR08  FRA   1830
622     05MAR08  FRA   1970
622     05MAR08  FRA   1441
622     05MAR08  FRA   1350
622     05MAR08  FRA   1352
132     06MAR08  YYZ   1333
132     06MAR08  YYZ   1890
132     06MAR08  YYZ   1414
132     06MAR08  YYZ   1475
132     06MAR08  YYZ   1437
132     06MAR08  YYZ   1111
219     06MAR08  LON   1106
219     06MAR08  LON   1118
219     06MAR08  LON   1425
```

```
219     06MAR08  LON   1434
219     06MAR08  LON   1555
219     06MAR08  LON   1332
132     07MAR08  YYZ   1407
132     07MAR08  YYZ   1118
132     07MAR08  YYZ   1094
132     07MAR08  YYZ   1555
132     07MAR08  YYZ   1350
132     07MAR08  YYZ   1111
219     07MAR08  LON   1905
219     07MAR08  LON   1478
219     07MAR08  LON   1124
219     07MAR08  LON   1434
219     07MAR08  LON   1983
219     07MAR08  LON   1332
271     07MAR08  PAR   1410
271     07MAR08  PAR   1777
271     07MAR08  PAR   1103
271     07MAR08  PAR   1574
271     07MAR08  PAR   1115
271     07MAR08  PAR   1443
622     07MAR08  FRA   1107
622     07MAR08  FRA   1890
622     07MAR08  FRA   1425
622     07MAR08  FRA   1475
622     07MAR08  FRA   1433
622     07MAR08  FRA   1352
;
```

PROCLIB.STAFF

```
data proclib.staff;
   input idnum $4. +3 lname $15. +2 fname $15. +2 city $15. +2
         state $2. +5 hphone $12.;
   datalines;
1919   ADAMS           GERALD          STAMFORD        CT     203/781-1255
1653   ALIBRANDI       MARIA           BRIDGEPORT      CT     203/675-7715
1400   ALHERTANI       ABDULLAH        NEW YORK        NY     212/586-0808
1350   ALVAREZ         MERCEDES        NEW YORK        NY     718/383-1549
1401   ALVAREZ         CARLOS          PATERSON        NJ     201/732-8787
1499   BAREFOOT        JOSEPH          PRINCETON       NJ     201/812-5665
1101   BAUCOM          WALTER          NEW YORK        NY     212/586-8060
1333   BANADYGA        JUSTIN          STAMFORD        CT     203/781-1777
1402   BLALOCK         RALPH           NEW YORK        NY     718/384-2849
1479   BALLETTI        MARIE           NEW YORK        NY     718/384-8816
1403   BOWDEN          EARL            BRIDGEPORT      CT     203/675-3434
1739   BRANCACCIO      JOSEPH          NEW YORK        NY     212/587-1247
1658   BREUHAUS        JEREMY          NEW YORK        NY     212/587-3622
1428   BRADY           CHRISTINE       STAMFORD        CT     203/781-1212
1782   BREWCZAK        JAKOB           STAMFORD        CT     203/781-0019
1244   BUCCI           ANTHONY         NEW YORK        NY     718/383-3334
1383   BURNETTE        THOMAS          NEW YORK        NY     718/384-3569
1574   CAHILL          MARSHALL        NEW YORK        NY     718/383-2338
```

1789	CARAWAY	DAVIS	NEW YORK	NY	212/587-9000
1404	COHEN	LEE	NEW YORK	NY	718/384-2946
1437	CARTER	DOROTHY	BRIDGEPORT	CT	203/675-4117
1639	CARTER-COHEN	KAREN	STAMFORD	CT	203/781-8839
1269	CASTON	FRANKLIN	STAMFORD	CT	203/781-3335
1065	COPAS	FREDERICO	NEW YORK	NY	718/384-5618
1876	CHIN	JACK	NEW YORK	NY	212/588-5634
1037	CHOW	JANE	STAMFORD	CT	203/781-8868
1129	COUNIHAN	BRENDA	NEW YORK	NY	718/383-2313
1988	COOPER	ANTHONY	NEW YORK	NY	212/587-1228
1405	DACKO	JASON	PATERSON	NJ	201/732-2323
1430	DABROWSKI	SANDRA	BRIDGEPORT	CT	203/675-1647
1983	DEAN	SHARON	NEW YORK	NY	718/384-1647
1134	DELGADO	MARIA	STAMFORD	CT	203/781-1528
1118	DENNIS	ROGER	NEW YORK	NY	718/383-1122
1438	DABBOUSSI	KAMILLA	STAMFORD	CT	203/781-2229
1125	DUNLAP	DONNA	NEW YORK	NY	718/383-2094
1475	ELGES	MARGARETE	NEW YORK	NY	718/383-2828
1117	EDGERTON	JOSHUA	NEW YORK	NY	212/588-1239
1935	FERNANDEZ	KATRINA	BRIDGEPORT	CT	203/675-2962
1124	FIELDS	DIANA	WHITE PLAINS	NY	914/455-2998
1422	FUJIHARA	KYOKO	PRINCETON	NJ	201/812-0902
1616	FUENTAS	CARLA	NEW YORK	NY	718/384-3329
1406	FOSTER	GERALD	BRIDGEPORT	CT	203/675-6363
1120	GARCIA	JACK	NEW YORK	NY	718/384-4930
1094	GOMEZ	ALAN	BRIDGEPORT	CT	203/675-7181
1389	GOLDSTEIN	LEVI	NEW YORK	NY	718/384-9326
1905	GRAHAM	ALVIN	NEW YORK	NY	212/586-8815
1407	GREGORSKI	DANIEL	MT. VERNON	NY	914/468-1616
1114	GREENWALD	JANICE	NEW YORK	NY	212/588-1092
1410	HARRIS	CHARLES	STAMFORD	CT	203/781-0937
1439	HASENHAUER	CHRISTINA	BRIDGEPORT	CT	203/675-4987
1409	HAVELKA	RAYMOND	STAMFORD	CT	203/781-9697
1408	HENDERSON	WILLIAM	PRINCETON	NJ	201/812-4789
1121	HERNANDEZ	ROBERTO	NEW YORK	NY	718/384-3313
1991	HOWARD	GRETCHEN	BRIDGEPORT	CT	203/675-0007
1102	HERMANN	JOACHIM	WHITE PLAINS	NY	914/455-0976
1356	HOWARD	MICHAEL	NEW YORK	NY	212/586-8411
1545	HERRERO	CLYDE	STAMFORD	CT	203/781-1119
1292	HUNTER	HELEN	BRIDGEPORT	CT	203/675-4830
1440	JACKSON	LAURA	STAMFORD	CT	203/781-0088
1368	JEPSEN	RONALD	STAMFORD	CT	203/781-8413
1369	JONSON	ANTHONY	NEW YORK	NY	212/587-5385
1411	JOHNSEN	JACK	PATERSON	NJ	201/732-3678
1113	JOHNSON	LESLIE	NEW YORK	NY	718/383-3003
1704	JONES	NATHAN	NEW YORK	NY	718/384-0049
1900	KING	WILLIAM	NEW YORK	NY	718/383-3698
1126	KIMANI	ANNE	NEW YORK	NY	212/586-1229
1677	KRAMER	JACKSON	BRIDGEPORT	CT	203/675-7432
1441	LAWRENCE	KATHY	PRINCETON	NJ	201/812-3337
1421	LEE	RUSSELL	MT. VERNON	NY	914/468-9143
1119	LI	JEFF	NEW YORK	NY	212/586-2344
1834	LEBLANC	RUSSELL	NEW YORK	NY	718/384-0040
1777	LUFKIN	ROY	NEW YORK	NY	718/383-4413
1663	MARKS	JOHN	NEW YORK	NY	212/587-7742
1106	MARSHBURN	JASPER	STAMFORD	CT	203/781-1457

1103	MCDANIEL	RONDA	NEW YORK	NY	212/586-0013
1477	MEYERS	PRESTON	BRIDGEPORT	CT	203/675-8125
1476	MONROE	JOYCE	STAMFORD	CT	203/781-2837
1379	MORGAN	ALFRED	STAMFORD	CT	203/781-2216
1104	MORGAN	CHRISTOPHER	NEW YORK	NY	718/383-9740
1009	MORGAN	GEORGE	NEW YORK	NY	212/586-7753
1412	MURPHEY	JOHN	PRINCETON	NJ	201/812-4414
1115	MURPHY	ALICE	NEW YORK	NY	718/384-1982
1128	NELSON	FELICIA	BRIDGEPORT	CT	203/675-1166
1442	NEWKIRK	SANDRA	PRINCETON	NJ	201/812-3331
1417	NEWKIRK	WILLIAM	PATERSON	NJ	201/732-6611
1478	NEWTON	JAMES	NEW YORK	NY	212/587-5549
1673	NICHOLLS	HENRY	STAMFORD	CT	203/781-7770
1839	NORRIS	DIANE	NEW YORK	NY	718/384-1767
1347	O'NEAL	BRYAN	NEW YORK	NY	718/384-0230
1423	OSWALD	LESLIE	MT. VERNON	NY	914/468-9171
1200	OVERMAN	MICHELLE	STAMFORD	CT	203/781-1835
1970	PARKER	ANNE	NEW YORK	NY	718/383-3895
1521	PARKER	JAY	NEW YORK	NY	212/587-7603
1354	PARKER	MARY	WHITE PLAINS	NY	914/455-2337
1424	PATTERSON	RENEE	NEW YORK	NY	212/587-8991
1132	PEARCE	CAROL	NEW YORK	NY	718/384-1986
1845	PEARSON	JAMES	NEW YORK	NY	718/384-2311
1556	PENNINGTON	MICHAEL	NEW YORK	NY	718/383-5681
1413	PETERS	RANDALL	PRINCETON	NJ	201/812-2478
1123	PETERSON	SUZANNE	NEW YORK	NY	718/383-0077
1907	PHELPS	WILLIAM	STAMFORD	CT	203/781-1118
1436	PORTER	SUSAN	NEW YORK	NY	718/383-5777
1385	RAYNOR	MILTON	BRIDGEPORT	CT	203/675-2846
1432	REED	MARILYN	MT. VERNON	NY	914/468-5454
1111	RHODES	JEREMY	PRINCETON	NJ	201/812-1837
1116	RICHARDS	CASEY	NEW YORK	NY	212/587-1224
1352	RIVERS	SIMON	NEW YORK	NY	718/383-3345
1555	RODRIGUEZ	JULIA	BRIDGEPORT	CT	203/675-2401
1038	RODRIGUEZ	MARIA	BRIDGEPORT	CT	203/675-2048
1420	ROUSE	JEREMY	PATERSON	NJ	201/732-9834
1561	SANDERS	RAYMOND	NEW YORK	NY	212/588-6615
1434	SANDERSON	EDITH	STAMFORD	CT	203/781-1333
1414	SANDERSON	NATHAN	BRIDGEPORT	CT	203/675-1715
1112	SANYERS	RANDY	NEW YORK	NY	718/384-4895
1390	SMART	JONATHAN	NEW YORK	NY	718/383-1141
1332	STEPHENSON	ADAM	BRIDGEPORT	CT	203/675-1497
1890	STEPHENSON	ROBERT	NEW YORK	NY	718/384-9874
1429	THOMPSON	ALICE	STAMFORD	CT	203/781-3857
1107	THOMPSON	WAYNE	NEW YORK	NY	718/384-3785
1908	TRENTON	MELISSA	NEW YORK	NY	212/586-6262
1830	TRIPP	KATHY	BRIDGEPORT	CT	203/675-2479
1882	TUCKER	ALAN	NEW YORK	NY	718/384-0216
1050	TUTTLE	THOMAS	WHITE PLAINS	NY	914/455-2119
1425	UNDERWOOD	JENNY	STAMFORD	CT	203/781-0978
1928	UPCHURCH	LARRY	WHITE PLAINS	NY	914/455-5009
1480	UPDIKE	THERESA	NEW YORK	NY	212/587-8729
1100	VANDEUSEN	RICHARD	NEW YORK	NY	212/586-2531
1995	VARNER	ELIZABETH	NEW YORK	NY	718/384-7113
1135	VEGA	ANNA	NEW YORK	NY	718/384-5913
1415	VEGA	FRANKLIN	NEW YORK	NY	718/384-2823

```
1076   VENTER          RANDALL         NEW YORK        NY     718/383-2321
1426   VICK            THERESA         PRINCETON       NJ     201/812-2424
1564   WALTERS         ANNE            NEW YORK        NY     212/587-3257
1221   WALTERS         DIANE           NEW YORK        NY     718/384-1918
1133   WANG            CHIN            NEW YORK        NY     212/587-1956
1435   WARD            ELAINE          NEW YORK        NY     718/383-4987
1418   WATSON          BERNARD         NEW YORK        NY     718/383-1298
1017   WELCH           DARIUS          NEW YORK        NY     212/586-5535
1443   WELLS           AGNES           STAMFORD        CT     203/781-5546
1131   WELLS           NADINE          NEW YORK        NY     718/383-1045
1427   WHALEY          CAROLYN         MT. VERNON      NY     914/468-4528
1036   WONG            LESLIE          NEW YORK        NY     212/587-2570
1130   WOOD            DEBORAH         NEW YORK        NY     212/587-0013
1127   WOOD            SANDRA          NEW YORK        NY     212/587-2881
1433   YANCEY          ROBIN           PRINCETON       NJ     201/812-1874
1431   YOUNG           DEBORAH         STAMFORD        CT     203/781-2987
1122   YOUNG           JOANN           NEW YORK        NY     718/384-2021
1105   YOUNG           LAWRENCE        NEW YORK        NY     718/384-0008
;
```

PROCLIB.STAFF2

```
data proclib.staff;
   input Name & $16. IdNumber $ Salary
         Site $ HireDate date7.;
   format hiredate date7.;
   datalines;
Capalleti, Jimmy  2355 21163 BR1 30JAN09
Chen, Len         5889 20976 BR1 18JUN06
Davis, Brad       3878 19571 BR2 20MAR84
Leung, Brenda     4409 34321 BR2 18SEP94
Martinez, Maria   3985 49056 US2 10JAN93
Orfali, Philip    0740 50092 US2 16FEB03
Patel, Mary       2398 35182 BR3 02FEB90
Smith, Robert     5162 40100 BR5 15APR06
Sorrell, Joseph   4421 38760 US1 19JUN11
Zook, Carla       7385 22988 BR3 18DEC10
;
```

PROCLIB.SUPERV

```
data proclib.superv;
   input supid $4. +8 state $2. +5  jobcat  $2.;
   label supid='Supervisor Id' jobcat='Job Category';
   datalines;
1677        CT     BC
1834        NY     BC
1431        CT     FA
1433        NJ     FA
1983        NY     FA
```

```
1385        CT     ME
1420        NJ     ME
1882        NY     ME
1935        CT     NA
1417        NJ     NA
1352        NY     NA
1106        CT     PT
1442        NJ     PT
1118        NY     PT
1405        NJ     SC
1564        NY     SC
1639        CT     TA
1401        NJ     TA
1126        NY     TA
;
```

RADIO

この DATA ステップは、INFILE ステートメントを使用して、外部ファイルに保存されているデータを読み込みます。

```
data radio;
   infile 'input-file' missover;
   input /(time1-time7) ($1. +1);
   listener=_n_;
run;
```

外部ファイルに保存されているデータを次に示します。

```
967 32 f 5 3 5
7 5 5 5 7 0 0 0 8 7 0 0 8 0
781 30 f 2 3 5
5 0 0 0 5 0 0 0 4 7 5 0 0 0
859 39 f 1 0 5
1 0 0 0 1 0 0 0 0 0 0 0 0 0
859 40 f 6 1 5
7 5 0 5 7 0 0 0 0 0 0 5 0 0
467 37 m 2 3 1
1 5 5 5 5 4 4 8 8 0 0 0 0 0
220 35 f 3 1 7
7 0 0 0 7 0 0 0 7 0 0 0 0 0
833 42 m 2 2 4
7 0 0 0 7 5 4 7 4 0 1 4 4 0
967 39 f .5 1 7
7 0 0 0 7 7 0 0 0 0 0 0 8 0
677 28 m .5 .5 7
7 0 0 0 0 0 0 0 0 0 0 0 0 0
833 28 f 3 4 1
1 0 0 0 0 1 1 1 1 0 0 0 1 1
677 24 f 3 1 2
2 0 0 0 0 0 0 0 2 0 8 8 0 0 0
688 32 m 5 2 4
5 5 0 4 8 0 0 5 0 8 0 0 0 0
542 38 f 6 8 5
```

```
5 0 0 5 5 5 0 5 5 5 5 5 5 0
677 27 m 6 1 1
1 1 0 4 4 0 0 1 4 0 0 0 0 0
779 37 f 2.5 4 7
7 0 0 0 7 7 0 7 7 4 4 7 8 0
362 31 f 1 2 2
8 0 0 0 8 0 0 0 0 0 0 8 8 0 0
859 29 m 10 3 4
4 4 0 2 2 0 0 4 0 0 0 4 4 0
467 24 m 5 8 1
7 1 1 1 7 1 1 0 1 7 1 1 1 1
851 34 m 1 2 8
0 0 0 0 8 0 0 0 4 0 0 0 8 0
859 23 f 1 1 8
8 0 0 0 8 0 0 0 0 0 0 0 0 8
781 34 f 9 3 1
2 1 0 1 4 4 4 0 1 1 1 1 4 4
851 40 f 2 4 5
5 0 0 0 5 0 0 5 0 0 5 5 0 0
783 34 m 3 2 4
7 0 0 0 7 4 4 0 0 4 4 0 0 0
848 29 f 4 1.5 7
7 4 4 1 7 0 0 0 7 0 0 7 0 0
851 28 f 1 2 2
2 0 2 0 2 0 0 0 0 2 2 2 0 0
856 42 f 1.5 1 2
2 0 0 0 0 0 0 2 0 0 0 0 0 0
859 29 m .5 .5 5
5 0 0 0 1 0 0 0 0 0 8 8 5 0
833 29 m 1 3 2
2 0 0 0 2 2 0 0 4 2 0 2 0 0
859 23 f 10 3 1
1 5 0 8 8 1 4 0 1 1 1 1 1 4
781 37 f .5 2 7
7 0 0 0 1 0 0 0 1 7 0 1 0 0
833 31 f 5 4 1
1 0 0 0 1 0 0 0 4 0 4 0 0 0
942 23 f 4 2 1
1 0 0 0 1 0 1 0 1 1 0 0 0 0
848 33 f 5 4 1
1 1 0 1 1 0 0 0 1 1 1 0 0 0
222 33 f 2 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0
851 45 f .5 1 8
8 0 0 0 8 0 0 0 0 0 8 0 0 0
848 27 f 2 4 1
1 0 0 0 1 1 0 0 4 1 1 1 1 1
781 38 m 2 2 1
5 0 0 0 1 0 0 0 0 0 1 1 0 0
222 27 f 3 1 2
2 0 2 0 2 2 0 0 2 0 0 0 0 0
467 34 f 2 2 1
1 0 0 0 0 1 0 1 0 0 0 0 1 0
833 27 f 8 8 1
7 0 1 0 7 4 0 0 1 1 1 4 1 0
677 49 f 1.5 0 8
```

8 0 8 0 8 0 0 0 0 0 0 0 0 0
849 43 m 1 4 1
1 0 0 0 4 0 0 0 4 0 1 0 0 0
467 28 m 2 1 7
7 0 0 0 7 0 0 7 0 0 1 0 0 0
732 29 f 1 0 2
2 0 0 0 2 0 0 0 0 0 0 0 0 0
851 31 m 2 2 2
2 5 0 6 0 0 8 0 2 2 8 2 0 0
779 42 f 8 2 2
7 2 0 2 7 0 0 0 0 0 0 0 2 0
493 40 m 1 3 3
3 0 0 0 5 3 0 5 5 0 0 0 1 1
859 30 m 1 0 7
7 0 0 0 7 0 0 0 0 0 0 0 0 0
833 36 m 4 2 5
7 5 0 5 0 5 0 0 7 0 0 0 5 0
467 30 f 1 4 1
0 0 0 0 1 0 6 0 0 1 1 1 0 6
859 32 f 3 5 2
2 2 2 2 2 2 6 6 2 2 2 2 2 6
851 43 f 8 1 5
7 5 5 5 0 0 0 4 0 0 0 0 0 0
848 29 f 3 5 1
7 0 0 0 7 1 0 0 1 1 1 1 1 0
833 25 f 2 4 5
7 0 0 0 5 7 0 0 7 5 0 0 5 0
783 33 f 8 3 8
8 0 8 0 7 0 0 0 8 0 5 4 0 5
222 26 f 10 2 1
1 1 0 1 1 0 0 0 3 1 1 0 0 0
222 23 f 3 2 2
2 2 2 2 7 0 0 2 2 0 0 0 0 0
859 50 f 1 5 4
7 0 0 0 7 0 0 5 4 4 4 7 0 0
833 26 f 3 2 1
1 0 0 1 1 0 0 5 5 0 1 0 0 0
467 29 m 7 2 1
1 1 1 1 1 0 0 1 1 1 0 0 0 0
859 35 m .5 2 2
7 0 0 0 2 0 0 7 5 0 0 4 0 0
833 33 f 3 3 6
7 0 0 0 6 8 0 8 0 0 0 8 6 0
221 36 f .5 1 5
0 7 0 0 0 7 0 0 7 0 0 7 7 0
220 32 f 2 4 5
5 0 5 0 5 5 5 0 5 5 5 5 5 5
684 19 f 2 4 2
0 2 0 2 0 0 0 0 0 2 2 0 0 0
493 55 f 1 0 5
5 0 0 5 0 0 0 0 7 0 0 0 0 0
221 27 m 1 1 7
7 0 0 0 0 0 0 0 5 0 0 0 5 0
684 19 f 0 .5 1
7 0 0 0 0 1 1 0 0 0 0 0 1 1
493 38 f .5 .5 5

```
0 8 0 0 5 0 0 0 5 0 0 0 0 0
221 26 f .5 2 1
0 1 0 0 0 1 0 0 5 5 5 1 0 0
684 18 m 1 .5 1
0 2 0 0 0 0 1 0 0 0 0 1 1 0
684 19 m 1 1 1
0 0 0 1 1 0 0 0 0 0 1 0 0 0
221 29 m .5 .5 5
0 0 0 0 0 5 5 0 0 0 0 0 5 5
683 18 f 2 4 8
0 0 0 0 8 0 0 0 8 8 8 0 0 0
966 23 f 1 2 1
1 5 5 5 1 0 0 0 0 1 0 0 1 0
493 25 f 3 5 7
7 0 0 0 7 2 0 0 7 0 2 7 7 0
683 18 f .5 .5 2
1 0 0 0 0 0 5 0 0 1 0 0 0 1
382 21 f 3 1 8
0 8 0 0 5 8 8 0 0 8 8 0 0 0
683 18 f 4 6 2
2 0 0 0 2 2 2 0 2 0 2 2 2 0
684 19 m .5 2 1
0 0 0 0 1 1 0 0 0 1 1 1 1 5
684 19 m 1.5 3.5 2
2 0 0 0 2 0 0 0 0 0 2 5 0 0
221 23 f 1 5 1
7 5 1 5 1 3 1 7 5 1 5 1 3 1
684 18 f 2 3 1
2 0 0 1 1 1 1 7 2 0 1 1 1 1
683 19 f 3 5 2
2 0 0 2 0 6 1 0 1 1 2 2 6 1
683 19 f 3 5 1
2 0 0 2 0 6 1 0 1 1 2 0 2 1
221 35 m 3 5 5
7 5 0 1 7 0 0 5 5 5 0 0 0 0
221 43 f 1 4 5
1 0 0 0 5 0 0 5 5 0 0 0 0 0
493 32 f 2 1 6
0 0 0 6 0 0 0 0 0 0 0 0 4 0
221 24 f 4 5 2
2 0 5 0 0 2 4 4 4 5 0 0 2 2
684 19 f 2 3 2
0 5 5 2 5 0 1 0 5 5 2 2 2 2
221 19 f 3 3 8
0 1 1 8 8 8 4 0 5 4 1 8 8 4
221 29 m 1 1 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5
221 21 m 1 1 1
1 0 0 0 0 0 5 1 0 0 0 0 0 5
683 20 f 1 2 2
0 0 0 0 2 0 0 0 2 0 0 0 0 0
493 54 f 1 1 5
7 0 0 5 0 0 0 0 0 0 5 0 0 0
493 45 m 4 6 5
7 0 0 0 7 5 0 0 5 5 5 5 5 5
850 44 m 2.5 1.5 7
```

```
7 0 7 0 4 7 5 0 5 4 3 0 0 4
220 33 m 5 3 5
1 5 0 5 1 0 0 0 0 0 0 0 5 5
684 20 f 1.5 3 1
1 0 0 0 1 0 1 0 1 0 0 1 1 0
966 63 m 3 5 3
5 4 7 5 4 5 0 5 0 0 5 5 4 0
683 21 f 4 6 1
0 1 0 1 1 1 1 0 1 1 1 1 1 1
493 23 f 5 2 5
7 5 0 4 0 0 0 0 1 1 1 1 1 0
493 32 f 8 8 5
7 5 0 0 7 0 5 5 5 0 0 7 5 5
942 33 f 7 2 5
0 5 5 4 7 0 0 0 0 0 0 7 8 0
493 34 f .5 1 5
5 0 0 0 5 0 0 0 0 0 6 0 0 0
382 40 f 2 2 5
5 0 0 0 5 0 0 5 0 0 5 0 0 0
362 27 f 0 3 8
0 0 0 0 0 0 0 0 0 0 0 0 8 0
542 36 f 3 3 7
7 0 0 0 7 1 0 0 0 7 1 1 0 0
966 39 f 3 6 5
7 0 0 0 7 5 0 0 7 0 5 0 5 0
849 32 m 1 .5 7
7 0 0 0 5 0 0 0 7 4 4 5 7 0
677 52 f 3 2 3
7 0 0 0 0 7 0 0 0 7 0 0 3 0
222 25 m 2 4 1
1 0 0 0 1 0 0 0 1 0 1 0 0 0
732 42 f 3 2 7
7 0 0 0 1 7 5 5 7 0 0 3 4 0
467 26 f 4 4 1
7 0 1 0 7 1 0 0 7 7 4 7 0 0
467 38 m 2.5 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0
382 37 f 1.5 .5 7
7 0 0 0 7 0 0 0 3 0 0 0 3 0
856 45 f 3 3 7
7 0 0 0 7 5 0 0 7 7 4 0 0 0
677 33 m 3 2 7
7 0 0 4 7 0 0 0 7 0 0 0 0 0
490 27 f .5 1 2
2 0 0 0 2 0 0 0 2 0 2 0 0 0
362 27 f 1.5 2 2
2 0 0 0 1 0 4 0 1 0 0 0 4 4
783 25 f 2 1 1
1 0 0 0 1 7 0 0 0 0 1 1 1 0
546 30 f 8 3 1
1 1 1 1 1 0 0 1 0 5 5 0 0 0
677 30 f 2 0 1
1 0 0 0 0 1 0 0 0 0 0 0 0 1
221 35 f 2 2 1
1 0 0 0 1 0 1 0 1 1 1 0 0 0
966 32 f 6 1 7
```

```
7 1 1 1 7 4 0 1 7 1 8 8 4 0
222 28 f 1 5 4
7 0 0 0 4 0 0 4 4 4 4 0 0 0
467 29 f 5 3 4
4 5 5 5 1 4 4 5 1 1 1 1 4 4
467 32 m 3 4 1
1 0 1 0 4 0 0 0 4 0 0 0 1 0
966 30 m 1.5 1 7
7 0 0 0 7 5 0 7 0 0 0 0 5 0
967 38 m 14 4 7
7 7 7 7 7 0 4 8 0 0 0 0 4 0
490 28 m 8 1 1
7 1 1 1 1 0 0 7 0 0 8 0 0 0
833 30 f .5 1 6
6 0 0 0 6 0 0 0 0 6 0 0 6 0
851 40 m 1 0 7
7 5 5 5 7 0 0 0 0 0 0 0 0 0
859 27 f 2 5 2
6 0 0 0 2 0 0 0 0 0 0 2 2 2
851 22 f 3 5 2
7 0 2 0 2 2 0 0 2 0 8 0 2 0
967 38 f 1 1.5 7
7 0 0 0 7 5 0 7 4 0 0 7 5 0
856 34 f 1.5 1 1
0 1 0 0 0 1 0 0 4 0 0 0 0 0
222 33 m .1 .1 7
7 0 0 0 7 0 0 0 0 0 7 0 0 0
856 22 m .50 .25 1
0 1 0 0 1 0 0 0 0 0 0 0 0 0
677 30 f 2 2 4
1 0 4 0 4 0 0 0 4 0 0 0 0 0
859 25 m 2 3 7
0 0 0 0 0 7 0 0 7 0 2 0 0 1
833 35 m 2 6 7
7 0 0 0 7 1 1 0 4 7 4 7 1 1
677 35 m 10 4 1
1 1 1 1 1 8 6 8 1 0 0 8 8 8
848 29 f 5 3 8
8 0 0 0 8 8 0 0 0 8 8 8 0 0
688 26 m 3 1 1
1 1 7 1 1 7 0 0 0 8 8 0 0 0
490 41 m 2 2 5
5 0 0 0 0 0 5 5 0 0 0 0 0 5
493 35 m 4 4 7
7 5 0 5 7 0 0 7 7 7 7 0 0 0
677 27 m 15 11 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
848 27 f 3 5 1
1 1 0 0 1 1 0 0 1 1 1 1 0 0
362 30 f 1 0 1
1 0 0 0 7 5 0 0 0 0 0 0 0 0
783 29 f 1 1 4
4 0 0 0 4 0 0 0 4 0 0 0 4 0
467 39 f .5 2 4
7 0 4 0 4 4 0 0 4 4 4 4 4 4
677 27 m 2 2 7
```

```
7 0 0 0 7 0 0 7 7 0 0 7 0 0
221 23 f 2.5 1 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0
677 29 f 1 1 7
0 0 0 0 7 0 0 0 7 0 0 0 0 0
783 32 m 1 2 5
4 5 5 5 4 2 0 0 0 0 3 2 2 0
833 25 f 1 0 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0
859 24 f 7 3 7
1 0 0 0 1 0 0 0 0 1 0 0 1 0
677 29 m 2 2 8
0 8 8 0 8 0 0 0 8 8 8 0 0 0
688 31 m 8 2 5
7 5 5 5 5 7 0 0 7 7 0 0 0 0
856 31 m 9 4 1
1 1 1 1 1 0 0 0 0 0 0 0 1 0
856 44 f 1 0 6
6 0 0 0 6 0 0 0 0 0 0 0 0 0
677 37 f 3 3 1
0 0 1 0 0 0 0 0 4 4 0 0 0 0
859 27 m 2 .5 2
2 2 2 2 2 2 2 2 0 0 0 0 0 2
781 30 f 10 4 2
2 0 0 0 2 0 2 0 0 0 0 0 0 2
362 27 m 12 4 3
3 1 1 1 1 3 3 3 0 0 0 0 3 0
362 33 f 2 4 1
1 0 0 0 7 0 0 7 1 1 1 1 1 0
222 26 f 8 1 1
1 1 1 1 0 0 0 1 0 0 0 0 0 0
779 37 f 6 3 1
1 1 1 1 1 0 0 1 1 0 0 0 1 0
467 32 f 1 1 2
2 0 0 0 0 0 0 0 2 0 0 2 0 0
859 23 m 1 1 1
1 0 0 0 1 1 0 1 0 0 0 0 1 1
781 33 f 1 .5 6
6 0 0 0 6 0 0 0 0 0 0 0 0 0
779 28 m 5 2 1
1 1 1 1 1 0 0 0 0 7 7 1 1 0
677 28 m 3 1 5
7 5 5 5 5 6 0 0 6 6 6 6 6 0
677 25 f 9 2 5
1 5 5 5 5 1 1 0 1 1 1 1 1 1
848 30 f 6 2 8
8 0 0 0 2 7 0 0 0 0 2 0 2 0
546 36 f 4 6 4
7 0 0 0 4 4 0 5 5 5 5 2 4 4
222 30 f 2 3 2
2 2 0 0 2 0 0 0 2 0 2 2 0 0
383 32 m 4 1 2
2 0 0 0 2 0 0 2 0 0 0 0 0 0
851 43 f 8 1 6
4 6 0 6 4 0 0 0 0 0 0 0 0 0
222 27 f 1 3 1
```

```
1 1 0 1 1 1 0 0 1 0 0 0 4 0
833 22 f 1.5 2 1
1 0 0 0 1 1 0 0 1 1 1 0 0 0
467 29 f 2 1 8
8 0 8 0 8 0 0 0 0 0 8 0 0 0
856 28 f 2 3 1
1 0 0 0 1 0 0 0 1 0 0 1 0 0
580 31 f 2.5 2.5 6
6 6 6 6 6 6 6 6 1 1 1 1 6 6
688 39 f 8 8 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3
677 37 f 1.5 .5 1
6 1 1 1 6 6 0 0 1 1 6 6 6 0
859 38 m 3 6 3
7 0 0 0 7 3 0 0 3 0 3 0 0 0
677 25 f 7 1 1
0 1 1 1 2 0 0 0 1 2 1 1 1 0
848 36 f 7 1 1
0 1 0 1 1 0 0 0 0 0 0 1 1 0
781 31 f 2 4 1
1 0 0 0 1 1 0 1 1 1 1 1 0 0
781 40 f 2 2 8
8 0 0 8 8 0 0 0 0 0 0 8 8 0 0
677 25 f 3 5 1
1 6 1 6 6 3 0 0 2 2 1 1 1 1
779 33 f 3 2 1
1 0 1 0 0 0 1 0 1 0 0 0 1 0
677 25 m 7 1.5 1
1 1 0 1 1 0 0 0 0 0 1 0 0 0
362 35 f .5 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0
677 41 f 6 2 7
7 7 0 7 7 0 0 0 0 0 8 0 0 0
677 24 m 5 1 5
1 5 0 5 0 0 0 0 1 0 0 0 0 0
833 29 f .5 0 6
6 0 0 0 6 0 0 0 0 0 0 0 0 0
362 30 f 1 1 1
1 0 0 0 1 0 0 0 1 0 0 0 0 0
850 26 f 6 12 6
6 0 0 0 2 2 2 6 6 6 0 0 6 6
467 25 f 2 3 1
1 0 0 6 1 1 0 0 0 0 1 1 1 1
967 29 f 1 2 7
7 0 0 0 7 0 0 7 7 0 0 0 0 0
833 31 f 1 1 7
7 0 7 0 7 3 0 0 3 3 0 0 0 0
859 40 f 7 1 5
1 5 0 5 5 1 0 0 1 0 0 0 0 0
848 31 m 1 2 1
1 0 0 0 1 1 0 0 4 4 1 4 0 0
222 32 f 2 3 3
3 0 0 0 0 7 0 0 3 0 8 0 0 0
783 33 f 2 0 4
7 0 0 0 7 0 0 0 4 0 4 0 0 0
856 28 f 8 4 2
```

0 2 0 2 2 0 0 0 2 0 2 0 4 0
781 30 f 3 5 1
1 1 1 1 1 1 0 0 1 1 1 1 1 0
850 25 f 6 3 1
7 5 0 5 7 1 0 0 7 0 1 0 1 0
580 33 f 2.5 4 2
2 0 0 0 2 0 0 0 0 0 8 8 0 0
677 38 f 3 3 1
1 0 0 0 1 0 1 1 1 0 1 0 0 4
677 26 f 2 2 1
1 0 1 0 1 0 0 0 1 1 1 0 0 0
467 52 f 3 2 2
2 6 6 6 6 2 0 0 2 2 2 2 0 0
542 31 f 1 3 1
1 0 1 0 1 0 0 0 1 1 1 1 1 0
859 50 f 9 3 6
6 6 6 6 6 6 6 6 6 3 3 3 6 6
779 26 f 1 2 1
7 0 1 0 1 1 4 1 4 1 1 1 4 4
779 36 m 1.5 2 4
1 4 0 4 4 0 0 4 4 4 4 0 0 0
222 31 f 0 3 7
1 0 0 0 7 0 0 0 0 0 0 0 0 0
362 27 f 1 1 1
1 0 1 0 1 4 0 4 4 1 0 4 4 0
967 32 f 3 2 7
7 0 0 0 7 0 0 0 1 0 0 1 0 0
362 29 f 10 2 2
2 2 2 2 2 2 2 2 2 2 2 7 0 0
677 27 f 3 4 1
0 5 1 1 0 5 0 0 0 1 1 1 0 0
546 32 m 5 .5 8
8 0 0 0 8 0 0 0 8 0 0 0 0 0
688 38 m 2 3 2
2 0 0 0 2 0 0 0 2 0 0 0 1 0
362 28 f 1 1 1
1 0 0 0 1 1 0 4 0 0 0 0 4 0
851 32 f .5 2 4
5 0 0 0 4 0 0 0 0 0 0 0 2 0
967 43 f 2 2 1
1 0 0 0 1 0 0 1 7 0 0 0 1 0
467 44 f 10 4 6
7 6 0 6 6 0 6 0 0 0 0 0 0 6
467 23 f 5 3 1
0 2 1 2 1 0 0 0 1 1 1 1 1 1
783 30 f 1 .5 1
1 0 0 0 1 0 0 0 0 0 0 7 0 0
677 29 f 3 1 2
2 2 2 2 2 0 0 0 0 0 0 0 0 0
859 26 f 9.5 1.5 2
2 2 2 2 2 0 0 2 2 0 0 0 0 0
222 28 f 3 0 2
2 0 0 0 2 0 0 0 0 0 2 0 0 0
966 37 m 2 1 1
7 1 1 1 7 0 0 0 7 0 0 0 0 0
859 31 f 10 10 1

```
0 1 1 1 1 0 0 0 1 1 0 0 1 0
781 27 f 2 1 2
2 0 0 0 1 0 0 0 4 0 0 0 0 0
677 31 f .5 .5 6
7 0 0 0 0 0 0 0 6 0 0 0 0 0
848 28 f 5 1 2
2 2 0 2 0 0 0 0 2 0 0 0 0 0
781 24 f 3 3 6
1 6 6 6 1 6 0 0 0 0 1 0 1 1
856 27 f 1.5 1 6
2 6 6 6 2 5 0 2 0 0 5 2 0 0
382 30 m 1 2 7
7 0 0 0 7 0 4 7 0 0 0 7 4 4
848 25 f 9 3 1
7 1 1 5 1 0 0 0 1 1 1 1 1 0
382 30 m 1 2 4
7 0 0 0 7 0 4 7 0 0 0 7 4 4
688 40 m 2 3 1
1 0 0 0 1 3 1 0 5 0 4 4 7 1
856 40 f .5 5 5
3 0 0 0 3 0 0 0 0 0 5 5 0 0
966 25 f 2 .5 2
1 0 0 0 2 6 0 0 4 0 0 0 0 0
859 30 f 2 4 2
2 0 0 0 0 2 0 0 0 0 2 0 0 0
849 29 m 10 1 5
7 5 5 5 7 5 5 0 0 0 0 0 7 0
781 28 m 1.5 3 4
1 0 0 0 1 4 4 0 4 4 1 1 4 0
467 35 f 4 2 6
7 6 7 6 6 7 6 7 7 7 7 7 7 6
222 32 f 10 5 1
1 1 0 1 1 0 0 1 1 1 0 0 1 0
677 32 f 1 0 1
1 0 1 0 0 0 0 0 0 0 0 0 0 0
222 54 f 21 4 3
5 0 0 0 7 0 0 7 0 0 0 0 0 0
677 30 m 4 6 1
7 0 0 0 0 1 1 1 7 1 1 0 8 1
683 29 f 1 2 8
8 0 0 0 8 0 0 0 0 8 8 0 0 0
467 38 m 3 5 1
1 0 0 0 1 0 0 1 1 0 0 0 0 0
781 29 f 2 3 8
8 0 0 0 8 8 0 0 8 8 0 8 8 0
781 30 f 1 0 5
5 0 0 0 0 5 0 0 0 0 0 0 0 0
783 40 f 1.5 3 1
1 0 0 0 1 4 0 0 1 1 1 0 0 0
851 30 f 1 1 6
6 0 0 0 6 0 0 0 6 0 0 6 0 0
851 40 f 1 1 5
5 0 0 0 5 0 0 0 0 1 0 0 0 0
779 40 f 1 0 2
2 0 0 0 2 0 0 0 0 0 0 0 0 0
467 37 f 4 8 1
```

1 0 0 0 1 0 3 0 3 1 1 1 0 0
859 37 f 4 3 3
0 3 7 0 0 7 0 0 0 7 8 3 7 0
781 26 f 4 1 2
2 2 0 2 1 0 0 0 2 0 0 0 0 0
859 23 f 8 3 3
3 2 0 2 3 0 0 0 1 0 0 3 0 0
967 31 f .5 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0
851 38 m 4 2 5
7 5 0 5 4 0 4 7 7 0 4 0 8 0
467 30 m 2 1 2
2 2 0 2 0 0 0 0 2 0 2 0 0 0
848 33 f 2 2 7
7 0 0 0 0 7 0 7 7 0 0 0 7 0
688 35 f 5 8 3
2 2 2 2 2 0 0 3 3 3 3 3 0 0
467 27 f 2 3 1
1 0 1 0 0 1 0 0 1 1 1 0 0 0
783 42 f 3 1 1
1 0 0 0 1 0 0 0 1 0 1 1 0 0
687 40 m 1.5 2 1
7 0 0 0 1 1 0 0 1 0 7 0 1 0
779 30 f 4 8 7
7 0 0 0 7 0 6 7 4 2 2 0 0 6
222 34 f 9 0 8
8 2 0 2 8 0 0 0 0 0 0 0 0 0
467 28 m 3 1 2
2 0 0 0 2 2 0 0 0 2 2 0 0 0
222 28 f 8 4 2
1 2 1 2 2 0 0 1 2 2 0 0 2 0
542 35 m 2 3 2
6 0 7 0 7 0 7 0 0 0 2 2 0 0
677 31 m 12 4 3
7 3 0 3 3 4 0 0 4 4 4 0 0 0
783 45 f 1.5 2 6
6 0 0 0 6 0 0 6 6 0 0 0 0 0
942 34 f 1 .5 4
4 0 0 0 1 0 0 0 0 0 2 0 0 0
222 30 f 8 4 1
1 1 1 1 1 0 0 0 1 1 0 0 0 0
967 38 f 1.5 2 7
7 0 0 0 7 0 0 7 1 1 1 1 0 0
783 37 f 2 1 1
6 6 1 1 6 6 0 0 6 1 1 1 6 0
467 31 f 1.5 2 2
2 0 7 0 7 0 0 7 7 0 0 0 7 0
859 48 f 3 0 7
7 0 0 0 0 0 0 0 0 0 7 0 0 0 0
490 35 f 1 1 7
7 0 0 0 7 0 0 0 0 0 0 0 8 0
222 27 f 3 2 3
8 0 0 0 3 8 0 3 3 0 0 0 0 0
382 36 m 3 2 4
7 0 5 4 7 4 4 0 7 7 4 7 0 4
859 37 f 1 1 2

```
7 0 0 0 0 2 0 2 2 0 0 0 0 2
856 29 f 3 1 1
1 0 0 0 1 1 1 1 0 0 1 1 0 1
542 32 m 3 3 7
7 0 0 0 0 7 7 7 0 0 0 0 7 7
783 31 m 1 1 1
1 0 0 0 1 0 0 0 1 1 1 0 0 0
833 35 m 1 1 1
5 4 1 5 1 0 0 1 1 0 0 0 0 0
782 38 m 30 8 5
7 5 5 5 5 0 0 4 4 4 4 4 0 0
222 33 m 3 3 1
1 1 1 1 1 1 1 1 4 1 1 1 1 1
467 24 f 2 4 1
0 0 1 0 1 0 0 0 1 1 1 0 0 0
467 34 f 1 1 1
1 0 0 0 1 0 0 1 1 0 0 0 0 0
781 53 f 2 1 5
5 0 0 0 5 5 0 0 0 0 5 5 5 0
222 30 m 2 5 3
6 3 3 3 6 0 0 0 3 3 3 3 0 0
688 26 f 2 2 1
1 0 0 0 1 0 0 0 1 0 1 1 0 0
222 29 m 8 5 1
1 6 0 6 1 0 0 1 1 1 1 0 0 0
783 33 m 1 2 7
7 0 0 0 7 0 0 0 7 0 0 0 7 0
781 39 m 1.5 2.5 2
2 0 2 0 2 0 0 0 2 2 2 0 0 0
850 22 f 2 1 1
1 0 0 0 1 1 1 0 5 0 0 1 0 0
493 36 f 1 0 5
0 0 0 0 7 0 0 0 0 0 0 0 0 0
967 46 f 2 4 7
7 5 0 5 7 0 0 0 4 7 4 0 0 0
856 41 m 2 2 4
7 4 0 0 7 4 0 4 0 0 0 7 0 0
546 25 m 5 5 8
8 8 0 0 0 0 0 0 0 0 0 0 0 0
222 27 f 4 4 3
2 2 2 3 7 7 0 2 2 2 3 3 3 0
688 23 m 9 3 3
3 3 3 3 3 7 0 0 3 0 0 0 0 0
849 26 m .5 .5 8
8 0 0 0 8 0 0 0 0 8 0 0 0 0
783 29 f 3 3 1
1 0 0 0 4 0 0 4 1 0 1 0 0 0
856 34 f 1.5 2 1
7 0 0 0 7 0 0 7 4 0 0 7 0 0
966 33 m 3 5 4
7 0 0 0 7 4 5 0 7 0 0 7 4 4
493 34 f 2 5 1
1 0 0 0 1 0 0 0 7 0 1 1 8 0
467 29 m 2 4 2
2 0 0 0 2 0 0 2 2 2 2 2 2 2
677 28 f 1 4 1
```

```
1 1 1 1 1 0 0 0 1 0 1 0 0 0
781 27 m 2 2 1
1 0 1 0 4 2 4 0 2 2 1 0 1 4
467 24 m 4 4 1
7 1 0 1 1 1 0 7 1 0 0 0 0 0
859 26 m 5 5 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
848 27 m 7 2 5
7 5 0 5 4 5 0 0 0 7 4 4 0 4
677 25 f 1 2 8
8 0 0 0 0 5 0 0 8 0 0 0 2 0
222 26 f 3.5 0 2
2 0 0 0 2 0 0 0 0 0 0 0 0 0
833 32 m 1 2 1
1 0 0 0 1 0 0 0 5 0 1 0 0 0
781 28 m 2 .5 7
7 0 0 0 7 0 0 0 4 0 0 0 0 0
783 28 f 1 1 1
1 0 0 0 1 0 0 0 0 0 1 1 0 0
222 28 f 5 5 2
2 6 6 2 2 0 0 0 2 2 0 0 2 2
851 33 m 4 5 3
1 0 0 0 7 3 0 3 3 3 3 3 7 5
859 39 m 2 1 1
1 0 0 0 1 0 0 0 0 0 0 1 0 0
848 45 m 2 2 7
7 0 0 0 7 0 0 0 7 0 0 0 0 0
467 37 m 2 2 7
7 0 0 0 0 7 0 0 0 7 0 0 7 0
859 32 m .25 .25 1
1 0 0 0 0 0 0 0 1 0 0 0 0 0
```

SALES

```
data sales;
   input Region $ CitySize $ Population Product $ SaleType $ Units NetSales;
   cards;
NC S 25000 A100 R 150 3750.00
NC M  125000 A100 R 350 8650.00
NC L 837000 A100 R 800 20000.00
NC S 25000 A100 W 150 3000.00
NC M 125000 A100 W 350 7000.00
NC M 625000 A100 W 750 15000.00
TX M 227000 A100 W 350 7250.00
TX L 5000 A100 W 750 5000.00
;
```

付録 4

ICU ライセンス

ICU ライセンス - ICU 1.8.1 以降

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

サードパーティのソフトウェアライセンス

本セクションの内容は、サードパーティのソフトウェア通知、および ICU ライブラリに含まれライセンスを受けているサードパーティのソフトウェアコンポーネントに対する追加条件です。

1.ユニコードのデータファイルおよびソフトウェア

別紙 1

UNICODE, INC.LICENSE AGREEMENT - DATA FILES AND SOFTWARE

ユニコードのデータファイルには、次のディレクトリ以下のすべてのデータファイルが含まれます。http://www.unicode.org/Public/、および http://www.unicode.org/reports/、および http://www.unicode.org/cldr/data/.次のディレクトリの下の PDF オンラインコードチャートは、ユニコードデータファイルには含まれません。http://www.unicode.org/Public/.ソフトウェアには、ユニコード規格で公表されているソースコード、または次のディレクトリ以下のソースコードが含まれます。http://www.unicode.org/Public/、および http://www.unicode.org/reports/、および http://www.unicode.org/cldr/data/.

NOTICE TO USER:Carefully read the following legal agreement.BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"), YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT.IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2014 Unicode, Inc. All rights reserved.Distributed under the Terms of Use in http://www.unicode.org/copyright.html.

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that (a) the above copyright notice(s) and this permission notice appear with all copies of the Data Files or Software, (b) both the above copyright notice(s) and this permission notice appear in associated documentation, and (c) there is clear notice in each modified Data File or in the Software as well as in the documentation associated with the Data File(s) or Software that the data or software has been modified.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY

RIGHTS.IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

Unicode and the Unicode logo are trademarks of Unicode, Inc. in the United States and other countries.All third party trademarks referenced herein are the property of their respective owners.

2.Chinese/Japanese Word Break Dictionary Data (cjdict.txt)

```
#  The Google Chrome software developed by Google is licensed under
#  the BSD license. Other software included in this distribution is provided
#  under other licenses, as set forth below.
#
#    The BSD License
#    http://opensource.org/licenses/bsd-license.php
#    Copyright (C) 2006-2008, Google Inc.
#
#    All rights reserved.
#
#    Redistribution and use in source and binary forms, with or without
#  modification, are permitted provided that the following conditions are met:
#
#    Redistributions of source code must retain the above copyright notice, this
#  list of conditions and the following disclaimer.
#    Redistributions in binary form must reproduce the above copyright notice,
#  this list of conditions and the following disclaimer in the documentation
#  and/or other materials provided with the distribution.
#    Neither the name of Google Inc. nor the names of its contributors may be
#  used to endorse or promote products derived from this software without
#  specific prior written permission.
#
#
#  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
#  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
#  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
#  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
#  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
#  CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
#  SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
#  INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
#  CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
#  THE POSSIBILITY OF SUCH DAMAGE.
#
#
#    The word list in cjdict.txt are generated by combining three word lists
```

```
#  listed below with further processing for compound word breaking. The
#  frequency is generated with an iterative training against Google
#  web corpora.
#
#    * Libtabe (Chinese)
#      - https://sourceforge.net/project/?group_id=1519
#      - Its license terms and conditions are shown below.
#
#    * IPADIC (Japanese)
#      - http://chasen.aist-nara.ac.jp/chasen/distribution.html
#      - Its license terms and conditions are shown below.
#
#    ---------COPYING.libtabe ---- BEGIN--------------------
#
#    /*
#     * Copyrighy (c) 1999 TaBE Project.
#     * Copyright (c) 1999 Pai-Hsiang Hsiao.
#     * All rights reserved.
#     *
#     * Redistribution and use in source and binary forms, with or without
#     * modification, are permitted provided that the following conditions
#     * are met:
#     *
#     * . Redistributions of source code must retain the above copyright
#     *   notice, this list of conditions and the following disclaimer.
#     * . Redistributions in binary form must reproduce the above copyright
#     *   notice, this list of conditions and the following disclaimer in
#     *   the documentation and/or other materials provided with the
#     *   distribution.
#     * . Neither the name of the TaBE Project nor the names of its
#     *   contributors may be used to endorse or promote products derived
#     *   from this software without specific prior written permission.
#     *
#     * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#     * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#     * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
#     * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#     * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#     * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#     * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
#     * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
#     * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
#     * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#     * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
#     * OF THE POSSIBILITY OF SUCH DAMAGE.
#     */
#
#    /*
#     * Copyright (c) 1999 Computer Systems and Communication Lab,
#     *                    Institute of Information Science, Academia Sinica.
#     * All rights reserved.
#     *
#     * Redistribution and use in source and binary forms, with or without
#     * modification, are permitted provided that the following conditions
#     * are met:
#     *
```

```
#     * . Redistributions of source code must retain the above copyright
#     *   notice, this list of conditions and the following disclaimer.
#     * . Redistributions in binary form must reproduce the above copyright
#     *   notice, this list of conditions and the following disclaimer in
#     *   the documentation and/or other materials provided with the
#     *   distribution.
#     * . Neither the name of the Computer Systems and Communication Lab
#     *   nor the names of its contributors may be used to endorse or
#     *   promote products derived from this software without specific
#     *   prior written permission.
#     *
#     * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#     * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#     * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
#     * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#     * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#     * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#     * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
#     * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
#     * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
#     * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#     * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
#     * OF THE POSSIBILITY OF SUCH DAMAGE.
#     */
#
#    Copyright 1996 Chih-Hao Tsai @ Beckman Institute, University of Illinois
#    c-tsai4@uiuc.edu  http://casper.beckman.uiuc.edu/~c-tsai4
#
#    ---------------COPYING.libtabe-----END-----------------------------------
#
#
#    ---------------COPYING.ipadic-----BEGIN----------------------------------
#
#    Copyright 2000, 2001, 2002, 2003 Nara Institute of Science
#    and Technology.  All Rights Reserved.
#
#    Use, reproduction, and distribution of this software is permitted.
#    Any copy of this software, whether in its original form or modified,
#    must include both the above copyright notice and the following
#    paragraphs.
#
#    Nara Institute of Science and Technology (NAIST),
#    the copyright holders, disclaims all warranties with regard to this
#    software, including all implied warranties of merchantability and
#    fitness, in no event shall NAIST be liable for
#    any special, indirect or consequential damages or any damages
#    whatsoever resulting from loss of use, data or profits, whether in an
#    action of contract, negligence or other tortuous action, arising out
#    of or in connection with the use or performance of this software.
#
#    A large portion of the dictionary entries
#    originate from ICOT Free Software.  The following conditions for ICOT
#    Free Software applies to the current dictionary as well.
#
#    Each User may also freely distribute the Program, whether in its
#    original form or modified, to any third party or parties, PROVIDED
```

```
#    that the provisions of Section 3 ("NO WARRANTY") will ALWAYS appear
#    on, or be attached to, the Program, which is distributed substantially
#    in the same form as set out herein and that such intended
#    distribution, if actually made, will neither violate or otherwise
#    contravene any of the laws and regulations of the countries having
#    jurisdiction over the User or the intended distribution itself.
#
#    NO WARRANTY
#
#    The program was produced on an experimental basis in the course of the
#    research and development conducted during the project and is provided
#    to users as so produced on an experimental basis.  Accordingly, the
#    program is provided without any warranty whatsoever, whether express,
#    implied, statutory or otherwise.  The term "warranty" used herein
#    includes, but is not limited to, any warranty of the quality,
#    performance, merchantability and fitness for a particular purpose of
#    the program and the nonexistence of any infringement or violation of
#    any right of any third party.
#
#    Each user of the program will agree and understand, and be deemed to
#    have agreed and understood, that there is no warranty whatsoever for
#    the program and, accordingly, the entire risk arising from or
#    otherwise connected with the program is assumed by the user.
#
#    Therefore, neither ICOT, the copyright holder, or any other
#    organization that participated in or was otherwise related to the
#    development of the program and their respective officials, directors,
#    officers and other employees shall be held liable for any and all
#    damages, including, without limitation, general, special, incidental
#    and consequential damages, arising out of or otherwise in connection
#    with the use or inability to use the program or any product, material
#    or result produced or otherwise obtained by using the program,
#    regardless of whether they have been advised of, or otherwise had
#    knowledge of, the possibility of such damages at any time during the
#    project or thereafter.  Each user will be deemed to have agreed to the
#    foregoing by his or her commencement of use of the program.  The term
#    "use" as used herein includes, but is not limited to, the use,
#    modification, copying and distribution of the program and the
#    production of secondary products from the program.
#
#    In the case where the program, whether in its original form or
#    modified, was distributed or delivered to or received by a user from
#    any person, organization or entity other than ICOT, unless it makes or
#    grants independently of ICOT any specific warranty to the user in
#    writing, such person, organization or entity, will also be exempted
#    from and not be held liable to the user for any such damages as noted
#    above as far as the program is concerned.
#
#    ---------------COPYING.ipadic-----END-----------------------------------
```

Lao Word Break Dictionary Data (laodict.txt)

```
#    Copyright (c) 2013 International Business Machines Corporation
#    and others. All Rights Reserved.
#
```

```
#    Project:    http://code.google.com/p/lao-dictionary/
#    Dictionary: http://lao-dictionary.googlecode.com/git/Lao-Dictionary.txt
#    License:    http://lao-dictionary.googlecode.com/git/Lao-Dictionary-LICENSE.txt
#                (copied below)
#
#    This file is derived from the above dictionary, with slight modifications.
#    --------------------------------------------------------------------------------
#    Copyright (C) 2013 Brian Eugene Wilson, Robert Martin Campbell.
#    All rights reserved.
#
#    Redistribution and use in source and binary forms, with or without modification,
#    are permitted provided that the following conditions are met:
#
#        Redistributions of source code must retain the above copyright notice, this
#        list of conditions and the following disclaimer. Redistributions in binary
#        form must reproduce the above copyright notice, this list of conditions and
#        the following disclaimer in the documentation and/or other materials
#        provided with the distribution.
#
#    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
#    ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
#    WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
#    DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
#    ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#    (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
#    LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
#    ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
#    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
#    SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#    --------------------------------------------------------------------------------
```

3.Time Zone Database

ICU uses the public domain data and code derived from Time Zone Database for its time zone support.The ownership of the TZ database is explained in BCP 175:Procedure for Maintaining the Time Zone Database section 7.

```
7.  Database Ownership

    The TZ database itself is not an IETF Contribution or an IETF
    document.  Rather it is a pre-existing and regularly updated work
    that is in the public domain, and is intended to remain in the public
    domain.  Therefore, BCPs 78 [RFC5378] and 79 [RFC3979] do not apply
    to the TZ Database or contributions that individuals make to it.
    Should any claims be made and substantiated against the TZ Database,
    the organization that is providing the IANA Considerations defined in
    this RFC, under the memorandum of understanding with the IETF,
    currently ICANN, may act in accordance with all competent court
    orders.  No ownership claims will be made by ICANN or the IETF Trust
    on the database or the code.  Any person making a contribution to the
    database or code waives all rights to future claims in that
    contribution or in the TZ Database.
```

推奨資料

このタイトルに関する推奨参照文献リスト:

- *Base SAS Utilities: リファレンス*
- *Carpenter's Complete Guide to the SAS® REPORT Procedure*
- *Learning SAS® by Example:A Programmer's Guide*
- *Little SAS® Book:A Primer, Fifth Edition*
- *Output Delivery System:The Basics and Beyond*
- *PROC REPORT by Example:Techniques for Building Professional Reports Using SAS*
- *PROC SQL:Beyond the Basics Using SAS,® Second Edition*
- *PROC TABULATE by Example*
- *SAS コンポーネントオブジェクト: リファレンス*
- *SAS データセットオプション: リファレンス*
- *SAS DS2 Language Reference*
- *SAS 出力形式と入力形式: リファレンス*
- *SAS 関数と CALL ルーチン: リファレンス*
- *SAS 言語リファレンス: 解説編*
- *SAS Output Delivery System: ユーザーガイド*
- *SAS SQL プロシジャユーザーガイド*
- *SAS ステートメント: リファレンス*
- *SAS システムオプション: リファレンス*
- SAS は、SAS Foundation の開始に役立つ、講師によるトレーニングと自分のペースで学習できる E ラーニングコースを提供しています。利用可能なコースの詳細については、support.sas.com/training を参照してください。

SAS 刊行物の一覧については、sas.com/store/books から入手できます。必要な書籍についての質問は SAS 担当者までお寄せください:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
電話: 1-800-727-0025
ファクシミリ: 1-919-677-4444

メール: sasbook@sas.com
Web アドレス: sas.com/store/books

キーワード

B

C

E

M

N

O

P

Q

R

S

U

X

あ

か

さ

た

な

は

ま

わ