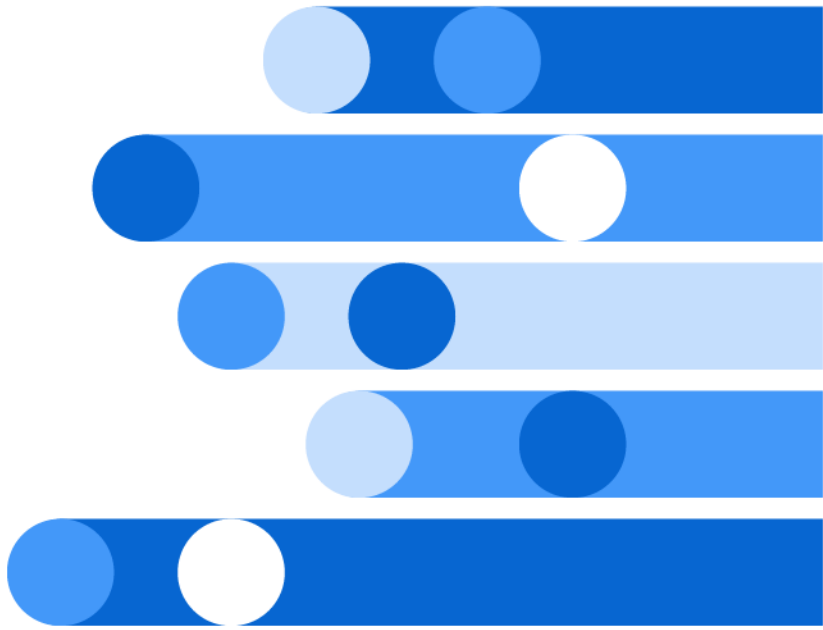
Base SAS®プロシジャガイド

2021.1.6 - 2026.03*



* このドキュメントは、ソフトウェアの追加バージョンに適用される場合があります。このドキュメントを SAS Help Center で開き、バナーのバージョンをクリックすると、使用できるすべてのバージョンが表示されます。

SAS® ドキュメント
2026 年 4 月 14 日

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2021. *Base SAS® プロシジャガイド*. Cary, NC: SAS Institute Inc.

Base SAS®プロシジャガイド

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

March 2026

v_002-P1:proc

目次

SAS 言語の構文規則

SAS 言語の構文規則の概要

SAS 言語要素の構文のドキュメントでは、標準の規則が使用されています。これらの規則により、SAS 構文の構成要素を簡単に識別できます。規則は、次の項目に分類されます。

- 構文の構成要素
- 書体に関する規則
- 特殊文字
- SAS ライブラリや外部ファイルへの参照

構文の構成要素

ほとんどの言語要素の構文のコンポーネントには、キーワードと引数があります。キーワードのみ必要な言語要素もあります。また、一部の言語要素では、キーワードの後に等号(=)を付加する必要があります。複数の引数を含む構文で区切り記号を使用する場合と使用しない場合を説明するために、引数の構文の形式が複数示されています。

keyword
プログラムを記述するときに使用する SAS 言語要素の名前を指定します。キーワードはリテラルであり、通常、構文の先頭の単語です。CALL ルーチンでは、最初の 2 つの単語がキーワードです。

これらの例の SAS 構文では、キーワードには太字が使用されています。

CHAR (*string, position*)

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

ALTER (*alter-password*)

BEST *w.*

REMOVE <*data-set-name*>

この例では、CALL ルーチンの最初の 2 つの単語がキーワードです。

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

一部の SAS ステートメントの構文は、引数のない 1 つのキーワードで構成されます。

DO;
... *SAS code* ...
END;

一部のシステムオプションでは、2 つあるキーワード値から 1 つを指定する必要があります。

DUPLEX | **NODUPLEX**

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

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

引数
数値または文字の定数、変数、式のいずれかを指定します。引数は、キーワードまたはキーワードの後の等号記号の後に続きます。SAS では、言語要素を処理するために引数が使用されます。引数は必須の場合と、省略可能な場合があります。構文では、オプションの引数は山かっこ(< >)で囲みます。

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

CHAR (*string, position*)

各引数には値があります。この例の SAS コードでは、引数 *string* の値は 'summer'、引数 *position* の値は 4 です。

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

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

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

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

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

MISSING *character(s)*;

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

たとえば、BY ステートメントはこの引数を含みます。

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

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

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

PICTURE name <(*format-options*)>
<*value-range-set-1* <(*picture-1-options*)>
<*value-range-set-2* <(*picture-2-options*)> ...>>;

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

たとえば、LABEL ステートメントは、この形式で複数の引数を含みます。

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

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

次に、この形式で指定された複数の引数の例を示します。

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

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

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

書体に関する規則

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

UPPERCASE BOLD
関数またはステートメントの名前など、SAS キーワードを示します。この例では、キーワード ERROR の表記には大文字太字が使用されています。

ERROR <*message*>;

UPPERCASE
リテラルである引数を表します。

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

CMPMODEL=BOTH | CATALOG | XML |

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

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

  LINK *label*;

- 引数に割り当てられる非リテラル値。

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

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

特殊文字

SAS 言語要素の構文には、次の特殊文字を使用できます。

=
等号記号は、システムオプションなどの一部の言語要素内のリテラルの値を示します。

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

MAPS=*location-of-maps*

< >
山かっこは省略可能な引数を示します。必須引数は山かっこで囲みません。

この CAT 関数の例では、少なくとも項目が 1 つ必要です。

CAT (*item-1* <, *item-2*, ...>)

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

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

CMPMODEL=BOTH | CATALOG | XML

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

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

CAT (*item-1 <, item-2, ...>*)

'*value*' or "*value*"
単一引用符や二重引用符付きの引数は、その値にも単一引用符または二重引用符を付ける必要があることを示します。

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

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

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

この例では、各ステートメントがセミコロンで終了しています。

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

SAS ライブラリと外部ファイルへの参照

SAS ステートメントおよびその他の言語要素の多くは、SAS ライブラリと外部ファイルを参照します。論理名(ライブラリ参照名またはファイル参照名)から参照を作成するのか、引用符付きの物理ファイル名を使用するかを選択できます。

論理名を使用する場合、通常、参照の作成に SAS ステートメント(LIBNAME または FILENAME)を使用するのか、動作環境のコントロール言語を使用するのかを選択します。SAS ライブラリと外部ファイルを参照する方法は複数あり、一部の方法は動作環境によって異なります。

SAS ドキュメント中の外部ファイルを使用する例では、*file-specification* という語句を斜体で使用しています。また、SAS ライブラリを使用する例には斜体フレーズ *SAS-library* を引用符で囲んで使用します。

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

1 部 コンセプト

1

適切なプロシジャの選択

Base SAS プロシジャの関数カテゴリ

レポート作成

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

FREQ[1]	QDEVICE[1]	SUMMARY[1]
MEANS[1]	REPORT[1]	TABULATE[1]
PRINT	SQL[1]	

1 これらのプロシジャによって、レポートが作成され、統計が計算されます。

統計

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

CORR	REPORT	TABULATE
FREQ	SQL	UNIVARIATE
MEANS	STANDARD	
RANK	SUMMARY	

ユーティリティ

これらのプロシジャによって、次の基本ユーティリティ 操作が実行されます。

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

APPEND	FCMP	OPTSAVE
CATALOG	FEDSQL	PRINTTO
CIMPORT	FONTREG	PWENCODE
COMPARE	GROOVY	REGISTRY
CONTENTS	HADOOP	SCAPROC
COPY	IMPORT[3]	SORT
CPORT	JAVAINFO	SOURCE
DATASETS	JSON	SQL[1]

DELETE	MIGRATE	TEMPLATE
DS2	OPTIONS	TRANSPOSE[1]
EXPORT[3]	OPTLOAD	TRANSTAB[2]

1 このプロシジャの説明については、ご利用の動作環境に応じた SAS ドキュメントを参照してください。
2 このプロシジャの説明については、*SAS Output Delivery System: ユーザーガイド*を参照してください。
3 このプロシジャの説明については、*SAS/ACCESS Interface to PC Files: SAS Viya プラットフォームリファレンス*を参照してください。

レポート作成プロシジャ

次のテーブルには、レポートの種類に応じたレポート作成プロシジャが一覧表示されています。

表 1.1 タスク別レポート作成プロシジャ

レポートの種類	プロシジャ	説明
詳細レポート	PRINT	データのリストをすばやく作成できます。タイトル、フットノートおよび列の合計も設定できます。
	REPORT	PROC PRINT よりも詳細に制御およびカスタマイズを行えます。列と行の両方の合計も計算でき、DATA ステップ計算機能も使用できます。
	SQL	SQL(Structured Query Language)と、形式設定などの SAS 機能が組み合わされます。データを操作し、レポートの作成と同じステップで SAS データセットを作成できます。列と行の統計を作成できます。PROC PRINT や PROC REPORT などの出力制御機能は提供されません。
要約レポート	MEANS または SUMMARY	数値変数の記述統計を計算します。印刷されたレポートを生成し、出力データセットを作成できます。
	PRINT	要約レポートを 1 つのみ作成します。BY 変数を合計できます。
	QDEVICE	グラフィックデバイスとユニバーサルプリンターに関するレポートを作成します。

レポートの種類	プロシジャ	説明
	REPORT	PRINT、MEANS および TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートを作成できる単一のレポート作成ツールに組み込みます。出力データセットも作成できます。
	SQL	1 つ以上の SAS データセットまたは DBMS テーブルに対して記述統計を計算します。印刷されたレポートを作成したり、SAS データセットを作成したりすることができます。
	TABULATE	記述統計を表形式で生成します。スタブアンドバナーレポート(記述統計が含まれた多次元表)を作成できます。出力データセットも作成できます。
詳細に形式設定された多種多様なレポート		
マルチパネルレポート(電話帳リスト)	REPORT	マルチパネルレポートを生成します。

統計プロシジャ

利用可能な統計プロシジャ

次のテーブルには、タスクに応じた統計プロシジャが一覧表示されています。表 A1.1 (2411 ページ)に、最も一般的な統計量と、それらを計算するプロシジャを表示します。

表 1.2　タスク別基本統計量プロシジャ

レポートの種類	プロシジャ	説明
記述統計	CORR	単純な記述統計を計算します。
	MEANS または SUMMARY	記述統計を計算します。印刷された出力および出力データセットを生成できますデフォルトで、PROC MEANS は印刷された出力を生成し、PROC SUMMARY は出力データセットを作成します。

レポートの種類	プロシジャ	説明
	REPORT	PROC TABULATE と大部分が同じである統計値を計算します。出力形式をカスタマイズできます。
	SQL	1 つ以上の DBMS テーブルのデータに関する記述統計値を計算します。印刷されたレポートを生成したり、SAS データセットを作成したりすることができます。
	TABULATE	記述統計に関する表レポートを生成します。出力データセットを作成できます。
	UNIVARIATE	最も広範囲にわたって一連の記述統計値を計算します。出力データセットを作成できます。
度数およびクロス集計表	FREQ	1 元から *n* 元の表を作成します。度数カウントをレポートします。カイ二乗検定を計算します。2 元から *n* 元のクロス集計表に対する関連性および一致の検定および測定値を計算します。正確検定および漸近検定を計算できます。出力データセットを作成できます。
	TABULATE	1 元および 2 元クロス集計表を作成します。出力データセットを作成できます。
	UNIVARIATE	1 元度数表を生成します。
相関分析	CORR	ピアソン、スピアマン、ケンドルの相関および偏相関を計算します。また、ヘフディングの従属(D)のメジャーおよびクローンバックのアルファ係数も計算します。
分布分析	UNIVARIATE	位置の検定と正規性の検定を計算します。
	FREQ	1 元表の二項比率の検定を計算します。1 元表の適合度検定を計算します。2 元表の等分布のカイ 2 乗検定を計算します。
ロバスト推定	UNIVARIATE	スケール、トリム平均、ウィンザー化平均のロバスト推定を計算します。
データ変換		
ランクの計算	RANK	SAS データセットのオブザベーションに対して 1 つ以上の変数のランクを計算し、出力データセットを作成します。正規スコアまたはその他のランクスコアを生成できます。

レポートの種類	プロシジャ	説明
データの標準化	STANDARD	所定の平均および標準偏差に標準化された変数を含む出力データセットを作成します。
低解像度グラフィック [1]		
	UNIVARIATE	幹葉図、箱ひげ図、正規確率プロットなどの記述プロットを作成します。

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

効率に関する問題

分位点

大規模なサンプルサイズ *n* の場合、中央値などの分位点の計算には *n*log(*n*)と比例する計算時間を要します。そのため、分位数を自動的に計算する UNIVARIATE などのプロシジャには、その他のデータ要約プロシジャよりも多くの時間が必要になることがあります。さらに、データはメモリに保持されるため、プロシジャには計算を実行するための記憶領域もさらに必要になります。デフォルトでは、レポートプロシジャ PROC MEANS、PROC SUMMARY、PROC TABULATE は分位数を自動的に計算しないため、必要なメモリは少なくなります。これらのプロシジャでは、新しい固定メモリ、通常はそれほどメモリ集中型ではない分位数推定方法を使用するオプションも使用できます。詳細については、PROC MEANS ドキュメントの“分位点” (1288 ページ)を参照してください。

オブザベーショングループの統計の計算

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

統計プロシジャに関する追加情報

付録 1, “基本的な SAS 統計プロシジャ” (2409 ページ)には、Base SAS プロシジャが頻繁に計算する統計に関する標準キーワード、統計的表記および式が一覧表示されています。個々の統計プロシジャのセクションには、プロシジャ出力の解釈に役立つ統計値の概念が説明されています。

ユーティリティプロシジャ

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

表 1.3 タスク別ユーティリティプロシジャ

タスク	プロシジャ	説明
情報を提供する	COMPARE	2 つの SAS データセットのコンテンツを比較します。
	CONTENTS	SAS ライブラリまたは特有ライブラリメンバーのコンテンツを説明します。
	JAVAINFO	SAS が使用している Java 環境に関する診断情報を通知します。
	OPTIONS	すべての SAS システムオプションの現在の値をリストします。
	SCAPROC	SAS コードアナライザーを実装します。これは、実行中に SAS ジョブからの入力、出力、マクロシンボルの使用に関する情報を取得します。
	SQL	個々の SAS データセット、および現在の SAS セッションでアクティブなすべての SAS ファイルのディクショナリテーブルによって情報を提供します。ディクショナリテーブルでは、マクロ、タイトル、インデックス、外部ファイルまたは SAS システムオプションに関する情報も提供できます。
SAS システムオプションを管理する	OPTIONS	すべての SAS システムオプションの現在の値をリストします。

タスク	プロシジャ	説明
	OPTLOAD	SAS レジストリや SAS データセットに保存されている SAS システムオプション設定を読み取ります。
	OPTSAVE	SAS システムオプション設定を SAS レジストリまたは SAS データセットに保存します。
	FONTREG	SAS レジストリにシステムフォントを追加します。
	FORMAT	データを表示および印刷するユーザー定義の出力形式を作成します。
	PRINTTO	プロシジャ出力をファイル、SAS カタログエントリ、プリンターに送ります。SAS ログをファイルにリダイレクトすることもできます。
	TEMPLATE[2]	ODS 出力をカスタマイズします。
データの作成、参照、編集	FCMP	SAS 関数およびサブルーチンを作成、検定および保存してから、他の SAS プロシジャで使用できます。
	SQL	Structured Query Language と SAS 機能を使用して SAS データセットを作成します。
	FORMAT	データを読み込むユーザー定義の入力形式と、データを表示するユーザー定義の出力形式を作成します。
	SORT	SAS データセットを 1 つ以上の変数によって並べ替えます。
	SQL	SAS データセットを 1 つ以上の変数によって並べ替えます。
	STREAM	SAS マクロ指定を含むことができる任意のテキストで構成される入力ストリームを処理できます。マクロコードを拡張してファイルに保存できます。
	TRANSPOSE	オブザベーションが変数になり、変数がオブザベーションになるように SAS データセットを変換します。
	TRANTAB[3]	カスタマイズされた変換テーブルを作成、編集、表示します。
	XSL	XML ドキュメントを HTML、テキスト、または他の種類の XML ドキュメントなどの別の出力形式に変換します。

タスク	プロシジャ	説明
SAS ファイルを管理する	APPEND	１つの SAS データセットを別のデータセットの最後に追加します。
	CATALOG	SAS カタログエントリを管理します。
	CIMPORT	PROC CPORT が(通常は別の動作環境で)作成する移送順次ファイルを、元の形式に SAS カタログ、SAS データセット、SAS ライブラリとしてリストアします。
	CONVERT[1]	BMDP システムファイル、OSIRIS システムファイル、SPSS ポータブルファイルを SAS データセットに変換します。
	COPY	SAS ライブラリまたはライブラリの特定メンバーをコピーします。
	CPORT	SAS カタログ、SAS データセットまたは SAS ライブラリを PROC CIMPORT がその元の形式にリストア可能な移送順次ファイルに(通常は別の動作環境で)変換します。
	DATASETS	SAS ファイルを管理します。
	DELETE	永続または一時 SAS ファイルを削除します。
	EXPORT[4]	データを SAS データセットから読み込み、外部データソースに書き込みます。
	IMPORT[4]	データを外部データソースから読み込み、SAS データセットに書き込みます。
	JSON	データを SAS データセットから読み込み、JSON 表現で外部データソースに書き込みます。
	MIGRATE	SAS ライブラリのメンバーを SAS の最新リリースに移行します。
	PROTO	C または C++プログラミング言語で記述される外部関数のバッチモードでの登録を行えるようにします。
	REGISTRY	レジストリ情報を SAS レジストリの USER 部分にインポートします。
	RELEASE[1]	z/OS 環境のディスクデータセットの最後の未使用スペースを 解放します。

タスク	プロシジャ	説明
	SOURCE[1]	ソースライブラリデータセットを簡単にバックアップして処理できます。
	SQL	SAS データセットを連結します。
	DS2	Base SAS セッションから DS2 言語ステートメントをサブミットします。
	FEDSQL	Base SAS セッションから FedSQL 言語ステートメントをサブミットします。
	GROOVY	Java 仮想マシン(JVM)で、SAS コードによって Groovy コードを実行できるようにします。
	HADOOP	SAS が Hadoop データに対して Apache Hadoop コードを実行できるようにします。
	PWENCODE	SAS プログラムで使用するパスワードをエンコードします。

1 これらプロシジャの説明については、ご利用の動作環境に対応する SAS ドキュメントを参照してください。
2 このプロシジャの説明については、*SAS Output Delivery System: ユーザーガイド*を参照してください。
3 このプロシジャの説明については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。
4 このプロシジャの説明については、*SAS/ACCESS Interface to PC Files: SAS Viya プラットフォームリファレンス*を参照してください。

Base SAS プロシジャの簡単な説明

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

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

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

プロシジャに加えて、PROC CPORT でも SAS ライブラリ、データセット、カタログを動作環境間で移動できます。

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

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

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

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

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

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

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

DOCUMENT プロシジャ
ODS 文書に保存されるプロシジャ出力を操作します。PROC DOCUMENT を使用すると、出力オブジェクトおよび階層の参照や編集を行ったり、任意のサポート対象 ODS 出力形式に再生したりすることができます。詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

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

DSTODS2 プロシジャ
SAS DATA ステップコードのサブセットを DS2 コードに変換できます。次に、プログラムを改訂して DS2 の機能を利用し、PROC DS2 を使用してプログラムをサブミットすることができます。

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

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

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

FMTC2ITM プロシジャ
1 つまたは複数のフォーマットカタログを、CAS が使用できる単一のアイテムストアに変換します。

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

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

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

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

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

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

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

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

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

LUA プロシジャ
SAS コード内で Lua プログラミング言語からステートメントを実行できるようにします。Lua ステートメントを外部 Lua スクリプトからサブミットしたり、SAS コードに直接入力したりできます。

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SCOREACCEL プロシジャ
DATA ステップおよび DS2 モデルのパブリッシュとスコアリングのために CAS サーバーへのインターフェイスを提供します。

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

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

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

SQOOP プロシジャ
データベースと HDFS の間でデータ転送を可能にするオプションを使って、Apache Sqoop へのアクセスを可能にします。

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

STREAM プロシジャ
SAS マクロ指定を含めることができる任意のテキストで構成された入力ストリームを処理できます。マクロコードを拡張してファイルに保存できます。マクロコードを拡張してファイルに保存できます。

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

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

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

TRANSPOSE プロシジャ
: データセットを転置し、オブザベーションは変数に、変数はオブザベーションに変換します。

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

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

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

2

Base SAS プロシジャの使用に必要な基本概念

言語の概念

一時 SAS データセットと永久 SAS データセット

SAS データセット名の指定

SAS データセットには、1 レベル名または 2 レベル名を指定できます。通常、一時 SAS データセットの名前は 1 レベルのみで、WORK ライブラリに保存されます。WORK ライブラリは SAS セッションの開始時に自動的に定義され、SAS セッションの終了時に自動的に削除されます。プロシジャでは、1 レベルの名前で指定される SAS データセットの読み取りと書き込みが WORK ライブラリに対して実行されるとみなされます。その他の場合を示すには、USER ライブラリを指定します。詳細については、“USER ライブラリ” (21 ページ)を参照してください。たとえば、次の PROC PRINT ステップは同等です。2 番目の PROC PRINT ステップは、DEBATE データセットが WORK ライブラリにあることを前提としています。

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

proc print data=debate;
run;
```

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

通常、2 レベルの名前は、永久 SAS データセットを表します。2 レベルの名前の形式は、*libref.SAS-data-set* となります。*libref* は、一時的に SAS ライブラリと関連付けられている名前です。SAS ライブラリは、1 つの単位として参照および保存される、SAS データセットを含む 1 つまたは複数の SAS ファイルのコレクションです。ディレクトリベースの環境では、SAS ライブラリは、同じフォルダーまたはディレクトリに格納されている SAS ファイルのグループです。LIBNAME ステートメントは、ライブラリ参照名と SAS ライブラリを関連付けます。次の PROC PRINT ステップでは、PROCLIB はライブラリ参照名で、EMP はライブラリ内の SAS データセットです。

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

USER ライブラリ

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

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

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

SAS システムオプション

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

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

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

データセットオプション

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

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

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

SAS データセットオプションは、次のとおりです。

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

KEEP=	WRITE=
LABEL=	

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

複数のプロシジャで同じ機能を提供するステートメント

多数の Base SAS プロシジャで、複数の Base ステートメントに同じ機能があります。一部のステートメントについては *SAS DATA ステップステートメント: リファレンス*にすべて記載され、その他は特定のプロシジャに記載されています。

グローバルステートメント

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

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

前述のステートメントで ODS ステートメント以外のものについては、SAS グローバルステートメント: リファレンスを参照してください。ODS ステートメントについては、“Output Delivery System” (71 ページ)および *SAS Output Delivery System: ユーザーガイド*.を参照してください。

AES 暗号化

ENCRYPT=AES2 データセットオプションで指定されたより強力な AES キー生成アルゴリズムがサポートされています。このより強力なアルゴリズムは、SAS の顧客によって要求される、より新しい基準を満たしています。ENCRYPTKEY=データセットオプションで指定された同じキー値パスフレーズは、どちらのアルゴリズムでも使用できます。

AES 暗号化で作成されたデータセットにアクセスするには、暗号化キー値に ENCRYPTKEY=オプションを付与してください。AES 保護データセットへのアクセス時に ENCRYPTKEY=キー値を省略すると、ダイアログボックスが表示され、ENCRYPTKEY=キー値の追加を求められます。詳細については、"暗号化" (*SAS プログラマガイド: エッセンシャル*)および "ENCRYPTKEY= データセットオプション" (*SAS データセットオプション: リファレンス*)を参照してください。

プロシジャの概念

入力データセット

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

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

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

Base SAS プロシジャのスレッド処理

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

計算された統計は、オブザベーションが処理される順序に応じてわずかに異なる場合があります。このような変動は、浮動小数点演算によって導入される数値エラーによるものであり、その結果は概算であり、正確ではないと見なす必要があります。オブザベーション処理の順序は、マルチスレッド処理または並列処理の非決定論的影響を受ける可能性があります。処理の順序は、ACCESS エンジンを介してクエリ結果を提供する DBMS などのデータソースによって 生成されるオブザベーションの一貫性のない、または非決定論的な並べ替えによっても 影響を受ける可能性があります。詳細については、“Base SAS のスレッド処理” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

次の Base SAS プロシジャはスレッド処理をサポートしています。

- 18 章, “DS2 プロシジャ,” (595 ページ)
- 23 章, “FEDSQL プロシジャ,” (789 ページ)
- 35 章, “MEANS プロシジャ,” (1221 ページ)
- 51 章, “REPORT プロシジャ,” (1741 ページ)
- 52 章, “S3 プロシジャ,” (1909 ページ)
- 55 章, “SORT プロシジャ,” (2027 ページ)
- 60 章, “SUMMARY プロシジャ,” (2141 ページ)
- 61 章, “TABULATE プロシジャ,” (2193 ページ)
- “SQL プロシジャ” (*SAS SQL プロシジャユーザーガイド*)

関連項目

システムオプション

- “CPUCOUNT= システムオプション” (*SAS システムオプション: リファレンス*)
- “THREADS システムオプション” (*SAS システムオプション: リファレンス*)

その他のドキュメント

- 並列処理のサポート
- *SAS Scalable Performance Data Server: ユーザーガイド*

データ値の順序の制御

データ値の並べ替え

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

- 動作環境固有の照合順序

- BY ステートメント
- 単一の分類変数
- 複数の分類変数
- 出力形式
- ORDER=オプション
- その他の並べ替えオプション

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

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

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

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

図 2.1 Sasuser.Houses データセット

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

BY ステートメントを使用したデータの並べ替え

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

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

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

図 2.2　2 つの BY グループのうち昇順で最初のグループ

The FREQ Procedure

style=CONDO

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

図 2.3　2 つの BY グループのうち昇順で 2 番目のグループ

The FREQ Procedure

style=RANCH

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

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

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

図 2.4　2 つの BY グループのうち降順で最初のグループ

The FREQ Procedure

style=TWOSTORY

bedrooms	Frequency	Cumulative Frequency
2	2	2
4	2	4

図 2.5　2 つの BY グループのうち降順で 2 番目のグループ

The FREQ Procedure

style=SPLIT

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

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

単一の分類変数を使用したデータの並べ替え

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

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

プロシジャ	ステートメント
FREQ	TABLES
MEANS	CLASS

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

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

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

図 2.6　単一の分類変数のデフォルトの昇順

The MEANS Procedure

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

PROC REPORT のデフォルトのデータ並べ替え動作については、“ORDER=オプションを使用したデータの並べ替え” (42 ページ)を参照してください。

複数の分類変数を使用したデータの並べ替え

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

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

プロシジャ	上位変数を指定するステートメント	例
FREQ	TABLES	proc freq; tables A * B * C * D; run;
MEANS	CLASS	proc means; class A B C D; run;

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

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

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

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

次の例を考えます。

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

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

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

図 2.7　ORDER=オプションを使用しない場合のメインの順序

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

次の例では、順序で度数カウントも考慮する場合を考えます。

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

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

次に、bedrooms のメインの順序が決定されます。bedrooms=2 の度数は 5、bedrooms=4 の度数は 4、bedrooms=3 の度数は 4、bedrooms=1 の度数は 2 です。bedrooms の順序は 2、4、3、1 です。出力は次のとおりです。

図 2.8　順序を使用した値の順序

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

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

図 2.9　度数カウントによる順序

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

フォーマットを使用したデータの順序: プロシジャのデフォルトの動作

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

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

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

プロシジャ	変数の種類	出力形式の指定	並べ替えの基準
FREQ	数値または文字	有りまたは無し	実際の値

プロシジャ	変数の種類	出力形式の指定	並べ替えの基準
MEANS	数値または文字	有りまたは無し	実際の値
REPORT	数値	有りまたは無し	フォーマットされた値 [1]
	文字	有り 無し	フォーマットされた値 実際の値
SUMMARY	数値または文字	有りまたは無し	実際の値
TABULATE	数値または文字	有りまたは無し	実際の値

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

フォーマットを使用したデータの順序: 最小の実際値の使用

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

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

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

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

図 2.10 出力形式範囲の最小値による並べ替え

The SAS System

bedrooms
GROUP B
GROUP A

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

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

```
data sample;

  length dept $ 5;
  input dept id;
  datalines;
PET   100
PET   110
PET   120
PET   199
PLANT 200
PLANT 210
PLANT 220
PLANT 299
;

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

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

図 2.11 実際の値の最小値による並べ替え

The SAS System

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

出力形式を使用したデータの並べ替え: 欠損値

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

1 出力形式グループ全体が無効として処理されます。

2 出力形式範囲全体が内部値の最小値となり、その結果、その出力形式範囲が並べ替えスキームで最初に表示される可能性があります。

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

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

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

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

data houses;
```

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

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

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

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

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

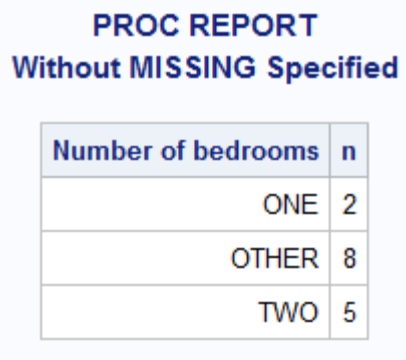
PROC PRINT
WORK.HOUSES

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

PROC FREQ
Without MISSING Specified

The FREQ Procedure

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



PROC REPORT
Without MISSING Specified

Number of bedrooms	n
ONE	2
OTHER	8
TWO	5

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

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

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

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

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

PROC FREQ
With MISSING Specified

The FREQ Procedure

bedrooms	Frequency
OTHER	9
ONE	2
TWO	5

PROC REPORT
With MISSING Specified

bedrooms	n
ONE	2
OTHER	9
TWO	5

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

出力形式を使用したデータの並べ替え: BY 変数

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

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

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

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

```
run;
```

アウトプット 2.3　フォーマットされた BY 変数を使用した並べ替え

ORDER=オプションを使用したデータの並べ替え

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

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

次のプロシジャでオプションを使用できます。

プロシジャ	SAS 製品	デフォルト値
CATMOD	SAS/STAT	INTERNAL
FREQ	Base SAS	INTERNAL
GLM	SAS/STAT	FORMATTED

プロシジャ	SAS 製品	デフォルト値
LIFEREG	SAS/STAT	FORMATTED
LOGISTIC	SAS/STAT	FORMATTED
MEANS	Base SAS	UNFORMATTED
PROBIT	SAS/STAT	FORMATTED
REPORT	Base SAS	FORMATTED
SUMMARY	Base SAS	UNFORMATTED
TABULATE	Base SAS	UNFORMATTED

次のトピックで ORDER=の各値について説明します。

ORDER=INTERNAL オプションを使用したデータの並べ替え

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

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

アウトプット 2.4　ORDER=INTERNAL の場合の出力

The SAS System

Number of bedrooms
ONE
TWO
THREE
FOUR

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

ORDER=FORMATTED オプションを使用したデータの並べ替え

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

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

アウトプット 2.5　ORDER=FORMATTED の場合の出力

The SAS System

The FREQ Procedure

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

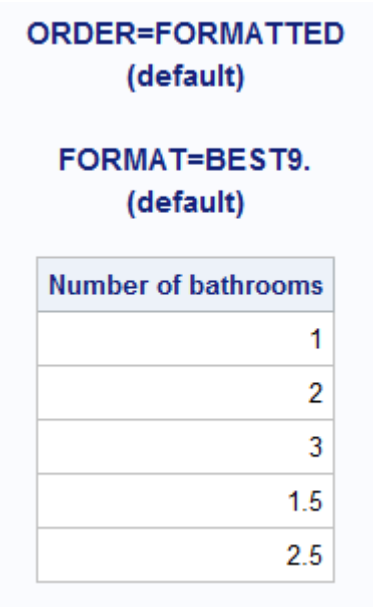
ORDER=FORMATTED を指定する場合、分類変数に関連付けられた出力形式がないと、その出力は内部値、すなわちフォーマットされていない値で並べられます。

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

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

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

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



ORDER=FORMATTED
(default)

FORMAT=BEST9.
(default)

Number of bathrooms
1
2
3
1.5
2.5

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

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

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

ORDER=FORMATTED
(default)

FORMAT=3.1
(specified)

Number of bathrooms
1.0
1.5
2.0
2.5
3.0

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

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

アウトプット 2.8　内部値に基づいた PROC FORMAT の出力

ORDER =INTERNAL
(specified)

FORMAT=BEST9.
(default)

Number of bathrooms
1
1.5
2
2.5
3

ORDER=DATA オプションを使用したデータの並べ替え

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

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

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

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

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

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

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

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

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

比較すると、Style の順序は RANCH、SPLIT、CONDO、TWOSTORY です。Bedrooms の順序は 2、1、4、3 です。Bedrooms の 4 つの値をすべて持つ STYLE の値がないため、これは出力からすぐにはわかりません。値の順序は、変数ごとに独立に作成され、サブグループには依存しません。このことは、TABLE ステートメントに PRINTMISS オプションを追加するか、Sasuser.Houses データセットの Style と Bedrooms の順序を比較することによって確認できます。“複数の分類変数を使用したデータの並べ替え” (30 ページ)を参照してください。

ORDER=FREQ オプションを使用したデータの並べ替え

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

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

PROC FREQ で度数カウントの重複が発生し、ORDER=FREQ が指定されていた場合、PROC FREQ では第 2 の並べ替えメソッドとして ORDER=FORMATTED を使用します。PROC FREQ で出力形式が適用されていなかった場合、ORDER=INTERNAL メソ

ッドを使用して同数の順序が決定されます。PROC REPORT を ORDER=FREQ および DESCENDING オプションとともに使用する場合、同数が発生すると、ORDER=DATA メソッドが使用されますが、データの発生とは逆順に分類水準が表示されます。

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

プロシジャ	並べ替え方向	出力形式	同数の場合の値の順序
FREQ	降順	いいえ	ORDER=INTERNAL
		はい	ORDER=FORMATTED
MEANS	降順	有りまたは無し	ORDER=DATA
REPORT	昇順	有りまたは無し	ORDER=DATA
	降順	有りまたは無し	ORDER=DATA[1]
SUMMARY	降順	有りまたは無し	ORDER=DATA
TABULATE	降順	有りまたは無し	ORDER=DATA

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

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

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

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

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

どちらの出力も、データ値を度数カウントの降順で表示します。

アウトプット 2.11　ORDER=FREQ の例

PROC TABULATE
Without Format

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

PROC FREQ
Without Format

The FREQ Procedure

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

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

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

アウトプット 2.12　出力形式を指定しない場合の PROC REPORT

PROC REPORT
Without Format
Without DESCENDING

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

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

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

PROC REPORT
Without Format
With DESCENDING

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

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

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

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

アウトプット 2.13　降順の出力例

PROC TABULATE
With Format

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

PROC REPORT
With Format
Without DESCENDING

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

PROC FREQ では、出力形式が指定されており、値が同一の場合には ORDER=FORMATTED を使用します。

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

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

PROC FREQ
With Format

The FREQ Procedure

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

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

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

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

PROC REPORT
With Format
With DESCENDING

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

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

次の例では、Baths の度数を検討します。

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

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

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

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

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

N 統計量では、組み合わせの度数は降順に表示されません。まず BEDROOMS の度数の降順で順序が設定され、次に BATHS で設定されます。これを確認するには、各変数の順序とアウトプット 2.11 (50 ページ)で使用されている順序を比較します。BEDROOMS の並べ替え順序は 2、4、3、1 です。BATHS の並べ替え順序は 1、3、2.5、1.5、2 です。

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

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

RUN グループ処理

RUN グループ処理によって、プロシジャを終了せずに、PROC ステップを RUN ステートメントと一緒にサブミットできます。別の PROC ステートメントを発行せずにプロシジャを続けて使用できます。プロシジャを終了するには、RUN CANCEL ステートメントか、QUIT ステートメントを使用します。RUN グループ処理は、複数の Base SAS プロシジャでサポートされています。

CATALOG DATASETS DS2 FEDSQL TRANTAB

詳細については、個別のプロシジャのセクションを参照してください。

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

BY グループの情報を含むタイトルの作成

BY グループ処理

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

デフォルトの BY 行を表示しない

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

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

- MEANS
- PRINT
- STANDARD
- SUMMARY

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

BY グループ情報をタイトルに挿入する

BY グループ情報をタイトルに挿入するための一般形式は、次のとおりです。

#BY-specification<.suffix>

BY-specification は、次のいずれかの指定です。

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

n
BY ステートメントの *n* 番目の BY 変数です。

BY-variable
値をタイトルに挿入する BY 変数の名前です。

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

n
BY ステートメントの *n* 番目の BY 変数です。

BY-variable
値をタイトルに挿入する BY 変数の名前です。

BYLINE
完全なデフォルトの BY 行をタイトルに挿入します。

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

例: タイトルに各 BY 変数の値を挿入する

この例では、次のアクションを説明します。

1 4 つの地域からの店舗に関するデータを含むデータセット GROC を作成します。店舗にはそれぞれ 4 つの部門があります。データセットを作成する DATA ステップについては、“GROC” (2491 ページ)を参照してください。

2 Region および Department 別にデータを並べ替えます。

3 SAS システムオプション NOBYLINE を使用して、BY グループ処理で生成される出力に通常表示される BY 行を非表示にします。

4 PROC REPORT を使用して、売上のチャートを Region および Department 別に作成します。最初の TITLE ステートメントで、#BYVAL2 は 2 番目の BY 変数、Department の値をタイトルに挿入します。2 番目の TITLE ステートメントで、#BYVAL(Region)は Region の値をタイトルに挿入します。Region の後の最初のピリオドは、接尾辞が続くことを示します。2 番目のピリオドは接尾辞です。

5 SAS システムオプション BYLINE を使用して、BY グループ処理によるデフォルトの BY 行の作成に返します。

```
data groc;                                1
  input Region $9. Manager $ Department $ Sales;
  datalines;
Southeast   Hayes     Paper     250
Southeast   Hayes     Produce    100
Southeast   Hayes     Canned     120
Southeast   Hayes     Meat       80
...more lines of data...
Northeast   Fuller    Paper     200
Northeast   Fuller    Produce    300
Northeast   Fuller    Canned     420
Northeast   Fuller    Meat      125
;
proc sort data=groc;                      2
  by region department;
run;
options nobyline nodate pageno=1
      linesize=64 pagesize=20;            3
proc report data=groc;                    4
  by region department;
  title1 'This chart shows #byval2 sales';
  title2 'in the #byval(region)..';
run;
options byline;                           5
```

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

This report shows Canned sales
in the Northeast.

Region	Manager	Department	Sales
Northeast	Fuller	Canned	420

This report shows Meat sales
in the Northeast.

Region	Manager	Department	Sales
Northeast	Fuller	Meat	125

This report shows Paper sales
in the Northeast.

Region	Manager	Department	Sales
Northeast	Fuller	Paper	200

...more...

例: タイトルに BY 変数の名前を挿入する

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

1 SAS システムオプション NOBYLINE を使用して、BY グループ処理で生成される出力に通常表示される BY 行を非表示にします。

2 PROC REPORT を使用して売上のチャートを Region 別に作成します。最初の TITLE ステートメントで、#BYVAR(Region)は変数 Region の名前をタイトルに挿入します(Region にラベルがある場合、#BYVAR は名前の代わりにラベルを使用します)。接尾辞 al がラベルに追加されます。2 番目の TITLE ステートメントで、#BYVAL1 は最初の BY 変数、Region の値をタイトルに挿入します。

3 SAS システムオプション BYLINE を使用して、BY グループ処理によるデフォルトの BY 行の作成に返します。

```
options nobyline nodate pageno=1
      linesize=64 pagesize=20;          1
proc report data=groc;                  2
  by region;
  title1 '#byvar(region).al Analysis';
  title2 'for the #byval1';
run;
options byline;                         3
```

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

Regional Analysis
for the Northeast

Region	Manager	Department	Sales
Northeast	Fuller	Canned	420
Northeast	Fuller	Meat	125
Northeast	Fuller	Paper	200
Northeast	Fuller	Produce	300

Regional Analysis
for the Southeast

Region	Manager	Department	Sales
Southeast	Hayes	Canned	120
Southeast	Hayes	Meat	80
Southeast	Hayes	Paper	250
Southeast	Hayes	Produce	100

例: タイトルに完全な BY 行を挿入する

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

1 SAS システムオプション NOBYLINE を使用して、BY グループ処理で生成される出力に通常表示される BY 行を非表示にします。

2 PROC REPORT を使用して、売上のチャートを Region および Department 別に作成します。TITLE ステートメントで、#BYLINE は完全な BY 行をタイトルに挿入します。

3 SAS システムオプション BYLINE を使用して、BY グループ処理によるデフォルトの BY 行の作成に返します。

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

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

Information for Region=Northeast Department=Canned

Region	Manager	Department	Sales
Northeast	Fuller	Canned	420

Information for Region=Northeast Department=Meat

Region	Manager	Department	Sales
Northeast	Fuller	Meat	125

Information for Region=Northeast Department=Paper

Region	Manager	Department	Sales
Northeast	Fuller	Paper	200

Information for Region=Northeast Department=Produce

Region	Manager	Department	Sales
Northeast	Fuller	Produce	300

...more observations...

BY グループの指定のエラー処理

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

変数名リストを指定するショートカット

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

表記	意味
x1-x*n*	変数 X1 から X*n* を指定します。番号は連続している必要があります。

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

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

詳細については、*SAS プログラマガイド: エッセンシャル*を参照してください。

フォーマットされた値

フォーマットされた値の使用

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

例: データセットのフォーマットされた値を印刷する

次の例では、データセット PROCLIB.PAYROLL のフォーマットされた値を印刷します。(このデータセットを作成する DATA ステップの詳細については、“PROCLIB.PAYROLL” (2500 ページ)を参照してください。)PROCLIB.PAYROLL で、変数 Jobcode は従業員のジョブとレベルを示します。たとえば、**TA1** は、従業員がチケット取扱店の開始レベルであることを示します。

```
options nodate pageno=1
```

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

次の例は、PROCLIB.PAYROLL の印刷の一部です。

PROCLIB.PAYROLL
First 10 Observations Only

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

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

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

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

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

```
proc means data=proclib.payroll mean max;
  class jobcode;
  var salary;
  format jobcode $jobfmt.;
  title 'Summary Statistics for';
  title2 'Each Job Code';
run;
```

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

Summary Statistics for Each Job Code

The MEANS Procedure

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

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

例: フォーマットされたデータをグループ化/分類する

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

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

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

PROC MEANS は次の出力を生成します。

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

The MEANS Procedure

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

例: 出力形式と変数を一時的に関連付ける

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

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

PROC PRINT は次の出力を生成します。

Temporarily Associating a Format
with the Variable Salary

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

例: 出力形式と変数の関連付けを一時的に解除する

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

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

```
proc format;
```

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

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

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

Average GPA

The MEANS Procedure

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

出力形式と BY グループ処理

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

consecutive 内部値に同じフォーマットされた値が含まれている場合は、同じ BY グループに含まれます。

出力形式とエラーチェック

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

注: ユーザー作成の出力形式を検索できるようにするには、SAS システムオプション FMTSEARCH=を使用します。出力形式の保存方法については、“入力形式と出力形式の保存 ” (864 ページ)を参照してください。

ライブラリのすべてのデータセットを処理する

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

“例 10: SAS ライブラリのすべてのデータセットを印刷する” (1480 ページ)は、ライブラリのすべてのデータセットの出力方法を示しています。同じマクロ定義を使用して、ライブラリのすべてのデータセットでプロシジャを 実行できます。プログラムの PROC PRINT の部分を適切なプロシジャコードと置き換えるだけです。

統計量の説明

次のテーブルでは、複数の Base SAS プロシジャで使用可能な共通の記述統計量を識別します。利用可能な統計量に関する詳細情報と理論情報については、“キーワードと式” (2410 ページ) を参照してください。

表 2.1 Base SAS プロシジャが計算する共通記述統計量

統計量	説明	プロシジャ
信頼区間		FREQ、MEANS または SUMMARY、TABULATE、UNIVARIATE
CSS	修正済み平方和	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE

統計量	説明	プロシジャ
CV	変動係数	MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
適合度検定		FREQ、UNIVARIATE
KURTOSIS	尖度	MEANS または SUMMARY、TABULATE、UNIVARIATE
MAX	最大(最大)値	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
MEAN	Mean	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
MEDIAN	中央値(50番目のの分位数)	CORR(ノンパラメトリックな相関指標の場合)、MEANS または SUMMARY、TABULATE、UNIVARIATE
MIN	最小(最小)値	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
MODE	最も頻度の高い値(一意でない場合、最小モードが使用されます)	UNIVARIATE
N	計算の基準となるオブザベーション数	CORR、FREQ、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
NMISS	欠損値の数	FREQ、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
NOBS	オブザベーション数	MEANS または SUMMARY、UNIVARIATE
PCTN	合計度数に対するセルまたは行の度数パーセント	REPORT、TABULATE
PCTSUM	総合計に対するセルまたは行合計比	REPORT、TABULATE
Pearson 相関		CORR
パーセント点		FREQ、MEANS または SUMMARY、REPORT、TABULATE、UNIVARIATE
RANGE	範囲	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE

統計量	説明	プロシジャ
ロバスト統計量	トリム平均、ウィンザー化平均	UNIVARIATE
SKEWNESS	歪度	MEANS または SUMMARY、TABULATE、UNIVARIATE
Spearman 相関		CORR
STD	標準偏差	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
STDERR	平均の標準誤差	MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
SUM	合計	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
SUMWGT	重みの合計	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
位置の検定		UNIVARIATE
USS	無修正平方和	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE
VAR	分散	CORR、MEANS または SUMMARY、REPORT、SQL、TABULATE、UNIVARIATE

統計量の計算の必要条件

次に、表 2.1 (68 ページ)にリストされている統計量の計算要件を示します。推奨サンプルサイズは説明されていません。

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

統計量は、計算できない場合は欠損値としてレポートされます。

Output Delivery System

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

- 共通して使用されるプロシジャの一部は、出力データセットを生成しません。

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

3

Base プロシジャの In-Database 処理

In-Database 処理向けに拡張される Base プロシジャ

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

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

Base SAS プロシジャは、複数のデータソース内のデータを処理できます。詳細については、“In-Database Procedures” (*SAS In-Database Products: User's Guide*)を参照してください。

次の Base SAS プロシジャでは、In-Database 処理がサポートされています。

表 3.1 In-Database Base プロシジャ

プロシジャ	説明
PROC COPY (303 ページ)	1 つ以上のテーブルを、1 つの SAS ライブラリから別の SAS ライブラリにコピーします。
*Base SAS プロシジャガイド: 統計プロシジャ*の PROC FREQ	1 次元から *n* 次元の表を作成します。度数カウントをレポートします。2 次元から *n* 次元のクロス集計テーブルに対する関連性および一致の検定および統計量を計算します。正確検定および漸近検定を計算できます。出力データセットを作成できます。
PROC MEANS (1229 ページ)	記述統計量を計算します。印刷済出力および出力データセットを生成できます。デフォルトでは、PROC MEANS は印刷済出力を生成します。
PROC RANK (1695 ページ)	SAS データセットのオブザベーションにわたる数値変数の順位を計算します。一部の順位スコアを生成できます。
PROC REPORT (1834 ページ)	PRINT プロシジャ、MEANS プロシジャ、TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートの作成が可能な 1 つのレポート作成ツールに組み合わせます。
PROC SORT (2065 ページ)	SAS データセットオブザベーションを 1 つ以上の文字変数または数値変数の値を基準にして並べ替えます。
PROC SUMMARY (2141 ページ)	記述統計量を計算します。印刷済レポートを生成し、出力データセットを作成できます。デフォルトでは、PROC SUMMARY は出力データセットを作成します。
PROC TABULATE (2278 ページ)	データセットの一部またはすべての変数を使用して、記述統計量を表形式で表示します。

4

Base プロシジャの CAS 処理

CAS アクションを使用するプロシジャ

次の Base SAS プロシジャで CAS アクションを実行できます。

プロシジャ	説明
PROC APPEND (119 ページ)	CAS テーブルの行を SAS データセットの末尾に追加し、SAS データセットの行を CAS テーブルの末尾に追加します。
PROC CONTENTS (283 ページ)	CAS テーブルの内容を表示して、caslib のディレクトリを出力します。
PROC COPY (303 ページ)	SAS ライブラリ全体またはライブラリの特定のメンバーをコピーします。
PROC DATASETS (359 ページ)	CAS テーブルを管理します。
PROC DELETE (583 ページ)	SAS データセットと CAS テーブルを削除します。
PROC DS2[1]	DS2 言語ステートメントでデータを操作します。

プロシジャ	説明
PROC FCMP (667 ページ)	SAS 関数、CALL ルーチン、およびサブルーチンを作成、テストおよび保存してから、他の SAS プロシジャまたは DATA ステップで使用するようにします。
PROC FEDSQL[2]	FedSQL 言語ステートメントでデータを操作し、レポートを実行します。
PROC FORMAT (859 ページ)	データを読み込むユーザー定義の入力形式と、データを表示するユーザー定義の出力形式を作成します。
PROC LUA (1163 ページ)	SAS コード内で Lua プログラミング言語からステートメントを実行できるようにします。
PROC MEANS (1231 ページ)	記述統計量を計算します。印刷済出力および出力データセットを生成できます。デフォルトでは、PROC MEANS は印刷済出力を生成します。
PROC REPORT (1835 ページ)	PRINT プロシジャ、MEANS プロシジャ、TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートの作成が可能な 1 つのレポート作成ツールに組み合わせます。
PROC SORT (2027 ページ)	データセット内の重複するオブザベーションを管理する機能を提供します。SORT プロシジャは、CAS deduplicate アクションを使用して、PROC SORT の NODUPKEY または NOUNIKEY オプションと同等の機能を実行します。
PROC SCOREACCEL (1957 ページ)	DATA ステップおよび DS2 モデルのパブリッシュとスコアリングのために CAS サーバーへのインターフェイスを提供します。
PROC SUMMARY (2141 ページ)	記述統計量を計算します。印刷済レポートを生成し、出力データセットを作成できます。デフォルトでは、PROC SUMMARY は出力データセットを作成します。
PROC TABULATE (2276 ページ)	データセットの一部またはすべての変数を使用して、記述統計量を表形式で表示します。

プロシジャ	説明
PROC TRANSPOSE (2359 ページ)	オブザベーションが変数になり、変数がオブザベーションになるように SAS データセットを変換します。

1 DS2 プロシジャは、インメモリテーブルへのアクセスに CAS LIBNAME エンジンを使用しません。代わりに、プロシジャは caslib と名前によってテーブルにアクセスします。詳細と制限については、*"CAS の DS2: 概念" (SAS DS2 プログラマガイド)*を参照してください。

2 FEDSQL プロシジャは、インメモリテーブルのアクセスに CAS LIBNAME エンジンを使用しません。代わりに、プロシジャは caslib と名前によってテーブルにアクセスします。詳細と制限については、*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*を参照してください。

CAS 処理が使用できない場合

各プロシジャのドキュメントに、サポートされている集計およびサポートされているステートメントとオプションが示されています。プログラムに CAS アクションで処理できないプロシジャステートメントまたはオプションが含まれている場合、インメモリテーブルのオブザベーションレベルのデータが SAS クライアントに転送されます。次に、このプロシジャは、CAS ではなく SAS Compute Server 上の SAS に転送されたデータに対して実行されます。

デフォルトでは、CAS LIBNAME エンジンはデータ転送を 100 MB に制限します。制限に達した場合:

1 SAS Visual Statistics プロシジャまたは SAS Visual Data Mining and Machine Learning プロシジャを使用して、希望する結果を達成できる場合があります。

2 要約がサーバーによって実行されるように、CAS アクションを使用してプログラムできる場合があります。

3 CASDATALIMIT=システムオプション、DATALIMIT=LIBNAME オプション、またはデータセットオプションを使用して、制限を増やすことができます。

BY グループ処理

BY ステートメントをサポートするプロシジャの場合、情報は CAS サーバーに渡されます。これにより、2 つの最適化が可能になります。

1 データは、指定された変数で事前に並べ替える必要はありません。

2 結果がサーバーによって SAS クライアントに転送されるとき、グループはすでに形成されています。これらの結果は、PROC MEANS の場合のように結果を要約することも、PROC PRINT の場合のようにオブザベーションレベルにすることもできます。

BY グループ処理を実行することがあらかじめわかっている場合は、特にインメモリテーブルが大きい場合は、インメモリテーブルをさらに効率的に分割できます。BY グループ処理に使用する変数と同じ変数を使用してインメモリテーブルを分割すると、テーブルにアクセスするたびにグループを 作成する際のパフォーマンスの低下を回避できます。

オブザベーションのフィルタリング

WHERE ステートメントをサポートするプロシジャでは、式がサーバーに送信されます。サーバーは式を解決し、分析のためにデータをサブセット化します。これにより、処理時間とサーバーから SAS クライアントへのデータ転送が大幅に削減されます。

CAS LIBNAME エンジンで使用する場合は、WHERE=データセットオプションについても同じです。式はサーバーに送信され、データをサブセット化します。

```
cas casauto host="cloud.example.com" port=5570;
libname mycas cas sessref=casauto;

proc casutil;
  load data=sashelp.prdsale;
quit;

/* View informational messages in the SAS log.      */
options msglevel=i;

/* The WHERE statement is processed by the server.   */
proc means data=mycas.prdsale;
  where region eq 'EAST';
run;

/* These results are identical to the preceding use  */
/* of PROC MEANS, but uses a WHERE= data set option. */
proc means data=mycas.prdsale(where=(region eq 'EAST'));
run;
```

関連ドキュメント

- *SAS Cloud Analytic Services: 基本的な構成*
- *SAS Cloud Analytic Services: ユーザーガイ ド*
- *SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*
- *SAS DS2 プログラマガイド*

5

他のドキュメントで説明されている Base SAS プロシジャ

他のドキュメントで説明されている Base SAS プロシジャ

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

プロシジャ	説明	パブリケーション
CALLRFC	SAP システムのリモートファンクションコール (RFC)や RFC 互換機能を実行します。	*SAS/ACCESS Interface to R/3: ユーザーガイド*
CORR	ピアソン相関係数、アソシエーションの 3 つのノンパラメトリック測定、これらの統計量に関連する確率を計算します。	Base SAS プロシジャガイド: 統計プロシジャ
DOCUMENT	ODS ドキュメント内のプロシジャやデータベースクエリの結果出力の再配置、複製または削除を可能にします。	*SAS Output Delivery System: ユーザーガイド*
FedSQL	後続の入力が FedSQL ステートメントであることを示します。	*SAS FedSQL 言語リファレンス*

プロシジャ	説明	パブリケーション
FREQ	1 次元度数から N 次元の度数および分割(クロス集計)表を作成します。	Base SAS プロシジャガイド: 統計プロシジャ
GCHART	次の 6 種類のグラフ、 BLOCK チャート、横棒グラフと縦棒グラフ、円グラフとドーナツグラフ、スターチャートを作成します。	*SAS/GRAPH: リファレンス*
GEOCODE	国勢調査ブロックなどの地理座標(緯度と経度の値)と属性値を住所に追加します。	*SAS/GRAPH and Base SAS: マッピングリファレンス*
GINSIDE	X 座標と Y 座標のデータセットを、マップポリゴンを含むマップデータセットと比較します。各ポイントの X 座標と Y 座標が、マップポリゴンの内側にあるか外側にあるかを判別します。結果の出力マップデータセットには、マップポリゴン内のポイントが含まれます。SAS/GRAPH の GMAP プロシジャ、または Base SAS の ODS Graphics SGMAP プロシジャによる入力として使用できます。	*SAS/GRAPH and Base SAS: マッピングリファレンス*
GPLOT	ひと組の座標軸(X および Y)の 2 つ以上の変数の値をプロットします。	*SAS/GRAPH and Base SAS: マッピングリファレンス*
GPROJECT	GMAP および SGMAP プロシジャで使用するために、球面座標(経度と緯度)をデカルト座標に変換することにより、マップデータセットを処理します。	*SAS/GRAPH and Base SAS: マッピングリファレンス*
GREDUCE	マップデータセットを処理して、より少ない境界点でより単純なマップを描画できるようにします。DENSITY 変数が追加された、結果の出力マップデータセットは、SAS/GRAPH の GMAP プロシジャ、または Base SAS の ODS Graphics SGMAP プロシジャによって 入力マップデータセットとして使用できます。	*SAS/GRAPH and Base SAS: マッピングリファレンス*
GREMOVE	元のユニットエリア間の内部境界を削除することにより、マップデータセットで定義されたユニットエリアをより大きなユニットエリアにまとめます。結果の出力マップデータセットは、SAS/GRAPH の GMAP プロシジャ、または Base SAS の ODS Graphics SGMAP プロシジャによる入力として使用できます。	*SAS/GRAPH and Base SAS: マッピングリファレンス*

プロシジャ	説明	パブリケーション
MAPIMPORT	Esri シェープファイル(空間データ形式)をインポートし、SHP ファイルを SAS/GRAPH により、またはサードパーティのソースから利用できるマップデータセットに処理できます。	*SAS/GRAPH and Base SAS: マッピングリファレンス*
ODSLIST	リポートの個別コンポーネントを保存し、レポートの変更および再生を可能にします。	*SAS Output Delivery System: ユーザーガイド*
ODSTABLE	表形式出力テンプレートを作成します。	*SAS Output Delivery System: ユーザーガイド*
ODSTEXT	無限にカスタマイズやネストを実施できるパラグラフとリストを作成します。	*SAS Output Delivery System: ユーザーガイド*
SGMAP	マップエリア、マップ応答値、およびオーバーレイプロットに必要なデータセットを識別します。	*SAS ODS Graphics: プロシジャガイド*
SGPANEL	1 つ以上の分類変数の値に対するグラフセルのパネルを作成します。	*SAS ODS Graphics: プロシジャガイド*
SGPIE	プロット変数を含むデータ セットを識別します。このステートメントには、説明を指定したり、自動凡例を制御したり、グラフの背景を不透明にするか透明にするかを指定するオプションもあります。	*SAS ODS Graphics: プロシジャガイド*
SGPLOT	1 つ以上のプロットを作成し、単一の軸の組み合わせに重ね合わせます。	*SAS ODS Graphics: プロシジャガイド*
SGRENDER	Graph Template Language (GTL)で作成されたテンプレートからグラフ出力を作成します。	*SAS ODS Graphics: プロシジャガイド*
SGSCATTER	使用するプロットステートメントに応じて、複数の変数の組み合わせに対する散布図のパネルグラフを作成します。	*SAS ODS Graphics: プロシジャガイド*
SQL	SAS に SQL(Structured Query Language)を実装します。	*SAS SQL プロシジャユーザーガイド*
TEMPLATE	SAS 出力の表示のユーザー設定を可能にします。	*SAS Output Delivery System: ユーザーガイド*

プロシジャ	説明	パブリケーション
TRANTAB	ユーザー設定された変換テーブルを作成、編集、表示して、SAS で提供される変換テーブルの参照と変更を可能にします。	*SAS 各国語サポート (NLS): リファレンスガイド*
UNIVARIATE	数値変数の統計分布を要約、ビジュアル化、分析およびモデル化するためのさまざまな記述測定、グラフィカル表示および統計方法を提供します。	Base SAS プロシジャガイド: 統計プロシジャ

すべての SAS プロシジャについては、*名前と製品順の SAS プロシジャ*を参照してください。*名前と製品順の SAS プロシジャ*の情報は、プロシジャ名とその製品名のアルファベット順に並んでいます。

6

サポートが終了した Base SAS プロシジャ

SAS Viya プラットフォームでサポートされていない Base SAS プロシジャ

次の SAS 9.4 プロシジャは SAS Viya プラットフォームではサポートされていません。

表 6.1　SAS Viya プラットフォームでサポートされていない Base SAS プロシジャ

Base SAS プロシジャ
AUTHLIB
CALENDAR
CHART
DISPLAY
FORMS
FSLIST
HDMD
PLOT

Base SAS プロシジャ
PMENU
PRESENV
PRODUCT_STATUS
SOAP
TIMEPLOT

2 部

プロシジャ

7

ACCELERATOR プロシジャ

概要: ACCELERATOR プロシジャ

ACCELERATOR プロシジャの機能

PROC ACCELERATOR では、モデルのパブリッシュとスコアリングのために外部データソースへのインターフェイスを 提供する対話型プロシジャです。

- PROC ACCELERATOR は、SCOREACCEL プロシジャの代替です。この 2 つのプロシジャの違いについては、“PROC ACCELERATOR と PROC SCOREACCEL の機能比較” (94 ページ)を参照してください。
- モデルは、DATA ステップコード、DS2 コード、または分析ストアの形式でサポートされます。
- このプロシジャでは、次のデータソースに対応しています。
  - 2025.12 以降、Microsoft Azure Synapse Analytics がサポートされています。
  - 2024.11 以降、AWS または Microsoft Azure 上の Databricks がサポートされています。
  - 2025.07 以降、SingleStore がサポートされています。製品 SAS SpeedyStore のライセンスを取得して配置する必要があります。
- モデルが外部データソースにパブリッシュされた後、モデルを実行すると、SAS Embedded Process が呼び出されます。
- Databricks の場合、PROC ACCELERATOR を使用して、SAS Embedded Process 連続セッションを開始および停止します。SingleStore の場合、セッションを開始または停止することはありません。
- モデルは、DELETEMODEL ステートメントを使用して、外部データソースから削除できます。
- サポートされているスコアリングモデルとその他の情報の一覧については、“In-Database Model Scoring” (*SAS In-Database Products: User’s Guide*)を参照してください。

概念: ACCELERATOR プロシジャ

1 部、2 部、または 3 部のテーブル名の指定

テーブル名ルールの概要

データソースは、2 部または 3 部のテーブル名をサポートしています。データソースに応じて、LIBNAME 割り当てでカタログ、データベース、スキーマ、またはこれらの値の組み合わせを指定します。PROC ACCELERATOR は、LIBNAME 値を MODELTABLENAME=、INPUTTABLENAME=、および OUTPUTTABLENAME=オプションのカタログ、データベースまたはスキーマのデフォルト値として使用します。

簡単にするために、PROC ACCELERATOR オプションで 1 部構成の名前を指定し、カタログ、データベース、またはスキーマ値には LIBNAME 割り当てからデフォルト値を取得することをお勧めします。

PROC ACCELERATOR を使用すると、MODELTABLENAME=、INPUTTABLENAME=、および OUTPUTTABLENAME=オプションで 1 部、2 部、または 3 部のテーブル名を指定できます。LIBNAME 割り当てのデフォルトをオーバーライドする必要がある場合は、2 部または 3 部の名前を指定できます。たとえば、入力データとは別のデータベースに出力データを作成するとします。Synapse は例外です。Synapse では、RUNMODEL=ステートメントでデータベースをオーバーライドできますが、PUBLISHMODEL または DELETEMODEL ステートメントではデータベースをオーバーライドできません。各データソースの動作については、次のトピックで説明します。

PROC ACCELERATOR は、LIBNAME で指定された認証情報を使用して、データソースに接続します。

有効なテーブル名を生成できない場合、PROC ACCELERATOR はエラーを SAS ログに書き込みます。

すべてのデータソースで、名前にピリオド、予約語、または特殊文字が含まれている場合は、名前をバックティック(`)で囲む必要があります。次の例では、テーブル名 **my.table** にピリオドが含まれているため、その前後の目盛が必要です。

```
modeltablename="myschema.`my.table`"
```

Databricks のテーブル名

Databricks は、次の形式の 3 部構成のテーブル名をサポートしています。

```
catalog.schema.table
```

Databricks の Spark LIBNAME ステートメントでは、SCHEMA=オプションを使用し、DATABASE=オプションは使用しないでください。2025.12 以降では、CATALOG=LIBNAME オプションを使用して、Databricks Unity Catalog を指定できます。

2025.12 より前のリリースでは、JDBC ドライバープロパティを 使用してカタログを指定する必要があります。必要に応じて、JDBC ドライバープロパティを 引き続き使用してカタログを指定したり、カタログとスキーマの両方を指定したりできます。ConnCatalog=および ConnSchema=プロパティは、URI=または PROPERTIES= LIBNAME オプションでサポートされています。

PROPERTIES=オプションでカタログとスキーマを指定し、それらを CATALOG=および SCHEMA LIBNAME オプションでも指定すると、PROPERTIES=オプションの値が優先されます。

次の表は、Databricks の PROC ACCELERATOR の動作を示しています。

表 7.1 Databricks 用の 3 つの部分からなるテーブル名の作成

MODELTABLENAME=、INPUTTABLENAME=、または OUTPUTTABLENAME=オプションの値	PROC ACCELERATOR の動作
1 部構成の名前	オプション値はテーブル名として読み込みます。CATALOG= LIBNAME オプションまたは ConnCatalog=ドライバープロパティのカタログを使用します。SCHEMA= LIBNAME オプションまたは ConnSchema=ドライバープロパティからのスキーマを使用します。
2 部構成の名前	オプション値を *schema.table* として読み取ります。CATALOG= LIBNAME オプションまたは ConnCatalog=ドライバープロパティのカタログを使用します。
3 部構成の名前	値をそのまま使用します。

SingleStore のテーブル名(SAS SpeedyStore)

SingleStore は、次の形式の 2 部構成のテーブル名をサポートします。

database.table

SingleStore LIBNAME ステートメントで、DATABASE=オプションにデータベースを指定します。SingleStore はスキーマ名をサポートしていません。

次の表は、SingleStore の PROC ACCELERATOR の動作を示しています。

表 7.2 SingleStore 用の 2 つの部分からなるテーブル名の作成

MODELTABLENAME=、INPUTTABLENAME=、または OUTPUTTABLENAME=オプションの値	PROC ACCELERATOR の動作
1 部構成の名前	LIBNAME のデータベースを使用します。
2 部構成の名前	オプションの値をそのまま使用します。
3 部構成の名前	3 部構成の名前は SingleStore ではサポートされていません。SAS ログにエラーを書き込みます。

Synapse のテーブル名

Microsoft Azure Synapse Analytics は、次の形式で 3 部構成のテーブル名を必要とします。

database.schema.table

接続には、Microsoft SQL サーバーの LIBNAME ステートメントを使用します。Synapse ワークスペースの 専用の SQL プールに接続する必要があります。プールは一時停止解除する必要があります。SCHEMA=LIBNAME オプションにスキーマを指定します。NOPROMPT=オプション内で Database=接続オプションを指定する必要があります。データベース値は、専用 SQL プールの名前です。DATASRC=オプションは使用しないでください。

次の表は、Synapse の PROC ACCELERATOR の動作を示しています。モデルをパブリッシュまたは削除する場合、LIBNAME 割り当てのデータベース値を上書きすることはできません。ただし、モデルを実行する際には、MODELTABLENAME=、INPUTTABLENAME=、または OUTPUTTABLENAME=の値に別のデータベースを指定できます。

表 7.3　Synapse 用の 3 部構成のテーブル名の作成

MODELTABLENAME=、INPUTTABLENAME=、または OUTPUTTABLENAME= オプションの値	PROC ACCELERATOR の動作	
	PUBLISHMODEL または DELETEMODEL ステートメント	RUNMODEL ステートメント
1 部構成の名前	LIBNAME のスキーマとデータベースを使用します。	オプション値はテーブル名として読み込みます。LIBNAME のスキーマとデータベースを使用します。
2 部構成の名前	値を *schema.table* として使用します。LIBNAME のデータベースを使用します。	オプション値を *schema.table* として読み取ります。LIBNAME のデータベースを使用します。
3 部構成の名前	3 部構成の名前のデータベースが LIBNAME のデータベースと一致する場合に値を使用します。データベースが一致しない場合は、SAS ログにエラーを書き込みます。	オプションの値をそのまま使用します。

PROC ACCELERATOR と PROC SCOREACCEL の機能比較

PROC ACCELERATOR は 2024.11 で追加されました。次のテーブルに、引き続きサポートされている以前のプロシジャである PROC SCOREACCEL との違いをまとめます。

表 7.4 In-Database スコアリングのタスクと機能のサポート

タスクまたは機能	PROC ACCELERATOR	PROC SCOREACCEL
SAS Embedded Process を配置します。	Databricks または Synapse の場合、accelserver RPM ファイルを配置する必要があります。[1]	Databricks または Synapse の場合、sepcoresparkRPM ファイルを配置する必要があります。[1]
	SingleStore の場合、製品 SAS SpeedyStore のライセンスを取得して配置する必要があります。[2]	SingleStore の場合、製品 SAS SpeedyStore のライセンスを取得して配置する必要があります。[2]
SAS Embedded Process 連続セッションを開始または停止します。	Databricks または Synapse の場合、STARTSESSION または STOPSESSION ステートメントを使用してプロシジャをサブミットします。 SingleStore の場合、セッションを開始または停止することはありません。	Databricks または Synapse の場合、CAS アクション sparkEmbeddedProcess.startSparkEP.startSparkEP または sparkEmbeddedProcess.startSparkEP.stopSparkEP をサブミットします。 SingleStore の場合、セッションを開始または停止することはありません。
データソースへの接続を確立します。	LIBNAME ステートメントをサブミットするか、他のメソッドを使用して SAS ライブラリを割り当てます。	CASLIB ステートメントをサブミットするか、他のメソッドを使用して CAS ライブラリを割り当てます。
プロシジャ内でデータソースを参照します。	ライブラリ参照名を指定します。	casref を指定します。

タスクまたは機能	PROC ACCELERATOR	PROC SCOREACCEL
入力データまたは出力データを参照します。	テーブルは、1 部構成、2 部構成、または 3 部構成の名前として指定できます。	データソースによって 、テーブルは 1 部構成の名前または 2 部構成の名前で指定できます。
	SingleStore の場合、SELECT ステートメントをサブミットして、入力データのサブセットを参照できます。	一部のデータソースでは、SELECT ステートメントをサブミットして、入力データのサブセットを参照できます。
	Databricks または Synapse の場合、カスタムスカラーコードを使用して、入力データを読み取り、出力データを書き込むことができます。	Scala コードはサポートされていません。
CAS サーバーと対話します。	CAS との相互作用はありません。	PROC SCOREACCEL は、CAS アクションを呼び出してタスクを完了します。PROC SCOREACCEL を使用する代わりに、モデルのパブリッシュとスコアリングアクションセットを直接使用できます。
非推奨の機能	非推奨のオプションは省略されています。	非推奨のオプションが使用可能です。
サポートされているデータソース	AWS Databricks[1] Azure Databricks[1] Azure Synapse[1] SingleStore[2]	CAS AWS Databricks Azure Databricks Azure Synapse Hadoop SingleStore Teradata

1 詳細については、“Databricks と Synapse の配置の詳細” (95 ページ)を参照してください。
2 詳細については、“SingleStore の配置の詳細” (96 ページ)を参照してください。

Databricks と Synapse の配置の詳細

PROC SCOREACCEL と比較して、PROC ACCELERATOR の配置の違いに注意することが重要です。詳細は次のとおりです。

- SAS Embedded Process for Spark で追加の RPM ファイルが利用できるようになりました。新しい配置は、*accelserver* RPM ファイルと呼ばれます。以前の配置は引き続きサポートされ、*sepcorespark* RPM ファイルと呼ばれます。
- PROC ACCELERATOR は、accelserver RPM ファイルから Python ホイールをインストールした場合にサポートされます。sepcorespark RPM ファイルからの Python ホイールは PROC ACCELERATOR をサポートしていません。
- sepcorespark RPM ファイルから Python ホイールをインストールすると、PROC SCOREACCEL、Model Publishing and Scoring アクションセット、SAS Embedded Process for Spark アクションセットがサポートされます。accelserver RPM ファイルの Python ホイールは PROC SCOREACCEL をサポートしていません。
- 同じクラスターに両方の RPM をインストールすることはできません。両方の RPM の機能が必要な場合は、別々のクラスターにインストールする必要があります。

配置の情報については、*SAS Viya In-Database Technologies: 配置と管理ガイド*を参照してください。

SingleStore の配置の詳細

製品 SAS SpeedyStore のライセンスを取得して配置した場合、プロシジャは SingleStore をサポートします。このプロシジャでは、SingleStore データベースの統合インスタンスのみがサポートされます。このプロシジャは外部データベースをサポートしていません。配置の詳細については、*SAS SpeedyStore: 管理および構成ガイド*を参照してください。

構文: ACCELERATOR プロシジャ

PROC ACCELERATOR*required-argument*;
STARTSESSION*required-arguments*<*optional-arguments*>;
STOPSESSION;
PUBLISHMODEL*required-arguments*<*optional-argument*>;
RUNMODEL*required-arguments*<*optional-arguments*>;
DELETEMODEL*required-argument*;

ステートメント	タスク	例
“PROC ACCELERATOR ステートメント”	連続セッションの開始や停止、データソース内のモデルのパブリッシュ、実行、削除を行うステートメントをサブミットします	Ex. 1, Ex. 2, Ex. 3, Ex. 4

ステートメント	タスク	例
“STARTSESSION ステートメント”	SAS Embedded Process 連続セッションを開始します	Ex. 1, Ex. 2, Ex. 4
“STOPSESSION ステートメント”	SAS Embedded Process 連続セッションを停止します	Ex. 1, Ex. 2, Ex. 4
“PUBLISHMODEL ステートメント”	データソースにモデルをパブリッシュします	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“RUNMODEL ステートメント”	データソースでモデルを実行します	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DELETEMODEL ステートメント”	データソースからモデルを削除します	Ex. 2, Ex. 1, Ex. 3, Ex. 4

PROC ACCELERATOR ステートメント

連続セッションを開始または停止するか、データソース内のモデルをパブリッシュ、実行、または削除します。

構文

PROC SCOREACCELLIBRARY=*libref*;

必須引数

LIBRARY=*libref*
モデルを実行する SAS ライブラリの名前を指定します。 libref は、SAS ライブラリを参照するために使用する名前です。このオプションでライブラリ参照名を指定するには、SAS ライブラリを割り当てる必要があります。LIBNAME ステートメントは、SAS ライブラリを割り当てる一般的な方法です。詳細については、*SAS/ACCESS for Relational Databases: リファレンス*を参照してください。

STARTSESSION ステートメント

SAS Embedded Process 連続セッションを開始します。

注: SingleStore で STARTSESSION および STOPSESSION ステートメントを使用しないでください。

構文

STARTSESSION
PLATFORM=DATABRICKS | SYNAPSE
PLATFORMENDPOINT="*rest-url*"
PLATFORMPASSWORD=*endpoint-password*
PLATFORMUSER=*endpoint-user-identifier*
<*startsession-optional-arguments*>;

必須引数

PLATFORM=DATABRICKS | SYNAPSE
モデルのパブリッシュとスコアリングのターゲットプラットフォームを指定します。 PROC ACCELERATOR は LIBNAME 割り当てからプラットフォームを決定できないため、このオプションは Azure Synapse で必要です。その場合、PLATFORM=オプションを指定しなかった場合、エラーが SAS ログに書き込まれます。

要件 PLATFORM=オプションは Azure Synapse では必須ですが、他のデータソースではオプションです。

PLATFORMENDPOINT="*rest-url*"
Apache Livy や Databricks 実行コンテキストなどのサービスに REST インターフェイスを介して実行要求をサブミットするために使用される URL を指定します。 Databricks の場合、この値はワークスペース URL です。Synapse の場合、この値は Livy REST URL です。

参照項目: “Specifying the Workspace URL for Databricks” (*SAS In-Database Products: User’s Guide*)

“Forming the Livy REST URL for Synapse” (*SAS In-Database Products: User’s Guide*)

PLATFORMPASSWORD=*endpoint-password*
Livy エンドポイントアプリケーションの秘密を指定します。

要件 Azure Synapse では、PLATFORMPASSWORD=オプションが必要です。他のデータソースにはこのオプションを使用しないでください。

PLATFORMUSER=*endpoint-user-identifier*
Livy エンドポイントユーザー名を指定します。

この値はアプリケーション ID です。

要件 Azure Synapse では、PLATFORMUSER=オプションが必要です。他のデータソースにはこのオプションを使用しないでください。

オプション引数

AUTORESTART=
タイムアウトやその他のイベントによってサーバー側でセッションが終了した場合、セッションを自動的に再開します。

デフォルト TRUE

制限事項 AUTORESTART=オプションは、Synapse の場合にのみサポートされます。

MAXUNBOUNDEDLENGTH=*maximum-unbounded-column-length*
制限されていないデータ型(BLOB、CLOB、TEXT、または長い STRING 型など)が SAS Embedded Process に読み込まれるときに、その最大長を指定します。 このオプションは、データベースからの非常に大きな文字列またはバイナリ列の一貫した処理を保証します。入力列のデータが MAXUNBOUNDEDLENGTH=値を超えると、読み取り中に切り捨てが発生する可能性があります。切り捨ては、入力列でもある出力列の場合に限られます。このオプションは、入力列ではない出力列には適用されません。

MAXUNBOUNDEDEDLENGTH=オプションは、STARTSESSION ステートメントまたは RUNMODEL ステートメントのいずれかで設定できます。RUNMODEL でオプションを指定すると、STARTSESSION の値がオーバーライドされます。DBMAX_TEXT= LIBNAME オプションは PROC ACCELERATOR での処理には影響しません。

デフォルト 32767

注 MAXUNBOUNDEDLENGTH=オプションは、2025.09 に追加されました。

PROPERTIES="*property-1=value-1*<;...*property-n=value-n*>"
名前と値のペアとして Spark 構成プロパティを指定するか、または名前と値のペアとしてプロパティのリストを指定します。 等号(=)を使用して、プロパティ名とその値を区切ります。複数の *property=value* ペアを指定する場合は、ペアの間にセミコロン(;)を使用します。

制限事項 PROPERTIES=オプションは Databricks ではサポートされていません。

TIMEOUT=*n*
Livy セッションが実行要求を待機する時間(分)を指定します。 タイムアウト時間に達すると、Livy セッションが終了します。値ゼロはタイムアウトを無効にします。値がゼロでない場合、少なくとも 15 分である必要があります。

デフォルト 60 分

制限事項 TIMEOUT=オプションは、Synapse の場合にのみサポートされます。

TRACE
SAS Embedded Process がトレースをオンにして実行されるかどうかを指定します。

STOPSESSION ステートメント

SAS Embedded Process 連続セッションを停止します。

注: Databricks の場合、モデルの実行が終了したら、STOPSESSION ステートメントで PROC ACCELERATOR をサブミットすることにより、SAS Embedded Process の連続セッションを停止できます。セッションを停止しない場合、SAS を終了した数分後にセッションが終了します。

SingleStore で STARTSESSION および STOPSESSION ステートメントを使用しないでください。

構文

STOPSESSION;

PUBLISHMODEL ステートメント

モデルをデータソースにパブリッシュします。

構文

PUBLISHMODEL
MODELTABLENAME="*fully-qualified-table-name*"
MODELTYPE=DATASTEP | DS2
PLATFORM=DATABRICKS | SINGLESTORE | SYNAPSE
STOREFILES=*file-specification* | PROGRAMFILE=*file-specification*
VARSFILE=*file-path*
<*publishmodel-optional-argument*>;

必須引数

MODELTABLENAME="*fully-qualified-table-name*"
モデルのパブリッシュ先のテーブルの名前を指定します。 1 部構成、2 部構成、または 3 部構成の名前を指定できます。詳細については、“1 部、2 部、または 3 部のテーブル名の指定” (90 ページ)を参照してください。

要件 名前にピリオド、予約語、または特殊文字が含まれる場合は、名前をバックティック(`)で囲みます。

例 次の例では、カタログ、スキーマ、およびテーブルを参照します。
modeltablename="mycatalog.myschema.mytable"

次の例では、スキーマとテーブルを参照します。
modeltablename="myschema.mytable"

次の例では、テーブル名 **my.table** にピリオドが含まれているため、その前後の目盛が必要です。
modeltablename="myschema.`my.table`"

MODELTYPE=DATASTEP | DS2
入力モデルプログラムの種類を指定します。

DATASTEP
入力モデルプログラムが DATA ステップコードであることを指定します。

DS2
入力モデルプログラムが DS2 コードであることを指定します。

デフォルト DS2

注 MODELTYPE=オプションは、DATA ステップモデルの場合のみ必要です。DATA ステップコードは、アイテムストアにバンドルされる前に DS2 コードに変換されます。

PLATFORM=DATABRICKS | SINGLESTORE | SYNAPSE
モデルのパブリッシュとスコアリングのターゲットプラットフォームを指定します。 PROC ACCELERATOR は LIBNAME 割り当てからプラットフォームを決定できないため、このオプションは Azure Synapse で必要です。その場合、PLATFORM=オプションを指定しなかった場合、エラーが SAS ログに書き込まれます。

要件 PLATFORM=オプションは Azure Synapse では必須ですが、他のデータソースではオプションです。

STOREFILES=("*file-path-1*" | *fileref-1* <,"*file-path-2*" | *fileref-2*, ...>)
分析ストア.sasast ファイルのリストを指定します。ファイルは、分析ストアごとに 1 つずつパブリッシュされます。 ファイルのリストには、ファイルパス名とファイル参照名が混在している場合があります。

STOREFILES=または PROGRAMFILE=またはその両方のオプションを指定する必要があります。一部の分析ストアモデルでは、STOREFILES=オプションのみが必要です。一部の DATA ステップおよび DS2 モデルでは、PROGRAMFILE=オプションのみが必要です。一部のモデルは、DATA ステップコード、DS2 コード、および 1 つ以上の ASTORE ファイルで構成されています。その場合、STOREFILES=オプションと PROGRAMFILE=オプションの両方を指定します。

モデルプログラムなしで分析ストアをパブリッシュすると、DS2 モデルプログラムが分析ストアから生成されます。

関連するモデルプログラムを指定するには、PROGRAMFILE=オプションを使用します。分析ストアがパブリッシュされるとき、関連付けられたモデルプログラムは通常、DS2 コードです。

操作 STOREFILES=または PROGRAMFILE=またはその両方のオプションを指定する必要があります。

STOREFILES=オプションは MODELTYPE=DS2 の場合にのみ使用されます。MODELTYPE=DATASTEP の場合はこのオプションは使用されません。

PROGRAMFILE="*file-path*" | *fileref*
STOREFILES=または PROGRAMFILE=またはその両方のオプションを指定する必要があります。一部の分析ストアモデルでは、STOREFILES=オプションのみが必要です。一部の DATA ステップおよび DS2 モデルでは、PROGRAMFILE=オプションのみが必要です。一部のモデルは、DATA ステップコード、DS2 コード、お

よび 1 つ以上の ASTORE ファイルで構成されています。その場合、STOREFILES=オプションと PROGRAMFILE=オプションの両方を指定します。

操作 STOREFILES=または PROGRAMFILE=またはその両方のオプションを指定する必要があります。

VARSFILE="*file-path*" | *fileref*
入力 DATA ステップモデルプログラムを DS2 に変換する際に使用される変数メタデータ XML を含むファイルを指定します。

操作 MODELTYPE=DATASTEP の場合、一部のモデルでは VARSFILE=オプションが必要です。MODELTYPE=DS2 の場合はこのオプションは使用されません。

オプション引数

FORMATSFILE="*file-path*" | *fileref*
パブリッシュするユーザー定義の形式 XML 定義を含むファイルを指定します。

注 このオプションは、DS2 モデルまたは DATA ステップモデルに有効です。

RUNMODEL ステートメント

データソースでモデルを実行します。

構文

RUNMODEL
INPUTQUERY='*sql-query*' | INPUTSCALACODE="*scala-code*" |
INPUTTABLENAME="*fully-qualified-table-name*"
MODELTABLENAME="*fully-qualified-table-name*"
OUTPUTSCALACODE="*scala-code*" | OUTPUTTABLENAME="*fully-qualified-table-name*"
PLATFORM=DATABRICKS | SINGLESTORE | SYNAPSE
<*runmodel-optional-arguments*>;

必須引数

INPUTQUERY='*sql-query*' | INPUTSCALACODE="*scala-code*" |
INPUTTABLENAME="*fully-qualified-table-name*"

INPUTQUERY='*sql-query*'
テーブル結果セットを返す SELECT ステートメントです。

SQL クエリはデータソースでサポートされている必要があります。

制限事項 INPUTQUERY=オプションは、Databricks ではサポートされていません。

注 このオプションは、 2025.07 に追加されました。

例 SingleStore の例を次に示します。
inputquery='select a, b, c from mytable where a > 1'

例 “例 3: SingleStore でモデルをパブリッシュ、実行、および削除する” (113 ページ)

INPUTSCALACODE="*scala-code*"
Scala コードの文字列です。

カスタム Scala コードを使用して入力データを読み取る場合は、このオプションを使用します。Scala コードは、Spark データセットを返す単一のステートメントである必要があります。

INPUTSCALACODE=文字列には、変数の宣言は含めないでください。

制限事項 Scala コードは SingleStore ではサポートされていません。

注 このオプションは、 2025.07 に追加されました。

ヒント Scala コード文字列内で二重引用符 (") を使用しており、文字列全体が二重引用符 (") で囲まれている場合、内部引用符をエスケープする必要があります。引用符をエスケープするには、1 つの引用符(")ではなく、一対の引用符("")を使用します。

例 Parquet ファイルを読み取るために Databricks ノートブックでサブミットする Scala コードの例を次に示します。
val data = spark.read.parquet("/data/myparquet")

PROC ACCELERATOR で Databricks から同じファイルを読み取るコードは次のとおりです。文字列は一重引用符(')で囲まれているため、二重引用符(")はエスケープされません。
inputscalacode='spark.read.parquet("/data/myparquet")'

JSON ファイルを読み取るために Databricks ノートブックで使用される Scala コードの例を次に示します。
val df = spark.read.format("json").load("myfile.json")

PROC ACCELERATOR で Databricks から同じファイルを読み取るコードは次のとおりです。文字列内に含まれる引用符は、一対の引用符を使用してエスケープされていることに注意してください。
inputscalacode="spark.read.format(""json"").load(""myfile.json"")"

例 “例 4: スコアリング用データの読み取りと書き込みのための Scala コードの使用” (115 ページ)

INPUTTABLENAME="*fully-qualified-table-name*"
入力テーブルの名前を指定します。

1 部構成、2 部構成、または 3 部構成の名前を指定できます。詳細については、“1 部、2 部、または 3 部のテーブル名の指定” (90 ページ)を参照してください。

制限事項 INPUTQUERY=オプションまたは INPUTSCALACODE=オプションまたは INPUTTABLENAME=オプションのいずれかを指定する必要があります。複数の入力オプションを指定することはできません。

要件 名前にピリオド、予約語、または特殊文字が含まれる場合は、名前をバックティック(`)で囲みます。

MODELTABLENAME="*fully-qualified-table-name*"
モデルのパブリッシュ先のテーブルの名前を指定します。 1 部構成、2 部構成、または 3 部構成の名前を指定できます。詳細については、“1 部、2 部、または 3 部のテーブル名の指定” (90 ページ)を参照してください。

要件 名前にピリオド、予約語、または特殊文字が含まれる場合は、名前をバックティック(`)で囲みます。

例 次の例では、カタログ、スキーマ、およびテーブルを参照します。
modeltablename="mycatalog.myschema.mytable"

次の例では、スキーマとテーブルを参照します。
modeltablename="myschema.mytable"

次の例では、テーブル名 **my.table** にピリオドが含まれているため、その前後の目盛が必要です。
modeltablename="myschema.`my.table`"

OUTPUTSCALACODE="*scala-code*" | OUTPUTTABLENAME="*fully-qualified-table-name*"

OUPUTSCALACODE="*scala-code*"
Scala コードの文字列です。

カスタム Scala コードを使用して出力データを書き込む場合は、このオプションを使用します。

文字列は関数で始まる必要があります。

制限事項 Scala コードは SingleStore ではサポートされていません。

注 このオプションは、 2025.07 に追加されました。

ヒント Scala コード文字列内で二重引用符 (") を使用しており、文字列全体が二重引用符 (") で囲まれている場合、内部引用符をエスケープする必要があります。引用符をエスケープするには、1 つの引用符(")ではなく、一対の引用符("")を使用します。

例 WRITE 関数を使用する Scala コードの例を次に示します。このコードは、Databricks ノートブックでサブミットします。
outputdataset.write.parquet("/data/myparquet_output")

Databricks の PROC ACCELERATOR で WRITE 関数を使用するためのコードは次のとおりです。
outputscalacode='write.parquet("/data/myparquet_output")'

例 “例 4: スコアリング用データの読み取りと書き込みのための Scala コードの使用” (115 ページ)

OUTPUTTABLENAME="*fully-qualified-table-name*"
出力テーブルの名前を示します。

1 部構成、2 部構成、または 3 部構成の名前を指定できます。詳細については、“1 部、2 部、または 3 部のテーブル名の指定” (90 ページ)を参照してください。

制限事項 OUTPUTSCALACODE=オプションまたは OUTPUTTABLENAME=オプションのいずれかを指定する必要があります。両方のオプションを指定することはできません。

要件 名前にピリオド、予約語、または特殊文字が含まれる場合は、名前をバックティック(`)で囲みます。

PLATFORM=DATABRICKS | SINGLESTORE | SYNAPSE
モデルのパブリッシュとスコアリングのターゲットプラットフォームを指定します。 PROC ACCELERATOR は LIBNAME 割り当てからプラットフォームを決定できないため、このオプションは Azure Synapse で必要です。その場合、PLATFORM=オプションを指定しなかった場合、エラーが SAS ログに書き込まれます。

要件 PLATFORM=オプションは Azure Synapse では必須ですが、他のデータソースではオプションです。

オプション引数

KEEPLISTCOLUMNS=("*column-1*"<..."*column-n*">)
出力データに保持する 1 つ以上の列の名前を指定します。 列が指定されていない場合、その列は出力データから削除されます。出力データにすべての列を保持する場合、KEEPLISTCOLUMNS=オプションを使用しないでください。

注 KEEPLISTCOLUMNS=オプションは、 2025.06 に追加されました。

MAXUNBOUNDEDLENGTH=*maximum-unbounded-column-length*
制限されていないデータ型(BLOB、CLOB、TEXT、または長い STRING 型など)が SAS Embedded Process に読み込まれるときに、その最大長を指定します。 このオプションは、データベースからの非常に大きな文字列またはバイナリ列の一貫した処理を保証します。入力列のデータが MAXUNBOUNDEDLENGTH=値を超えると、読み取り中に切り捨てが発生する可能性があります。切り捨ては、入力列でもある出力列の場合に限られます。このオプションは、入力列ではない出力列には適用されません。

MAXUNBOUNDEDEDLENGTH=オプションは、STARTSESSION ステートメントまたは RUNMODEL ステートメントのいずれかで設定できます。RUNMODEL でオプションを指定すると、STARTSESSION の値がオーバーライドされます。DBMAX_TEXT= LIBNAME オプションは PROC ACCELERATOR での処理には影響しません。

デフォルト 32767

注 MAXUNBOUNDEDLENGTH=オプションは、2025.09 に追加されました。

詳細

入力オブジェクトと出力オブジェクトを組み合わせることができます。

- 入力オプション INPUTQUERY=または INPUTSCALACODE=または INPUTTABLENAME=のうち、1 つだけを指定する必要があります。
- 出力オプション OUTPUTSCALACODE=または OUTPUTTABLENAME=のうち 1 つだけを指定する必要があります。
- 入力オプションと出力オプションを組み合わせることができます。たとえば、INPUTSCALACODE=を使用して入力データを読み取り、OUTPUTTABLENAME=を使用して出力データを書き込むことができます。

DELETEMODEL ステートメント

データソースからモデルを削除します。

構文

DELETEMODEL
MODELTABLENAME=*fully-qualified-table-name*
PLATFORM=DATABRICKS | SINGLESTORE | SYNAPSE;

必須引数

MODELTABLENAME="*fully-qualified-table-name*"
モデルのパブリッシュ先のテーブルの名前を指定します。 1 部構成、2 部構成、または 3 部構成の名前を指定できます。詳細については、“1 部、2 部、または 3 部のテーブル名の指定” (90 ページ)を参照してください。

要件 名前にピリオド、予約語、または特殊文字が含まれる場合は、名前をバックティック(`)で囲みます。

例 次の例では、カタログ、スキーマ、およびテーブルを参照します。
modeltablename="mycatalog.myschema.mytable"

次の例では、スキーマとテーブルを参照します。
modeltablename="myschema.mytable"

次の例では、テーブル名 **my.table** にピリオドが含まれているため、その前後の目盛が必要です。

```
modeltablename="myschema.`my.table`"
```

PLATFORM=DATABRICKS | SINGLESTORE | SYNAPSE
モデルのパブリッシュとスコアリングのターゲットプラットフォームを指定します。 PROC ACCELERATOR は LIBNAME 割り当てからプラットフォームを決定できないため、このオプションは Azure Synapse で必要です。その場合、PLATFORM=オプションを指定しなかった場合、エラーが SAS ログに書き込まれます。

要件 PLATFORM=オプションは Azure Synapse では必須ですが、他のデータソースではオプションです。

例: ACCELERATOR プロシジャ

例 1: Azure Databricks でモデルをパブリッシュ、実行、および削除する

要素: LIBNAME ステートメント
PROC ACCELERATOR ステートメント
PUBLISHMODEL ステートメント
STARTSESSION ステートメント
RUNMODEL ステートメント
STOPSESSION ステートメント
DELETEMODEL ステートメント

詳細

この例では、PROC ACCELERATOR はモデルを Microsoft Azure 上の Databricks の Spark テーブルにパブリッシュします。次に、PROC ACCELERATOR は、SAS Embedded Process for Spark を使用して Databricks でモデルを実行します。プロシジャはモデルをファイルシステムではなく Spark テーブルにパブリッシュするため、PROC ACCELERATOR を使用した In-Database モデルスコアリングには、AZURETENANTID=システムオプションは必要ありません。

重要 SAS Embedded Process for Spark では、PROC SCOREACCEL の配置は PROC ACCELERATOR の場合と異なる配置が必要です。両方の RPM の機

能が必要な場合は、別々のクラスターにインストールする必要があります。詳細については、“Databricks と Synapse の配置の詳細” (95 ページ)を参照してください。

注: 2023.10 において SAS は、SAS/ACCESS Interface to Hadoop および SAS/ACCESS Interface to Spark によって使用される再配布された JDBC ドライバーを変更しました。接続オプションで変更が必要になる場合があります。詳細については、“Updates Might Be Needed Due to Driver Change” (*SAS In-Database Products: User's Guide*)を参照してください。

プログラム

```
libname spklib spark platform=databricks
      user=token
      password="authentication-token"
      server="myserver.azuredatabricks.net"
      catalog="my-catalog-name"
      schema="my-schema-name"
      port=10016
      httppath="my-http-path";

proc accelerator library=spklib;
  publishModel
     modeltablename="my-model-table"
     programfile="my-program-file"
     storefiles="my-analytic-store-file";
run;
quit;


proc accelerator library=spklib;
  startsession
     platformendpoint="https://my-identifier.databricks.net";
run;
quit;

proc accelerator library=spklib;
  runmodel
     modeltablename="my-model-table"
     inputtablename="catalog.schema.input_table"
     outputtablename="catalog.schema.output_table";
run;
quit;

proc accelerator library=spklib;
  stopsession;
run;
quit;

proc accelerator library=spklib;
  deletemodel
     modeltablename="my-model-table"
     ;
```

```
run;
quit;
```

プログラムの説明

SAS/ACCESS Interface to Spark を使用して SAS ライブラリを割り当てます。 PLATFORM=DATABRICKS、接続オプション、および必要なその他の LIBNAME オプションを指定します。この SAS ライブラリの割り当ての詳細については、"SAS/ACCESS Interface to Spark" (*SAS/ACCESS for Relational Databases: Reference*) を参照してください。

```
libname spklib spark platform=databricks
    user=token
    password="authentication-token"
    server="myserver.azuredatabricks.net"
    catalog="my-catalog-name"
    schema="my-schema-name"
    port=10016
    httppath="my-http-path";
```

モデルをパブリッシュします。 ライブラリ参照名を参照するには、PROC ACCELERATOR ステートメントで LIBRARY=オプションを使用します。

```
proc accelerator library=spklib;
  publishModel
    modeltablename="my-model-table"
    programfile="my-program-file"
    storefiles="my-analytic-store-file";
run;
quit;
```

SAS Embedded Process 連続セッションを開始します。 プロシジャは連続セッションを開始し、指定されたエンドポイントと SAS ライブラリの場所を関連付けます。PLATFORMENDPOINT=値は、Databricks ワークスペース URL です。

```
proc accelerator library=spklib;
  startsession
    platformendpoint="https://my-identifier.databricks.net";
run;
quit;
```

モデルを実行します。 モデルを実行する前に SAS Embedded Process 連続セッションを開始する必要があります。

```
proc accelerator library=spklib;
  runmodel
    modeltablename="my-model-table"
    inputtablename="catalog.schema.input_table"
    outputtablename="catalog.schema.output_table";
run;
quit;
```

SAS Embedded Process 連続セッションを停止します。 モデルの実行が終了したら、STOPSESSION ステートメントを使用して PROC ACCELERATOR をサブミット

することにより、SAS Embedded Process の連続セッションを停止できます。セッションを停止しない場合、SAS を終了した数分後にセッションが終了します。

```
proc accelerator library=spklib;
  stopsession;
run;
quit;
```

モデルを削除します。

```
proc accelerator library=spklib;
  deletemodel
    modeltablename="my-model-table"
    ;
run;
quit;
```

例 2: Azure Synapse でモデルをパブリッシュ、実行、および削除する

要素:

LIBNAME ステートメント
PROC ACCELERATOR ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この例では、PROC ACCELERATOR はモデルを Microsoft Azure Synapse Analytics の SQL サーバーテーブル(専用 SQL プールテーブルとも呼ばれます)にパブリッシュします。次に、PROC ACCELERATOR は、SAS Embedded Process for Spark を使用してモデルを実行します。接続には SAS/ACCESS Interface to Microsoft SQL サーバーが使用されます。

プログラム

```
libname mylib sqlsvr
      user=my-user-name
      password="my-password"
      schema="my-schema"
      noprompt="DRIVER=SAS ACCESS to MS SQL Server;
            Hostname=synapse-workspace-name.sql.azuresynapse.net;
            Port=my-port-number;
            Database=my-database-name
```

```
            ...;";
proc accelerator library=mylib;
  publishModel
    platform=synapse
    modeltablename="my-model-table"
    programfile="my-program-file"
    storefiles="my-analytic-store-file";
run;
quit;

proc accelerator library=mylib;
  startsession
    platform=synapse
    platformendpoint=""
    platformuser="endpoint/livyApi/versions/livyApiVersion/sparkPools/sparkPoolName"
    platformpassword="my-secret";
run;
quit;

proc accelerator library=mylib;
  runmodel
    platform=synapse
    modeltablename="my-model-table"
    inputtablename="my-input-table"
    outputtablename="my-output-table";
run;
quit;

proc accelerator library=mylib;
  stopsession;
run;
quit;

proc accelerator library=mylib;
  deletemodel
    platform=synapse
    modeltablename="my-model-table"
    ;
run;
quit;
```

プログラムの説明

SAS/ACCESS Interface to Microsoft SQL サーバーを使用して SAS ライブラリを割り当てます。 Synapse ワークスペースの 専用の SQL プールに接続する必要があります。プールは一時停止解除する必要があります。SCHEMA=LIBNAME オプションにスキーマを指定します。NOPROMPT=オプション内で Database=接続オプションを指定する必要があります。データベース値は、専用 SQL プールの名前です。DATASRC=オプションは使用しないでください。追加のオプションについては、*“SAS/ACCESS Interface to Microsoft SQL Server” (SAS/ACCESS for Relational Databases: Reference)*を参照してください。

```
libname mylib sqlsvr
      user=my-user-name
      password="my-password"
```

```
    schema="my-schema"
    noprompt="DRIVER=SAS ACCESS to MS SQL Server;
        Hostname=synapse-workspace-name.sql.azuresynapse.net;
        Port=my-port-number;
        Database=my-database-name
        ...;";
```

モデルをパブリッシュします。 ライブラリ参照名を参照するには、PROC ACCELERATOR ステートメントで LIBRARY=オプションを使用します。Synapse の場合、STOPSESSION を除くすべての PROC ACCELERATOR ステートメントで PLATFORM=オプションが必要です。モデルに応じて、PROGRAMFILE=または STOREFILES=または両方のオプションを指定する必要があります。詳細については、“PUBLISHMODEL ステートメント” (100 ページ)の構文を参照してください。

```
proc accelerator library=mylib;
  publishModel
    platform=synapse
    modeltablename="my-model-table"
    programfile="my-program-file"
    storefiles="my-analytic-store-file";
run;
quit;
```

SAS Embedded Process 連続セッションを開始します。 プロシジャは連続セッションを開始し、指定されたエンドポイントと SAS ライブラリの場所を関連付けます。Synapse には、PLATFORM=、PLATFORMENDPOINT=、PLATFORMUSER=、および PLATFORMPASSWORD=オプションが必要です。エンドポイントの詳細については、“Forming the Livy REST URL for Synapse” (*SAS In-Database Products: User’s Guide*)を参照してください。

```
proc accelerator library=mylib;
  startsession
    platform=synapse
    platformendpoint=""
    platformuser="endpoint/livyApi/versions/livyApiVersion/sparkPools/sparkPoolName"
    platformpassword="my-secret";
run;
quit;
```

モデルを実行します。 モデルを実行する前に SAS Embedded Process 連続セッションを開始する必要があります。

```
proc accelerator library=mylib;
  runmodel
    platform=synapse
    modeltablename="my-model-table"
    inputtablename="my-input-table"
    outputtablename="my-output-table";
run;
quit;
```

SAS Embedded Process 連続セッションを停止します。 モデルの実行が終了したら、STOPSESSION ステートメントを使用して PROC ACCELERATOR をサブミットすることにより、SAS Embedded Process の連続セッションを停止できます。セッションを停止しない場合、SAS を終了した数分後にセッションが終了します。

```
proc accelerator library=mylib;
  stopsession;
run;
```

```
quit;
```

モデルを削除します。

```
proc accelerator library=mylib;
  deletemodel
    platform=synapse
    modeltablename="my-model-table"
    ;
run;
quit;
```

例 3: SingleStore でモデルをパブリッシュ、実行、および削除する

要素: LIBNAME ステートメント
PROC ACCELERATOR ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この例では、PROC ACCELERATOR はモデルを SingleStore データベースのテーブルにパブリッシュします。次に、PROC ACCELERATOR は、SAS Embedded Process を使用して SingleStore でモデルを実行します。製品 SAS SpeedyStore に含まれる SingleStore インスタンスがサポートされています。外部 SingleStore データベースはサポートされていません。詳細については、“SingleStore の配置の詳細” (96 ページ)を参照してください。

注: SingleStore の場合、モデルを実行する前に SAS Embedded Process 連続セッションを開始しないでください。

プログラム

```
libname mylib sstore user=myusr1 password=mypwd1 database=mydb
      server=mysrv1 port=9876;

proc accelerator library=mylib;
  publishModel
    modeltablename="model_table"
    programfile="my_program_file"
    storefiles="my-analytic-store-file";
```

```
run;
quit;

proc accelerator library=mylib;
  runmodel
    modeltablename="model_table"                    1
    inputquery="select sales, salary from mydb.mydata" 2
    outputtablename="output_database.output_table";   3
run;
quit;

proc accelerator library=mylib;
  deletemodel
    modeltablename="model_table";
run;
quit;
```

プログラムの説明

SAS/ACCESS Interface to SingleStore を使用して SAS ライブラリを割り当てます。 この LIBNAME ステートメントについては、“SAS/ACCESS Interface to SingleStore” (*SAS/ACCESS for Relational Databases: Reference*)を参照してください。

```
libname mylib sstore user=myusr1 password=mypwd1 database=mydb
      server=mysrv1 port=9876;
```

モデルをパブリッシュします。 PROC ACCELERATOR ステートメントの LIBRARY= オプションは、ライブラリ参照名を参照します。

```
proc accelerator library=mylib;
  publishModel
    modeltablename="model_table"
    programfile="my_program_file"
    storefiles="my-analytic-store-file";
run;
quit;
```

モデルを実行します。

```
proc accelerator library=mylib;
  runmodel
    modeltablename="model_table"                    1
    inputquery="select sales, salary from mydb.mydata" 2
    outputtablename="output_database.output_table";   3
run;
quit;
```

1 MODELTABLENAME=オプションは 1 レベルの名前を指定するため、Mylib LIBNAME ステートメントで指定された Mydb データベースを参照します。

2 INPUTQUERY=オプションは、SELECT ステートメントを指定します。SELECT クエリは SingleStore でサポートされている必要があります。LIBNAME ステートメントと同じデータベースの場合でも、SELECT ステートメントでデータベースを指定する必要があります。

3 OUTPUTTABLENAME=オプションは 2 レベルの名前を指定するため、LIBNAME ステートメントの Mylib データベースとは異なるデータベースにテーブルを保存します。

モデルを削除します。

```
proc accelerator library=mylib;
  deletemodel
    modeltablename="model_table";
run;
quit;
```

例 4: スコアリング用データの読み取りと書き込みのための Scala コードの使用

要素:

LIBNAME ステートメント
PROC ACCELERATOR ステートメント
PUBLISHMODEL ステートメント
STARTSESSION ステートメント
RUNMODEL ステートメント
STOPSESSION ステートメント
DELETEMODEL ステートメント

詳細

この Databricks の例では、入力データはカスタム Scala コードを使用して読み取られ、出力データはカスタム Scala コードを使用して書き込まれます。

> 重要　SAS Embedded Process for Spark では、PROC SCOREACCEL の配置は PROC ACCELERATOR の場合と異なる配置が必要です。両方の RPM の機能が必要な場合は、別々のクラスターにインストールする必要があります。詳細については、“Databricks と Synapse の配置の詳細” (95 ページ)を参照してください。

SAS/ACCESS Interface to Spark を使用して SAS ライブラリを割り当てます。
PLATFORM=DATABRICKS、接続オプション、および必要なその他の LIBNAME オプションを指定します。この SAS ライブラリの割り当ての詳細については、“SAS/ACCESS Interface to Spark” (*SAS/ACCESS for Relational Databases: Reference*) を参照してください。

```
libname spklib spark platform=databricks
    user=token
    password="authentication-token"
    server="myserver.azuredatabricks.net"
```

```
    catalog="my-catalog-name"
    schema="my-schema-name"
    port=10016
    httppath="my-http-path";
```

モデルをパブリッシュします。 ライブラリ参照名を参照するには、PROC ACCELERATOR ステートメントで LIBRARY=オプションを使用します。

```
proc accelerator library=spklib;
  publishModel
    modeltablename="my-model-table"
    programfile="my-program-file"
    storefiles="my-analytic-store-file";
run;
quit;
```

SAS Embedded Process 連続セッションを開始します。 プロシジャは連続セッションを開始し、指定されたエンドポイントと SAS ライブラリの場所を関連付けます。PLATFORMENDPOINT=値は、Databricks ワークスペース URL です。

```
proc accelerator library=spklib;
  startsession
    platformendpoint="https://my-identifier.databricks.net";
run;
quit;
```

モデルを実行します。 INPUTSCALACODE=オプションは、Spark テーブルを読み取ります。OPUTSCALACODE=オプションは、Spark テーブルを書き込みます。入力 Scala 文字列は、Spark データセットを返す単一のステートメントである必要があります。出力 Scala 文字列は、関数で始まる必要があります。ここでは、その関数は WRITE です。

```
%let mydatabase=september;                                    1
proc accelerator library=spklib;
  runmodel
    modeltablename="my-model-table"
    inputscalacode="spark.table(""&mydatabase..input_table"").select(""col1"", ""col2"")"  2

outputscalacode="write.mode(""overwrite"").saveAsTable(""&mydatabase..output_table"")";
3
run;
```

1 ユーザーが毎月異なるデータベースに対してモデルの実行を望むため、LET ステートメントはマクロ変数 mydatabase を定義します。

2 INPUTSCALACODE=オプションはマクロ変数を参照するため、文字列全体を二重引用符で囲む必要があります。文字列内では、マクロ変数を含む値を囲む引用符を避ける必要があります。文字列内の引用符をエスケープするには、1 つの引用符(")のかわりに一対の引用符(" ")を使用します。列名の前後の引用符もエスケープされています。

3 OUTPUTSCALACODE=文字列内の引用符は、入力文字列と同様にエスケープされます。

SAS Embedded Process 連続セッションを停止します。 モデルの実行が終了したら、STOPSESSION ステートメントを使用して PROC ACCELERATOR をサブミットすることにより、SAS Embedded Process の連続セッションを停止できます。セッションを停止しない場合、SAS を終了した数分後にセッションが終了します。

```
proc accelerator library=spklib;
  stopsession;
run;
quit;
```

モデルを削除します。

```
proc accelerator library=spklib;
  deletemodel
    modeltablename="my-model-table";
run;
quit;
```

8

APPEND プロシジャ

概要: APPEND プロシジャ

APPEND プロシジャの機能

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

詳細については、4 章, “Base プロシジャの CAS 処理” (75 ページ)を参照してください。

大まかに言えば、APPEND プロシジャは DATASETS プロシジャの APPEND ステートメントと同様に機能します。APPEND プロシジャと PROC DATASETS の APPEND ステートメントの違いは、BASE=オプションと DATA=オプションの *libref* のデフォ

ルト値のみです。PROC APPEND のデフォルトは WORK か USER です。APPEND ステートメントのデフォルトは、プロシジャの入力ライブラリのライブラリ参照名です。

構文: APPEND プロシジャ

要件: BASE=データセットは、更新処理をサポートする SAS ライブラリのメンバーである必要があります。

CAS libname エンジンは更新処理をサポートしていないため、既存の CAS テーブルを BASE=オプションで指定することはできません。

注: NOTE: _LAST_ は、セッションで作成された最後のデータセットの名前です。BASE=データセットが存在せず、PROC APPEND がそれを作成する場合、PROC APPEND は_LAST_を BASE=データセットの名前に設定します。

BASE=データセットで DROP=、KEEP=、または RENAME=オプションを使用する場合、それらのオプションは APPEND 処理のみに影響し、追加された BASE=データセットの変数を変更しません。DROP=オプションと KEEP=オプションを使用してドロップされる、または保持されない変数は、追加された BASE=データセットに存在しています。RENAME=オプションを使用して名前が変更される変数は、追加された BASE=データセットにその元の名前で存在しています。

ヒント: APPEND プロシジャに関する詳細なドキュメントは、APPEND ステートメント (383 ページ)の DATASETS プロシジャの中にあります。

ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (23 ページ)を参照してください。

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

PROC APPENDBASE=<*libref.*>
<*options*>*SAS-data-set*;
ATTRIB *variable-list(s) attribute-list(s)*;
FORMAT *variable-1*<...*variable-n*> <*format*> <DEFAULT=*default-format*>;
LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;
WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク	例
“APPEND ステートメント”	ある SAS データセットのオブザベーションを別の SAS データセットの末尾に追加する	Ex. , Ex. 3

PROC APPEND ステートメント

1 つの SAS データセットから別の SAS データセットの最後にオブザベーションを追加します。

構文

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

必須引数

BASE=<*libref.*>*SAS-data-set*
オブザベーションの追加先となるデータセットを指定します。

libref
SAS データセットを含むライブラリを指定します。ライブラリ参照名を省略すると、デフォルトはプロシジャ入力ライブラリのライブラリ参照名となります。PROCAPPEND を使用している場合、ライブラリ参照名のデフォルトは WORK または USER のいずれかになります。

SAS-data-set
SAS データセットを指定します。APPEND ステートメントは、この名前の既存データセットが見つからない場合、ライブラリに新しいデータセットを作成します。つまり、APPEND ステートメントを使用して、BASE=引数で新しいデータセット名を指定することによりデータセットを作成できます。

新しいデータセットを作成する、既存データセットに追加するにかかわらず、BASE=データセットはすべての追加操作の後の最新の SAS データセットとなります。

別名 OUT=

例 “例 6: 2 つの SAS データセットを連結する” (508 ページ)

オプション引数

APPENDVER=V6
オブザベーションを BASE=データセットに追加するためのバージョン 6 の動作を使用します。一度に 1 つのオブザベーションが追加されます。バージョン 7 からは、パフォーマンスを向上させるため、データセットの処理後にすべてのオブザベーションが追加されるようにデフォルトの動作が変わりました。

参照項目: “インデックス付きデータセットへの追加 — 高速追加メソッド” (390 ページ)

DATA=<*libref.*> *SAS-data-set*
BASE=引数で指定される SAS データセットの最後に追加するオブザベーションを含む SAS データセットを指定します。

libref
SAS データセットを含むライブラリを指定します。ライブラリ参照名を省略すると、デフォルトはプロシジャ入力ライブラリのライブラリ参照名となります。DATA=データセットは、SAS ライブラリのものとなります。データセットがプロシジャ入力ライブラリ以外のライブラリに存在する場合、2 レベル名を使用する必要があります。

SAS-data-set
SAS データセットを指定します。APPEND ステートメントはこの名前の既存データセットが見つからなかった場合、処理を停止します。

別名 NEW=

デフォルト SAS ライブラリからの、最も新しく作成された SAS データセット

参照項目: “世代グループを使用した追加” (394 ページ)

例 “例 6: 2 つの SAS データセットを連結する” (508 ページ)

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

参照項目: “AES 暗号化データセットの追加” (389 ページ)

FORCE
DATA=データセットに次の基準のうちいずれかを満たす変数が含まれている場合、APPEND ステートメントによるデータセットの連結を強制的に行います。

- BASE=データセットにありません。
- BASE=データセットの変数と同じ種類がありません。
- BASE=データセットの変数より長いです。

ヒント GENNUM=データセットオプションを使用して、世代グループの特定のバージョン間での追加が可能です。次に、いくつかの例を示します。

```
/* appends historical version to base A */
proc datasets;
  append base=a
    data=a (gennum=2);

/* appends current version of A to historical version */
proc datasets;
  append base=a (gennum=1)
    data=a;
```

参照項目: “変数が異なるデータセットへの追加” (391 ページ)および“属性が異なる変数を含むデータセットへの追加” (392 ページ)

例 “例 6: 2 つの SAS データセットを連結する” (508 ページ)

GETSORT
ソートインジケーターを DATA=データセットから BASE=データセットにコピーします。ソートインジケーターは、次の基準が満たされる場合に PROC SQL の PROC SORT 句または ORDERBY 句によって作成されます。

- BASE=データセットは、次の基準を満たしていなければなりません。
  - オブザベーションが含まれていない
  - ソートインジケーターを受け入れる

注意
DATA=データセットが並べ替えられない場合でも、BASE=データセット上の既存するソートインジケーターが警告なしで上書きされます。

- DATA=データセットは、次の基準を満たしていなければなりません。
  - PROC SORT によって作成されたソートインジケーターを含む
  - BASE=データセットと同じデータ表現

制限事項 BASE=データセットが監査証跡と関連付けられている場合、GETSORT オプションはそのデータセットに影響しません。この制約は、APPEND プロセスが続行される間、出力における WARNING の原因となります。

DATA=データファイルにドロップ、保持、または名前変更された変数がある場合、GETSORT オプションはそのデータセットに影響しません。

NOWARN
FORCE オプションと使用した場合、異なる変数を持つ 2 つのデータセットを連結する時に、警告を非表示にします。

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。“出力形式の取り消し” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、*"PUT ステートメント: リスト" (SAS DATA ステップステートメント: リファレンス)*を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=' '>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項
ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える

場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: *“文字定数と文字変数” (SAS プログラマガイド: エッセンシャル)*.

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

使用法: APPEND プロシジャ

APPEND プロシジャの使用

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

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

proc contents data=dataset1;
run;
```

例: APPEND プロシジャ

例 1: 2 つの SAS データセットを連結する

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

OPTIONS ステートメント

PRINT プロシジャ

データセット: EXP ライブラリ

詳細

この例は、次のタスクを示しています。

- ライブラリの印刷の抑制
- 2 つのデータセットの追加
- 追加後の新しいデータセットの出力

Exp.Results および Exp.Sur の各データセットを作成し、それらを出力してから、この例に従ってそれらを連結するには、“EXP ライブラリ” (2489 ページ)を参照してください。

プログラム

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

LIBNAME exp 'SAS-library';

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

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

プログラムの説明

この例では、あるデータセットの末尾に別のデータセットを追加します。

データセット Exp.Sur には変数 Wt6Mos が含まれていますが、Exp.Results データセットには含まれていません。

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

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

LIBNAME ステートメントでライブラリを割り当てます。

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

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

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

データセットを出力します。 アウトプット 8.20 (133 ページ)を参照してください。

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

出力: 2 つのデータセットを連結する

アウトプット 8.1　Results データセット

The RESULTS Data Set

ID	TREAT	INITWT	WT3MOS	AGE
1	Other	166.28	146.98	35
2	Other	214.42	210.22	54
3	Other	172.46	159.42	33
5	Other	175.41	160.66	37
6	Other	173.13	169.40	20
7	Other	181.25	170.94	30
10	Other	239.83	214.48	48
11	Other	175.32	162.66	51
12	Other	227.01	211.06	29
13	Other	274.82	251.82	31

アウトプット 8.2　Sur データセット

The EXP.SUR Data Set

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

アウトプット 8.3　Results データセットと Sur データセットを連結する

The RESULTS Data Set

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

例 2: CAS テーブルを SAS データセットに連結する

要素:

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

OPTIONS ステートメント

CONTENTS プロシジャ

詳細

この例は、次のタスクを示しています。

- CAS テーブルを SAS データセットに追加する
- テーブル、データセット、および追加後の新しいデータセットの内容

プログラム

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

libname sascas1 cas;

libname saleslib 'directory-name';

proc contents data=saleslib.monthly;
run;

proc contents data=sascas1.lastmonth;
run;

proc append base=saleslib.monthly data=sascas1.lastmonth force;
run;

proc sql outobs=5;
  select store_id, address, city, state, zipcode, totalsales
  format dollar12.
  from saleslib.monthly(obs=4)
  where totalsales gt 2000000;
quit;
```

プログラムの説明

この例では、SAS データセットの末尾に CAS テーブルを追加します。

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

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

LIBNAME ステートメントは、CAS Engine および BASE エンジンライブラリを割り当てます。

```
libname sascas1 cas;

libname saleslib 'directory-name';
```

テーブルとデータセットの内容を確認します。 PROC CONTENTS を使用して、データセットとテーブルを表示します。

```
proc contents data=saleslib.monthly;
run;

proc contents data=sascas1.lastmonth;
run;
```

SasCas1.LastMonth テーブルを SalesLib.Monthly データセットに追加します。 CAS テーブルの先月の売り上げデータが、SAS データセットに保存された売り上げ累計データに追加されます。CAS テーブルは、VARCHAR を使用して、city と address の値を保存します。SAS データセットは、その値を文字変数で保存します。2 つの値の属性が異なるため、PROC APPEND で FORCE オプションが使用されます。

```
proc append base=saleslib.monthly data=sascas1.lastmonth force;
run;
```

売り上げ合計を取得します。 PROC SQL を使用して、5 つの変数と、$2,000,000 よりも大きい売り上げを取得します。

```
proc sql outobs=5;
  select store_id, address, city, state, zipcode, totalsales
  format dollar12.
  from saleslib.monthly(obs=4)
  where totalsales gt 2000000;
quit;
```

ログ内の警告

ログに送信された警告に注意してください。

```
117       proc append base=saleslib.monthly data=sascas1.lastmonth force;
 118       run;

 NOTE: Appending SASCAS1.LASTMONTH to SALESLIB.MONTHLY.
 WARNING: Variable city has different lengths on BASE and DATA files
       (BASE 100 DATA 21).
 WARNING: Variable address has different lengths on BASE and DATA files
       (BASE 160 DATA 19).
 NOTE: There were 12 observations read from the data set SASCAS1.LASTMONTH.
 NOTE: 12 observations added.
 NOTE: The data set SALESLIB.MONTHLY has 24 observations and 7 variables.
```

出力: CAS テーブルを SAS データセットに連結する

アウトプット 8.4　CAS テーブルの内容

LastMonth Table

The CONTENTS Procedure

Data Set Name	SASCAS1.LASTMONTH	Observations	12
Member Type	DATA	Variables	7
Engine	CAS	Indexes	0
Created	08/16/2017 18:56:09	Observation Length	80
Last Modified	08/16/2017 18:56:09	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Limit	100MB
Caslib	CASUSER
Scope	Session

Alphabetic List of Variables and Attributes

#	Variable	Type	Len		Max Bytes Used
			Bytes	Chars	
6	address	Varchar	160	40	19
4	city	Varchar	100	25	21
7	month	Char	10		
5	state	Char	2		
1	store_id	Char	3		
2	totalsales	Num	8		
3	zipcode	Char	5		

アウトプット 8.5　Monthly データセットの内容

Concatenated Data Set

The CONTENTS Procedure

Data Set Name	SALESLIB.MONTHLY	Observations	24
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	08/16/2017 13:56:07	Observation Length	288
Last Modified	08/16/2017 13:56:17	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	227
Obs in First Data Page	24
Number of Data Set Repairs	0
Filename	/r/ge
Release Created	9.0401M5
Host Created	Linux
Inode Number	71586336
Access Permission	rw-r--r--
Owner Name	
File Size	128KB
File Size (bytes)	131072

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
6	address	Char	160
4	city	Char	100
7	month	Char	10
5	state	Char	2
1	store_id	Char	3
2	totalsales	Num	8
3	zipcode	Char	5

アウトプット 8.6　連結データセット

Concatenated Data Set

The CONTENTS Procedure

Data Set Name	SALESLIB.MONTHLY	Observations	24
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	08/16/2017 13:56:07	Observation Length	288
Last Modified	08/16/2017 13:56:17	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	227
Obs in First Data Page	24
Number of Data Set Repairs	0
Filename	/r/ge
Release Created	9.0401M5
Host Created	Linux
Inode Number	71586336
Access Permission	rw-r--r--
Owner Name	
File Size	128KB
File Size (bytes)	131072

アウトプット 8.7　総売上

store_id	address	city	state	zipcode	totalsales
843	318 S Barnes St	What Cheer	IA	50268	$2,218,497
486	2345 Lindeman Ln	Nameless	TX	78641	$2,101,480
814	115 Tuscarora Pike	Shanghai	WV	25427	$2,195,378
842	318 S Lamar St	What Cheer	IA	50268	$2,218,497

例 3: ソートインジケーター情報の取得

要素:

PROC APPEND ステートメントオプション
　GETSORT
BY ステートメント
CONTENTS プロシジャ
ODS ステートメント
SORT プロシジャ

詳細

この例は、次のタスクを示しています。

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

プログラム

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

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

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

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

ods select sortedby;

proc contents data=mtea;
run;

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

ods select sortedby;

proc contents data=mysort;
run;

data mysort;
   length var1 8.;
```

```
   do var1=1 to 10;
   output;
end;
run;

proc sort data=mysort;
   by var1;
run;

ods select sortedby;

proc contents data=mysort;
run;
```

プログラムの説明

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

オブザベーションを含まない"シェル"データセットを作成します。

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

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

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

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

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

ods select sortedby;

proc contents data=mtea;
run;
```

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

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

```
run;

ods select sortedby;

proc contents data=mysort;
run;
```

ソートインジケーターが PROC SORT で作成されます。

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

proc sort data=mysort;
   by var1;
run;

ods select sortedby;

proc contents data=mysort;
run;
```

出力例

アウトプット 8.8　降順の並べ替え情報

The CONTENTS Procedure

Sort Information	
Sortedby	DESCENDING var1
Validated	YES
Character Set	ANSI

アウトプット 8.9　SORTEDBY=データセットオプションを使用したソートインジケーター情報

The CONTENTS Procedure

Sort Information	
Sortedby	var1
Validated	NO
Character Set	ANSI

アウトプット 8.10　SORT プロシジャを使用したソートインジケーター情報

The CONTENTS Procedure

Sort Information	
Sortedby	var1
Validated	YES
Character Set	ANSI

9

CATALOG プロシジャ

概要: CATALOG プロシジャ

CATALOG プロシジャの機能

CATALOG プロシジャは、SAS カタログのエントリを管理します。PROC CATALOG は対話型のステートメント主導のプロシジャで、次を実行できます。

- カタログのコンテンツのリストの作成
- カタログ、またはカタログ内の選択エントリのコピー
- カタログ内のエントリの名前変更、交換、削除
- カタログエントリ名の変更
- カタログエントリの説明の変更または削除による修正

概念: CATALOG プロシジャ

コンセプト: CATALOG プロシジャ

SAS カタログのエントリの管理に SAS ウィンドウ環境を使用する方法については、SAS オンラインヘルプの**エクスプローラー**ウィンドウを参照してください。PROC CATALOG を使用するかわりに**エクスプローラー**ウィンドウを使用することもできます。**エクスプローラー**ウィンドウでは、プロシジャが行うほとんどの操作を実行することができます。

構文: CATALOG プロシジャ

制限事項: Compute Server で使用しますが、CAS では使用しません。

ヒント: CATALOG プロシジャは RUN グループ処理をサポートします。

PROC CATALOG CATALOG=<*libref.*>*catalog* <ENTRYTYPE=*entry-type*>
<FORCE> <KILL>;

CONTENTS <OUT=*SAS-data-set*> <FILE=*fileref*>;

COPY OUT=<*libref.*>*catalog* <*options*>;

SELECT *entry-1 <entry-2 ...>* </ ENTRYTYPE=*entry-type* >;

EXCLUDE *entry-1 <entry-2 ...>* </ ENTRYTYPE=*entry-type*>;

CHANGE *old-name-1=new-name-1*
<old-name-2=new-name-2 ...>
</ ENTRYTYPE=*entry-type*>;

EXCHANGE *name-1=other-name-1*
<name-2=other-name-2 ...>
</ ENTRYTYPE=*entry-type*>;

DELETE *entry-1 <entry-2 ...>* </ ENTRYTYPE=*entry-type*>;

MODIFY *entry* (DESCRIPTION=<<'>*entry-description*<'>>)
</ ENTRYTYPE=*entry-type*>;

SAVE *entry-1 <entry-2 ...>* </ ENTRYTYPE=*entry-type* >;

ステートメント	タスク	例
9 章, “CATALOG プロシジャ,”	SAS カタログ間でエントリをコピーします	Ex. 1, Ex. 2, Ex. 3
“CHANGE ステートメント”	カタログエントリの名前を変更します	Ex. 2
“CONTENTS ステートメント”	カタログのコンテンツを印刷します	Ex. 2
“COPY ステートメント”	カタログ間で一部のエントリまたはすべてのエントリをコピーします	Ex. 1
“DELETE ステートメント”	指定エントリを削除します	Ex. 1
“EXCHANGE ステートメント”	2 つのカタログエントリの名前を切り替えます	
“EXCLUDE ステートメント”	エントリをコピー対象から除外します	Ex. 1
“MODIFY ステートメント”	カタログエントリの説明を変更します	Ex. 2
“SAVE ステートメント”	指定されているエントリ以外はすべて削除します	
“SELECT ステートメント”	選択エントリのみコピーします	Ex. 1

PROC CATALOG ステートメント

エントリを SAS カタログ間でコピーします。

構文

PROC CATALOG CATALOG=<*libref*.>*catalog* <ENTRYTYPE=*entry-type*>
<FORCE> <KILL>;

オプション引数の要約

ENTRYTYPE=*entry-type*
: 現在の PROC CATALOG ステップの処理を 1 つのエントリの種類に制限します。

FORCE
: 別のリソース環境で開かれているカタログに対してステートメントの実行を強制します。

KILL

必須引数

CATALOG=<*libref*.>*catalog*
: 処理する SAS カタログを指定します。

別名: CAT=
C=

デフォルト: PROC CATALOG はカタログのすべてのエントリを処理します。

例: “例 3: FORCE オプションと KILL オプションの併用” (170 ページ)

オプション引数

ENTRYTYPE=*entry-type*
: 現在の PROC CATALOG ステップの処理を 1 つのエントリの種類に制限します。

別名: ET=

デフォルト: PROC CATALOG はカタログのすべてのエントリを処理します。

操作 指定したエントリの種類は、従属ステートメントで使用される 1 レベルのエントリ名に適用されます。この指定は従属ステートメントでオーバーライドできません。

ENTRYTYPE=オプションは KILL オプションの影響を制限しません。

ヒント 単一の PROC CATALOG ステップで複数のエントリの種類を処理するには、ENTRYTYPE=を PROC CATALOG ステートメントではなく、従属ステートメントで使用します。

参照項目: “エントリの種類の指定” (159 ページ)

例 “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)

“例 2: コンテンツの表示、名前の変更、説明の変更” (168 ページ)

FORCE
別のリソース環境で開かれているカタログに対してステートメントの実行を強制します。

カタログのコンテンツを根本的に変えるような一部の CATALOG ステートメントおよび引数には、操作するカタログに対する排他アクセスが必要です。排他アクセスを取得できない場合、アクションは失敗します。FORCE オプションの影響を受けるステートメントと引数およびカタログは、次のとおりです。

COPY	OUT=カタログに影響します。
COPY MOVE	IN=カタログと OUT=カタログに影響します。
SAVE	指定したカタログに影響します。

ヒント 排他アクセスが取得できない場合にもステートメントを実行するには、FORCE オプションを使用します。

例 “例 3: FORCE オプションと KILL オプションの併用” (170 ページ)

KILL
指定されたカタログに影響し、SAS カタログ内のすべてのエントリを削除します。

SAS カタログのすべてのエントリを削除します。

注意
KILL オプションの影響を制限しないでください。 このオプションは、その他のオプションまたはステートメントが有効になる前に SAS カタログのすべてのエントリを削除します。

操作 ENTRYTYPE=が指定されている場合でも、KILL オプションはすべてのカタログエントリを削除します。

KILL オプションはその他のステートメントが処理される前に SAS カタログのすべてのエントリを削除するため、SAVE ステートメントは影響しません。

ヒント KILL オプションはすべてのエントリを削除しますが、SAS ライブラリからの空のカタログは削除しません。空の SAS カタログを削除するには、PROC DATASETS、**DIR** ウィンドウなどの別の方法を使用する必要があります。

例 “例 3: FORCE オプションと KILL オプションの併用” (170 ページ)

CHANGE ステートメント

1 つ以上のカタログエントリの名前を変更します。

ヒント: 単一の CHANGE ステートメントで複数の名前を変更することも、複数の CHANGE ステートメントを使用することもできます。

例: “例 2: コンテンツの表示、名前の変更、説明の変更” (168 ページ)

構文

CHANGE *old-name-1=new-name-1*
<*old-name-2=new-name-2 ...*>
</ ENTRYTYPE=*entry-type*>;

必須引数

old-name=new-name
カタログエントリの現在の名前と、それに割り当てる新しい名前を指定します。有効な SAS 名を指定します。

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目: “ENTRYTYPE=オプション” (160 ページ)

“エントリの種類の指定” (159 ページ)

CONTENTS ステートメント

プロシジャ出力のカタログのコンテンツをリストするか、コンテンツのリストを SAS データセット、外部ファイル、またはその両方に書き込みます。

注: CONTENTS ステートメントには、ENTRYTYPE=オプションを使用できません。

例: “例 2: コンテンツの表示、名前の変更、説明の変更” (168 ページ)

構文

CONTENTS <CATALOG=<*libref*.>*catalog* > <OUT=*SAS-data-set*> <FILE=*fileref*>;

引数なし

出力はプロシジャ出力に送信されます。

オプション引数

CATALOG=<*libref*.>*catalog*
処理する SAS カタログを指定します。

別名 CAT=
C=

デフォルト なし

FILE=*fileref*
SAS ファイル参照名で識別される、外部ファイルにコンテンツを送信します。

操作 *fileref* に割り当て済みのファイルがない場合は、動作環境での命名規則に従った名前の外部ファイルが作成されます。

OUT=*SAS-data-set*
コンテンツを SAS データセットに送信します。ステートメントが実行されると、データセットが作成されたことを示すメッセージが SAS ログに書き込まれます。データセットには、次の順序で 6 つの変数が含まれています。

表 9.1 OUT=出力

LIBNAME	ライブラリ参照名
MEMNAME	カタログ名
NAME	エントリ名

TYPE	エントリの種類
DESC	エントリの説明
DATE	エントリが最後に変更された日付

COPY ステートメント

カタログ間で一部のエントリまたはすべてのエントリをコピーします。

制限事項: COPY ステートメントの影響は RUN ステートメントで、または SELECT または EXCLUDE ステートメント以外のステートメントの開始時に終了します。

ヒント: COPY ステートメントの後に、SELECT または EXCLUDE ステートメントのいずれか(両方ではなく)を使用して、コピーするエントリを制限します。

単一の PROC ステップで(PROC CATALOG ステートメントで指定した 1 つのカタログだけでなく)複数のカタログからエントリをコピーできます。

COPY ステートメントでは、ENTRYTYPE=オプションにスラッシュ(/)は不要です。

例: “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)

構文

COPY OUT=<*libref.*>*catalog* <*options*>;

必須引数

OUT=<*libref.*>*catalog*
エントリのコピー先のカタログに名前を付けます。

オプション引数

ENTRYTYPE=*entry-type*
現在の COPY ステートメントと後続の SELECT または EXCLUDE ステートメントについて、処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目: “ENTRYTYPE=オプション” (160 ページ)

“エントリの種類の指定” (159 ページ)

IN=<*libref.*>*catalog*
コピーするカタログを指定します。

操作 IN=オプションは、PROC CATALOG ステートメントで指定された CATALOG=引数をオーバーライドします。

例 “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)

LOCKCAT=EXCLUSIVE | SHARE
複数のユーザーが同時に同じカタログにコピーできるかどうかを指定します。LOCKCAT=SHARE を使用することにより、カタログ全体ではなく個別のエントリがロックされ、これによりスループットが高まります。デフォルトは LOCKCAT=EXCLUSIVE で、カタログ全体を 1 人のユーザーにロックします。LOCKCAT=SHARE オプションの使用により、単一ユーザー環境で使用される場合はパフォーマンスが低下する可能性があります。各エントリのロックとロック解除と関連付けられているオーバーヘッドのためです。

MOVE
新しいコピーの作成後に元のカタログまたはエントリを削除します。

操作 MOVE オプションによってカタログからすべてのエントリが 削除されると、プロシジャはライブラリからカタログを削除します。

NEW
カタログ(OUT=オプションで指定)がすでに存在する場合は上書きします。NEW オプションを省略すると、PROC CATALOG はカタログを更新します。

参照項目: 連結カタログでの NEW オプションの使用については、“カタログ連結” (161 ページ)を参照してください。

NOEDIT
次の SAS/AF エントリの種類のコピー済バージョンが BUILD プロシジャによって編集されないようにします。

CBT
FRAME
HELP
MENU
PROGRAM
SCL
SYSTEM

制限事項 これらの種類以外のエントリに NOEDIT オプションを指定しても、無視されます。

ヒント 他のユーザー用の SAS/AF アプリケーションを作成する場合は、NOEDIT オプションを使用して特定のカタログエントリが変更されないようにして、アプリケーションを保護します。

例 “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)

NOSOURCE
SAS/AF PROGRAM、FRAME または SCL エントリをコピーする場合にソース行のコピーを省略します。

別名 NOSRC

制限事項 このオプションを PROGRAM、FRAME、SCL エントリ以外のエントリに指定しても、無視されます。

DELETE ステートメント

エントリを SAS カタログから削除します。

制限事項: このプロシジャは、CAS サーバーではサポートされません。

ヒント: いくつかのエントリのみを削除するには、DELETE ステートメントを使用します。削除しないエントリを指定した方が効率的であれば、SAVE を使用します。

複数のエントリを指定できます。複数の DELETE ステートメントを使用することもできます。

参照項目: SAVE ステートメント (155 ページ)

例: “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)

構文

DELETE *entry-1 <entry-2 ...>* </ ENTRYTYPE=*entry-type*>;

必須引数

entry-1 <entry-2 ...>
SAS カタログエントリの名前を 1 つ以上指定します

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目: “ENTRYTYPE=オプション” (160 ページ)

“エントリの種類の指定” (159 ページ)

EXCHANGE ステートメント

2 つのカタログエントリの名前を交換します。

制限事項: EXCHANGE ステートメントの使用時には、カタログエントリの種類は同じである必要があります。

構文

EXCHANGE *name-1=other-name-1*
<*name-2=other-name-2 ...*>
</ ENTRYTYPE=*entry-type*>;

必須引数

name=other-name
プロシジャが交換する 2 つのカタログエントリ名を指定します。

操作 ENTRYTYPE=オプションを PROC CATALOG ステートメントまたは EXCHANGE ステートメントで使用する場合、エントリの種類なしでエントリ名のみ指定できます。

参照項目: “エントリの種類の指定” (159 ページ)

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目: “ENTRYTYPE=オプション” (160 ページ)

“エントリの種類の指定” (159 ページ)

EXCLUDE ステートメント

COPY ステートメントがコピーしないエントリを指定します。

制限事項: EXCLUDE ステートメントには COPY ステートメントが必要です。
EXCLUDE ステートメントを SELECT ステートメントと一緒に使用しないでください。

ヒント: 単一の EXCLUDE ステートメントで複数のエントリを指定できます。
複数の EXCLUDE ステートメントを RUN グループ内の単一の COPY ステートメントと使用できます。

参照項目: COPY ステートメント (150 ページ)および SELECT ステートメント (156 ページ)

例: “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)

構文

EXCLUDE *entry-1 <entry-2 ...>* </ ENTRYTYPE=*entry-type*>;

必須引数

entry-1 <entry-2 ...>
SAS カタログエントリの名前を 1 つ以上指定します

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

参照項目: “エントリの種類の指定” (159 ページ)

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目: “ENTRYTYPE=オプション” (160 ページ)

“エントリの種類の指定” (159 ページ)

MODIFY ステートメント

カタログエントリの説明を変更します。

例: “例 2: コンテンツの表示、名前の変更、説明の変更” (168 ページ)

構文

MODIFY *entry* (DESCRIPTION=<<'>*entry-description*<'>>)
</ ENTRYTYPE=*entry-type*>;

必須引数

entry
SAS カタログエントリの名前を 1 つ指定します。エントリの名前と一緒に種類を指定できます(*entry-name.entry-type*)。

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

参照項目: “エントリの種類の指定” (159 ページ)

DESCRIPTION='*entry-description*'
カタログエントリの説明を最大 256 文字の新しい説明に置き換えるか、すべて削除して変更します。説明を単一引用符または二重引用符で囲みます。

別名 DESC

ヒント MODIFY ステートメントを CATALOG プロシジャで使用している場合に現在の説明を削除するには、DESCRIPTION=オプションをテキストなしで使用します。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目: “ENTRYTYPE=オプション” (160 ページ)

“エントリの種類の指定” (159 ページ)

SAVE ステートメント

SAS カタログから削除しないエントリを指定します。

制限事項: SAVE ステートメントでは KILL オプションの影響を制限できません。

ヒント: SAVE ステートメントを使用して、カタログ内の一部を除くすべてのエントリを削除します。削除するエントリを指定した方が効率的であれば、DELETE ステートメントを使用します。

複数のエントリを指定することも、複数の SAVE ステートメントを使用することもできます。

参照項目: DELETE ステートメント (152 ページ)

構文

SAVE *entry-1 <entry-2 ...>* </ ENTRYTYPE=*entry-type* >;

必須引数

entry-1 <entry-2...>
SAS カタログエントリの名前を 1 つ以上指定します

制限事項 SAVE ステートメントで名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目: “ENTRYTYPE=オプション” (160 ページ)

“エントリの種類の指定” (159 ページ)

SELECT ステートメント

COPY ステートメントがコピーするエントリを指定します。

制限事項: SELECT ステートメントには COPY ステートメントが必要です。
SELECT ステートメントは、EXCLUDE ステートメントと併用できません。

ヒント: 単一の SELECT ステートメントで複数のエントリを指定できます。
複数の SELECT ステートメントを RUN グループ内の単一の COPY ステートメントと使用できます。

参照項目: COPY ステートメント (150 ページ) および EXCLUDE ステートメント (153 ページ)

例: “例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)

構文

SELECT *entry-1 <entry-2 ...>* </ ENTRYTYPE=*entry-type* >;

必須引数

entry-1 <entry-2 ...>
SAS カタログエントリの名前を 1 つ以上指定します

制限事項 名前(*entry-name.entry-type*)を指定する場合や ENTRYTYPE=オプションを使用する場合は、エントリの種類を指定する必要があります。

オプション引数

ENTRYTYPE=*entry-type*
処理を 1 つのエントリの種類に制限します。

別名 ET=

参照項目: “ENTRYTYPE=オプション” (160 ページ).

“エントリの種類の指定” (159 ページ).

QUIT ステートメント

実行されなかったステートメントを実行し、プロシジャを終了します。

構文

QUIT;

使用法: CATALOG プロシジャ

RUN グループを使用した対話型処理

定義

CATALOG プロシジャは対話型です。PROC CATALOG ステートメントをサブミットすると、PROC CATALOG ステートメントを繰り返さずにステートメントまたはステートメントグループを続けてサブミット、実行できます。

RUN ステートメントで終わる一連のプロシジャステートメントは、*RUN グループ*と呼ばれます。指定ステートメントグループで指定される変更は、RUN ステートメントの使用時に有効になります。

PROC CATALOG ステップの終了方法

DATA ステップおよび、ほとんどの SAS プロシジャでは、RUN ステートメントはステップの境界で、ステップを終了します。次のリストには、PROC CATALOG ステップを強制終了するための方法を示します。ただし、単純な RUN ステートメントは対話型プロシジャを終了しません。次のリストには、PROC CATALOG ステップを強制終了するための方法を示します。

- QUIT ステートメントをサブミットします
- RUN ステートメントを CANCEL オプションでサブミットします
- 別の DATA または PROC ステートメントをサブミットします
- セッションを終了します

注: QUIT、DATA、または PROC ステートメントを入力すると、最後の RUN グループの後のステートメントが CATALOG プロシジャの強制終了前に実行されます。RUN ステートメントを CANCEL オプションで入力すると、残りのステートメントはプロシジャの終了前に*実行されません。*

“例 2: コンテンツの表示、名前の変更、説明の変更” (168 ページ)を参照してください。

エラー処理と RUN グループ

エラー処理は、RUN グループへのステートメント分割の部分に基づいています。構文エラーが発生すると、現在の RUN グループのステートメントの 1 つが実行され、次の RUN グループの実行に進みます。

たとえば、次のステーテメントには、スペルミスのある DELETE ステートメントが含まれています。

```
proc catalog catalog=misc entrytype=help;
     copy out=drink;
       select coffee tea;
     del juices;       /* INCORRECT!!! */
     exchange glass=plastic;
  run;
     change calstats=nutri;
  run;
quit;
```

DELETE ステートメントが誤って DEL と指定されているため、PROC CATALOG ステートメント自体を除く、RUN グループのステートメントはすべて 実行されません。CHANGE ステートメントは実行されますが、これは異なる RUN グループにあるためです。

注: 1 つの RUN グループのステートメントが前の RUN グループの影響に依存するバッチジョブの設定時は注意が必要です。特にエントリの削除時と名前変更時です。

エントリの種類の指定

エントリの種類を指定する 4 つの方法

エントリの種類にデフォルト値はないため、入力しないと PROC CATALOG はエラーを生成します。エントリの種類は、次のテーブルの 4 つの方法のうちいずれかを使用して入力できます。

表 9.2 エントリの種類の指定

エントリの種類	例
エントリ名	delete test1.*program* test1.*log* test2.*log*;
ET= (かっこで囲む)	delete test1 *(et=program)*;
ET= (スラッシュの後ろ)[1]	delete test1 (et=program) test1 test2 / *et=log*;
ENTRYTYPE= (スラッシュなし)[2]	proc catalog catalog=mycat et=log; delete test1 test2;

1 従属ステートメントに入力
2 PROC CATALOG または COPY ステートメントに入力

注: CONTENTS ステートメント以外のすべてのステートメントは、ENTRYTYPE=オプションを受け入れます。

ENTRYTYPE=オプションを使用する利点

同じ種類のエントリを複数処理する場合、ENTRYTYPE=オプションによってキー入力を省略することができます。

現在のステップのすべてのステートメントに対してエントリの種類のデフォルトを作成するには、ENTRYTYPE=オプションを PROC CATALOG ステートメントで使用します。現在のステートメントに対してのみデフォルトを設定するには、ENTRYTYPE=オプションを従属ステートメントで使用します。

ある種類のエントリを多数指定し、その他の種類のエントリを少数しか指定しない場合があります。ENTRYTYPE=オプションを使用してデフォルトを指定し、デフォルトを使用するエントリの後に、デフォルトをオーバーライドする個別のエントリを(ENTRYTYPE=)のようにかっこで囲んで指定します。

よくあるエラーの回避

ENTRYTYPE=オプションは、PROC CATALOG ステートメントと従属ステートメントの両方で指定することはできません。たとえば、次のステートメントはエラーを生成し、エントリを削除しません。ENTRYTYPE=オプションの指定が相互に矛盾するためです。

```
  /* THIS IS INCORRECT CODE. */
proc catalog cat=sample et=help;
  delete a b c / et=program;
run;
quit;
```

ENTRYTYPE=オプション

ENTRYTYPE=オプションは、CATALOG プロシジャの CONTENTS ステートメントを除くすべてのステートメントで利用可能です。

ENTRYTYPE=*entry-type*
*(かっこで囲まない場合)*PROC CATALOG ステートメントで使用されると、全体の PROC ステップに対するデフォルトのエントリの種類を設定します。その他すべてのステートメントで、このオプションは現在のステートメントのデフォルトのエントリの種類を設定します。ENTRYTYPE=オプションを省略すると、PROC CATALOG はカタログのすべてのエントリを処理します。NEW オプションを省略すると、PROC CATALOG はカタログを更新します。

別名　ET=

デフォルト　PROC CATALOG はカタログのすべてのエントリを処理します。

操作　ENTRYTYPE=オプションを PROC CATALOG ステートメントで指定する場合、従属ステートメントで ENTRYTYPE=または(ENTRYTYPE=)を指定しないでください。

エントリ名の直後にかっこで囲んだ(ENTRYTYPE=)を指定すると、同じステートメントの ENTRYTYPE=オプションをオーバーライドします。

ヒント PROC CATALOG と COPY ステートメント以外のすべてのステートメントで、ENTRYTYPE=オプションはスラッシュの後に続きます。

単一の PROC CATALOG ステップで複数のエントリの種類を処理するには、ENTRYTYPE=オプションを PROC CATALOG ステートメントではなく、従属ステートメントで使用します。

参照項目: “エントリの種類の指定” (159 ページ)および“例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)

(ENTRYTYPE=*entry-type*)
*(かっこで囲んだ場合)*直前のエントリの種類を特定します。

別名 (ET=)

制限事項 (ENTRYTYPE=)オプションを従属ステートメントのエントリ名の直後に指定しても、PROC CATALOG ステートメントの ENTRYTYPE=オプションをオーバーライドできません。構文エラーが生成されます。

操作 エントリ名の直後に(ENTRYTYPE=)オプションを指定すると、同じステートメントの ENTRYTYPE=オプションをオーバーライドします。

ヒント この形式は主に、従属ステートメントで使用される ENTRYTYPE=オプションの例外を指定する場合に便利です。次のステートメントは、A.Help、B.Format および C.Help を削除します。

```
delete a b (et=format) c / et=help;
```

CHANGE および EXCHANGE ステートメントの場合は、次の例にあるように、各名前ペアに対して 1 回のみ(2 つ目の名前の後ろに)、かっこで囲んだ(ENTRYTYPE=)オプションを指定します。次に、例を示します。

```
change old1=new1 (et=log)
     old1=new2 (et=help);
```

参照項目: “エントリの種類の指定” (159 ページ)、“例 1: 複数のカタログからカタログエントリをコピー、削除、移動する” (164 ページ)および“例 2: コンテンツの表示、名前の変更、説明の変更” (168 ページ)

カタログ連結

カタログ連結について

CATALOG 連結には 2 つの種類があります。1 つ目は LIBNAME ステートメントによって指定され、2 つ目はグローバル CATNAME ステートメントによって指定されます。単一(未連結)のカタログで使用可能なすべてのステートメントとオプションは、カタログ連結で使用できます。

制限事項

連結カタログのコピーに CATALOG プロシジャを使用し、NEW オプションを使用する場合、次のルールが適用されます。

- 入力カタログが連結カタログであり、出力カタログが入力の連結カタログのいずれかのレベルに存在する場合、コピーは実行できません。
- 出力カタログが連結カタログであり、入力カタログが出力の連結カタログの最初のレベルに存在する場合、コピーは実行できません。

たとえば、次のコードは、これらの 2 つのルールとコピーの失敗を示します。

```
libname first 'SAS-library-1';
libname second 'SAS-library-2';
  /* create concat.x */
libname concat (first second);

  /* fails rule #1 */
proc catalog c=concat.x;
  copy out=first.x new;
run;
quit;

  /* fails rule #2 */
proc catalog c=first.x;
  copy out=concat.x new;
run;
quit;
```

次のテーブルに、コピーが可能な場合を示します。テーブルでは、A と B がライブラリで、それぞれにカタログ X が含まれています。 カタログ C は A と B の自動連結で、カタログ D は B と A の自動連結です。

表 9.3　コピーの許可

入力カタログ	出力カタログ	コピー可能か
C.X	B.X	いいえ
C.X	D.X	いいえ
D.X	C.X	いいえ
A.X	A.X	いいえ
A.X	B.X	はい
B.X	A.X	はい
C.X	A.X	いいえ

B.X	C.X	はい
A.X	C.X	いいえ

結果: CATALOG プロシジャ

結果: CATALOG プロシジャ

CATALOG プロシジャは、CONTENTS ステートメントがオプションなしで実行されるときに出力を生成します。このプロシジャ出力には、名前が割り当てられます。この名前を使用して、Output Delivery System (ODS)の使用時にテーブルを参照し、テーブルを選択して出力データセットを作成できます。詳細については、“ODS Table Names Produced by Base SAS Procedures” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

表 9.4　CATALOG プロシジャによって 生成される ODS テーブル

テーブル名	ライブラリの種類
Catalog_Random	カタログがランダムアクセスライブラリにある場合
Catalog_Sequential	カタログが順次ライブラリにある場合

例: CATALOG プロシジャ

例 1: 複数のカタログからカタログエントリをコピー、削除、移動する

要素: PROC CATALOG ステートメントオプション
CAT=
COPY ステートメント
DELETE ステートメント

詳細

この例は、次のタスクを示しています。

- 少数の除外対象エントリの指定によるエントリのコピー
- 少数のコピー対象エントリの指定によるエントリのコピー
- 編集からのエントリの保護
- エントリの移動
- エントリの削除
- 複数のカタログのエントリの処理
- 複数の RUN グループでのエントリの処理

SAS カタログ Perm.Sample には、次のエントリが含まれています。

```
DEFAULT    FORM       Default form for printing
FSLETTER   FORM       Standard form for letters (HP Laserjet)
LOAN       FRAME      Loan analysis application
LOAN       HELP       Information about the application
BUILD      KEYS       Function Key Definitions
LOAN       KEYS       Custom key definitions for application
CREDIT     LOG        credit application log
TEST1      LOG        Inventory program
TEST2      LOG        Inventory program
TEST3      LOG        Inventory program
LOAN       PMENU      Custom menu definitions for applicaticm
CREDIT     PROGRAM    credit application pgm
TEST1      PROGRAM    testing budget applic.
TEST2      PROGRAM    testing budget applic.
```

```
TEST3      PROGRAM    testing budget applic.
LOAN       SCL        SCL code for loan analysis application
PASSIST    SLIST      User profile
```

SAS カタログ Perm.Formats には、次のエントリが含まれています。

```
REVENUE    FORMAT     FORMAT:MAXLEN=16,16,12
DEPT       FORMATC    FORMAT:MAXLEN=1,1,14
```

プログラム

```
libname perm 'SAS-library';

proc catalog cat=perm.sample;
  delete credit.program credit.log;
run;

  copy out=tcatall;
run;

  copy out=testcat;
    exclude test1 test2 test3  passist (et=slist) / et=log;
run;

  copy out=logcat move;
    select test1 test2 test3 / et=log;
run;

  copy out=perm.finance noedit;
    select loan.frame loan.help loan.keys loan.pmenu;
run;

  copy in=perm.formats out=perm.finance;
    select revenue.format dept.formatc;
run;
quit;
```

プログラムの説明

ライブラリ参照を SAS ライブラリに割り当てます。 LIBNAME ステートメントは、ライブラリ参照名 Perm を永続 SAS カタログを含む SAS ライブラリに割り当てます。

```
libname perm 'SAS-library';
```

Perm.Sample カタログから 2 つのエントリを削除します。

```
proc catalog cat=perm.sample;
  delete credit.program credit.log;
run;
```

Perm.Sample カタログのすべてのエントリを Work.TCatAll カタログにコピーします。

```
  copy out=tcatall;
```

```
run;
```

3 つの LOG エントリと Paassist.Slist 以外のすべてを Perm.Sample から Work.TestCat にコピーします。 EXCLUDE ステートメントは、コピーしないエントリを指定します。ET=は、デフォルトの種類を指定します。(ET=)は、デフォルトの種類の例外を指定します。

```
  copy out=testcat;
    exclude test1 test2 test3  passist (et=slist) / et=log;
run;
```

3 つの LOG エントリを Perm.Sample から Work.LogCat に移動させます。 SELECT ステートメントは、移動するエントリを指定します。ET=は、処理を LOG エントリに制限します。

```
  copy out=logcat move;
    select test1 test2 test3 / et=log;
run;
```

4 つの SAS/AF ソフトウェアエントリを Perm.Sample から Perm.Finance にコピーします。 NOEDIT オプションは、これらのエントリが Perm.Finance 内で PROC BUILD によって編集されないように保護します。

```
  copy out=perm.finance noedit;
    select loan.frame loan.help loan.keys loan.pmenu;
run;
```

2 つの出力形式を Perm.Formats から Perm.Finance にコピーします。 IN=オプションを使用して、PROC CATALOG ステートメントで指定されたものとは異なるカタログからコピーできます。COPY および SELECT ステートメントは、QUIT ステートメントが PROC CATALOG ステップを終了する前に実行されます。数値の出力形式と文字の出力形式のエントリの種類に注意してください。REVENUE.FORMAT は数値の出力形式で、DEPT.FORMATC は文字の出力形式です。COPY および SELECT ステートメントは、QUIT ステートメントが PROC CATALOG ステップを終了する前に実行されます。

```
  copy in=perm.formats out=perm.finance;
    select revenue.format dept.formatc;
run;
quit;
```

SAS ログ

例のコード 9.1 PROC CATALOG を使用したエントリのコピー、保護、削除、処理

```
1   libname perm 'SAS-library';
NOTE: Libref PERM was successfully assigned as follows:
      Engine:       V9
      Physical Name: SAS-library\perm
2   proc catalog cat=perm.sample;
NOTE: Writing HTML Body file: sashtml.htm
3      delete credit.program credit.log;
4   run;

NOTE: Deleting entry CREDIT.PROGRAM in catalog PERM.SAMPLE.
NOTE: Deleting entry CREDIT.LOG in catalog PERM.SAMPLE.
5      copy out=tcatall;
6   run;

NOTE: Copying entry DEFAULT.FORM from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry FSLETTER.FORM from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry BUILD.KEYS from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST1.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST2.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST3.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST1.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST2.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST3.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry LOAN.SCL from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry PASSIST.SLIST from catalog PERM.SAMPLE to catalog WORK.TCATALL.
7      copy out=testcat;
8         exclude test1 test2 test3  passist (et=slist) / et=log;
9   run;

NOTE: Copying entry DEFAULT.FORM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry FSLETTER.FORM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry BUILD.KEYS from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry TEST1.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry TEST2.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry TEST3.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry LOAN.SCL from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
10     copy out=logcat move;
11        select test1 test2 test3 / et=log;
12  run;

NOTE: Moving entry TEST1.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.
NOTE: Moving entry TEST2.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.
NOTE: Moving entry TEST3.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.
13     copy out=perm.finance noedit;
14        select loan.frame loan.help loan.keys loan.pmenu;
15  run;
```

```
NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog PERM.FINANCE.
NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog PERM.FINANCE.
NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog PERM.FINANCE.
NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog PERM.FINANCE.
16     copy in=perm.formats out=perm.finance;
17       select revenue.format dept.formatc;
18   quit;

NOTE: Copying entry REVENUE.FORMAT from catalog PERM.FORMATS to catalog PERM.FINANCE.
NOTE: Copying entry DEPT.FORMATC from catalog PERM.FORMATS to catalog PERM.FINANCE.
```

例 2: コンテンツの表示、名前の変更、説明の変更

要素:

PROC CATALOG ステートメントオプション
- CATALOGS=
- CHANGE ステートメント
- CONTENTS ステートメント
- MODIFY ステートメント

TITLE ステートメント

詳細

この例は、次のタスクを示しています。

- カタログのエントリをリストして出力をファイルに送る
- エントリの名前の変更
- エントリの説明の変更
- 複数の RUN グループでのエントリの処理

プログラム

```
libname perm 'SAS-library';

proc catalog catalog=perm.finance;
  contents;
title1 'Contents of PERM.FINANCE before changes are made';
run;

  change dept=deptcode (et=formatc);
run;

  modify loan.frame (description='Loan analysis app. - ver1');
  contents;
title1 'Contents of PERM.FINANCE after changes are made';
run;
```

```
quit;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME ステートメントは、ライブラリ参照を永続 SAS カタログを含む SAS ライブラリに割り当てます。

```
libname perm 'SAS-library';
```

カタログのエントリをリストし、ファイルに出力を送ります。 CONTENTS ステートメントは、SAS カタログ Perm.Finance のコンテンツのリストを作成し、ファイルに出力を送ります。

```
proc catalog catalog=perm.finance;
  contents;
title1 'Contents of PERM.FINANCE before changes are made';
run;
```

エントリ名を変更します。 CHANGE ステートメントは、ユーザー作成の文字の出力形式を含むエントリの名前を変更します。(ET=)は、エントリの種類を指定します。

```
  change dept=deptcode (et=formatc);
run;
```

複数の RUN グループでエントリを処理します。 MODIFY ステートメントはエントリの説明を変更します。すべての変更の適用後、CONTENTS ステートメントは Perm.Finance のコンテンツのリストを作成します。QUIT はプロシジャを終了します。

```
  modify loan.frame (description='Loan analysis app. - ver1');
  contents;
title1 'Contents of PERM.FINANCE after changes are made';
run;

quit;
```

出力例

アウトプット 9.1　変更前後の Perm.Finance のコンテンツ

Contents of PERM.FINANCE before changes are made

Contents of Catalog PERM.FINANCE					
#	Name	Type	Create Date	Modified Date	Description
1	REVENUE	FORMAT	10/16/1996 13:48:11	10/16/1996 13:48:11	FORMAT:MAXLEN=16,16,12
2	DEPTCODE	FORMATC	10/30/1996 13:40:42	10/30/1996 13:40:42	FORMAT:MAXLEN=1,1,14
3	LOAN	FRAME	10/30/1996 13:40:43	02/18/2014 13:31:25	Loan analysis app. - ver1
4	LOAN	HELP	10/16/1996 13:48:10	10/16/1996 13:48:10	Information about the application
5	LOAN	KEYS	10/16/1996 13:48:10	10/16/1996 13:48:10	Custom key definitions for application
6	LOAN	PMENU	10/16/1996 13:48:10	10/16/1996 13:48:10	Custom menu definitions for application
7	LOAN	SCL	10/16/1996 13:48:10	10/16/1996 13:48:10	SCL code for loan analysis application

Contents of PERM.FINANCE after changes are made

Contents of Catalog PERM.FINANCE					
#	Name	Type	Create Date	Modified Date	Description
1	REVENUE	FORMAT	10/16/1996 13:48:11	10/16/1996 13:48:11	FORMAT:MAXLEN=16,16,12
2	DEPTCODE	FORMATC	10/30/1996 13:40:42	10/30/1996 13:40:42	FORMAT:MAXLEN=1,1,14
3	LOAN	FRAME	10/30/1996 13:40:43	03/28/2014 10:26:50	Loan analysis app. - ver1
4	LOAN	HELP	10/16/1996 13:48:10	10/16/1996 13:48:10	Information about the application
5	LOAN	KEYS	10/16/1996 13:48:10	10/16/1996 13:48:10	Custom key definitions for application
6	LOAN	PMENU	10/16/1996 13:48:10	10/16/1996 13:48:10	Custom menu definitions for application
7	LOAN	SCL	10/16/1996 13:48:10	10/16/1996 13:48:10	SCL code for loan analysis application

例 3: FORCE オプションと KILL オプションの併用

要素:

PROC CATALOG ステートメントオプション
- CATALOG=
- FORCE
- KILL

%MACRO ステートメント

%MEND ステートメント

%PUT ステートメント

詳細

この例は、次のタスクを示しています。

- リソース環境の作成
- KILL オプションによってすべてのカタログエントリの 削除を試行するが、エラーを受け取る
- KILL オプションによってすべてのカタログエントリが 正常に削除されるように、FORCE オプションを指定する

プログラム

```
 %macro matt;
   %put &syscc;
%mend matt;

proc catalog c=work.sasmacr kill;
run;
quit;

proc catalog c=work.sasmacr kill force;
run;
quit;
```

プログラムの説明

プロセスを開始します(リソース環境)。 これは、Work.Sasmacr カタログのカタログエントリ MATT を開いて実行します。

```
 %macro matt;
   %put &syscc;
%mend matt;
```

KILL オプションを指定して、Work.Sasmacr のすべてのカタログエントリを 削除します。 リソース環境(カタログを使用したプロセス)があるため、KILL は機能せず、エラーがログに送信されます。

```
proc catalog c=work.sasmacr kill;
run;
quit;
```

カタログエントリを削除するために、KILL オプションに FORCE オプションを指定します。

```
proc catalog c=work.sasmacr kill force;
run;
quit;
```

ログの例

例のコード 9.2　KILL オプションによる SAS ログへのエラー送信

```
1    %macro matt;
2      %put &syscc;
3     %mend matt;
4
5   proc catalog c=work.sasmacr kill;
NOTE: Writing HTML Body file: sashtml.htm
6   run;

ERROR: You cannot open WORK.SASMACR.CATALOG for update access because WORK.SASMACR.CATALOG is
in
use by you in resource environment _O_TAGS.
WARNING: Command CATALOG not processed because of errors noted above.
7   quit;

NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE CATALOG used (Total process time):
      real time          6.46 seconds
      cpu time           0.62 seconds
```

例のコード 9.3　KILL オプションへの FORCE オプションの追加によるカタログエントリの削除

```
8   proc catalog c=work.sasmacr kill force;
9   run;

NOTE: Deleting entry MATT.MACRO in catalog WORK.SASMACR.
10  quit;

NOTE: PROCEDURE CATALOG used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
```

10

CIMPORT プロシジャ

概要: CIMPORT プロシジャ

CIMPORT プロシジャの機能

注: このセクションでは、PROC CPORT と PROC CIMPORT が SAS Viya プラットフォームで実行されると述べている場合、これらのプロシジャは計算サーバーで実行されるという意味になります。

CIMPORT プロシジャは、CPORT プロシジャによって作成(*エクスポート*)された移送ファイルを *インポート*します。PROC CIMPORT は、移送ファイルを SAS カタログ、SAS データセットまたは SAS ライブラリの元の形式に戻します。*移送ファイル*は SAS ライブラリ、SAS カタログ、または SAS データセットを移送出力形式で保持している順編成ファイルです。PROC CPORT が書き出す移送出力形式は、すべての環境およびすべての SAS リリースを通して同じです。

PROC CIMPORT は、SAS ファイルの *変換*も行います。つまり、SAS ファイルの出力形式を SAS のバージョン間で適切な SAS 出力形式に変更します。たとえば、 PROC CPORT と PROC CIMPORT を使用して、ファイルを SAS の以前のリリースから以降のリリースに(SAS®9 から SAS Viya など) 、または同じバージョン間(SAS Viya 動作環境間など)で移動できます。PROC CIMPORT は自動で移送ファイルを変換して、それをインポートします。

注: PROC CIMPORT と PROC CPORT を使用して、グラフィックカタログをバックアップできます。PROC COPY はグラフィックカタログのバックアップには使用できません。

PROC CIMPORT は出力を生成しませんが、ノートを SAS ログに書き込みます。

移送ファイルの作成処理と読み込み処理

次に、ソースコンピューターで移送ファイルを作成して、それをターゲットコンピューターで読み込むプロセスを示します。

1 移送ファイルは、ソースコンピューターで PROC CPORT を使って作成されます。

2 移送ファイルは、ソースコンピューターからターゲットコンピューターへ移送されます。

3 移送ファイルは、ターゲットコンピューターで PROC CIMPORT を使って読み込まれます。

   注: 移送ファイルは、PROC CPORT を使って作成されたものと、XPORT エンジンを使って作成されたものとで、互換性はありません。

移送ファイルの作成(PROC CPORT)、移送ファイルの転送、移送ファイルの復元(PROC CIMPORT)のステップの詳細については、*SAS ファイルの移動とアクセス*を参照してください。

UTF-8 へのデータの変換

注: このセクションでは、PROC CPORT と PROC CIMPORT が SAS Viya プラットフォームで実行されると述べている場合、これらのプロシジャは計算サーバーで実行されるという意味になります。

計算サーバーは、UTF-8 およびその他の従来のエンコーディングをサポートしています。CIMPORT プロシジャは、移送ファイルに十分なエンコード情報があれば、UTF-8 ではないデータセットを UTF-8 セッションに正常に変換できます。UTF-8 は他のエンコーディングよりも多くの可変長を必要とするため、EXTENDVAR=オプションを指定すると、データが切り捨てられるのを防ぐのに役立ちます。EXTENDFORMAT=オプションを指定すると、データの変換に十分な出力形式の幅を確保できます。

注: サポートされている形式は SAS 出力形式のみです。ユーザー定義出力形式はサポートされません。

データを UTF-8 エンコーディングに移行する方法の詳細については、次のリソースを参照してください。

- “UTF-8 Encoding” (*Migrating Data to UTF-8 for the SAS Viya Platform*)
- “Read External Files” (*Migrating Data to UTF-8 for the SAS Viya Platform*)

SAS Studio でエンコーディングを設定する方法については、“Opening a Program” (*SAS Studio: User's Guide*)を参照してください。

構文: CIMPORT プロシジャ

ヒント: グラフィックカタログをバックアップするには、PROC CIMPORT または PROC CPORT を使用してください。PROC COPY はグラフィックカタログのバックアップには使用できません。

参照項目: 移送ファイルの作成(PROC CPORT)、移送ファイルの転送、移送ファイルの復元(PROC CIMPORT)のステップの詳細については、*SAS ファイルの移動とアクセス*を参照してください。

PROC CIMPORT*destination=libref | <libref.> member-name*<*options*>;

EXCLUDE *SAS file(s) | catalog entry(s)*< / MEMTYPE=*mtype*> < / ENTRYTYPE=*entry-type*>;

SELECT *SAS file(s) | catalog entry(s)*< / MEMTYPE=*mtype*> < / ENTRYTYPE=*entry-type*>;

ステートメント	タスク	例
“PROC CIMPORT ステートメント”	移送ファイルをインポートします	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“EXCLUDE ステートメント”	1 つ以上の指定ファイルをインポートプロセスから除外します	
“SELECT ステートメント”	1 つ以上のファイルまたはエントリをインポートプロセスに指定します。	Ex. 2

PROC CIMPORT ステートメント

移送ファイルをインポートします。

例: “例 1: ライブラリ全体のインポート” (196 ページ)
“例 2: 個々のカタログエントリのインポート” (197 ページ)
“例 3: 単一インデックス付き SAS データセットのインポート” (198 ページ)
“例 4: PROC CIMPORT を使用してフランス語のデータセットを UTF-8SAS セッションにインポートする” (200 ページ)

構文

PROC CIMPORT*destination=libref | <libref.> member-name*<*options*>;

オプション引数の要約

COMPRESS=NO | CHAR | BINARY
: 生成した CIMPORT データセットを、どのように圧縮するかを指定します。

FORCE
: ロックされているカタログへのアクセスが可能になります。

NEW
: インポートされた移送ファイルに対し新しいカタログを作成し、同じ名前の既存カタログを削除します。

NOEDIT
: SAS/AF PROGRAM および SCL エントリを編集機能なしでインポートします。

NOSRC
: コンパイルされた SCL コードを含む SAS/AF エントリに対するソースコードのインポートを行いません。

SORT
: PROC CIMPORT を使用して並べ替えられたデータセットをインポートするときに必要に応じて出力データセットをもう一度並べ替えます。

Control the contents of the transport file

EXTENDFORMAT=YES | NO
: 文字出力形式の幅を拡張するかどうかを指定します。

EXTENDSN=YES | NO
: 整数型(short-integer) (8 バイト未満)の長さをインポート時に 1 バイト拡張するかどうかを指定します。

EXTENDVAR=*multiplier* | AUTO
: 文字変数の長さを拡張する乗数値を指定します。

UPCASE
: アルファベット文字を大文字で出力ファイルに書き込みます。

Identify the input transport file

INFILE=*fileref* | '*filename*'
: 読み込む移送ファイルの事前に定義されているファイル参照名またはファイル名を指定します。

TAPE
: 入力移送ファイルをテープから読み込みます。

Look at the encoding of the transport file

ENCODINGINFO=ALL | *n*
: データセットのエンコーディング値をログに出力するために読み込むデータセットヘッダーの数を指定します。

ISFILEUTF8=YES | NO
: ファイルが UTF-8 形式にエンコードされるかどうかを指定します。

Select files to import

EET=(*etype(s)*)
: 指定したエントリの種類をインポートプロセスから除外します。

ET=(*etype(s)*)
: インポートするエントリの種類を指定します。

MEMTYPE=*mtype*
: ライブラリのインポート時にデータセットのみ、カタログのみ、または両方が移動させるように指定します。

必須引数

destination=libref | *<libref.>member-name*
: インポートするファイルの種類を特定し、インポートするカタログ、SAS データセット、または SAS ライブラリを指定します。

destination
: 単一カタログ、単一 SAS データセット、または SAS ライブラリのメンバーとして移送ファイルのファイルを特定します。*destination* 引数は、次のうちいずれかになります。

CATALOG=
: ファイルの種類が SAS カタログであることを指定します。

別名 C=

CAT=

DATA=
: ファイルの種類が SAS データセットであることを指定します。

別名 D=

DS=

LIBRARY=
: ファイルの種類が SAS ライブラリあることを指定します。

別名 L=

LIB=

libref
<libref.>member-name
移送ファイルの出力先として、特定のカタログ、SAS データセット、SAS ライブラリを指定します。*destination* 引数が CATALOG または DATA の場合、*libref* とメンバー名の両方を指定できます。*libref* が省略された場合、PROC CIMPORT はデフォルトのライブラリ(通常は WORK ライブラリ)を *libref* として使用します。*destination* 引数が LIBRARY の場合、*libref* のみを指定します。

参照項目: 名前とメンバー名に使用できる命名規則については、"SAS 名" (*SAS プログラマガイド: エッセンシャル*)を参照してください。

オプション引数

COMPRESS=NO | CHAR | BINARY
生成した CIMPORT データセットを圧縮するかどうかを指定します。使用されている圧縮タイプを指定できます。

NO
CIMPORT によって生成されたデータセットが圧縮されないことを指定します。

別名 N

OFF

CHAR
新しく作成された SAS データセット内でオブザベーションは SAS により RLE (Run Length Encoding)を使用して圧縮されること(可変長レコードの作成)を指定します。RLE では、連続する同じ文字(空白を含む)を 2 バイトまたは 3 バイトの表現に削減することでオブザベーションが圧縮されます。

別名 ON

YES

Y

BINARY
新しく作成された SAS データセット内でオブザベーションは RDC (Ross Data Compression)を使用して圧縮されること(可変長レコードの作成)を指定します。RDC では、Run Length Encoding とスライディングウィンドウ圧縮を組み合わせて反復バイトパターンをより効果的に表現することでファイルが圧縮されます。

注: この方式は、中サイズから大サイズ(数百バイトまたはそれ以上)のバイナリデータ(文字変数と数値変数)のブロックを圧縮する場合に有効です。この

圧縮関数は一度に 1 つのレコードに対してのみ動作するため、効果的に圧縮するには数百バイト以上のレコード長が必要です。

参照項目: 移送ファイルの作成と復元については、*SAS ファイルの移動とアクセス*を参照してください。

例
```
proc cimport file=transportFile lib=myLib compress=char;
run;
```

EET=(*etype(s)*)
指定したエントリの種類をインポートプロセスから除外します。*etype* が単一エントリの種類の場合は、かっこは省略できません。複数の値はスペースで区切ります。

操作 同じ PROC CIMPORT ステップで、EET=オプションと ET=オプションを同時に使用できません。

ENCODINGINFO=ALL | *n*
データセットのエンコーディング値をログに出力するために読み込むデータセットヘッダーの数を指定します。

ALL
すべてのデータセットヘッダーを読み込むことを指定します。データセットヘッダーの格納されたエンコーディング値は SAS ログに出力されます。

n
ゼロより大きい整数を指定します。この値は、読み込むデータセットヘッダーの数を表します。データセットヘッダーの格納されたエンコーディング値は SAS ログに出力されます。

範囲 ゼロより大きい整数を指定します。上限はシステムの制約によって異なります。*n* が移送ファイルに含まれるデータセットの数よりも大きい場合、すべてのデータが処理されます。

注 データセットヘッダーの読み取り中にライブラリにカタログが見つかった場合、カタログはスキップされます。カタログヘッダーはエンコーディング値を含んでいません。

参照項目: 移送ファイルの作成と復元については、*SAS ファイルの移動とアクセス*を参照してください。

例
```
proc cimport lib=work file='mixed' encodinginfo=3;
run;

NOTE: The CPORTed data set ASCII transport file encoding=us-ascii.
NOTE: The CPORTed data set JISDS transport file encoding=wlatin1.
NOTE: The CPORTed data set UTF8 transport file encoding=wlatin1.
NOTE: The CPORTed data set WLATIN1DS transport file encoding=wlatin1.
```

ET=(*etype(s)*)
インポートするエントリの種類を指定します。*etype* が単一エントリの種類の場合は、かっこは省略できません。複数の値はスペースで区切ります。

操作 同じ PROC CIMPORT ステップで、EET=オプションと ET=オプションを同時に使用できません。

EXTENDSN=YES | NO
整数型(short-integer) (8 バイト未満)の長さをインポート時に 1 バイト拡張するかどうかを指定します。拡張すると、整数型(short-integer)を IBM 形式または IEEE 形式で移送するときに精度の損失を回避できます。8 バイトの整数型(short-integer)の長さは拡張できません。

デフォルト YES

制限事項 このオプションは、データセットにのみ適用されます。

ヒント 小数値を整数型(short-integer)として保存しないでください。

EXTENDFORMAT=YES | NO
文字出力形式の幅を拡張するかどうかを指定します。

別名 Y

N

デフォルト YES

制限事項 このオプションは SAS Viya でサポートされていますが、SAS 9 ではサポートされていません。

EXTENDFORMAT=は、EXTENDVAR=オプションとともに使用されます。

EXTENDVAR=*multiplier* | AUTO
文字変数の長さを拡張する乗数値を指定します。拡張された長さは、トランスコーディングが必要な SAS データファイルを処理するときに、文字データが切り捨てられないようにするのに役立ちます。

multiplier
1 から 5 までの乗数値を指定するか、AUTO を指定できます。文字変数の長さは、指定した値を現在の長さに乗算して増やします。デフォルト値は 1 です。1 を使用すると、可変長は拡張されません。AUTO を指定すると、乗数が自動的に設定されます。

Auto
AUTO は、*乗数*を設定するために使用されます。この値は、セッションのエンコーディングと移送ファイルのデータセットによって異なります。INFILE=で指定されたファイルが ANSI エンコードであるか、文字エンコード ID(CEI)情報がない場合、乗数値として 1 が使用されます。それ以外の場合は、拡張された長さが自動的に計算されます。

文字列が切り捨てられると、警告メッセージが生成されます。このメッセージは、EXTENDVAR オプションを使用するように促します。

デフォルト 1.デフォルトを使用する場合、可変長は拡張されません。

制限事項 このオプションは SAS Viya でサポートされていますが、SAS 9 ではサポートされていません。

このオプションは、データセットにのみ適用されます。

SAS 文字変数のデータ型に NOTRANSCODE 属性が指定されている場合、このオプションは有効になりません。

PROC CIMPORT は VARCHAR をサポートしていません。

注 PROC CPORT で ASIS オプションが指定されている場合、EXTENDVAR= オプションは無視されます。

ISFILEUTF8=YES | NO
UTF-8 として移送ファイルに含まれるデータセットのエンコーディングを明示的に指定します。データセットエンコーディングは SAS 9.2 移送ファイルに記録(またはスタンプ)されますが、エンコーディングは 9.2 より前の SAS リリースを使用して作成された移送ファイルにはスタンプされません。そのため、UTF-8 エンコーディングの指定は、次の状況下で有用です。

- 移送ファイルのデータセットが、9.2 より前の SAS リリースを使用して作成された。
- データセットが UTF-8 としてエンコードされている。

ターゲット環境の移送ファイルを復元する場合、その復元作業の前に移送ファイルの説明がなければなりません。

yes
移送ファイルのデータセットが UTF-8 としてエンコードされるように指定します。

別名 YES

Y

TRUE

T

no
移送ファイルのデータセットが UTF-8 としてエンコードされないように指定します。NO がデフォルトです。

別名 NO

N

FALSE

F

ターゲット SAS セッションの移送ファイルを正常にインポートするために、移送ファイルに関する次の情報がなければなりません。

- ソース動作環境。たとえば、**Linux**。
- SAS リリース。たとえば、**SAS Viya 4.0**。

- トランスポートファイルの名前。たとえば、 **tport.dat** 。
- 文字データのエンコーディング。たとえば、**wlatin1**
- 文字データの各国語。たとえば、アメリカ英語(または **en_US**)

デフォルト NO

制限事項 PROC CIMPORT がこのオプションを使用するのは、移送ファイルがデータセットのエンコーディングによってスタンプされない 場合のみです。エンコーディングは、9.2 より前の SAS リリースでは記録されませんでした。エンコーディングが移送ファイルに記録され、ISFILEUTF8=オプションが PROC CIMPORT で指定されると、ISFILEUTF8=は無視されます。

参照項目: “CIMPORT の問題: トランスポートファイルのインポート” (187 ページ)

移送ファイルの作成と復元については、*SAS ファイルの移動とアクセス* を参照してください。

FORCE
ロックされているカタログへのアクセスが可能になります。PROC CIMPORT はデフォルトで、更新中のカタログにその他のユーザーがアクセスしないようにするために、更新中のカタログをロックします。FORCE オプションはこのロックを無効にします。これにより、その他のユーザーはインポート中のカタログにアクセスできるようになります。また、その他のユーザーが現在アクセスしているカタログをインポートできるようになります。

注意
FORCE オプションにより、予期せぬ結果が発生する可能性があります。 FORCE オプションを指定すると、複数のユーザーが同じカタログエントリに同時にアクセスできるようになります。

INFILE=*fileref* | '*filename*'
読み込む移送ファイルの事前に定義されているファイル参照名またはファイル名を指定します。INFILE=オプションを省略すると、PROC CIMPORT はファイル参照名 SASCAT で移送ファイルから読み込もうとします。ファイル参照名 SASCAT が存在しない場合、PROC CIMPORT は、SASCAT.DAT という名前のファイルから読み込もうとします。

別名 FILE=

例 “例 1: ライブラリ全体のインポート” (196 ページ)

MEMTYPE=*mtype*
データセットのみ、カタログのみ、または両方が移送ファイルからインポートされるように指定します。*mtype* に可能な値は、次のとおりです。

別名 MT=

ALL
カタログとデータセットの両方をインポートすることを指定します。

CATALOG
カタログのみをインポートするように指定します。

別名 CAT

DATA
データセットのみをインポートするように指定します。

別名 DS

NEW
指定する出力先の名前が既存カタログと同じ場合に、インポートされた移送ファイルのコンテンツを含む新しいカタログを作成します。NEW は、インポートの出力先として指定する名前と同じ名前の既存カタログを削除します。NEW を指定せず、指定する出力先名が既存カタログ名と同じ場合、PROC CIMPORT はインポートされた移送ファイルを既存カタログに追加します。

NOEDIT
SAS/AF PROGRAM および SCL エントリを編集機能なしでインポートします。

SAS/AF ソフトウェアの BUILD プロシジャで MERGE ステートメントで NOEDIT オプションを使用して SCL コードを含む新規カタログを作成する場合と同じ結果が得られます。

注: NOEDIT オプションは SAS/AF PROGRAM と SCL エントリにのみ影響します。FSEDIT SCREEN および FSVIEW FORMULA エントリには影響しません。

別名 NEDIT

NOSRC
コンパイルされた SCL コードを含む SAS/AF エントリに対するソースコードのインポートを行いません。

SAS/AF ソフトウェアの BUILD プロシジャで MERGE ステートメントで NOSOURCE オプションを使用して SCL コードを含む新規カタログを作成する場合と同じ結果が得られます。

別名 NSRC

操作 NOSRC オプションは FRAME、PROGRAM、SCL 以外のエントリの種類と使用した場合、PROC CIMPORT に無視されます。

SORT
PROC CIMPORT を使用して並べ替えられたデータセットをインポートするときに必要に応じて出力データセットをもう一度並べ替えます。インポートするデータセットが 1 つ以上の文字変数によって並べ替えられ、ターゲット SAS セッションの文字値の順序がソースセッションと異なる場合、もう一度並べ替える必要があります。

注: CIMPORT SORT オプションは、並べ替え情報を含んでいないデータセットに影響しません。このオプションは、以前に並べ替えられたデータセットにのみ適用されます。

文字値の順序、つまり、照合順序は、SAS および SORT プロシジャを起動するときに使用されるオプションによって使用されるセッションエンコーディングと変換テーブルに応じて異なります。必要に応じて、インポート時にもう一度並べ替えを実行します。データセットのソートインジケーターは保持されます。

詳細については、"照合順序" (*SAS 各国語サポート(NLS): リファレンスガイド*)および 55 章, "SORT プロシジャ," (2027 ページ)を参照してください。

PROC CIMPORT がデータセットをもう一度並べ替えると、次の情報が出力されます。

```
NOTE: PROC CIMPORT re-sorted the data set WORK.TEMP because it contained
    character variables in the sort key, and the collating sequence of
    the sort differs from the local host.
```

例 proc cimport '*transportfile*.tpt' lib=*library* sort; run;

TAPE
入力移送ファイルをテープから読み込みます。

デフォルト PROC CIMPORT はディスクから読み込みます。

UPCASE
移送ファイルから読み込み、アルファベット文字を大文字で出力ファイルに書き込みます。

制限事項 UPCASE オプションは、ダブルバイト文字セット(DBCS)拡張が SAS に付随している場合のみ使用できます。

ヒント PROC CPORT は、すべての大文字を含む移送ファイルの作成に使用できます。詳細については"OUTTYPE=UPCASE" (344 ページ)を参照してください。

EXCLUDE ステートメント

指定したファイルまたはエントリをインポートプロセスから除外します。

操作: PROC CIMPORT ステップでは EXCLUDE ステートメントまたは SELECT ステートメントを使用できますが、両方を同時には使用できません。

ヒント: PROC CIMPORT 一回の起動で、使用できる EXCLUDE ステートメントの数に制限はありません。

構文

EXCLUDE *SAS file(s)* | *catalog entry(s)*< / MEMTYPE=*mtype*>
< / ENTRYTYPE=*entry-type*>;

必須引数

SAS file(s) | catalog entry(s)
インポートプロセスから除外する SAS ファイルまたはカタログエントリを指定します。SAS ライブラリのインポート時は SAS ファイル名を指定し、個別 SAS カタログのインポート時はカタログエントリ名を指定します。複数のファイル名またはエントリ名はスペースで区切ります。EXCLUDE ステートメントでは、ショートカットを使って、多数の似ている名前をリストすることができます。詳細については、"SAS ファイル" (*SAS プログラマガイド: エッセンシャル*)を参照してください。

オプション引数

ENTRYTYPE=*entry-type*
EXCLUDE ステートメントにリストされる 1 つ以上のカタログエントリに対し、単一のエントリの種類を指定します。参照: "Catalog Names and Catalog Entries" (*SAS V9 LIBNAME Engine: Reference*).

別名 ETYPE=

ET=

制限事項 ENTRYTYPE=は個別の SAS カタログをインポートする時にのみ有効です。

MEMTYPE=*mtype*
EXCLUDE ステートメントにリストされている 1 つ以上の SAS ファイルに対して、単一のメンバーの種類を指定します。*mtype* の値は、次のどれかになります。

ALL
カタログとデータセットの両方をインポートすることを指定します。

CATALOG
カタログのみをインポートするように指定します。

DATA
データセットのみをインポートするように指定します。

ファイル名の後に、MEMTYPE= option をかっこで囲んで指定することもできます。かっこの中で、MEMTYPE=はそのすぐ前にあるファイル名の種類を識別します。オプションのこの形式を使用する場合は、EXCLUDE ステートメントでスラッシュの後に続く MEMTYPE=オプションを無効にしますが、PROC CIMPORT ステートメントの MEMTYPE=オプションと一致している必要があります。

別名 MTYPE=

MT=

デフォルト ALL

制限事項 MEMTYPE=は、SAS ライブラリのインポート時にのみ有効です。

SELECT ステートメント

インポートする個別のファイルまたはエントリを指定します。

操作: PROC CIMPORT ステップでは EXCLUDE ステートメントまたは SELECT ステートメントを使用できますが、両方を同時には使用できません。

ヒント: PROC CIMPORT 一回の起動で、使用できる SELECT ステートメントの数に制限はありません。

例: “例 2: 個々のカタログエントリのインポート” (197 ページ)

構文

SELECT *SAS file(s) | catalog entry(s)*< / MEMTYPE=*mtype*>
< / ENTRYTYPE=*entry-type*>;

必須引数

SAS file(s) | catalog entry(s)
インポートする SAS ファイルまたはカタログエントリを指定します。SAS ライブラリのインポート時は SAS ファイル名を指定し、個別 SAS カタログのインポート時はカタログエントリ名を指定します。複数のファイル名またはエントリ名はスペースで区切ります。SELECT ステートメントでは、ショートカットを使って、多数の似ている名前をリストすることができます。詳細については、“SAS ファイル” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

オプション引数

ENTRYTYPE=*entry-type*
SELECT ステートメントにリストされる 1 つ以上のカタログエントリに対し、単一のエントリの種類を指定します。参照: “Catalog Names and Catalog Entries” (*SAS V9 LIBNAME Engine: Reference*).

別名 ETYPE=

ET=

制限事項 ENTRYTYPE=は個別の SAS カタログをインポートする時にのみ有効です。

MEMTYPE=*mtype*
SELECT ステートメントにリストされている 1 つ以上の SAS ファイルに対して、単一のメンバーの種類を指定します。有効な値は CATALOG または CAT、DATA、または ALL です。

ファイル名の後に、MEMTYPE= option をかっこで囲んで指定することもできます。かっこの中で、MEMTYPE=はそのすぐ前にあるファイル名の種類を識別します。オプションのこの形式を使用する場合は、SELECT ステートメントでスラッ

シュの後に続く MEMTYPE=オプションを無効にしますが、PROC CIMPORT ステートメントの MEMTYPE=オプションと一致している必要があります。

ALL
カタログとデータセットの両方をインポートすることを指定します。

CATALOG
カタログのみをインポートするように指定します。

DATA
データセットのみをインポートするように指定します。

別名 MTYPE=

MT=

デフォルト ALL

制限事項 MEMTYPE=は、SAS ライブラリのインポート時にのみ有効です。

使用法: CIMPORT プロシジャ

CIMPORT の問題: トランスポートファイルのインポート

移送ファイルとエンコーディングについて

移送ファイルの文字データは、次の種類のエンコーディングで作成されます。

- 移送ファイルが作成される SAS セッションの UTF-8 エンコーディング。
- UTF-8 セッションで PROC CIMPORT を使用してデータセットをインポートすると、UTF-8 エンコーディングが維持されます。
- PROC CIMPORT は、非 UTF-8 SAS セッションで作成されたデータセットを UTF-8 SAS セッションにインポートする機能をサポートしています。
- PROC CPORT を使用して移送ファイル(ASCII プラットフォームでは US-ASCII でエンコードされる)を作成すると、セッションエンコーディングに関係なく、その移送ファイルに対する US-ASCII エンコーディングが保持されます。その後、PROC CIMPORT を使用して、そのデータセットを ASCII プラットフォームに移送した場合、その移送ファイルに対する US-ASCII エンコーディングが保持されます。作成されたデータセットには、セッションエンコーディングではなく、

US-ASCII エンコーディングが含まれます。たとえば、セッションエンコーディングが WLATIN1 の場合に、PROC CPORT を使用して、US-ASCII のエンコーディングを含むデータセットを作成したとします。移送ファイルでは、WLATIN1 エンコーディングではなく、US-ASCII エンコーディングが保持されます。このデータセットに PROC CIMPORT を使用した場合もこの保持が行われます。PROC CIMPORT を使用してデータセットを ASCII プラットフォームに移送する場合、US-ASCII エンコーディングが保持されトランスコードされません。

注: US-ASCII エンコーディングの保持は、ASCII プラットフォームでのみ行われます。

次の例に、エンコーディングの移送ファイルへの適用方法を示します。

SAS Studio でエンコーディングを設定する方法については、"Opening a Program" (*SAS Studio: User's Guide*)を参照してください。

表 10.1　移送ファイルへのエンコーディングの割り当て

移送ファイルのエンコーディング値	SAS 起動	説明
utf-8 または us-ascii	エンコーディングは utf8	SAS セッションは、UTF-8 エンコーディングを使用して起動されます。セッションエンコーディングは、移送ファイルに適用されます。データセットが US-ASCII の場合、セッションエンコーディングではなく、US-ASCII エンコーディングが保持されます。
wlatin2	ロケールは pl_PL	SAS セッションは、Polish Poland ロケールに関連付けられているエンコーディング、LATIN2 を使用して起動されます。

各ロケールに関連付けられているエンコーディングの完全なリストについては、ロケールテーブル(*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

移送ファイルをインポートするためには、移送元と移送先の SAS セッションのエンコーディングに互換性がなければなりません。次に、互換性のあるソースおよびターゲット SAS セッションの例を示します。

表 10.2 互換性のあるエンコーディング s

ソース SAS セッション			ターゲット SAS セッション	
ロケール	Linux の SAS セッションエンコーディング	移送ファイルエンコーディング	ロケール	SAS セッションエンコーディング
es_MX (Spanish Mexico)	latin1	wlatin1	it_IT (Italian Italy)	wlatin1

ソースおよびターゲット SAS セッションのエンコーディングは、互換性があります。es_MX ロケールに対するデフォルトエンコーディングが WLATIN1 で、ターゲット SAS セッションのエンコーディングが WLATIN1 であるためです。

ただし、ソースおよびターゲット SAS セッションのエンコーディングに互換性がない場合、移送ファイルが正常にインポートされないことがあります。次に、互換性のないエンコーディングの例を示します。

表 10.3 互換性のないエンコーディング

ソース SAS セッション			ターゲット SAS セッション	
ロケール	LINUX SAS セッションエンコーディング	移送ファイルエンコーディング	ロケール	エンコーディング
cs_CZ (Czech Czechoslovakia)	latin2	wlatin2	de_DE (German Germany)	latin1

ソースおよびターゲット SAS セッションのエンコーディングは、互換性がありません。cs_CZ ロケールのデフォルトエンコーディングが WLATIN2 で、ターゲット SAS セッションのエンコーディングが WLATIN1 であるためです。移送ファイルは、これらのロケール間でインポートできません。

ターゲットセッションエンコーディングは latin1 です。CPORT 移送ファイルは WLATIN2 であり、このセッションでインポートできます。ただし、一部の文字は不適切に変更される場合があります。次の警告メッセージが表示されます。

```
警告: この移送ファイルはエンコーディング wlatin2 で作成されました。 このセッションエンコーディング
wlatin1 で読み込み、作成できる移送ファイルのエンコーディングは wlatin1 です。 データが正しくインポート
されない可能性があります。
```

SAS Viya プラットフォームで PROC CIMPORT を使用すると生成される警告メッセージ

SAS Viya プラットフォームは、従来のエンコーディングと UTF-8 エンコーディングをサポートしています。PROC CIMPORT を使用して SAS Viya プラットフォームにデータセットをインポートすると、データのインポート時に警告が発生する場合があります。これらの警告の一部を 表 10.18 (190 ページ)に示します。

WARNING: The Dataset Cannot Be Imported

次のテーブルに、PROC CIMPORT を使用してデータセットをインポートできない場合の警告のリスト、問題が発生する理由、および PROC CIMPORT が問題を処理する方法に関する情報を示します。

表 10.4　SAS Viya プラットフォームで PROC CIMPORT を使用すると生成される警告メッセージ

PROC CIMPORT トランスコーディングの問題			
トランスコーディングの例外	SAS Viya プラットフォームが例外を処理する方法	表示されるメッセージ	いつ例外が発生する可能性があるのか
データセットの名前にサポートされていない文字が含まれている	メッセージを表示し、現在のデータセットのインポートをスキップする	WARNING: The dataset cannot be imported because unsupported characters are found in the dataset “*mylib.mydata*”.	UTF-8 から非 UTF-8 へのトランスコード時
データセットインデックスの名前にサポートされていない文字が含まれている	メッセージを表示し、現在のデータセットのインポートをスキップする	WARNING: The dataset cannot be imported because unsupported characters are found in the name of index "**indexString**" in the dataset "*mylib.mydata*".	UTF-8 から非 UTF-8 へのトランスコード時
データセットの変数名にサポートされていない文字が含まれている	メッセージを表示し、現在のデータセットのインポートをスキップする	WARNING: The dataset cannot be imported because unsupported characters are found in one or more variable names.	UTF-8 から非 UTF-8 へのトランスコード時
データセット変数ラベルがターゲッ	メッセージを表示し、現在のデータセ	WARNING: The dataset cannot be imported	非 UTF-8 から UTF-8 へのト

トエンコーディングに対して長すぎる	ットのインポートをスキップする	because the variable label "**labelString**" is truncated.	ランスコード時
データセットラベルがターゲットエンコーディングに対して長すぎる	メッセージを表示し、現在のデータセットのインポートをスキップする	WARNING: The dataset cannot be imported because the dataset label "lableString" is truncated.	非 UTF-8 から UTF-8 へのトランスコード時
サポートされていない文字がデータセットのオブザベーションに存在する	メッセージを表示し、現在のデータセットのインポートをスキップする	WARNING: The dataset cannot be imported because unsupported characters are found in one or more observations.	UTF-8 から非 UTF-8 へのトランスコード時
サポートされていない文字がデータセットの変数ラベルに存在する	メッセージを表示し、現在のデータセットのインポートをスキップする	WARNING: The dataset cannot be imported because unsupported characters are found in one or more variable labels.	UTF-8 から非 UTF-8 へのトランスコード時
サポートされていない文字がデータセットのラベルに存在する	メッセージを表示し、現在のデータセットのインポートをスキップする	WARNING: The dataset cannot be imported because unsupported characters are found in the dataset label.	UTF-8 から非 UTF-8 へのトランスコード時

SAS のバージョン 9.2〜9.4 を使用して作成された移送ファイルの問題

概要

文字データのエンコーディングは、SAS のバージョン 9.2 以降を使用して作成される移送ファイルにスタンプされます。したがって、CIMPORT プロシジャはエラー状態を検出できます。たとえば、移送ファイル内の UTF-8 データセットは、UTF-8 セッションではないセッションにインポートできます。UTF-8 移送ファイルデータセットのすべての文字がターゲットセッションでサポートされている場合、移送ファイルをインポートできます。そうでない場合は、無効な文字を含むデータセットのインポートに失敗し、次の警告メッセージが表示されます。

```
WARNING: The dataset cannot be imported because unsupported characters are found in one or more
observations.
```

SAS のバージョン 9.2 以降は、移送ファイルのエンコーディングとターゲット SAS セッションのロケールの間の非互換性の状況を検出できます。一部の顧客の SAS アプリケーションが SAS 9.2 より前のリリースを使用して正常に実行されたため、PROC CIMPORT は警告のみレポートしますが、インポート処理の続行を許可します。

次に、復元アクションを含む警告およびエラーメッセージを示します。

- “警告: ターゲットセッションは UTF-8 を使用しません: トランスポートファイルは UTF-8 ではありません” (193 ページ)

エラー: ターゲット セッションは UTF-8 を使用しています: トランスポート ファイルは UTF-8 ではありません

SAS 9.4M3 以降では、このエラーメッセージは表示されません。このエラーメッセージでは、次の情報が提供されます。

- ターゲット SAS セッションで UTF-8 エンコーディングが使用されています。
- 移送ファイルに、UTF-8 以外の特定のエンコーディングが含まれています。移送ファイルとターゲット SAS セッションのエンコーディングは、互換性がありません。

> **注:** PROC CIMPORT は、非 UTF-8 SAS セッションで作成されたデータセットを UTF-8 SAS セッションにインポートする機能をサポートしています。

このエラーが発生した場合、ターゲットの SAS セッションのエンコーディングを UTF-8 にすることはできません。ソースとターゲットの SAS セッションのロケールは同じにする必要があります。

次に、SAS 9.2 WLATIN2 移送ファイルと UTF-8 ターゲット SAS セッションの例を示します。

1 回復するには、ターゲット SAS セッションで別の SAS セッションを開始し、ロケールを変更して、移送ファイルを作成したソース SAS セッションで使用されたロケールにします。

   LOCALE=値は、ENCODING=値よりも優先されます。ENCODING=、DFLANG=、DATESTYLE=、PAPERSIZE=オプションに対するデフォルト値を自動的に設定するためです。

   ソースセッション(または移送ファイル)のロケールが不明な場合は、移送ファイルの文字データで使用されている言語から推測できます。

   たとえば、その言語がポーランド語だとわかれば、新しいターゲット SAS セッションで pl_PL (Polish Poland)ロケールを指定します。Linux の pl_PL (Polish Poland)ロケールに関連付けられているエンコード値は、latin2 です。

   新しい SAS セッションで pl_PL ロケールを指定する例を次に示します。

   ```
   set locale=pl_PL;
   ```

   詳細については、ロケールテーブルを参照してください。

注: UTF-8 エンコーディングの指定がすでに含まれている SAS 起動コマンドがないことを確認します。たとえば、sas9 -encoding utf8 などです。存在する場合は、新しいロケール指定に関係なく、UTF-8 エンコーディングが持続します。

2 ファイルを再度インポートします。次に、例を示します。

```
filename importin 'transport-file';
libname target 'SAS-data-library';
proc cimport infile=importin library=target memtype=data;
run;
```

PROC CIMPORT が続きます。

警告: ターゲットセッションは UTF-8 を使用しません: トランスポートファイルは UTF-8 ではありません

この警告メッセージでは、次の情報が提供されます。

- ターゲット SAS セッションで、特定されたエンコーディングを使用している。
- 移送ファイルのエンコーディングが特定されている。移送ファイルとターゲット SAS セッションのエンコーディングは、互換性がありません。

次のテーブルに、互換性のないソースおよびターゲット SAS セッションのロケールとエンコーディング値を示します。

表 10.5 Czech および German ロケールの値のエンコーディング

SAS セッション	Posix ロケール	Linux エンコーディング
ソース SAS セッション	cs_CZ (Czech Czechoslovakia)	latin2
ターゲット SAS セッション	de_DE (German Germany)	latin9

移送ファイルはインポートされますが、ファイルのコンテンツには問題があります。メッセージは、互換性のないエンコーディング形式を特定します。回復するには、インポートされたファイルのコンテンツを読み込みます。ファイルが読み込めない場合、次のステップを実行します。

1 ターゲット SAS セッションで、新しい SAS セッションを開始し、ロケール(エンコーディングではなく)をソース SAS セッションで使用されるロケールに変更します。

LOCALE=値は、ENCODING=値よりも優先されます。ENCODING=、DFLANG=、DATESTYLE=、PAPERSIZE=オプションに対するデフォルト値を自動的に設定するためです。

ソースセッション(または移送ファイル)のロケールが不明な場合、移送ファイルの各言語から推測できます。

たとえば、チェコ語が言語である場合、新しいターゲット SAS セッションで **cs_CZ** ロケールを指定します。

新しい SAS セッションで **cs_CZ** ロケールを設定します。

```
-locale cs_CZ;
```

ターゲット SAS セッションと移送ファイルは、互換性のあるエンコーディングを使用します。両方ともに wlatin2 を使用します。

詳細については、ロケールテーブル(*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

2 ファイルを再度インポートします。次に例を示します。

```
filename importin 'transport-file';
libname target 'sas-library';
proc cimport infile=importin library=target memtype=data;
run;
```

PROC CIMPORT が続きます。

SAS 9.2 より前の SAS リリースを使用して作成された移送ファイルの問題

概要: 9.2 より前の SAS リリース

9.2 より前の SAS リリースで作成された移送ファイルには、エンコーディング値がスタンプされません。そのため、CIMPORT プロシジャは、移送ファイルのエンコーディングの ID を認識せず、特定の警告やエラーの詳細をレポートできません。移送ファイルのエンコーディングは、復元アクションの実行時に推測する必要があります。

ただし、移送ファイルに関する知識を使用して、移送問題を解決できる必要があります。ターゲット SAS セッションでの移送ファイルのインポートに有用な情報については、ヒント: ISFILEUTF8 オプション (181 ページ)を参照してください。移送ファイルの作成と復元については、*SAS ファイルの移動とアクセス*を参照してください。

次に、復元アクションを含む警告およびエラーメッセージを示します。

- “エラー: トランスポートファイルのエンコーディングが不明です: ISFILEUTF8=オプションを使用してください” (194 ページ)
- “Warning:Transport File Encoding Is Unknown” (195 ページ)

エラー: トランスポートファイルのエンコーディングが不明です: ISFILEUTF8=オプションを使用してください

このエラーメッセージでは、次の情報が提供されます。

- 移送ファイルが 9.2 より前の SAS リリースを使用して作成されている。
- エンコーディングが移送ファイルにスタンプされていないため、エンコーディングが不明である。
- ターゲット SAS セッションで UTF-8 エンコーディングが使用されています。

注: 復元ステップを実行するため、移送ファイルのエンコーディングがわかっている必要があります。

移送ファイルが UTF-8 としてエンコードされていることがわかっていれば、ファイルを再度インポートして、ISFILEUTF8=YES オプションを PROC CIMPORT で使用できます。

次に、UTF-8 移送ファイルと UTF-8 ターゲット SAS セッションの例を示します。UTF-8 移送ファイルが 9.2 より前の SAS リリースを使用して作成されている。

```
filename importin 'transport-file';
libname target 'sas-library';
proc cimport isfileutf8=yes infile=importin library=target memtype=data;
run;
```

構文の詳細については、ISFILEUTF8=オプション (181 ページ)を参照してください。

PROC CIMPORT が続きます。

Warning:Transport File Encoding Is Unknown

この警告メッセージでは、次の情報が提供されます。

- 移送ファイルが 9.2 より前の SAS リリースを使用して作成されている。
- エンコーディングが移送ファイルにスタンプされていないため、エンコーディングが不明である。

インポートされたデータセットから文字データを読み込みます。データを読み込めない場合、ターゲット SAS セッションのロケールが移送ファイルのエンコーディングと互換していないことが推測できます。

注: 復元ステップを実行するため、移送ファイルのエンコーディングがわかっている必要があります。

たとえば、移送ファイルが、9.2 より前の SAS リリースを使用して、ソース SAS セッションで作成されたとします。トランスポートファイルも、Polish Poland ロケールを使用して作成されました。ターゲット SAS セッションでは German ロケールを使用します。

1 ターゲット SAS セッションで、別の SAS セッションを開始し、移送ファイルを作成したソース SAS セッションのロケールにロケールを変更します。

   この例では、新しい SAS セッションを Polish Poland ロケールで開始します。

   ```
   sas9 -locale pl_PL;
   ```

2 ファイルを再度インポートします。次に例を示します。

   ```
   filename importin 'transport-file';
   libname target 'sas-library';
   proc cimport infile=importin library=target memtype=data;
   run;
   ```

PROC CIMPORT が続いて、Polish_Poland ロケールを使用した SAS セッションでデータが読み込み可能になります。

例: CIMPORT プロシジャ

例 1: ライブラリ全体のインポート

要素: PROC CIMPORT ステートメントオプション
INFILE=

詳細

この例では、PROC CPORT が別の動作環境で SAS ライブラリから作成した TRANFILE という名前の移送ファイルを PROC CIMPORT を使用してディスクから読み込む方法を示します。移送ファイルが新しい動作環境に移動されました。PROC CIMPORT は、移送ファイルを新しい動作環境の NEWLIB という名前の SAS ライブラリにインポートします。

プログラム

```
libname newlib 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;

proc cimport library=newlib infile=tranfile;
run;
```

プログラムの説明

ライブラリ名とファイル名を指定します。 LIBNAME ステートメントは、新しい SAS ライブラリに対し LIBNAME を指定します。FILENAME ステートメントは、PROC CPORT が作成した移送ファイルのファイル名を指定し、ファイル特性に対し動作環境オプションを指定できるようにします。

```
libname newlib 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;
```

NEWLIB ライブラリの SAS ライブラリをインポートします。 PROC CIMPORT は、SAS ライブラリを NEWLIB という名前のライブラリにインポートします。

```
proc cimport library=newlib infile=tranfile;
run;
```

ログの例

例のコード 10.1　ライブラリ全体のインポート

```
NOTE: Proc CIMPORT begins to create/update catalog NEWLIB.FINANCE
NOTE: Entry LOAN.FRAME has been imported.
NOTE: Entry LOAN.HELP has been imported.
NOTE: Entry LOAN.KEYS has been imported.
NOTE: Entry LOAN.PMENU has been imported.
NOTE: Entry LOAN.SCL has been imported.
NOTE: Total number of entries processed in catalog NEWLIB.FINANCE: 5

NOTE: Proc CIMPORT begins to create/update catalog NEWLIB.FORMATS
NOTE: Entry REVENUE.FORMAT has been imported.
NOTE: Entry DEPT.FORMATC has been imported.
NOTE: Total number of entries processed in catalog NEWLIB.FORMATS: 2
```

例 2: 個々のカタログエントリのインポート

要素:

PROC CIMPORT ステートメントオプション
　INFILE=
SELECT ステートメント

詳細

この例では、PROC CIMPORT を使用して個別のカタログエントリ LOAN.PMENU と LOAN.SCL を単一の SAS カタログから作成された移送ファイル TRANS2 からインポートする方法を示します。

プログラム

```
libname newlib 'sas-library';
filename trans2 'transport-file';

host-options-for-file-characteristics;

proc cimport catalog=newlib.finance infile=trans2;
```

```
  select loan.pmenu loan.scl;
run;
```

プログラムの説明

ライブラリ名、ファイル名、動作環境オプションを指定します。 LIBNAME ステートメントは、新しい SAS ライブラリに対し LIBNAME を指定します。FILENAME ステートメントは、PROC CPORT が作成した移送ファイルのファイル名を指定し、ファイル特性に対し動作環境オプションを指定できるようにします。

```
libname newlib 'sas-library';
filename trans2 'transport-file';

host-options-for-file-characteristics;
```

指定したカタログエントリを新しい SAS カタログにインポートします。 PROC CIMPORT は、個別のカタログエントリを TRANS2 移送ファイルからインポートして、NEWLIB.FINANCE という名前の新しい SAS カタログに保存します。SELECT ステートメントは、新しいカタログにインポートする移送ファイルから 2 つの指定されたエントリのみ選択します。

```
proc cimport catalog=newlib.finance infile=trans2;
  select loan.pmenu loan.scl;
run;
```

ログの例

例のコード 10.2　個々のカタログエントリのインポート

```
NOTE: Proc CIMPORT begins to create/update catalog NEWLIB.FINANCE
NOTE: Entry LOAN.PMENU has been imported.
NOTE: Entry LOAN.SCL has been imported.
NOTE: Total number of entries processed in catalog NEWLIB.FINANCE: 2
```

例 3: 単一インデックス付き SAS データセットのインポート

要素:　PROC CIMPORT ステートメントオプション
　INFILE=

詳細

この例では、PROC CIMPORT を使用して、単一 SAS データセットからの PROC CPORT によって作成された移送ファイルからインデックス付けされた SAS データセットをインポートする方法を示します。

プログラム

```
libname newdata 'sas-library';
filename trans3 'transport-file';

host-options-for-file-characteristics;

proc cimport data=newdata.times infile=trans3;
run;
```

プログラムの説明

ライブラリ名、ファイル名、動作環境オプションを指定します。 LIBNAME ステートメントは、新しい SAS ライブラリに対し LIBNAME を指定します。FILENAME ステートメントは、PROC CPORT が作成した移送ファイルのファイル名を指定し、ファイル特性に対し動作環境オプションを指定できるようにします。

```
libname newdata 'sas-library';
filename trans3 'transport-file';

host-options-for-file-characteristics;
```

SAS データセットをインポートします。 PROC CIMPORT は、PROC CIMPORT ステートメントの DATA=指定によって特定する単一の SAS データセットをインポートします。PROC CPORT は、移送ファイル TRANS3 のデータセット NEWDATA.TIMES をエクスポートしました。

```
proc cimport data=newdata.times infile=trans3;
run;
```

ログの例

例のコード 10.3　単一インデックス付き SAS データセットのインポート

```
NOTE: Proc CIMPORT begins to create/update data set NEWDATA.TIMES
NOTE: The data set index x is defined.
NOTE: Data set contains 2 variables and 2 observations.
      Logical record length is 16
```

例 4: PROC CIMPORT を使用してフランス語のデータセットを UTF-8SAS セッションにインポートする

要素:

PROC CPORT ステートメントオプション
- DATA=
- FILE=

PROC CIMPORT ステートメントオプション
- DATA=
- INFILE=
- EXTENDVAR=
- EXTENDFORMAT=

詳細

この例は、フランス語のデータセットを UTF-8SAS セッションにインポートする方法を示しています。フランス語のデータセットには、文字形式が$8 の 8 バイトの文字変数 a があります。SAS Viya プラットフォームの配置を使用して、PROC CPORT を使用して myfile という名前の移送ファイルを生成するフランス語の SAS セッションを開始します。

別の **SAS Viya** プラットフォームウィンドウで、UTF-8 エンコーディングを使用する SAS セッションを開始します。次に、PROC CIMPORT を使用して移送ファイル(非 UTF-8)を UTF-8SAS セッションにインポートします。最初に、インポートされたデータセットが切り捨てられることを示し、次に EXTENDVAR=オプションを使用して可変長を拡張して切り捨てを回避する方法を示します。

プログラム

PROC CPORT を使用して、フランス語のデータセットに対し移送ファイルを作成します。 SAS Viya プラットフォームの配置から、フランス語を使用してデフォルトの SAS セッションを起動します。次の例では、データセットに 8 バイトの文字変数 a(a の値はアクセント付きの e が付いた premiere)があり、文字フォーマットが$8 に設定されているフランス語のデータセットを使用しています。PROC CPORT を使用してトランスポートファイルを作成します。この例では、エンコーディングはデフォルトの wlatin-1 です。

```
data test;
  a = 'première';
  format a $8.;
run;
proc cport data=test file='myfile';
run;
```

アウトプット 10.1　PROC CPORT ファイルで使用されるフランス語のデータセット

Obs	a
1	premièr

PROC CIMPORT を使用して、UTF-8 SAS セッションに移送ファイルをインポートします。 UTF-8 エンコーディングを使用している別の SAS Viya プラットフォームセッションから、PROC CIMPORT を使用してトランスポートファイル myfile を UTF-8 SAS セッションにインポートします。SAS データセットと PROC CONTENTS を出力します。アクセント付きの e を UTF-8 エンコーディングで格納するには 2 バイト必要であるため、値の premiere には値を表示するために 9 バイトが必要です。形式は$8 です。したがって、値は切り捨てられます。

```
proc cimport data=test infile='myfile';
run;
proc print data=test;
run;
proc contents data=test;
run;
```

アウトプット 10.2　非 UTF-8 トランスポートファイルを UTF-8 - Truncated のエンコーディングセッションに CIMPORT

Obs	a
1	premièr

The CONTENTS Procedure

Data Set Name	WORK.TEST	Observations	1
Member Type	DATA	Variables	1
Engine	V9	Indexes	0
Created	04/21/2020 15:01:17	Observation Length	9
Last Modified	04/21/2020 15:01:17	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64, LINUX_POWER_64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	7177
Obs in First Data Page	1
Number of Data Set Repairs	0
Filename	/viya-directory/test.sas7bdat
Release Created	V.0400M0
Host Created	Linux
Inode Number	105502840
Access Permission	rw-r--r--
Owner Name	UNKNOWN
File Size	128KB
File Size (bytes)	131072

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
1	a	Char	9	$8.

切り捨てを回避するために可変長を拡張します。 切り捨てがあったため、EXTENDVAR=オプションを使用して、可変長とフォーマット幅を拡張します (EXTENDFORMAT=のデフォルトは YES です)。

```
proc cimport data=test infile='myfile' extendvar=1.5;
run;
proc print data=test;
run;
proc contents data=test;
run;
```

出力データセットに変数全体が含まれていることに注意してください。また、PROC CONTENTS の出力で、可変長とフォーマット幅が拡張されていることに注意してください。

アウトプット 10.3　非 UTF-8 トランスポートファイルを UTF-8-EXTENDVAR=のエンコーディングセッションに CIMPORT

Obs	a
1	première

The CONTENTS Procedure

Data Set Name	WORK.TEST	**Observations**	1
Member Type	DATA	**Variables**	1
Engine	V9	**Indexes**	0
Created	04/21/2020 15:23:51	**Observation Length**	14
Last Modified	04/21/2020 15:23:51	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64, LINUX_POWER_64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	4636
Obs in First Data Page	1
Number of Data Set Repairs	0
Filename	/viya-directory/test.sas7bdat
Release Created	V.0400M0
Host Created	Linux
Inode Number	105502841
Access Permission	rw-r--r--
Owner Name	UNKNOWN
File Size	128KB
File Size (bytes)	131072

Alphabetic List of Variables and Attributes				
#	**Variable**	**Type**	**Len**	**Format**
1	a	Char	14	$12.

11

COMPARE プロシジャ

概要: COMPARE プロシジャ

COMPARE プロシジャの機能

COMPARE プロシジャは、2 つの SAS データセットのコンテンツ、異なるデータセットの選択変数、または同一データセット内の変数を比較します。

PROC COMPARE は、基準データセットと比較データセットという 2 つのデータセットを比較します。プロシジャは、マッチング変数とマッチングオブザベーションを決定します。マッチング変数は、同名の変数か、または VAR および WITH ステートメントを使用してペアにする変数です。マッチング変数は同じ種類にする必要があります。マッチングオブザベーションは、指定したすべての ID 変数の値が同じオブザベーションで、ID ステートメントを使用していない場合は、データセットの同じ位置に発生するオブザベーションです。オブザベーションを ID 変数によってマッチングさせる場合は、すべての ID 変数を基準にして両方のデータセットを並べ替えておく必要があります。

PROC COMPARE から提供される情報について

PROC COMPARE は、比較対象の 2 つのデータセットについて次の情報を生成します。

- マッチング変数の値が異なるかどうか
- 一方のデータセットのオブザベーションが他方よりも多いかどうか
- 2 つのデータセットに共通する変数はどれか
- 一方のデータセットには存在するが他方のデータセットには存在しない変数はいくつあるか
- マッチング変数の出力形式、ラベルまたは種類が異なるかどうか
- マッチングオブザベーションの値の比較

注: 同じ変数名を持つ 2 つの列があるビューを作成できます。重複する変数名がビューに存在する場合、PROC COMPARE は、基準データセットのどの列を比較データセットと比較するのかを決定できません。PROC COMPARE では、重複する変数名が見つかった場合、エラーが発生します。

さらに、PROC COMPARE は、比較する変数のオブザベーション間の差異について詳細を示す 2 種類の出力データセットを作成します。

次の例では、データセット Proclib.One と Proclib.Two を比較しています。このデータセットには、学生についての類似したデータが含まれています。

```
data proclib.one(label='First Data Set');
  input student year $ state $ gr1 gr2;
  label year='Year of Birth';
  format gr1 4.1;
  datalines;
1000 1990 NC 85 87
1042 1991 MS 90 92
1095 1989 TN 78 92
1187 1990 MA 87 94
;

data proclib.two(label='Second Data Set');
  input student $ year $ state $ gr1
       gr2 major $;
  label state='Home State';
  format gr1 5.2;
  datalines;
1000 1990 NC 85 87 Math
1042 1991 MS 90 92 History
1095 1989 TN 78 92 Physics
1187 1990 MA 87 94 Music
1204 1991 NC 82 96 English
;
```

PROC COMPARE では、各比較データセットの同じ値についての情報は生成されません。同じ値ではなく、異なる値についての情報が生成されます。

PROC COMPARE では、比較データセットの 1 つには存在するけれど別の比較データセットには存在しないオブザベーションや、両方の比較データセットに存在するオブザベーションを含むデータセットは生成されません。COMPARE ステートメントのオプションで、この情報のほとんどを生成できますが、データセットは生成されません。この情報を含むデータセットを生成する場合は、MERGE ステートメントを含む DATA ステップを使用します。DATA ステップで MERGE ステートメントを使用してデータセットを作成する例を次に示します。

```
data inone intwo inboth;
  merge a (in=ina) b(in=inb);
  by byvar;
    if ina and not inb then output inone;
    if inb and not ina then output intwo;
    if ina and inb then output inboth;
run;
```

概念: COMPARE プロシジャ

PROC COMPARE を使用した比較

PROC COMPARE はまず次の比較を行います。

- データセット属性(データセットオプション TYPE=および LABEL=により設定)。
- 変数。PROC COMPARE は、一方のデータセットの各変数をチェックして、他方のデータセットの変数と一致するかどうかを決定します。
- マッチング変数の属性(種類、長さ、ラベル、出力形式、入力形式)。
- オブザベーション。PROC COMPARE は、一方のデータセットの各オブザベーションをチェックして、他方のデータセットのオブザベーションと一致するかどうかを決定します。PROC COMPARE は、データセット内の位置かまたは ID 変数の値を基準にしてオブザベーションをマッチングします。

これらの比較を行った後で、PROC COMPARE はデータセットのマッチ部分の値を比較します。PROC COMPARE は、オブザベーションの位置かまたは ID 変数の値を基準にしてデータを比較します。

オブザベーションの位置による比較

次の図は、2 つのデータセットを示しています。網かけボックス内のデータは、データセットのうちプロシジャが比較する部分を示しています。同じ名前の変数は同じ種類であると仮定します。

図 11.1 観測位置による比較

Data Set ONE

Obs	idnum	name	year	state	grade1	grade2
1	1000	Emily	1990	NC	85.0	87
2	1042	Dan	1991	MS	90.0	92
3	1095	Julia	1989	TN	78.0	92
4	1187	Robin	1990	MA	87.0	94

Data Set TWO

Obs	idnum	name	year	state	grade1	grade2	major
1	1000	Emily	1990	NC	85.00	87	Math
2	1042	Dan	1991	MS	90.00	92	History
3	1095	Julia	1989	TN	78.00	92	Physics
4	1187	Robin	1990	MA	87.00	94	Music
5	1204	Keith	1991	NC	82.00	96	English
6	1198	Ted	1990	PA	84.00	93	Music
7	1054	Lisa	1989	GA	81.00	94	French

PROC COMPARE を使用してデータセット TWO とデータセット ONE を比較する場合、プロシジャはデータセット ONE の 1 番目のオブザベーションとデータセット TWO の 1 番目のオブザベーションを比較してから、1 番目のデータセットの 2 番目のオブザベーションと 2 番目のデータセットの 2 番目のオブザベーションを比較し、その後も同じように処理を続けます。比較対象のオブザベーションごとに、プロシジャは idnum、name、year、state、grade1、grade2 値を比較します。

プロシジャは、データセット TWO の最後の 3 つのオブザベーションや変数 major の値についてはレポートしません。これはデータセット ONE に比較対象が存在しないからです。

ID 変数を使用した比較

単純な比較では、PROC COMPARE は、オブザベーション番号を使用して、どのオブザベーションを比較するかを決定します。ID 変数を使用する場合、PROC COMPARE は、ID 変数の値を使用して、どのオブザベーションを比較するかを決定します。ID 変数は、値が一意で、種類が同じである必要があります。

次の図で示されている 2 つのデータセットについては、IDNUM が ID 変数で、なおかつ両データセットの IDNUM の種類が同じだと仮定します。プロシジャは、IDNUM の値が同じオブザベーションを比較します。網かけボックス内のデータは、データセットのうちプロシジャが比較する部分を示しています。

図 11.2　ID 変数の値による比較

Data Set ONE

Obs	idnum	name	year	state	grade1	grade2
1	1000	Emily	1990	NC	85.0	87
2	1042	Dan	1991	MS	90.0	92
3	1095	Julia	1989	TN	78.0	92
4	1187	Robin	1990	MA	87.0	94

Data Set TWO

Obs	idnum	name	year	state	grade1	grade2	major
1	1000	Emily	1990	NC	85.00	87	Math
2	1042	Dan	1991	MS	90.00	92	History
3	1095	Julia	1989	TN	78.00	92	Physics
4	1187	Robin	1990	MA	87.00	94	Music
5	1204	Keith	1991	NC	82.00	96	English
6	1198	Ted	1990	PA	84.00	93	Music
7	1054	Lisa	1989	GA	81.00	94	French

データセットには、name、year、state、grade1、grade2 という 5 つのマッチング変数が含まれています。また、idnum 値が 1000、1042、1095、1187 の 4 つのマッチングオブザベーションも含まれています。

データセット TWO には、データセット ONE にマッチングオブザベーションが含まれていない 3 つのオブザベーション(idnum=1204、idnum=1198、idnum=1054)が含まれています。同様に、データセット ONE には、データセット TWO の変数“major”とマッチする変数はありません。

ID 変数の使用例については、“例 5: ID 変数を使用してオブザベーションを比較する” (269 ページ)を参照してください。

同等性の基準

CRITERION=オプションの使用

COMPARE プロシジャは、METHOD=オプションに従って測定された差異の大きさが CRITERION=オプションの値を上回る場合、数値が不等であると判断します。PROC COMPARE では、CRITERION=を適用するメソッドが 4 つあります。

- EXACT メソッドでは、完全に同等かが検証されます。
- ABSOLUTE メソッドでは、絶対差と CRITERION=で指定した値が比較されます。

- RELATIVE メソッドでは、絶対的な相対差と CRITERION=で指定した値が比較されます。
- PERCENT メソッドでは、絶対的なパーセント表示の差異と CRITERION=で指定した値が比較されます。

数値変数を比較する場合、基準データセットの値を *x*、比較データセットの値を *y* とします。*x* と *y* が両方とも欠損値ではない場合、次のように METHOD=の値と CRITERION= (γ)の値に従って値が不等と判断されます。

- METHOD=EXACT では、*y* と *x* が等しくない場合、値は不等です。
- METHOD=ABSOLUTE では、次の場合、値は不等です。

  $$\mathrm{ABS}(y-x) > \gamma$$

- METHOD=RELATIVE では、次の場合、値は不等です。

  $$\mathrm{ABS}(y-x)/((\mathrm{ABS}(x)+\mathrm{ABS}(y))/2+\delta) > \gamma$$

  x=*y*=0 の場合、値は同等です。
- METHOD=PERCENT では、次の場合、値は不等です。

  $$100(\mathrm{ABS}(y-x)/\mathrm{ABS}(x)) > \gamma \ \text{ for } \ x \neq 0$$

  または

  $$y \neq 0 \ \text{ for } \ \mathrm{x} = 0$$

x または *y* が欠損している場合、比較は NOMISSING オプションによって変わります。NOMISSING オプションが有効な場合、欠損値は常にいずれの値とも等しいと判断されます。NOMISSING オプションが無効な場合、欠損値は同じ種類の欠損値(すなわち、.=.、.^=.A、.A=.A、.A^=.B など)とのみ等しいと判断されます。

CRITERION=に対して指定した値が負の場合、実際に使用する基準 γ は、指定した基準の絶対値に、コンピューターの数値精度に応じたきわめて小さい数字 ε(イプシロン)を掛けた値と等しくなります。この数字 ε は、マシンで計算できる最小の正の浮動小数点値(1−ε<1<1+ε など)として定義されます。浮動小数点計算での四捨五入または切り捨てによる誤差は、通常 ε よりも数桁大きくなります。多くの場合、CRITERION=−1000 にすると、マシンレベルの精度で計算結果の同等性を合理的にテストできます。

RELATIVE メソッドで分母に加える値 δ は、METHOD=RELATIVE(δ)のように、メソッド名の後にかっこで囲んで指定します。METHOD=で指定をしない場合、δ はデフォルトで 0 になります。δ の値は、*x* と *y* の両方が非常に 0 に近い場合に、誤差の測定動作を制御するために使用されます。δ が与えられず、*x* と *y* の両方が非常に 0 に近い場合は、大きな相対誤差が生じます(限界は 2)。

δ の値を指定すると、RELATIVE メソッドが、小さな値に対して極度に反応することを避けられます。*x* と *y* が両方ともの絶対値よりもはるかに小さい場合に METHOD=RELATIVE(δ) CRITERION=γ を指定すると、METHOD=ABSOLUTE CRITERION=δγ を指定した場合と同じように比較されます。ただし、*x* または *y* のどちらかが δ の絶対値よりもはるかに大きい場合は、METHOD=RELATIVE CRITERION=γ の場合と同じように比較されます。*x* と *y*x と y の値が適度な場合は、METHOD=RELATIVE(δ) CRITERION=γ は、実質的には、METHOD=ABSOLUTE CRITERION=δ γ と METHOD=RELATIVE CRITERION=γ の比較になります。

文字変数については、一方の値が他方より長い場合、比較のために短い方の値にブランクが埋め込まれます。ブランク以外の文字値は、各文字が一致する場合のみ同

等と判断されます。NOMISSING オプションが有効な場合、ブランクの文字値はいずれの値とも同等と判断されます。

差異とパーセント表示の差異の定義

PROC COMPARE は、値比較のレポートと OUT=データセットに、比較した数字の差異値とパーセント表示の差異値を表示します。この数量は、基準データセットの値を使用して、参照値として定義されます。数値変数を比較する場合、基準データセットの値を *x*、比較データセットの値を *y* とします。*x* と *y* が両方とも欠損値ではない場合、差異およびパーセント表示の差異は、次のように定義されます。

- 差異 = $y - x$
- パーセント表示の差異 = $(y - x)/x * 100 \text{ for } x \neq 0$
- パーセント表示の差異 = 欠損$x = 0$.

PROC COMPARE での変数の出力形式の処理法

PROC COMPARE は、フォーマットされていない値を比較します。2 つのマッチング変数に異なるフォーマットされている場合、PROC COMPARE は変数の出力形式をリストにします。出力の比較セクションに一致しない値が見つかると、フォーマットされた値が値比較セクションに表示されます。

構文: COMPARE プロシジャ

制限事項: WITH ステートメントの使用時には、VAR ステートメントも使用する必要があります。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化し、ログに適切に書き込まれるようにしてください。

LABEL、ATTRIB、FORMAT、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (23 ページ)を参照してください。

OPTIONS ステートメントの LINESIZE オプションが、データセットのラベル情報を表示するのに適切な長さを指定していることを確認してください。

CAS で 2 つの不均等なデータセットを比較するときに CHAR 変数の長さに問題が発生した場合、LIBNAME ステートメントの NCHARMULTIPLIER オプションの値を変更する必要があります。詳細については、“NCHARMULTIPLIER= LIBNAME ステートメントオプション” (*SAS Cloud Analytic Services: ユーザーガイド*)および“LIBNAME ステートメント: CAS エンジン” (*SAS Cloud Analytic Services: ユーザーガイド*)を参照してください。

PROC COMPARE<*options*>;
ATTRIB *variable-list(s) attribute-list(s)*;
BY<DESCENDING>*variable-1*
<... <DESCENDING> *variable-n*>
<NOTSORTED>;
FORMAT *variable-1*<...*variable-n*> <*format*> <DEFAULT=*default-format*>;
ID<DESCENDING>*variable-1*
<<DESCENDING> *variable-2* ...>
<NOTSORTED>;
LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;
VAR *variable(s)*;
WHERE *where-expression-1*
<*logical-operator where-expression-n*>;
WITH *variable(s)*;

ステートメント	タスク	例
“PROC COMPARE ステートメント”	SAS データセットのコンテンツを比較するか、または 2 つの変数を比較します。	Ex. 1, Ex. 2, Ex. 4, Ex. 6, Ex. 7
“BY ステートメント”	BY グループごとに個別の比較を行います。	
“ID ステートメント”	オブザベーションのマッチングに使用する変数を識別します。	Ex. 5
“VAR ステートメント”	比較を指定変数の値に制限します。	Ex. 2, Ex. 3, Ex. 4
“WITH ステートメント”	名前の異なる変数を比較します。	Ex. 2, Ex. 3, Ex. 4
“WITH ステートメント”	同一データセット内の 2 つの変数を比較します。	Ex. 4

PROC COMPARE ステートメント

2 つの SAS データセットのコンテンツ、異なるデータセットの選択変数、または同一データセット内の変数を比較します。

制限事項: COMPARE=を省略する場合は、WITH および VAR ステートメントを使用する必要があります。

ヒント: OPTIONS ステートメントの LINESIZE オプションが、データセットのラベル情報を表示するのに適切な長さを指定していることを確認してください。

CAS で 2 つの不均等なデータセットを比較するときに CHAR 変数の長さに問題が発生した場合、LIBNAME ステートメントの NCHARMULTIPLIER オプションの値を変更する必要があります。詳細については、“NCHARMULTIPLIER= LIBNAME ステー

トメントオプション” (*SAS Cloud Analytic Services: ユーザーガイド*)および “LIBNAME ステートメント: CAS エンジン” (*SAS Cloud Analytic Services: ユーザーガイド*)を参照してください。

BASE=オプションと COMPARE=オプションではデータセットオプションが使用できます。

例: “例 1: 差異の完全なレポートを作成” (256 ページ)
“例 2: 異なるデータセットでの変数の比較” (263 ページ)
“例 4: 同一データセット内での変数の比較” (267 ページ)
“例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する” (275 ページ)
“例 7: 統計量の出力データセット(OUTSTATS=)の作成” (279 ページ)

構文

PROC COMPARE <*options*>;

オプション引数の要約

Control the details in the default report

ALLOBS
: すべてのマッチングオブザベーションの値を含めます。

ALLSTATS
: マッチング変数のすべてのペアについて要約統計量テーブルを印刷します。

ALLVARS
: すべてのマッチング変数の値と差異をレポートに含めます。

BRIEFSUMMARY
: 簡潔な比較要約のみを印刷します。

FUZZ=*number*
: 0 と 1 の間の数字のレポートを変更します。

MAXPRINT=*total* | (*per-variable, total*)
: 印刷される差異の数を制限します。

NODATE
: 作成日および最終変更日の印刷を抑制します。

NOPRINT
: すべての印刷出力を抑制します。

NOSUMMARY
: データセット、変数、オブザベーション、値比較要約レポートを非表示にします。

NOVALUES
: 値比較結果レポートを非表示にします。

PRINTALL
: 値と差異の完全リストを作成します。

STATS
: 不等と判断されたマッチング数値変数のすべてのペアの要約統計量テーブルを印刷します。

TRANSPOSE
: 変数ごとではなくオブザベーションごとに値差異のレポートを印刷します。

Control the listing of variables and observations

LISTALL
: 一方のデータセットにしかない変数とオブザベーションをすべてリストに出力します。

LISTBASE
: 基準データセットにしかない変数とオブザベーションをすべてリストに出力します。

LISTBASEOBS
: 基準データセットにしかないオブザベーションをすべてリストに出力します。

LISTBASEVAR
: 基準データセットにしかない変数をすべてリストに出力します。

LISTCOMP
: 比較データセットにしかない変数とオブザベーションをすべてリストに出力します。

LISTCOMPOBS
: 比較データセットにしかないオブザベーションをすべてリストに出力します。

LISTCOMPVAR
: 比較データセットにしかない変数をすべてリストに出力します。

LISTEQUALVAR
: 値が同等と判断された変数をリストに出力します。

LISTOBS
: 一方のデータセットにしかないオブザベーションをすべてリストに出力します。

LISTVAR
: 一方のデータセットにしかない変数をすべてリストに出力します。

Control the output data set

OUT=*SAS-data-set*
: 出力データセットを作成します。

OUTALL
: BASE=データセットと COMPARE=データセットのオブザベーションごとに 1 つずつオブザベーションを書き込みます。

OUTBASE
: 基準データセットの各オブザベーションに対するオブザベーションを書き込みます。

OUTCOMP
: 比較データセットのオブザベーションごとに 1 つずつオブザベーションを書き込みます。

OUTDIF

マッチングオブザベーションのペアごとに 1 つずつオブザベーションを出力データセットに書き込みます。

OUTNOEQUAL
すべての値が等しい場合、オブザベーションの書き込みを抑制します。

OUTPERCENT
マッチングオブザベーションのペアごとに 1 つずつオブザベーションを出力データセットに書き込みます。

Create an output data set that contains summary statistics

OUTSTATS=*SAS-data-set*
マッチング変数のすべてのペアの要約統計量を、指定した *SAS-data-set* に書き込みます。

Display a warning message in the SAS log

WARNING
差異が見つかると、SAS ログに警告メッセージを表示します。

Display an error message in the SAS log

ERROR
差異が見つかると、SAS ログにエラーメッセージを表示します。

Specify how the values are compared

CRITERION=γ
数値の同等性を判断する基準を指定します。

METHOD=ABSOLUTE | EXACT | PERCENT | RELATIVE<(δ)>
数値の同等性を判断するメソッドを指定します。

NOMISSBASE
基準データセットの欠損値をすべての値と同等であると判断します。

NOMISSCOMP
比較データセットの欠損値をすべての値と同等であると判断します。

NOMISSING
基準データセットと比較データセットの両方にある欠損値を、すべての値と同等であると判断します。

Specify the data sets to compare

BASE=*SAS-data-set*
基準データセットとして使用するデータセットを指定します。

COMPARE=*SAS-data-set*
比較データセットとして使用するデータセットを指定します。

Write notes to the SAS log

NOTE
比較結果を説明する注釈を SAS ログに表示します。

オプション引数

ALLOBS
すべてのマッチングオブザベーションについて、同等と判断された場合でも、値および(数値の場合は)差異を値比較結果のレポートに含めます。

デフォルト ALLOBS を省略すると、PROC COMPARE は不等と判断されたオブザベーションの値のみを印刷します。

操作 TRANSPOSE オプションと併せて指定すると、ALLOBS は ALLVARS オプションを呼び出して、すべてのマッチングオブザベーションとマッチング変数の値を表示します。

ALLSTATS
マッチング変数のすべてのペアについて 要約統計量テーブルを印刷します。

参照項目: “要約統計量のテーブル” (249 ページ)(生成された統計量について)

ALLVARS
マッチング変数のすべてのペアについて 、同等と判断された場合でも、値および(数値の場合は)差異を値比較結果のレポートに含めます。

デフォルト ALLVARS を省略すると、PROC COMPARE は不等と判断された変数の値のみを印刷します。

操作 TRANSPOSE オプションとあわせて指定すると、ALLVARS は他のマッチング変数の値と前後関係のある不等値のみ表示します。TRANSPOSE オプションを省略すると、ALLVARS は ALLOBS オプションを呼び出して、すべてのマッチングオブザベーションとマッチング 変数の値を表示します。

BASE=*SAS-data-set*
基準データセットとして使用するデータセットを指定します。

別名 DATA=

B=

デフォルト 一番最近に作成された SAS データセット

ヒント WHERE=データセットオプションを BASE=オプションと併せて使用すると、比較に使用可能なオブザベーションを限定できます。

BRIEFSUMMARY
簡潔な比較要約のみを生成して、4 つのデフォルト要約レポート(データセット要約レポート、変数要約レポート、オブザベーション要約レポート、値比較要約レポート)を抑制します。

別名 BRIEF

ヒント デフォルトでは、要約レポートに伴って値差異のリストが作成されます。リストの作成を抑制するには、NOVALUES オプションを使用します。

例 “例 4: 同一データセット内での変数の比較” (267 ページ)

COMPARE=*SAS-data-set*
比較データセットとして使用するデータセットを指定します。

別名 COMP=

C=

デフォルト　COMPARE=オプションを指定すると、比較データセットは基準データセットと同じになり、PROC COMPARE はデータセット内の変数を比較します。

制限事項　COMPARE=を省略する場合は、WITH および VAR ステートメントを使用する必要があります。

ヒント　比較に使用可能なオブザベーションを制限するには、WHERE=データセットオプションを COMPARE=とあわせて使用します。

CRITERION=γ
数値の同等性を判断する基準を指定します。 通常、γ(ガンマ)の値は正です。この場合、その数自体が同等性の基準となります。γ に負の値を使用すると、PROC COMPARE は、SAS を実行しているコンピューターの精度に比例した同等性基準を使用します。

別名　CRITERIA=

CRIT=

デフォルト　0.00001

参照項目:　“同等性の基準” (208 ページ)

ERROR
差異が見つかると、SAS ログにエラーメッセージを表示します。

操作　このオプションは、WARNING オプションよりも優先されます。

FUZZ=*number*
number より小さい数字の値比較結果を変更します。PROC COMPARE は次のとおりに印刷します。

- *number* よりも小さいすべての変数値の代わりに 0 を印刷
- *number* よりも小さい差異またはパーセント表示の差異の代わりにブランクを印刷
- *number* よりも小さいすべての要約統計量のかわりに 0 を印刷

デフォルト　0

範囲　0 - 1

ヒント　些細な差異を多く含むレポートは、この形式の方が読みやすくなります。

LISTALL
一方のデータセットにしかない変数とオブザベーションをすべてリストに出力します。

別名　LIST

操作 LISTALL の処理は、LISTBASEOBS、LISTCOMPOBS、LISTBASEVAR、LISTCOMPVAR の 4 オプションを使用した場合と同じです。

LISTBASE
基準データセットにあって比較データセットにはないオブザベーションと変数をすべてリストに出力します。

操作 LISTBASE の処理は、LISTBASEOBS および LISTBASEVAR オプションを使用する場合と同じです。

LISTBASEOBS
基準データセットにあって比較データセットにはないオブザベーションをすべてリストに出力します。

LISTBASEVAR
基準データセットにあって比較データセットにはない変数をすべてリストに出力します。

LISTCOMP
比較データセットにあって基準データセットにはないオブザベーションと変数をすべてリストに出力します。

別名 LISTCOMPARE

操作 LISTCOMP の処理は、LISTCOMPOBS および LISTCOMPVAR オプションを使用する場合と同じです。

LISTCOMPOBS
比較データセットにあって基準データセットにはないオブザベーションをすべてリストに出力します。

別名 LISTCOMPAREOBS

LISTCOMPVAR
比較データセットにあって基準データセットにはない変数をすべてリストに出力します。

別名 LISTCOMPAREVARS

LISTCOMPAREVAR

LISTEQUALVAR
値が不等と判断された変数のデフォルトリストに加えて、すべてのオブザベーションで値が同等と判断された変数のリストを印刷します。

別名 LISTEQ

LISTEQUAL

LISTEQVAR

LISTOBS
一方のデータセットにしかないオブザベーションをすべてリストに出力します。

操作 LISTOBS の処理は、LISTBASEOBS および LISTCOMPOBS オプションを使用する場合と同じです。

LISTVAR
一方のデータセットにしかない変数をすべてリストに出力します。

操作 LISTVAR の処理は、LISTBASEVAR と LISTCOMPVAR の両オプションを使用する場合と同じです。

MAXPRINT=*total* | (*per-variable, total*)
印刷される差異の最大数を指定します。

total
印刷する差異の最大合計数です。デフォルト値は 500 ですが、ALLOBS オプション(または ALLVAR と TRANSPOSE の両オプション)を使用する場合は別です。その場合、デフォルトは 32000 になります。

per-variable
BY グループ内の変数ごとに印刷する差異の最大数です。デフォルト値は 50 ですが、ALLOBS オプション(または ALLVAR と TRANSPOSE の両オプション)を使用する場合は別です。その場合、デフォルトは 1000 になります。

MAXPRINT=オプションは、データセットが大きく異なる場合に、出力が極端に大きくならないようにします。

METHOD=ABSOLUTE | EXACT | PERCENT | RELATIVE<(δ)>
数値の同等性を判断するメソッドを指定します。 定数 δ(デルタ)は、同等性測定の計算時に分母に加える値を指定する 0 と 1 の間の数です。デフォルトでは、δ は 0 です。

CRITERION=オプションを使用しなければ、デフォルトメソッドは EXACT です。CRITERION=オプションを使用する場合、デフォルトメソッドは RELATIVE(φ)です。この場合、φ(ファイ)は SAS を実行しているコンピューターの数値精度と CRITERION=の値に依存する小さい数です。

別名 M=

METH=

参照項目: “同等性の基準” (208 ページ)

NODATE
基準データセットと比較データセットの作成日と最終変更日を、データセット要約レポートに表示しないようにします。

NOMISSBASE
(デフォルトでは、欠損値は、.=.、.^=.A、.A=.A、.A^=.B などの同種の欠損値とのみ同等と判断されます)。

このオプションを使用すると、DATA ステップの UPDATE ステートメントで、基準データセットがメインデータセットとして使用され、比較データセットがトランザクションデータセットとして使用されている場合に、比較データセットのオブザベーションに加える変更を決定できます。UPDATE ステートメントの詳細については、“UPDATE ステートメント” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

別名 NOMISS1

NOMISSCOMP
比較データセットの欠損値をすべての値と同等であると判断します。 (デフォルトでは、欠損値は、.=.、.^=.A、.A=.A、.A^=.B などの同種の欠損値とのみ同等と判断されます)。

このオプションを使用すると、DATA ステップの UPDATE ステートメントで、基準データセットがメインデータセットとして使用され、比較データセットがトランザクションデータセットとして使用されている場合に、比較データセットのオブザベーションに加える変更を決定できます。UPDATE ステートメントの詳細については、"UPDATE ステートメント" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

別名 NOMISS2

NOMISSCOMPARE

NOMISSING
基準データセットと比較データセットの両方にある欠損値を、すべての値と同等であると判断します。 デフォルトでは、欠損値は、.=.、.^=.A、.A=.A、.A^=.B などの同種の欠損値とのみ同等と判断されます。

別名 NOMISS

操作 NOMISSING の処理は、NOMISSBASE と NOMISSCOMP の両方を使用する場合と同じです。

NOPRINT
すべての印刷出力を抑制します。

ヒント 出力データセットを 1 つ以上作成する場合に、このオプションを使用することがあります。

例 "例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する" (275 ページ)

NOSUMMARY
データセット、変数、オブザベーション、値比較要約レポートを非表示にします。

別名 NOSUM

ヒント NOSUMMARY は、マッチング値に差異がない場合、出力を作成しません。

例 "例 2: 異なるデータセットでの変数の比較" (263 ページ)

NOTE
差異が見つかった場合、比較結果を説明する注釈を SAS ログに表示します。

NOVALUES
値比較結果レポートを非表示にします。

別名 NOOBS

例 概要: COMPARE プロシジャ (204 ページ)

OUT=*SAS-data-set*
出力データセットを指定します。*SAS-data-set* が存在しない場合は、PROC COMPARE が作成します。*SAS-data-set* には、マッチング変数の差異が含まれます。

参照項目: "出力データセット(OUT=)" (253 ページ)

例 "例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する" (275 ページ)

OUTALL
基準データセットのオブザベーションと比較データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込みます。また、このオプションは、マッチングオブザベーションの値の差異およびパーセント表示の差異を含む出力データセットにもオブザベーションを書き込みます。

ヒント OUTALL の処理は、OUTBASE、 OUTCOMP、OUTDIF、OUTPERCENT の 4 オプションを使用した場合と同じです。

参照項目: "出力データセット(OUT=)" (253 ページ)

OUTBASE
基準データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込み、_TYPE_=BASE のオブザベーションを作成します。

参照項目: "出力データセット(OUT=)" (253 ページ)

例 "例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する" (275 ページ)

OUTCOMP
比較データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込み、_TYPE_=COMP のオブザベーションを作成します。

別名 OUTCOMPARE

参照項目: "出力データセット(OUT=)" (253 ページ)

例 "例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する" (275 ページ)

OUTDIF
マッチングオブザベーションのペアごとに 1 つずつオブザベーションを出力データセットに書き込みます。 オブザベーションの値には、オブザベーションのペアにおける値の差異を示す値が含まれます。各オブザベーションの_TYPE_の値は DIF です。

別名 OUTDIFF

デフォルト OUTDIF オプションがデフォルトですが、OUTBASE、OUTCOMP または OUTPERCENT オプションを指定した場合は別です。これらのオプションのいずれかを使用する場合は、OUTDIF オプションを指定して、

出力データセットに_TYPE_=DIF オブザベーションを作成する必要があります。

参照項目: “出力データセット(OUT=)” (253 ページ)

例 “例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する” (275 ページ)

OUTNOEQUAL
オブザベーションのすべての値が同等と判断された場合に、出力データセットへのオブザベーションの書き込みを抑制します。さらに、同等と判断された変数値も不等と判断された変数値も含まれるオブザベーションでは、OUTNOEQUAL オプションが特殊欠損値".E"を使用して、同等と判断された変数の差異とパーセント表示の差異を示します。

別名 OUTNOEQ

参照項目: “出力データセット(OUT=)” (253 ページ)

例 “例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する” (275 ページ)

OUTPERCENT
マッチングオブザベーションのペアごとに 1 つずつオブザベーションを出力データセットに書き込みます。 オブザベーションの値には、オブザベーションのペアにおける値のパーセント表示の差異を示す値が含まれます。各オブザベーションの_TYPE_の値は PERCENT です。

参照項目: “出力データセット(OUT=)” (253 ページ)

OUTSTATS=*SAS-data-set*
マッチング変数のすべてのペアの要約統計量を、指定した *SAS-data-set* に書き込みます。

ヒント プロシジャ出力の統計量テーブルを印刷する場合は、STATS、ALLSTATS または PRINTALL オプションを使用します。

参照項目: “出力統計量データセット(OUTSTATS=)” (255 ページ)

“要約統計量のテーブル” (249 ページ)

例 “例 7: 統計量の出力データセット(OUTSTATS=)の作成” (279 ページ)

PRINTALL
オプション ALLVARS、ALLOBS、ALLSTATS、LISTALL、WARNING を呼び出します。

別名 ALL

例 “例 1: 差異の完全なレポートを作成” (256 ページ)

STATS
不等と判断されたマッチング数値変数のすべてのペアの 要約統計量テーブルを印刷します。

別名 統計量

参照項目: “要約統計量のテーブル” (249 ページ) (生成された統計量について)。

TRANSPOSE
変数ごとではなくオブザベーションごとに値差異のレポートを印刷します。

別名 TRANS

操作 NOVALUES オプションも使用すると、TRANSPOSE オプションは、各オブザベーションについて値が不等と判断された変数の *names* のみをリストに出力し、値と差異は出力しません。

参照項目: “オブザベーションの比較結果(TRANSPOSE オプション使用)” (251 ページ).

WARNING
差異が見つかると、SAS ログに警告メッセージを表示します。

別名 WARN

操作 ERROR オプションが WARNING オプションよりも優先されます。

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

BY ステートメント

出力を BY グループ順に並べ替えます。

制限事項: BY ステートメントではグループ変数を指定しないでください。BY 変数をグループ変数と同じにすることはできません。

構文

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

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

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

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

詳細

BY グループ処理

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

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

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

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

- BY ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。

- PROC SORT の詳細については、55 章, “SORT プロシジャ,” (2027 ページ)を参照してください。
- CAS での BY グループ処理の詳細については、“グループ化と順序付け” (*SAS Cloud Analytic Services: DATA ステッププログラミング*)を参照してください。
- インデックスの作成の詳細については、“INDEX CREATE ステートメント” (435 ページ)を参照してください。

BY 変数値のフォーマティング

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

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

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

2 つのデータセットで長さが異なる BY 変数

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

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

- プロシジャで、BY ステートメントが常にセカンダリデータセットに適用される場合があります。この場合、BY ステートメントの変数が 1 つ以上、セカンダリデータセットに出現している必要があります。
- プロシジャで、BY ステートメントがセカンダリデータセットに一度も適用されない場合もあります。この場合、セカンダリデータセットには、BY ステートメントの変数は不要です。
- プロシジャで、セカンダリデータセットに BY 変数があるかどうかチェックされる場合があります。セカンダリデータセットに BY 変数が 1 つもない場合、セカンダリデータセットに対する BY 処理は発生しません。セカンダリデータセットに BY 変数が 1 つ以上あり、それがプライマリデータセットの BY 変数と一致する場合、セカンダリデータセットに対して BY 処理が行われます。セカンダリデータセットに全部ではなく一部の BY 変数がある場合、プロシジャでエラーメッ

セージが出力され、終了する場合があります。または、その特定プロシジャのドキュメントで説明しているその他のアクションが実行される場合もあります。

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

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

- A
- A、B
- A、B、C
- A、B、C、D

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

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブランクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデータセットの‘ABCD’とセカンダリデータセットの‘ABCD ’は一致します。セカンダリデータセットに‘ABCDEF’が含まれている場合は、一致しません。

BY ステートメントをサポートする Base SAS プロシジャ

COMPARE	SORT (必須)
CORR	STANDARD
FREQ	SUMMARY
MEANS	TABULATE
PRINT	TRANSPOSE

RANK	UNIVARIATE
REPORT (非ウィンドウ環境のみ)	ODSTEXT
ODSLIST	ODSTABLE

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

関連項目

"BY ステートメント" (*SAS DATA ステップステートメント: リファレンス*)

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。"出力形式の取り消し" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、"PUT

ステートメント: リスト" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

ID ステートメント

オブザベーションのマッチングに使用する変数のリストを表示します。

参照項目: "ID 変数を使用した比較" (207 ページ)

例: "例 5: ID 変数を使用してオブザベーションを比較する" (269 ページ)

構文

ID <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* ...>
<NOTSORTED>;

必須引数

variable
プロシジャがオブザベーションのマッチングに使用する変数を指定します。複数の変数を指定できますが、データセットは、1 つまたは複数の指定変数順に並べ替える必要があります。この変数が *ID 変数*です。ID 変数は、印刷したレポートや出力データセットのオブザベーションも識別します。

オプション引数

DESCENDING
データセットが ID ステートメントで文字 DESCENDING の直後に続く変数別に降順で並べ替えられるように指定します。

DESCENDING オプションを使用する場合は、データセットを並べ替える必要があります。DESCENDING オプションを指定した ID ステートメントの処理には、インデックスは使用されません。さらに、ID 変数に対する DESCENDING の使用は、データセットを並べ替えるために使用した PROC SORT ステップにおける BY ステートメントの DESCENDING オプションの使用と対応している必要があります。

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

参照項目: "並べ替えられていないデータの比較" (229 ページ)

詳細

ID 変数の必要条件

- ID 変数が BASE=データセットに存在する必要があります。 存在しなければ、PROC COMPARE は処理を停止します。
- ID 変数が COMPARE=データセットになければ、PROC COMPARE は SAS ログに警告メッセージを書き込み、その変数は使用せずに比較データセットのオブザベーションをマッチングします(ただし、OUT=データセットには書き込みます)。
- 両方のデータセットで ID 変数の種類を同一にする必要があります。
- NOTSORTED オプションを指定する場合を除いて、両方のデータセットを(もしあれば BY 変数内の)共通の ID 変数で並べ替える必要があります。

並べ替えられていないデータの比較

データを ID 変数で並べ替えない場合は、NOTSORTED オプションを使用できます。NOTSORTED オプションを指定した場合、または ID ステートメントを省略した場合、PROC COMPARE はオブザベーションを 1 対 1 でマッチングします。つまり、PROC COMPARE は、基準データセットの 1 番目のオブザベーションと比較データセットの 1 番目のオブザベーション、2 番目と 2 番目、というようにマッチングしていきます。NOTSORTED の使用時に、対応するオブザベーションの ID 値が同一ではなかった場合、PROC COMPARE はエラーメッセージを印刷して処理を停止します。

データセットを共通の ID 変数で並べ替えず、なおかつ NOTSORTED オプションを指定していない場合、PROC COMPARE は SAS ログに警告メッセージを書き込み、NOTSORTED の指定時と同様にデータセットの処理を続行します。

重複 ID 値の回避

各データセットのオブザベーションは、ID 変数値によって一意にラベル付けされている必要があります。PROC COMPARE は、データセットで同じ ID 値のオブザベーションを 2 つ連続で発見すると、次の処理を実行します。

- そのデータセットでの最初の発生時に警告 Duplicate Observations を印刷します。
- オブザベーション要約レポートに、データセットで発見された重複オブザベーションの合計数を印刷します。
- 基準データセットと比較データセットの重複オブザベーションを使用して、1 対 1 ベースでオブザベーションを比較します。

データセットが並べ替えられていない場合、PROC COMPARE は連続で発生した重複オブザベーションのみを検出します。

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=' '>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項 ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
```

```
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: “文字定数と文字変数” (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

VAR ステートメント

変数値の比較を、VAR ステートメントで指定した変数に限定します。

別名: VARIABLE

例: “例 2: 異なるデータセットでの変数の比較” (263 ページ)
“例 3: 変数を複数回比較する” (265 ページ)
“例 4: 同一データセット内での変数の比較” (267 ページ)

構文

VAR *variable(s)*;

必須引数

variable(s)
BASE=データセットと COMPARE=データセットの両方か、または BASE=データセットにのみ出現する 1 つ以上の変数。

詳細

- VAR ステートメントを使用しない場合、PROC COMPARE は、BY ステートメントと ID ステートメントに記述された変数を除く、すべてのマッチング変数の値を比較します。
- VAR ステートメントの変数が COMPARE=データセットに存在しない場合、PROC COMPARE は SAS ログに警告メッセージを書き込み、その変数を無視します。
- VAR ステートメントの変数が BASE=データセットに存在しない場合、PROC COMPARE は処理を停止して、SAS ログにエラーメッセージを書き込みます。
- VAR ステートメントは、マッチング変数の値の比較のみを制限します。PROC COMPARE はそれでもマッチング変数の合計数とその属性の比較をレポートします。ただし、これらの変数についてはエラーメッセージも警告メッセージも生成されません。

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

WITH ステートメント

基準データセットの変数と比較データセットの異なる名前の変数を比較します。また、同一データセット内の異なる変数を比較します。

制限事項: WITH ステートメントの使用時には、VAR ステートメントも使用する必要があります。

例: “例 2: 異なるデータセットでの変数の比較” (263 ページ)
“例 3: 変数を複数回比較する” (265 ページ)
“例 4: 同一データセット内での変数の比較” (267 ページ)

構文

WITH *variable(s)*;

必須引数

variable(s)
VAR ステートメントの変数と比較する 1 つ以上の変数。

詳細

選択された変数の比較

基準データセットの変数を比較データセットの異なる名前の変数と比較する場合は、VAR ステートメントで基準データセットの変数名を指定し、WITH ステートメントでマッチング変数名を指定します。WITH 変数リストに指定する最初の変数と VAR 変数リストに指定する最初の変数、2 番目と 2 番目、というように対応します。WITH 変数リストが VAR 変数リストよりも短い場合、PROC COMPARE は、VAR ステートメントの余った変数が、比較データセットでも基準データセットでも同じ名前であるとみなします。WITH 変数リストが VAR 変数リストよりも長い場合、PROC COMPARE は余った変数を無視します。

VAR ステートメントまたは WITH ステートメントでは 1 つの変数名を何度でも記述できます。VAR 変数リストおよび WITH 変数リストの選択によって、変数をどんな順列でも比較できます。

PROC COMPARE ステートメントで COMPARE=オプションを省略する場合は、WITH ステートメントを使用する必要があります。この場合、PROC COMPARE は、BASE=データセットにある異なる名前の変数の値を比較します。

使用法: COMPARE プロシジャ

PROC COMPARE 出力のカスタマイズ

PROC COMPARE は非常に長い出力を生成します。オプションを 1 つ以上使用すると、比較の種類やレポートの詳細度を決定できます。たとえば、次の PROC COMPARE ステップの NOVALUES オプションは、マッチング変数の値の差異を示す出力部分を非表示にします。

```
options nodate pageno=1 linesize=80 pagesize=40;

title 'The SAS System';

proc compare base=proclib.one
        compare=proclib.two novalues;
run;
```

アウトプット 11.1　WORK.ONE と WORK.TWO の比較(その 1)

The SAS System

```
                        The COMPARE Procedure
                  Comparison of WORK.ONE with WORK.TWO
                             (Method=EXACT)

                           Data Set Summary

Dataset            Created          Modified  NVar    NObs  Label

WORK.ONE   16MAR11:16:43:16  16MAR11:16:43:16     5       4  First Data Set
WORK.TWO   16MAR11:16:43:16  16MAR11:16:43:16     6       5  Second Data Set


                           Variables Summary

             Number of Variables in Common: 5.
             Number of Variables in WORK.TWO but not in WORK.ONE: 1.
             Number of Variables with Conflicting Types: 1.
             Number of Variables with Differing Attributes: 3.
```

```
             Listing of Common Variables with Conflicting Types

                      Variable  Dataset   Type  Length

                      student   WORK.ONE  Num        8
                                WORK.TWO  Char       8


              Listing of Common Variables with Differing Attributes

            Variable  Dataset   Type  Length  Format  Label

            year      WORK.ONE  Char       8          Year of Birth
                      WORK.TWO  Char       8
            state     WORK.ONE  Char       8
                      WORK.TWO  Char       8          Home State
```

アウトプット 11.2　WORK.ONE と WORK.TWO の比較(その 2)

The SAS System

```
                        The COMPARE Procedure
                 Comparison of WORK.ONE with WORK.TWO
                            (Method=EXACT)

          Listing of Common Variables with Differing Attributes

        Variable  Dataset   Type  Length  Format  Label

        gr1       WORK.ONE  Num        8  4.1
                  WORK.TWO  Num        8  5.2
```

```
                          Observation Summary

                     Observation      Base  Compare

                     First Obs           1        1
                     First Unequal       1        1
                     Last  Unequal       4        4
                     Last  Match         4        4
                     Last  Obs           .        5

      Number of Observations in Common: 4.
      Number of Observations in WORK.TWO but not in WORK.ONE: 1.
      Total Number of Observations Read from WORK.ONE: 4.
      Total Number of Observations Read from WORK.TWO: 5.

      Number of Observations with Some Compared Variables Unequal: 4.
      Number of Observations with All Compared Variables Equal: 0.
```

アウトプット 11.3　WORK.ONE と WORK.TWO の比較(その 3)

The SAS System

```
                        The COMPARE Procedure
                  Comparison of WORK.ONE with WORK.TWO
                             (Method=EXACT)

                      Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 3.
Total Number of Values which Compare Unequal: 6.
Maximum Difference: 20.


                   Variables with Unequal Values

       Variable  Type  Len   Compare Label  Ndif   MaxDif

       state     CHAR    8   Home State        2
       gr1       NUM     8                     2    1.000
       gr2       NUM     8                     2   20.000
```

“プロシジャ出力” (242 ページ) では、この 2 つのデータセットのデフォルト出力が示されます。“例 1: 差異の完全なレポートを作成” (256 ページ)では、この 2 つのデータセットの完全出力が示されます。

結果: COMPARE プロシジャ

結果レポート

PROC COMPARE は、次の方法で比較結果をレポートします。

- SAS ログ
- 自動マクロ SYSINFO に格納されたリターンコード
- プロシジャの出力
- 出力データセット

SAS ログ

WARNING、PRINTALL または ERROR オプションを使用する場合、PROC COMPARE は差異の説明を SAS ログに書き込みます。

マクロリターンコード(SYSINFO)

PROC COMPARE は自動マクロ変数 SYSINFO にリターンコードを格納します。リターンコードの値によって、比較結果についての情報が提供されます。SAS マクロは、PROC COMPARE の実行後、他のステップが開始される前に、SYSINFO の値をチェックすることにより、PROC COMPARE ステップの結果を使用して、どんなアクションを取るべきか、または SAS プログラムのどの部分を実行するべきかを決定できます。

PROC COMPARE からの SYSINFO リターンコードを解釈するためのキーを次のテーブルに示します。リストの各条件が真の場合、関連する値がリターンコードに追加されます。したがって、次のテーブルに記載された該当条件のコードの合計が SYSINFO リターンコードになります。

表 11.1 マクロリターンコード

ビット	条件	コード	16 進数	説明
1	DSLABEL	1	0001X	データセットラベルが異なります。
2	DSTYPE	2	0002X	データセットの種類が異なります。
3	INFORMAT	4	0004X	変数の入力形式が異なります。
4	FORMAT	8	0008X	変数の出力形式が異なります。
5	LENGTH	16	0010X	変数の長さが異なります。
6	LABEL	32	0020X	変数のラベルが異なります。
7	BASEOBS	64	0040X	基準データセットに、比較データセットにないオブザベーションがあります。
8	COMPOBS	128	0080X	比較データセットに、基準データセットにないオブザベーションがあります。
9	BASEBY	256	0100X	基準データセットに、比較データセットにない BY グループがあります。

ビット	条件	コード	16 進数	説明
10	COMPBY	512	0200X	比較データセットに、基準データセットにない BY グループがあります。
11	BASEVAR	1024	0400X	基準データセットに、比較データセットにない変数があります。
12	COMPVAR	2048	0800X	比較データセットに、基準データセットにない変数があります。
13	VALUE	4096	1000X	値の比較が不等でした。
14	TYPE	8192	2000X	変数の種類が一致しません。
15	BYVAR	16384	4000X	BY 変数が一致しません。
16	ERROR	32768	8000X	致命的なエラー:比較は行われません。

これらのコードは、データセットの差異の度合いを簡単にチェックできるよう段階的に並べられています。たとえば、同じ変数、オブザベーションおよび値を含む 2 つのデータセットをチェックするが、ラベルや出力形式などの差異について考慮しない場合は、次のステートメントを使用します。

```
proc compare base=SAS-data-set
          compare=SAS-data-set;
run;

%if &sysinfo >= 64 %then
  %do;
    handle error;
  %end;
```

コードに SYSINFO を含めるたびに、SYSINFO のその後の各インスタンスの値が前の値に加算されます。SYSINFO の最終値は、コード内の SYSINFO のすべてのインスタンスの累積値です。たとえば、次のサンプルコードを実行した場合、SYSINFO の最初のインスタンスでリターンコード 32 が生成され、2 番目のインスタンスでリターンコード 4096 が生成されます。2 つの値を合計して、最終リターンコードの 4128 が生成されます。

```
/*diff label -- RC is 32*/
data class1;
  set sashelp.class;
  label sex='Gender';
run;

data class2;
  set sashelp.class;
run;

proc compare base=class1 comp=class2;
run;

%let rc=&sysinfo
%put 'RC' &rc
```

```
/*diff label and value -- RC is 4128*/
data class1;
  set sashelp.class;
  label sex='Gender';
run;
data class2;
  set sashelp.class;
  if name="Jeffrey" then name="Jeff";
run;

proc compare base=class1 comp=class2;
run;

%let rc=&sysinfo
%put 'RC' &rc
```

DATA ステップのビットテスト機能を使用して SYSINFO 値の個々のビットを検証すると、特定の状況についてチェックできます。たとえば、比較データセットにはない基本データセットのオブザベーションの存在をチェックするには、次のステートメントを使用します。

```
proc compare base=SAS-data-set
          compare=SAS-data-set;
run;

%let rc=&sysinfo;
data _null_;
  /* Test for data set label */
  if &rc = '1'b then
    put '<<<< Data sets have different labels';
/* Test for data set types */
  if &rc = '1.'b then
    put '<<<< Data set types differ';
/* Test for label */
  if &rc = '1.....'b then
    put '<<<< Variable has different label';
/* Test for base observation */
if &rc = '1......'b then
    put '<<<< Base data set has observation not in comparison data set';
/* Test for length */
  if &rc = '1....'b then
    put '<<<< Variable has different lengths between the base data set
    and the comparison data set';
/* Variable in base data set not in compare data set */
  if &rc ='1.........'b then
    put '<<<< Variable in base data set not found in comparison data set';
/* Comparison data set has variable not in base data set */
  if &rc = '1..........'b then
    put '<<<< Comparison data set has variable not contained in the
    base data set';
/* Test for values */
  if &rc = '1..........'b then
    put '<<<< A value comparison was unequal';
/* Conflicting variable types */
  if &rc ='1.........'b then
    put '<<<< Conflicting variable types between the two data sets
```

```
    being compared';
run;
```

SYSINFO をチェックする前に PROC COMPARE を実行し、別の SAS ステップが開始する前に SYSINFO 値を取得する必要があります。 SAS ステップはどれも SYSINFO をリセットするからです。

プロシジャ出力

プロシジャ出力の概要

次のセクションでは、概要: COMPARE プロシジャ (204 ページ)で示した 2 つのデータセットのデフォルト出力を示して説明します。PROC COMPARE は非常に長い出力を生成するので、出力は 7 つに分けて提示されます。

```
options nodate pageno=1 linesize=80 pagesize=60;
proc compare base=proclib.one compare=proclib.two;
 run;
```

データセット要約

このレポートは、比較されているデータセットの属性をリストにします。これらの属性には、次が含まれます。

- データセット名
- もしあればデータセットの種類
- もしあればデータセットラベル
- 作成日と最終変更日
- 各データセットの変数の数
- 各データセットのオブザベーションの数

データセット要約を表示するには、アウトプット 11.35 (244 ページ)を参照してください。

注: COMPARE プロシジャは、ラインサイズが小さすぎる場合は、データセットラベルを省略します。

変数要約

このレポートでは、2 つのデータセットの変数が比較されます。レポートの最初の部分には、次が列挙されます。

- データセットに共通の変数の数
- 比較データセットにはなくて基準データセットにはある変数、およびその逆の変数の数
- 両データセットで種類が異なる変数の数
- その他の属性(長さ、ラベル、出力形式、入力形式)が異なる変数の数
- 比較のために指定した BY、ID、VAR、WITH 変数の数

レポートの 2 番目の部分には、属性の異なるマッチング変数のリストと、その属性の差異が示されます。(COMPARE プロシジャは、ページサイズが小さすぎた場合は、変数ラベルを省略します)。

次の出力は、データセット要約と変数要約を示しています。

アウトプット 11.4　データセット要約と変数要約を示す出力の一部

The SAS System

```
                        The COMPARE Procedure
                  Comparison of WORK.ONE with WORK.TWO
                             (Method=EXACT)

                           Data Set Summary

Dataset            Created          Modified  NVar    NObs  Label

WORK.ONE  18MAR11:11:22:21  18MAR11:11:22:21     5       4  First Data Set
WORK.TWO  18MAR11:11:22:22  18MAR11:11:22:22     6       5  Second Data Set


                           Variables Summary

             Number of Variables in Common: 5.
             Number of Variables in WORK.TWO but not in WORK.ONE: 1.
             Number of Variables with Conflicting Types: 1.
             Number of Variables with Differing Attributes: 3.
```

```
            Listing of Common Variables with Conflicting Types

                      Variable  Dataset   Type  Length

                      student   WORK.ONE  Num        8
                                WORK.TWO  Char       8


           Listing of Common Variables with Differing Attributes

          Variable  Dataset   Type  Length  Format  Label

          year      WORK.ONE  Char       8          Year of Birth
                    WORK.TWO  Char       8
          state     WORK.ONE  Char       8
                    WORK.TWO  Char       8          Home State
          gr1       WORK.ONE  Num        8  4.1
                    WORK.TWO  Num        8  5.2
```

オブザベーション要約

このレポートでは、基準データセットと比較データセットのオブザベーションについての情報が提供されます。最初に、各データセットの最初と最後のオブザベーション、最初と最後のマッチングオブザベーション、および最初と最後の異なるオブザベーションが示されます。それから、レポートに次が列挙されます。

- データセットに共通のオブザベーションの数

- 比較データセットにはなくて基準データセットにはあるオブザベーション、およびその逆のオブザベーションの数
- 各データセットのオブザベーションの合計数
- PROC COMPARE によって一部の変数が不等と判断されたマッチングオブザベーションの数
- PROC COMPARE によってすべての 変数が同等と判断されたマッチングオブザベーションの数

次の出力は、オブザベーション要約を示しています。

アウトプット 11.5　オブザベーション要約を示す出力の一部

```
                        Observation Summary

                    Observation      Base  Compare

                    First Obs           1        1
                    First Unequal       1        1
                    Last  Unequal       4        4
                    Last  Match         4        4
                    Last  Obs           .        5

Number of Observations in Common: 4.
Number of Observations in WORK.TWO but not in WORK.ONE: 1.
Total Number of Observations Read from WORK.ONE: 4.
Total Number of Observations Read from WORK.TWO: 5.

Number of Observations with Some Compared Variables Unequal: 4.
Number of Observations with All Compared Variables Equal: 0.
```

値の比較の要約 y

このレポートにはまず、次が列挙されます。

- すべてのオブザベーションが同等な比較変数の数
- 一部のオブザベーションが不等な比較変数の数
- もしあれば欠損値を含む差異のある変数の数
- 不等と判断された値の合計数
- マッチング変数のすべてのペアの 不等な値の最大差異測定(欠損値を含まない差異を対象とする)

さらに、一部のマッチングオブザベーションに不等な値がある変数について、レポートに次が列挙されます。

- 変数の名前
- 他の変数の属性
- PROC COMPARE がその変数を不等と判断した回数

- 値間に見られる最大差異測定(欠損値を含まない差異を対象とする)
- もしあれば、欠損値を含む比較によって起こる差異の数

次の出力は、値の比較の要約を示しています。

アウトプット 11.6　値の比較の要約を示す出力の一部

The SAS System

```
                        The COMPARE Procedure
                 Comparison of WORK.ONE with WORK.TWO
                             (Method=EXACT)

                      Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 3.
Total Number of Values which Compare Unequal: 6.
Maximum Difference: 20.


                   Variables with Unequal Values

Variable  Type  Len   Compare Label  Ndif   MaxDif

state     CHAR    8   Home State        2
gr1       NUM     8                     2    1.000
gr2       NUM     8                     2   20.000
```

値の比較の結果

このレポートは、1 つ以上のオブザベーションで不等と判断されたマッチング変数の各ペアのテーブルで構成されています。

注:

- PROC COMPARE は、文字値の比較時は、最初の 20 文字のみ表示します。TRANSPOSE オプションを使用した場合、PROC COMPARE は、最初の 12 文字のみ表示します。

  文字値を比較する場合、出力 PROC で COMPARE は 20 文字を超える文字列の最後にプラス記号を表示します。TRANSPOSE オプションを指定すると、PROC COMPARE は、出力において、12 文字より長い文字列の最後にプラス記号を表示します。どちらの場合も、プラス記号は文字列の上にあるテーブルのセル境界に表示されます。次に例を示します。

```
                The COMPARE Procedure
            Comparison of WORK.X with WORK.Y
                     (Method=EXACT)

          Value Comparison Results for Variables

_____________________________________________________________
            ||  Base Value           Compare Value
       Obs  ||  x                     x
________    ||  ____________________+  ____________________+
            ||
          1 ||  12345678901234567890   12345678901234567890
_____________________________________________________________
```

注: 文字列の上の境界線の最後にプラス記号が表示されます。

- LINESIZE オプションに指定した値が適切でない場合、SAS では、LINESIZE 値が小さすぎて値比較レポートに ID 変数をすべて出力できないという警告が出力される場合があります。PROC COMPARE は、フォーマットされたデータ値の幅に基づいて、値セクションのためのスペースを確保します。スペースの残り部分は、ID 値のために使用可能です。文字 ID 値では、その出力形式で指定された幅が使用され、数値変数では、約 10 スペースが使用されます。

各テーブルには、次が表示されます。

- オブザベーション番号、または ID 変数の値(ID ステートメントを使用している場合)
- 基準データセットの変数の値
- 比較データセットの変数の値
- 2 つの値の差異(数値変数のみ)
- 2 つの値のパーセント表示の差異(数値変数のみ)

次の出力は、変数の値比較結果を示しています。

アウトプット 11.7　変数の値比較結果を示す出力の一部

Value Comparison Results for Variables

Obs	Home State Base Value state	Compare Value state
2	MD	MA
4	MA	MD

Obs	Base gr1	Compare gr1	Diff.	% Diff
1	85.0	84.00	-1.0000	-1.1765
3	78.0	79.00	1.0000	1.2821

Obs	Base gr2	Compare gr2	Diff.	% Diff
3	72.0000	73.0000	1.0000	1.3889
4	94.0000	74.0000	-20.0000	-21.2766

NOVALUES オプションを使用して値比較結果を非表示にできます。NOVALUES オプションと TRANSPOSE オプションの両方を使用すると、PROC COMPARE は、値が不等と判断された変数の名前をオブザベーションごとにリストにしますが、値および差異は表示しません。

PROC COMPARE および TRANSPOSE オプションの表示制限は、PROC COMPARE によって生成される印刷出力にのみ適用されます。値全体を表示するには、次のオプションを使用して、PROC COMPARE で出力データセットを作成する必要があります。

- OUT=では、出力データセット名が指定されます。
- OUTNOEQUAL では、すべての変数がマッチするオブザベーションの書き込みが抑制されます。
- OUTBASE と OUTCOMP では、BASE=と COMPARE=のデータセットからのオブザベーションが含められます。
- NOPRINT では、デフォルトの印刷レポートが抑制されます。

次の例では、差異データセットを生成した後で、PROC PRINT を実行します。

```
data x;
```

```
a='aaaaaaaaaaaaaaaaaaaaa';
data y;
a='aaaaaaaaaaaaaaaaaaaab';
run;
proc compare base=x comp=y out=dif outbase outcomp outdif outnoequal;
run;
proc print data=dif;
run;
```

図 11.3　差異データセット

The SAS System

Obs	_TYPE_	_OBS_	a
1	BASE	1	aaaaaaaaaaaaaaaaaaaaa
2	COMPARE	1	aaaaaaaaaaaaaaaaaaaab
3	DIF	1	X

要約統計量のテーブル

STATS、ALLSTATS または PRINTALL オプションを使用すると、変数の値比較結果セクションに、比較されている数値変数の要約統計量が含まれます。STATS オプションは、値が不等と判断された数値変数についてのみこの統計量を生成します。ALLSTATS および PRINTALL オプションは、すべての値が同等と判断されても、すべての数値変数についてこの統計量を生成します。ALLSTATS セクションの統計量の生成に使用する式を次に示します。

```
(ystat-xstat)/x*100
```

注: いかなる場合でも、PROC COMPARE は、不等な値を含むマッチングオブザベーションだけでなく、欠損値を含まないすべてのマッチングオブザベーションに基づいて、要約統計量を計算します。

次の出力には、次に示す基準データセット値、比較データセット値、差異、パーセント表示の差異の要約統計量が表示されます。

N
　非欠損値の数。

　注: 出力の 4 列すべてにおいて、N 統計量の値は常に非欠損オブザベーションの数になります。Diff と%Diff の列では、N 統計量に対する計算は実行されません。

MEAN
　値の平均値。

STD
　標準偏差。

MAX
最大値。

MIN
最小値。

MISSDIFF
基準データセットと比較データセットのどちらかの欠損値数。

STDERR
平均値の標準誤差。

T
T 比率(MEAN/STDERR)。

PROB> | T |
真の母平均が 0 の場合に、絶対 T 値がより大きくなる確率。

NDIF
不等と判断されたマッチングオブザベーションの数、および不等と判断されたマッチングオブザベーションの割合。

DIFMEANS
基準値の平均と比較値の平均の差異。この行には 3 つの数字が含まれます。1 番目は、基準値平均のパーセント表示平均。2 番目は、比較値平均のパーセント表示平均。3 番目は、2 つの平均値の差異(比較平均値から基準平均値を引く)。

R
どちらのデータセットでも非欠損なマッチングオブザベーションの基準値と比較値の相関係数。

RSQ
どちらのデータセットでも非欠損なマッチングオブザベーションの基準値と比較値の相関係数の 2 乗。

次の出力は、ALLSTATS オプションで概要: COMPARE プロシジャ (204 ページ)に示した 2 つのデータセットを使用した結果です。

```
options nodate pageno=1
  linesize=80 pagesize=60;
proc compare base=proclib.one
    compare=proclib.two allstats;
  title 'Comparing Two Data Sets: Default Report';
run;
```

アウトプット 11.8　変数の値比較結果を示す出力の一部

Value Comparison Results for Variables

Obs	Home State Base Value state	Compare Value state
2	MD	MA
4	MA	MD

Obs	Base gr1	Compare gr1	Diff.	% Diff
1	85.0	84.00	-1.0000	-1.1765
3	78.0	79.00	1.0000	1.2821
N	4	4	4	4
Mean	85.5000	85.5000	0	0.0264
Std	5.8023	5.4467	0.8165	1.0042
Max	92.0000	92.0000	1.0000	1.2821
Min	78.0000	79.0000	-1.0000	-1.1765
StdErr	2.9011	2.7234	0.4082	0.5021
t	29.4711	31.3951	0.0000	0.0526
Prob>|t|	<.0001	<.0001	1.0000	0.9614
Ndif	2	50.000%		
DifMeans	0.000%	0.000%	0	
r, rsq	0.991	0.983		

注: PRINTALL でページサイズを大きくすると、PROC COMPARE は、文字変数の値比較結果を数値変数の結果の隣に印刷します。その場合、PROC COMPARE は、文字変数の NDIF のみ計算します。

オブザベーションの比較結果(TRANSPOSE オプション使用)

TRANSPOSE オプションは、比較結果を、変数別ではなくオブザベーション別に印刷します。比較結果は、オブザベーション要約レポートの前に置かれます。デフォルトでは、テーブルの各行の値のソースが次のラベルで表示されます。

_OBS_1=*number-1* _OBS_2=*number-2*

number-1 は変数の値が示されているデータベースセットのオブザベーションの番号で、*number-2* は、比較データセットのオブザベーションの番号です。

次の出力では、変数ではなくオブザベーションに関する、PROCLIB.ONE と PROCLIB.TWO との差を示しています。

```
options nodate pageno=1
  linesize=80 pagesize=60;
proc compare base=proclib.one
    compare=proclib.two transpose;
  title 'Comparing Two Data Sets: Default Report';
run;
```

アウトプット 11.9　オブザベーションの比較結果を示す出力の一部

Comparing Two Data Sets: Default Report

```
                        The COMPARE Procedure
                  Comparison of WORK.ONE with WORK.TWO
                             (Method=EXACT)

                  Comparison Results for Observations

_OBS_1=3 _OBS_2=3:
Variable    Base Value       Compare          Diff.        % Diff
     gr1          78.0         79.00       1.000000      1.282051
     gr2     72.000000     73.000000       1.000000      1.388889

_OBS_1=4 _OBS_2=4:
Variable    Base Value       Compare          Diff.        % Diff
     gr2     94.000000     74.000000    -20.000000    -21.276596
   state            MA            MD
```

ID ステートメントを使用すると、ラベルの識別は次の形式になります。

ID-1=ID-value-1 ... ID-n=ID-value-n

この場合、*ID* は ID 変数名で、*ID-value* は ID 変数値です。

注: TRANSPOSE オプションを使用した場合、PROC COMPARE は値の最初の 12 文字のみ印刷します。

ODS テーブル名

COMPARE プロシジャは、作成する各テーブルに名前を割り当てます。これらの名前を使用して、Output Delivery System (ODS)を使用してテーブルを選択し、出力データセットを作成する際にそのテーブルを参照できます。詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

表 11.2 COMPARE プロシジャによって生成される ODS テーブル

テーブル名	説明	テーブル生成時の条件
CompareData sets	1 つまたは複数のデータセットについての情報	デフォルトでは、NOSUMMARY または NOVALUES オプションが指定されていない場合
CompareDetails (オブザベーションの比較結果)	基準データデットと比較データセットに共通しないオブザベーションのリスト	PRINTALL オプションが指定されている場合
CompareDetails (ID 変数の注記と警告)	重複 ID 変数値に関する注記と警告のリスト	ID ステートメントが指定され、重複 ID 変数値がどちらかのデータセットに存在する場合
CompareDifferences	変数値差異のレポート	デフォルトでは、NOVALUES オプションが指定されていない場合
CompareSummary	不等値のオブザベーション、値および変数の要約レポート	デフォルト
CompareVariables	基準データセットと比較データセットの変数の種類または属性の差異のリスト	デフォルトでは、値が同一ではないか、または NOSUMMARY オプションが指定されていない場合

注: ODS 出力テーブルには、**出力**ウィンドウに書き込まれるものと同じ情報が含まれます。これらのテーブルには、追加情報は含まれません。また、その情報に対する異なる出力形式も提供されません。ODS 出力テーブルが役に立つ場合もありますが、それは実行するタスクによって決まります。**出力**ウィンドウでで生成される出力形式の例については、“PROC COMPARE 出力のカスタマイズ” (235 ページ)の出力を参照してください。

出力データセット(OUT=)

デフォルトでは、OUT=データセットには、マッチングオブザベーションのペアごとに 1 つずつオブザベーションが含まれます。OUT=データセットには、比較しているデータセットから次の変数が含まれます。

- BY ステートメントで指定したすべての変数
- ID ステートメントで指定したすべての変数
- すべてのマッチング変数か、または(VAR ステートメントを使用した場合は)VAR ステートメントに記述したすべての変数

さらに、データセットには、PROC COMPARE がマッチング変数値のソースを識別するために作成する 2 つの変数_TYPE_および_OBS_も含まれます。

TYPE
は長さ 8 の文字変数です。この値によって、そのオブザベーションのマッチング(VAR)変数の値のソースが識別されます。(比較されない ID 変数と BY 変数については、元のデータセットからの値がこの値になります)。_TYPE_のラベルは Type of Observation です。この変数の 4 つの可能な値は次のとおりです。

BASE
このオブザベーションの値の出所は、基準データセットのオブザベーションです。PROC COMPARE は、OUTBASE オプションを指定すると、OUT=データセットにこの種類のオブザベーションを書き込みます。

COMPARE
このオブザベーションの値の出所は、比較データセットのオブザベーションです。PROC COMPARE は、OUTCOMP オプションを指定すると、OUT=データセットにこの種類のオブザベーションを書き込みます。

DIF
このオブザベーションの値は、基準データセットと比較データセットの値の差異です。

文字変数の場合、PROC COMPARE は、同等な文字を表すにはピリオド(.)、不等な文字を表すには X を使用します。

数値変数の場合、E は差異がないことを意味します。それ以外の場合は、数値の差異が表示されます。

PROC COMPARE は、デフォルトで、OUT=データセットにこの種類のオブザベーションを書き込みます。ただし、OUTBASE、OUTCOMP または OUTPERCENT オプションを指定して他の種類のオブザベーションを要求する場合は、OUT=データセットにこの種類のオブザベーションを生成するには OUTDIF オプションを指定する必要があります。

等しい値と異なる値を表すためのピリオド、X、および E の使用例については、“例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する” (275 ページ)を参照してください。

PERCENT
このオブザベーションの値は、基準データセットと比較データセットの値のパーセント表示の差異です。文字変数の場合、種類 PERCENT のオブザベーションの値は、種類 DIF のオブザベーションの値と同じになります。

OBS
OUT=オブザベーションのソースをさらに識別する番号を含む数値変数です。

_TYPE_が BASE に等しいオブザベーションの場合、_OBS_は、VAR 変数値のコピー元の基準データセットのオブザベーション番号になります。同様に、_TYPE_が COMPARE に等しいオブザベーションの場合、_OBS_は、VAR 変数値のコピー元の比較データセットのオブザベーション番号になります。

_TYPE_が DIF または PERCENT に等しいオブザベーションの場合、_OBS_は、BY グループのマッチングオブザベーションに振られたシーケンス番号になります。

_OBS_のラベルは Observation Number です。

COMPARE プロシジャは、VAR 変数の長さを除き、ベースデータセットから OUT=データセットの変数名と属性を取得します。COMPARE プロシジャは VAR 変数のよ

り長いほうの長さを使用します。これは、その長さがどのデータセットから取得されるかに関係ありません。この動作によって 2 つの重要な影響が生じます。

- VAR ステートメントと WITH ステートメントを使用すると、OUT=データセットの変数名は VAR ステートメントから取得されます。したがって、_TYPE_が BASE に等しいオブザベーションには VAR 変数の値が含まれ、一方、_TYPE_が COMPARE に等しいオブザベーションは WITH 変数の値が含まれます。
- 1 つの変数を複数の変数と比較するために、VAR ステートメントにその変数を複数含めた場合、PROC COMPARE が OUT=データセットに含められるのは最初の比較のみです。これは、各変数の名前を一意にする必要があるためです。その他の比較については、警告メッセージが生成されます。

OUT=オプションの例については、“例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する” (275 ページ)を参照してください。

出力統計量データセット(OUTSTATS=)

OUTSTATS=オプションを使用すると、PROC COMPARE は比較される数値変数の各ペアに対して、ALLSTATS オプションと同じ要約統計量を計算します。詳細については、“要約統計量のテーブル” (249 ページ)を参照してください。OUTSTATS=データセットには、各変数ペアの各要約統計量ごとに 1 オブザベーションが含まれます。また、データセットには、比較に使用された BY 変数、および PROC COMPARE によって作成された複数の変数も含まれます。

VAR
オブザベーションの統計量の計算対象となった基準データセットの変数名が含まれる文字変数です。

WITH
オブザベーションの統計量の計算対象となった比較データセットの変数名が含まれる文字変数です。_WITH_変数は、WITH ステートメントを使用しない限り、OUTSTATS=データセットには含まれません。

TYPE
オブザベーションに含まれる統計量名を含む文字変数です。_TYPE_変数の値には、 N、MEAN、STD、MIN、MAX、STDERR、T、PROBT、NDIF、DIFMEANS、R、RSQ があります。

BASE
比較データセットにマッチングオブザベーションがある基準データセットのオブザベーションの_VAR_によって指定された変数の値から計算された統計量の値が含まれる数値変数です。

COMP
基準データセットにマッチングオブザベーションがある比較データセットのオブザベーションの_VAR_変数によって(WITH ステートメントを使用している場合は_WITH_変数によって)指定された変数の値から計算された統計量の値が含まれる数値変数です。

DIF
基準データセットの_VAR_変数によって指定された変数と比較データセットの(_VAR_変数または_WITH_変数によって指定された)マッチング変数の値差異から計算された統計量の値が含まれる数値変数です。

PCTDIF
基準データセットの_VAR_変数によって指定された変数と比較データセットの(_VAR_変数または_WITH_変数によって指定された)マッチング変数の値のパーセント表示の差異から計算された統計量の値が含まれる数値変数です。

注: どちらの種類の出力データセットについても、PROC COMPARE は次のデータセットラベルのいずれかを割り当てます。

Comparison of *base-SAS-data-set*
with *comparison-SAS-data-set*

Comparison of variables in *base-SAS-data-set*

ラベルは 40 文字に制限されています。

OUTSTATS=データセットの例については、“例 7: 統計量の出力データセット(OUTSTATS=)の作成” (279 ページ)を参照してください。

例: COMPARE プロシジャ

例 1: 差異の完全なレポートを作成

要素: PROC COMPARE ステートメントオプション
BASE=
PRINTALL
COMPARE=

データセット: Proclib.One、Proclib.Two

詳細

この例では、PROC COMPARE がプロシジャ出力として生成する最も包括的なレポートを示します。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;
```

```
proc compare base=proclib.one compare=proclib.two printall;
  title 'Comparing Two Data Sets: Full Report';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

2 つのデータセットの差異についての包括的なレポートを作成します。 BASE=および COMPARE=は比較するデータセットを指定します。PRINTALL は差異についての完全なレポートを印刷します。

```
proc compare base=proclib.one compare=proclib.two printall;
  title 'Comparing Two Data Sets: Full Report';
run;
```

HTML 出力

出力の A>は、完全なレポートにはあってデフォルトレポートにはない情報を示します。追加情報には、一方のデータセットにはあるが他方にはない変数のリスト、一方のデータセットにはあるが他方にはないオブザベーションのリスト、すべての同等な値を含む変数のリスト、要約統計量などがあります。統計量の説明については、“要約統計量のテーブル” (249 ページ)を参照してください。

アウトプット 11.10　2 つのデータセットの比較(その 1): 完全なレポート

Comparing Two Data Sets: Full Report

```
                        The COMPARE Procedure
                    Comparison of WORK.ONE with WORK.TWO
                              (Method=EXACT)

                           Data Set Summary

Dataset            Created          Modified  NVar    NObs  Label

WORK.ONE   16MAR11:17:52:15  16MAR11:17:52:15     5       4  First Data Set
WORK.TWO   16MAR11:17:52:15  16MAR11:17:52:15     6       5  Second Data Set


                           Variables Summary

              Number of Variables in Common: 5.
              Number of Variables in WORK.TWO but not in WORK.ONE: 1.
              Number of Variables with Conflicting Types: 1.
              Number of Variables with Differing Attributes: 3.
```

```
            Listing of Variables in WORK.TWO but not in WORK.ONE

                        Variable  Type  Length

                        major     Char       8


             Listing of Common Variables with Conflicting Types

                      Variable  Dataset   Type  Length

                      student   WORK.ONE  Num        8
                                WORK.TWO  Char       8
```

アウトプット 11.11　2 つのデータセットの比較(その 2): 完全なレポート

Comparing Two Data Sets: Full Report

```
                        The COMPARE Procedure
                  Comparison of WORK.ONE with WORK.TWO
                             (Method=EXACT)

          Listing of Common Variables with Differing Attributes

       Variable  Dataset   Type  Length  Format  Label

       year      WORK.ONE  Char       8          Year of Birth
                 WORK.TWO  Char       8
       state     WORK.ONE  Char       8
                 WORK.TWO  Char       8          Home State
       gr1       WORK.ONE  Num        8  4.1
                 WORK.TWO  Num        8  5.2
```

```
                  Comparison Results for Observations

      Observation 5 in WORK.TWO not found in WORK.ONE.
```

```
                       Observation Summary

                   Observation      Base  Compare

                   First Obs           1        1
                   First Unequal       1        1
                   Last  Unequal       4        4
                   Last  Match         4        4
                   Last  Obs           .        5

      Number of Observations in Common: 4.
      Number of Observations in WORK.TWO but not in WORK.ONE: 1.
      Total Number of Observations Read from WORK.ONE: 4.
      Total Number of Observations Read from WORK.TWO: 5.

      Number of Observations with Some Compared Variables Unequal: 4.
      Number of Observations with All Compared Variables Equal: 0.
```

アウトプット 11.12　2 つのデータセットの比較(その 3): 完全なレポート

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 3.
Total Number of Values which Compare Unequal: 6.
Maximum Difference: 20.

Variables with All Equal Values

Variable	Type	Len	Label
year	CHAR	8	Year of Birth

Variables with Unequal Values

Variable	Type	Len	Compare Label	Ndif	MaxDif
state	CHAR	8	Home State	2	
gr1	NUM	8		2	1.000
gr2	NUM	8		2	20.000

アウトプット 11.13　2 つのデータセットの比較(その 4): 完全なレポート

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Value Comparison Results for Variables

Obs	Year of Birth Base Value year	Compare Value year
1	1970	1970
2	1971	1971
3	1969	1969
4	1970	1970

Obs	Home State Base Value state	Compare Value state
1	NC	NC
2	MD	MA
3	PA	PA
4	MA	MD

アウトプット 11.14　2 つのデータセットの比較(その 5): 完全なレポート

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Value Comparison Results for Variables

Obs	Base gr1	Compare gr1	Diff.	% Diff
1	85.0	84.00	-1.0000	-1.1765
2	92.0	92.00	0	0
3	78.0	79.00	1.0000	1.2821
4	87.0	87.00	0	0
N	4	4	4	4
Mean	85.5000	85.5000	0	0.0264
Std	5.8023	5.4467	0.8165	1.0042
Max	92.0000	92.0000	1.0000	1.2821
Min	78.0000	79.0000	-1.0000	-1.1765
StdErr	2.9011	2.7234	0.4082	0.5021
t	29.4711	31.3951	0.0000	0.0526
Prob>|t|	<.0001	<.0001	1.0000	0.9614
Ndif	2	50.000%		
DifMeans	0.000%	0.000%	0	
r, rsq	0.991	0.983		

アウトプット 11.15　2 つのデータセットの比較(その 6): 完全なレポート

Comparing Two Data Sets: Full Report

The COMPARE Procedure
Comparison of WORK.ONE with WORK.TWO
(Method=EXACT)

Value Comparison Results for Variables

Obs	Base gr2	Compare gr2	Diff.	% Diff
1	87.0000	87.0000	0	0
2	92.0000	92.0000	0	0
3	72.0000	73.0000	1.0000	1.3889
4	94.0000	74.0000	-20.0000	-21.2766
N	4	4	4	4
Mean	86.2500	81.5000	-4.7500	-4.9719
Std	9.9457	9.4692	10.1776	10.8895
Max	94.0000	92.0000	1.0000	1.3889
Min	72.0000	73.0000	-20.0000	-21.2766
StdErr	4.9728	4.7346	5.0888	5.4447
t	17.3442	17.2136	-0.9334	-0.9132
Prob>|t|	0.0004	0.0004	0.4195	0.4285
Ndif	2	50.000%		
DifMeans	-5.507%	-5.828%	-4.7500	
r, rsq	0.451	0.204		

例 2: 異なるデータセットでの変数の比較

要素:
- PROC COMPARE ステートメントオプション
  - NOSUMMARY
- VAR ステートメント
- WITH ステートメント

データセット: Proclib.One、Proclib.Two

詳細

この例では、基準データセットの変数と比較データセットの変数を比較します。要約レポートはすべて非表示になります。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;

proc compare base=proclib.one compare=proclib.two nosummary;

  var gr1;
  with gr2;
  title 'Comparison of Variables in Different Data Sets';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

2 つのデータセットの差異についての要約レポートをすべて非表示にします。 BASE=は基準データセットを指定し、COMPARE=は比較データセットを指定します。NOSUMMARY はすべての要約レポートを非表示にします。

```
proc compare base=proclib.one compare=proclib.two nosummary;
```

基準データセットから 1 変数を指定して、比較データセットの 1 変数と比較します。 VAR ステートメントと WITH ステートメントは、比較する変数を指定します。この例では、基準データセットの GR1 と比較データセットの GR2 を比較しています。

```
  var gr1;
  with gr2;
  title 'Comparison of Variables in Different Data Sets';
run;
```

HTML 出力

アウトプット 11.16　異なるデータセットでの変数の比較

Comparison of Variables in Different Data Sets

```
                        The COMPARE Procedure
                    Comparison of WORK.ONE with WORK.TWO
                              (Method=EXACT)

NOTE: Data set WORK.TWO contains 1 observations not in WORK.ONE.
NOTE: Values of the following 1 variables compare unequal: gr1^=gr2
```

Value Comparison Results for Variables

Obs	Base gr1	Compare gr2	Diff.	% Diff
1	85.0	87.0000	2.0000	2.3529
3	78.0	73.0000	-5.0000	-6.4103
4	87.0	74.0000	-13.0000	-14.9425

例 3: 変数を複数回比較する

要素: VAR ステートメント
WITH ステートメント

データセット: Proclib.One、Proclib.Two

詳細

この例では、基準データセットの 1 つの変数と比較データセットの 2 つの変数を比較します。

プログラム

```
libname proclib 'SAS-library';
options nodate pageno=1 linesize=80 pagesize=40;
```

```
proc compare base=proclib.one compare=proclib.two nosummary;

  var gr1 gr1;
  with gr1 gr2;
  title 'Comparison of One Variable with Two Variables';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

2 つのデータセットの差異についての要約レポートをすべて非表示にします。 BASE=は基準データセットを指定し、COMPARE=は比較データセットを指定します。NOSUMMARY はすべての要約レポートを非表示にします。

```
proc compare base=proclib.one compare=proclib.two nosummary;
```

基準データセットから 1 つの変数を指定して、比較データセットの 2 つの変数と比較します。 VAR ステートメントと WITH ステートメントは、比較する変数を指定します。この例では、基準データセットの GR1 を、比較データセットの GR1 および GR2 と比較しています。

```
  var gr1 gr1;
  with gr1 gr2;
  title 'Comparison of One Variable with Two Variables';
run;
```

HTML 出力

値比較結果セクションは、比較結果を示しています。

アウトプット 11.17　1 変数と 2 変数の比較

Comparison of One Variable with Two Variables

```
                        The COMPARE Procedure
                   Comparison of WORK.ONE with WORK.TWO
                              (Method=EXACT)

NOTE: Data set WORK.TWO contains 1 observations not in WORK.ONE.
NOTE: Values of the following 2 variables compare unequal: gr1^=gr1 gr1^=gr2
```

Value Comparison Results for Variables

Obs	Base gr1	Compare gr1	Diff.	% Diff
1	85.0	84.00	-1.0000	-1.1765
3	78.0	79.00	1.0000	1.2821

Obs	Base gr1	Compare gr2	Diff.	% Diff
1	85.0	87.0000	2.0000	2.3529
3	78.0	73.0000	-5.0000	-6.4103
4	87.0	74.0000	-13.0000	-14.9425

例 4: 同一データセット内での変数の比較

要素:
- PROC COMPARE ステートメントオプション
  - ALLSTATS
  - BRIEFSUMMARY
- VAR ステートメント
- WITH ステートメント

データセット: Proclib.One

詳細

この例では、PROC COMPARE が同一データセット内の 2 つの変数を比較できることを示します。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;

proc compare base=proclib.one allstats briefsummary;

  var gr1;
  with gr2;
  title 'Comparison of Variables in the Same Data Set';
run;
```

プログラムの説明

Proclib SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

1 つのデータセット内の差異についての簡潔な要約レポートを作成します。 ALLSTATS は要約統計量を印刷します。BRIEFSUMMARY は簡潔な比較要約のみを印刷します。

```
proc compare base=proclib.one allstats briefsummary;
```

比較する 2 つの変数を基準データセットから指定します。 VAR ステートメントと WITH ステートメントは、比較する基準データセットの変数を指定します。この例では、GR1 と GR2 を比較しています。比較データセットがないので、変数 GR1 および GR2 が基準データセットに存在している必要があります。

```
  var gr1;
  with gr2;
  title 'Comparison of Variables in the Same Data Set';
run;
```

HTML 出力

アウトプット 11.18　1 変数と 2 変数の比較

Comparison of One Variable with Two Variables

```
                    The COMPARE Procedure
                Comparisons of variables in WORK.ONE
                        (Method=EXACT)

NOTE: Values of the following 1 variables compare unequal: gr1^=gr2
```

Value Comparison Results for Variables

Obs	Base gr1	Compare gr2	Diff.	% Diff
1	85.0	87.0000	2.0000	2.3529
3	78.0	72.0000	-6.0000	-7.6923
4	87.0	94.0000	7.0000	8.0460
N	4	4	4	4
Mean	85.5000	86.2500	0.7500	0.6767
Std	5.8023	9.9457	5.3774	6.5221
Max	92.0000	94.0000	7.0000	8.0460
Min	78.0000	72.0000	-6.0000	-7.6923
StdErr	2.9011	4.9728	2.6887	3.2611
t	29.4711	17.3442	0.2789	0.2075
Prob>|t|	<.0001	0.0004	0.7984	0.8489
Ndif	3	75.000%		
DifMeans	0.877%	0.870%	0.7500	
r, rsq	0.898	0.807		

例 5: ID 変数を使用してオブザベーションを比較する

要素: ID ステートメント

データセット: Proclib.Emp95
Proclib.Emp96

詳細

この例では、PROC COMPARE は、ID 変数のマッチング値があるオブザベーションのみを比較します。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;

data proclib.emp95;
  input #1 idnum $4. @6 name $15.
        #2 address $42.
        #3 salary 6.;
  datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
... more data lines...
3888 Kim Siu
5662 Magnolia Blvd Southeast Cary NC 27513
77558
;

data proclib.emp96;
  input #1 idnum $4. @6 name $15.
        #2 address $42.
        #3 salary 6.;
  datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
...more data lines...
6544 Roger Monday
3004 Crepe Myrtle Court Raleigh NC 27604
47007
;

proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
```

```
  by idnum;
run;

proc compare base=emp95_byidnum compare=emp96_byidnum;
  id idnum;
  title 'Comparing Observations that Have Matching IDNUMs';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Proclib.Emp95 データセットと Proclib.Emp96 データセットを作成します。 Proclib.Emp95 と Proclib.Emp96 には従業員データが含まれます。IDNUM の値は一意なので、ID 変数に適しています。最初の DATA ステップで、Proclib.Emp95 が作成されます。次の DATA ステップで Proclib.Emp96 が作成されます。

```
data proclib.emp95;
  input #1 idnum $4. @6 name $15.
        #2 address $42.
        #3 salary 6.;
  datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
... more data lines...
3888 Kim Siu
5662 Magnolia Blvd Southeast Cary NC 27513
77558
;

data proclib.emp96;
  input #1 idnum $4. @6 name $15.
        #2 address $42.
        #3 salary 6.;
  datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
...more data lines...
6544 Roger Monday
3004 Crepe Myrtle Court Raleigh NC 27604
```

```
47007
;
```

データセットを ID 変数順に並べ替えます。 両方のデータセットを、PROC COMPARE ステップで ID 変数として使用する変数順に並べ替える必要があります。OUT=は並べ替え後のデータの場所を指定します。

```
proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
  by idnum;
run;
```

ID 変数のマッチング値を含むオブザベーションを比較する要約レポートを作成します。 ID ステートメントは、IDNUM を ID 変数として指定します。

```
proc compare base=emp95_byidnum compare=emp96_byidnum;
  id idnum;
  title 'Comparing Observations that Have Matching IDNUMs';
run;
```

HTML 出力

PROC COMPARE は IDNUM の値によって特定のオブザベーションを識別します。PROC COMPARE は、Value Comparison Results for Variables セクションに、一致しない住所と一致しない給与を印刷します。給与については、PROC COMPARE は数値の差異とパーセント表示の差異を計算します。ADDRESS は文字変数なので、PROC COMPARE は最初の 20 文字のみを表示します。オブザベーションの IDNUM が 0987、2776、3888 の住所については、20 文字目より後ろに差異が発生しているので、出力には差異が表示されていません。出力のプラス記号は、値の一部が表示されていないことを示します。値全体を確認するには、出力データセットを作成します。“例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する” (275 ページ)を参照してください。

アウトプット 11.19　マッチング IDNUM を含むオブザベーションの比較(その 1)

Comparing Observations that Have Matching IDNUMs

```
                          The COMPARE Procedure
          Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
                                (Method=EXACT)

                            Data Set Summary

     Dataset                      Created          Modified  NVar    NObs

     WORK.EMP95_BYIDNUM  17MAR11:08:58:10  17MAR11:08:58:10     4      10
     WORK.EMP96_BYIDNUM  17MAR11:08:58:10  17MAR11:08:58:10     4      12


                            Variables Summary

                   Number of Variables in Common: 4.
                   Number of ID Variables: 1.
```

```
                            Observation Summary

                  Observation      Base  Compare  ID

                  First Obs           1        1  idnum=0987
                  First Unequal       1        1  idnum=0987
                  Last  Unequal      10       12  idnum=9857
                  Last  Obs          10       12  idnum=9857

Number of Observations in Common: 10.
Number of Observations in WORK.EMP96_BYIDNUM but not in WORK.EMP95_BYIDNUM: 2.
Total Number of Observations Read from WORK.EMP95_BYIDNUM: 10.
Total Number of Observations Read from WORK.EMP96_BYIDNUM: 12.

Number of Observations with Some Compared Variables Unequal: 5.
Number of Observations with All Compared Variables Equal: 5.
```

アウトプット 11.20　マッチング IDNUM を含むオブザベーションの比較(その 2)

Comparing Observations that Have Matching IDNUMs

```
                    The COMPARE Procedure
      Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
                          (Method=EXACT)

                  Values Comparison Summary

Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 2.
Total Number of Values which Compare Unequal: 8.
Maximum Difference: 2400.


                Variables with Unequal Values

             Variable  Type  Len  Ndif   MaxDif

             address   CHAR   42     4
             salary    NUM     8     4     2400
```

```
          Value Comparison Results for Variables

_____________________________________________________
       ||  Base Value           Compare Value
idnum  ||  address               address
_____  ||  ____________________+ ____________________+
       ||
0987   ||  2344 Persimmons Bran  2344 Persimmons Bran
2776   ||  12988 Wellington Far  12988 Wellington Far
3888   ||  5662 Magnolia Blvd S  5662 Magnolia Blvd S
9857   ||  1000 Taft Ave. Morri  100 Taft Ave. Morris
_____________________________________________________
```

アウトプット 11.21　マッチング IDNUM を含むオブザベーションの比較(その 3)

Comparing Observations that Have Matching IDNUMs

```
                    The COMPARE Procedure
     Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
                       (Method=EXACT)

            Value Comparison Results for Variables

     _______________________________________________________
              ||       Base    Compare
      idnum   ||     salary     salary      Diff.     % Diff
      _____   ||  _________  _________  _________  _________
              ||
      0987    ||      44010      45110       1100     2.4994
      3286    ||      87734      89834       2100     2.3936
      3888    ||      77558      79958       2400     3.0945
      9857    ||      38756      40456       1700     4.3864
     _______________________________________________________
```

例 6: 出力データセット(OUT=)を使用してオブザベーションの値を比較する

要素: PROC COMPARE ステートメントオプション
- NOPRINT
- OUT=
- OUTBASE
- OUTCOMP
- OUTDIF
- OUTNOEQUAL

PRINT プロシジャ

データセット: Proclib.Emp95
Proclib.Emp96

詳細

この例では、マッチングオブザベーションの差異を示す出力データセットの作成と印刷を行います。

“例 5: ID 変数を使用してオブザベーションを比較する” (269 ページ)では、20 文字目より後の差異は出力に表示されません。この例の出力データセットには、完全な値が表示されます。さらに、一方のデータセットにのみ出現するオブザベーションが表示されます。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=120 pagesize=40;

proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
  by idnum;
run;

proc compare base=emp95_byidnum compare=emp96_byidnum

          out=result outnoequal outbase outcomp outdif
   noprint;

  id idnum;
run;

proc print data=result noobs;
  by idnum;
  id idnum;
  title 'The Output Data Set RESULT';
run;
```

プログラムの説明

PROCLIB SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

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

```
options nodate pageno=1 linesize=120 pagesize=40;
```

データセットを ID 変数順に並べ替えます。 両方のデータセットを、PROC COMPARE ステップで ID 変数として使用する変数順に並べ替える必要があります。OUT=は並べ替え後のデータの場所を指定します。

```
proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
  by idnum;
run;
```

比較するデータセットを指定します。 BASE=および COMPARE=は比較するデータセットを指定します。

```
proc compare base=emp95_byidnum compare=emp96_byidnum
```

出力データセット Result を作成し、すべての不等なオブザベーションとその差異を含めます。 OUT=は、出力データセットの命名と作成を行います。NOPRINT は、プロシジャ出力の印刷を抑制します。OUTNOEQUAL は、不等と判断されたオブザベーションのみを含めます。OUTBASE は、基準データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込みます。OUTCOMP は、比較データセットのオブザベーションごとに 1 つずつオブザベーションを出力データセットに書き込みます。OUTDIF は、2 つのオブザベーションの差異を含む出力データセットにオブザベーションを書き込みます。

```
        out=result outnoequal outbase outcomp outdif
    noprint;
```

ID 変数を指定します。 ID ステートメントは、IDNUM を ID 変数として指定します。

```
  id idnum;
run;
```

出力データセット RESULT を印刷し、BY および ID ステートメントで ID 変数を指定します。 PROC PRINT は、出力データセットを印刷します。BY ステートメントと ID ステートメントで同じ変数を指定すると、出力が読み取りやすくなります。この方法の詳細については、PRINT プロシジャを参照してください。

```
proc print data=result noobs;
  by idnum;
  id idnum;
  title 'The Output Data Set RESULT';
run;
```

HTML 出力

文字変数の差異は、X またはピリオド(.)で示されます。X は文字が一致していないことを示します。ピリオドは文字が一致していることを示します。数値変数の場合、E は差異がないことを意味します。それ以外の場合は、数値の差異が表示されます。デフォルトでは、出力データセットは、比較データセットの 2 つのオブザベーションについて、基準データセットにマッチングオブザベーションがないことを示しています。これらのオブザベーションを出力データセットに表示するためにオプションを使用する必要はありません。

アウトプット 11.22　出力データセット RESULT(その 1)

The Output Data Set RESULT

idnum	_TYPE_	_OBS_	name	address	salary
0987	BASE	1	Dolly Lunford	2344 Persimmons Branch Apex NC 27505	44010
	COMPARE	1	Dolly Lunford	2344 Persimmons Branch Trail Apex NC 27505	45110
	DIF	1	.....	.....XXXXX.XXXXXXXXXXXX	1100

idnum	_TYPE_	_OBS_	name	address	salary
2776	BASE	5	Robert Jones	12988 Wellington Farms Ave. Cary NC 27512	29025
	COMPARE	5	Robert Jones	12988 Wellington Farms Ave. Cary NC 27511	29025
	DIF	5	.....	.....X.	E

idnum	_TYPE_	_OBS_	name	address	salary
3278	COMPARE	6	Mary Cravens	211 N. Cypress St. Cary NC 27512	35362

idnum	_TYPE_	_OBS_	name	address	salary
3286	BASE	6	Hoa Nguyen	2818 Long St. Cary NC 27513	87734
	COMPARE	7	Hoa Nguyen	2818 Long St. Cary NC 27513	89834
	DIF	6	.....	.....	2100

アウトプット 11.23　出力データセット RESULT(その 2)

idnum	_TYPE_	_OBS_	name	address	salary
3888	BASE	7	Kim Siu	5662 Magnolia Blvd Southeast Cary NC 27513	77558
	COMPARE	8	Kim Siu	5662 Magnolia Blvd Southwest Cary NC 27513	79958
	DIF	7	.....	.....XX.....	2400

idnum	_TYPE_	_OBS_	name	address	salary
6544	COMPARE	9	Roger Monday	3004 Crepe Myrtle Court Raleigh NC 27604	47007

idnum	_TYPE_	_OBS_	name	address	salary
9857	BASE	10	Kathy Krupski	1000 Taft Ave. Morrisville NC 27508	38756
	COMPARE	12	Kathy Krupski	100 Taft Ave. Morrisville NC 27508	40456
	DIF	10	.....	...XXXXXXXXXXXXX.XXXXX.XXXXXXXXXX.....	1700

例 7: 統計量の出力データセット(OUTSTATS=)の作成

要素: PROC COMPARE ステートメントオプション
NOPRINT
OUTSTATS=

データセット: Proclib.Emp95
Proclib.Emp96

詳細

この例では、比較された数値変数の要約統計量を含む出力データセットを作成します。

プログラム

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=80 pagesize=40;

proc sort data=proclib.emp95 out=emp95_byidnum;
  by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
  by idnum;
run;

proc compare base=emp95_byidnum compare=emp96_byidnum
        outstats=diffstat noprint;
  id idnum;
run;

proc print data=diffstat noobs;
  title 'The DIFFSTAT Data Set';
run;
```

プログラムの説明

Proclib SAS ライブラリを宣言します。

```
libname proclib 'SAS-library';
```

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

データセットを ID 変数順に並べ替えます。 両方のデータセットを、PROC COMPARE ステップで ID 変数として使用する変数順に並べ替える必要があります。OUT=は並べ替え後のデータの場所を指定します。

```
proc sort data=proclib.emp95 out=emp95_byidnum;
  by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
  by idnum;
run;
```

統計量の出力データセットを作成し、ID 変数のマッチング値があるオブザベーションを比較します。 BASE=および COMPARE=は比較するデータセットを指定します。OUTSTATS=は、出力データセット Diffstat.Noprint を作成し、プロシジャ出力を抑制します。ID ステートメントは、IDNUM を ID 変数として指定します。PROC COMPARE は、IDNUM の値を使用してオブザベーションのマッチングを行います。

```
proc compare base=emp95_byidnum compare=emp96_byidnum
        outstats=diffstat noprint;
  id idnum;
run;
```

出力データセット Diffstat を印刷します。 PROC PRINT は、出力データセット Diffstat を印刷します。

```
proc print data=diffstat noobs;
  title 'The DIFFSTAT Data Set';
run;
```

HTML 出力

変数については、“出力統計量データセット(OUTSTATS=)” (255 ページ)で説明しています。

アウトプット 11.24 Diffstat データセット

The DIFFSTAT Data Set

VAR	_TYPE_	_BASE_	_COMP_	_DIF_	_PCTDIF_
salary	N	10.00	10.00	10.00	10.0000
salary	MEAN	52359.00	53089.00	730.00	1.2374
salary	STD	24143.84	24631.01	996.72	1.6826
salary	MAX	92100.00	92100.00	2400.00	4.3864
salary	MIN	29025.00	29025.00	0.00	0.0000
salary	STDERR	7634.95	7789.01	315.19	0.5321
salary	T	6.86	6.82	2.32	2.3255
salary	PROBT	0.00	0.00	0.05	0.0451
salary	NDIF	4.00	40.00	.	.
salary	DIFMEANS	1.39	1.38	730.00	.
salary	R,RSQ	1.00	1.00	.	.

12

CONTENTS プロシジャ

概要: CONTENTS プロシジャ

CONTENTS プロシジャの機能

CONTENTS プロシジャは、SAS データセットの内容の表示、SAS ライブラリのディレクトリの出力を行います。

大まかに言えば、CONTENTS プロシジャは DATASETS プロシジャの CONTENTS ステートメントと同様に機能します。CONTENTS プロシジャと PROC DATASETS の CONTENTS ステートメントの違いを次に示します。

- PROC CONTENTS の DATA=オプションの *libref* のデフォルトは、Work です。CONTENTS ステートメントの場合、デフォルトはプロシジャ入力ライブラリのライブラリ参照名です。

- PROC CONTENTS は、順編成ファイルを読み込めます。CONTENTS ステートメントは、読めません。

概念: CONTENTS プロシジャ

コンセプト: CONTENTS プロシジャ

CONTENTS ステートメントの概念 (400 ページ)を参照してください。

PROC CONTENTS は、テーブルに関するメタデータと変数に関するメタデータをレポートします。

CAS エンジンは、VARCHAR をサポートする唯一の SAS エンジンです。テーブルに VARCHAR データ型がある場合、PROC CONTENTS は長さをバイト数と文字数で、また使用される最大バイト数で示します。

SAS エンジンのデータセットと同様に、PROC CONTENTS の上部にはテーブルに関する情報が表示されます。PROC CONTENTS(変数メタデータレポートに関連する)の下部は、計算サーバーセッションで表されるメタデータです。

エンコーディングは、CAS テーブルのエンコーディングを示します。CAS UTF-8 エンコーディングからセッションエンコーディングに移行するために必要なトランスコードのタイプに基づいて、セッションで使用される可変バイト長は、セッションのエンコーディングによっては、CAS テーブルのバイト長とは異なる場合があります

アウトプット 12.1　VARCHAR を使用した Mycas.French2 の出力

Data Set Name	MYCAS.FRENCH2	Observations	11
Member Type	DATA	Variables	2
Engine	CAS	Indexes	0
Created	08/16/2017 18:05:10	Observation Length	40
Last Modified	08/16/2017 18:05:10	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Limit	100MB
Caslib	CASUSER,
Scope	Session

Alphabetic List of Variables and Attributes

#	Variable	Type	Len		Max Bytes Used
			Bytes	Chars	
1	nterm	Varchar	80	20	19
2	term	Char	19		

次の PROC CONTENTS は、Hadoop エンジンからの出力を示しています。

アウトプット 12.2　Hadoop エンジン用の PROC CONTENTS

The CONTENTS Procedure

Data Set Name	X.CLASS	Observations	.
Member Type	DATA	Variables	5
Engine	HADOOP	Indexes	0
Created	07/13/2022 23:28:04	Observation Length	0
Last Modified	.	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	Default		
Encoding	Default		

Engine/Host Dependent Information	
Table Type	MANAGED_TABLE
Location	hdfs://fedvm171.unx.sas.com:8020/warehouse/tablespace/managed/hive/accesstesting.db/class
Partition Columns	.
Bucket Columns	.
Num Buckets	.
SerDe Library	org.apache.hadoop.hive.ql.io.orc.OrcSerde
Input Format	org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
Output Format	org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
3	age	Num	8			age
4	height	Num	8			height
1	name	Char	24	$24.	$24.	name
2	sex	Char	3	$3.	$3.	sex
5	weight	Num	8			weight

注意

CONTENTS の OUT=データセットの GENNUM 変数値と DICTIONARY テーブルからの GEN 変数値を混同しないでください。 CONTENTS プロシジャまたはステートメントからの GENNUM は、データセットの特定の世代を参照します。DICTIONARY テーブルからの GEN は、データセットの世代の合計数を参照します。 各 SAS プロシジャは、特定の情報を提供するように設計および構築されています。DICTIONARY.TABLES と PROC CONTENTS からの出力は互換性がありません。

構文: CONTENTS プロシジャ

制限事項:

PROC CONTENTS はオブザベーションを処理しないため、出力に作用する WHERE オプションは使用できません。

SAS データファイルが最大オブザベーション数に達すると、オブザベーション数 (PRINT プロシジャや CONTENTS プロシジャなど)を返す SAS プロシジャは、オブザベーション数に対して、ピリオド(.)で表される欠損値を返します。

SAS / ACCESS LIBNAME エンジンを使用してデータベースにアクセスする 場合、SAS データセットのヘッダーで使用できる情報の一部は使用できません。CONTENTS

や DATASETS などのプロシジャは、システムテーブルに対してクエリを実行しないため、索引、一貫性制約、オブザベーション数などは表示されません。

注: Parquet エンジンを使用する場合、出力の圧縮済みフィールドには COLUMNAR が含まれます。圧縮タイプについては、“COLUMN_COMPRESS= Data Set Option” (*SAS Viya LIBNAME Engines for ORC and Parquet: Reference*)を参照してください。

ATTRIB ステートメントは、PROC CONTENTS 出力には影響を与えません。PROC CONTENTS により、実際のメンバーのラベル、インフォーマット、フォーマットがレポートされます。

ヒント: CONTENTS プロシジャの詳細なドキュメントは、CONTENTS ステートメント (400 ページ)にあります。これらのオプションの説明については、下のテーブルのリンクをクリックして DATASETS プロシジャのドキュメントを参照してください。

PROC CONTENTS の使用時には、DATA=オプション、OUT=オプション、OUT2=オプションとともにデータセットオプションを使用できます。

ORDER=オプションは、OUT=と OUT2=データセットの順序には影響しません。

PROC CONTENTS <*options*>;

ステートメント	タスク	例
“CONTENTS ステートメント”	SAS データセット(複数可)のコンテンツを表示したり、SAS ライブラリのディレクトリを出力したりします。	Ex. 1, Ex. 2, Ex. 3

PROC CONTENTS ステートメント

1 つ以上の SAS データセットのコンテンツを記述し、SAS ライブラリのディレクトリを出力します。

制限事項: PROC CONTENTS はオブザベーションを処理しないため、出力に作用する WHERE オプションは使用できません。

注: ATTRIB ステートメントは、CONTENTS ステートメント出力に影響を与えません。CONTENTS は、実際のメンバーに関するラベル、入力形式、および出力形式を報告します。

SAS/ACCESS LIBNAME エンジンで CONTENTS ステートメントを使用する場合は、“SAS/ACCESSLIBNAME エンジンを使用する場合の DATASETS プロシジャ出力の違い” (369 ページ)を参照してください。

ヒント: DATA=、OUT=および OUT2=オプションではデータセットオプションが使用できます。グローバルステートメントも使用できます。

例: “例 5: SAS データセットを記述” (505 ページ)

構文

PROC CONTENTS<*options*>;

オプション引数

CENTILES
インデックス付き変数に関するパーセント点情報を出力します。

CENTILES オプションが選択され、インデックスがデータセット上に存在する場合、次の追加フィールドが PROC CONTENTS のデフォルトレポートに出力されます。追加フィールドは、インデックスが単一または複合かどうかによって異なります。

#	データセット上のインデックスの数。
インデックス	インデックス名。
Update Centiles	インデックス付き変数に対し CENTILES の前に変更が必要なデータ値のパーセントが自動的に更新されます。
現在の更新率	CENTILES が更新されてから更新されたインデックスのパーセント。
一意の値の数	一意のインデックス付き値の数です。
Variables	インデックスの作成に使用される変数の名前。パーセント点情報は、変数の下にリストされます。

DATA=*SAS-file-specification*
ライブラリ全体またはライブラリ内の特定の SAS データセットを指定します。*SAS-file-specification* は、次の形式のうちいずれかが可能です。

<libref.>SAS-data-set
処理する SAS データセットを 1 つ指定します。ライブラリ参照名のデフォルトは、プロシジャ入力ライブラリのライブラリ参照名です。たとえば、プロシジャ入力ライブラリから SAS データセット HtWt のコンテンツを取得するには、次の CONTENTS ステートメントを使用します。

```
contents data=HtWt;
```

世代グループから特定バージョンのコンテンツを取得するには、次の CONTENTS ステートメントにあるように GENNUM=データセットオプションを使用します。

```
contents data=HtWt(gennum=3);
```

***<libref.>*_ALL_**
MEMTYPE==オプションに指定した種類(複数可)のすべての SAS データセットに関する情報を提供します。*libref* は、SAS ライブラリを参照します。ライブラリ参照名のデフォルトは、プロシジャ入力ライブラリのライブラリ参照名です。

- _ALL_キーワードを使用している場合、SAS ライブラリの読み取り保護されているすべての SAS データセットへの読み取りアクセス権限が必要です。
- DATA=_ALL_は、SAS ライブラリに含まれる SAS ファイルのリストを自動的に出力します。SAS ビューの場合、そのビューと関連付けられているすべてのライブラリ参照名は、リストに対して処理されるように現在のセッションで割り当てられる必要があります。

デフォルト SAS ライブラリからの、ジョブまたはセッションで最も新しく作成されたデータセット。

ヒント DATA=オプションで読み取り保護されているデータセットを指定し、読み取りパスワードを与えない場合、デフォルトでプロシジャが PROC DATASETS ステートメントで読み取りパスワードを検索します。ただし、DATA=オプションを指定せず、デフォルトのデータセット(セッションで最後に作成されたもの)が読み取り保護されている場合、プロシジャは PROC DATASETS ステートメントで読み取りパスワードを検索しません。

例 “例 5: SAS データセットを記述” (505 ページ)

DETAILS | NODETAILS
オブザベーションの数、変数の数、インデックスの数、およびデータセットラベルに関する情報を出力に含めます。DETAILS には出力に情報の追加列が含まれますが、DIRECTORY も指定されている場合だけです。

デフォルト DETAILS も NODETAILS も指定されていない場合、デフォルトは次のようになります。CONTENTS プロシジャの場合、デフォルトはシステムオプション設定の NODETAILS です。CONTENTS ステートメントの場合、デフォルトは PROC DATASETS ステートメントで指定されている値で、これもシステムオプション設定となります。

参照項目: PROC DATASETS ステートメント (376 ページ) の Optional Argument セクションの追加列の説明

DIRECTORY
指定した SAS ライブラリのすべての SAS ファイルのリストを出力します。DETAILS も指定されている場合、DIRECTORY を使用すると DETAILS | NODETAILS (289 ページ)に記述されている追加列が出力されます。

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。詳細については、“ライブラリコンテンツと AES 暗号化” (404 ページ)を参照してください。

FMTLEN
入力形式または出力形式の長さを出力します。入力形式または出力形式の長さを変数と関連付けるときに指定しない場合、長さは FMTLEN オプションを使用しない限り CONTENTS ステートメントの出力に表示されません。長さは出力データセットの FORMATL 変数または INFORML 変数にも表示されます。

MEMTYPE=(*member-type(s)*)
処理を 1 つ以上のメンバーの種類に制限します。CONTENTS ステートメントは、メンバーの種類 DATA、VIEW および ALL(DATA と VIEW が含まれます)に対してのみ出力を生成します。

CONTENTS ステートメントの MEMTYPE=は、DATASETS プロシジャのその他の大部分のステートメントの MEMTYPE=と次の点が異なります。

- スラッシュはオプションの前に指定できません。
- MEMTYPE=オプションをかっこで囲み、その影響をその直前の SAS ファイルにのみ制限することはできません。

MEMTYPE=は結果として、DATA=メンバーが置かれているライブラリのディレクトリとなります。ただし、MEMTYPE=は、_ALL_キーワードが DATA=オプションで使用されない限り、コンテンツが表示されるメンバーの種類を制限しません。たとえば、次のステートメントは、メンバーの種類が DATA の SAS データセットのみのコンテンツを生成します。

```
proc datasets memtype=data;
  contents data=_all_;
run;
```

別名 MTYPE=

MT=

デフォルト DATA

NODS
DATA=オプションで_ALL_を指定する際に、個々のファイルの内容を出力しません。CONTENTS ステートメントは、SAS ライブラリディレクトリのみを出力します。DATA=オプションで SAS データセットを 1 つだけ指定する際に、NODS オプションは使用できません。

NOPRINT
CONTENTS ステートメントの出力を行いません。

ORDER=COLLATE | CASECOLLATE | IGNORECASE | VARNUM

COLLATE 変数のリストを大文字名、次に小文字名で始まるアルファベット順で出力します。

CASECOLLATE 変数の一覧を、大文字と小文字が混在する名前と数値が含まれている場合でも、アルファベット順で出力します。

IGNORECASE 変数のリストを大文字と小文字に関係なくアルファベット順で出力します。

VARNUM VARNUM オプションと同じです。

注 ORDER=オプションは、OUT=と OUT2=データセットの順序には影響しません。

参照項目: “VARNUM” (291 ページ)

例 ORDER=のデフォルトと 4 つのオプションを比較するには、“例 3: ORDER=オプションの使用” (296 ページ)を参照してください。

OUT=*SAS-data-set*
出力 SAS データセットを指定します。

ヒント OUT=は、ステートメントからの出力を非表示にしません。出力を非表示にする場合、NOPRINT オプションを使用する必要があります。

参照項目: “OUT=データセット” (480 ページ)には、OUT=データセットの変数の説明が記載されています。

OUT2=*SAS-data-set*
インデックスと一貫性制約に関する情報を含む出力データセットを指定します。

注 OUT2=*PermanentLibrary*_ALL_オプションを PROC CONTENTS または PROC DATASETS 内で CONTENTS ステートメントとともに使用する場合、REPLACE=YES データセットオプションまたは REPLACE システムオプションも設定する必要があります。

ヒント UPDATECENTILES がインデックス定義で指定されなかった場合、デフォルト値 5 が OUT2 データセットの RECREATE 変数で使用されます。

OUT2=は、ステートメントからの出力を非表示にしません。出力を非表示にするには、NOPRINT オプションを使用する必要があります。

参照項目: “OUT2=データセット” (486 ページ)には、OUT2=データセットの変数の説明が記載されています。

SHORT
SAS データセットの変数名のリスト、インデックス情報、並べ替え情報のみ出力します。

制限事項 変数のリストが 32,767 文字を超える場合、リストは切り捨てられ、WARNING が SAS ログに書き込まれます。変数の完全リストを取得するには、変数のアルファベット順リストを要求します。

VARNUM
変数名のリストをデータセットの論理位置の順序で出力します。デフォルトでは、CONTENTS ステートメントは変数をアルファベット順でリストします。データセットの変数の物理位置は、エンジン依存です。

使用法: CONTENTS プロシジャ

PROC CONTENTS の使用

“CONTENTS ステートメント” (400 ページ)を参照してください。

詳細については、4 章, “Base プロシジャの CAS 処理” (75 ページ)を参照してください。

例: CONTENTS プロシジャ

例 1: SAS データセットを記述

要素:

PROC CONTENTS ステートメントオプション
- DATA=
- OUT=

OPTIONS ステートメント

TITLE ステートメント

詳細

この例は、Class データセットに対する CONTENTS プロシジャからの出力を示しています。出力には、“例 4: SAS データセットの変更” (502 ページ)で Class データセットに行われた変更と、Class2 データセットの内容が示されています。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME v9 'SAS-library';

proc contents data=sashelp (read=green) out=Class3;
  title  'The Contents of the Class Data Set';
run;

proc contents data=shelp.class;
  title  'The Contents of the Class Data Set';
run;
```

プログラムの説明

システムオプションを設定します。 PAGESIZE=オプションは SAS ログと SAS 出力のページを構成する行数を指定します。LINESIZE=オプションでは、SAS ログと SAS プロシジャ出力の行サイズを指定します。NODATE オプションは、日時を印刷しな

いことを指定します。PAGENO=オプションでは、出力の次ページの開始ページ番号を指定します。

```
options pagesize=40 linesize=80 nodate pageno=1;
```

ライブラリ参照名を設定します。

```
LIBNAME v9 'SAS-library';
```

データセット Class から出力データセット Class3 を作成します。 Class データセットへの読み取りアクセス権を付与します。

```
proc contents data=sashelp (read=green) out=Class3;
  title  'The Contents of the Class Data Set';
run;
```

Class データセットのコンテンツが示されます。

```
proc contents data=shelp.class;
  title  'The Contents of the Class Data Set';
run;
```

出力例

アウトプット 12.3　Class データセットのコンテンツ

The Contents Using Directory

The CONTENTS Procedure

Directory	
Libref	WORK
Engine	V9
Physical Name	/opt/sas/viya/config/var/tmp/compsrv/default/c4f057e0-c534-4802-967d-5bf07f9a0901/SAS_work923C000002A6_launcher-b966616f-b389-4b4e-8f22-3c578bc9789c-bxwxt
Filename	/opt/sas/viya/config/var/tmp/compsrv/default/c4f057e0-c534-4802-967d-5bf07f9a0901/SAS_work923C000002A6_launcher-b966616f-b389-4b4e-8f22-3c578bc9789c-bxwxt
Inode Number	222690576
Access Permission	rwx------
Owner Name	UNKNOWN
File Size	4KB
File Size (bytes)	4096

Engine/Host Dependent Information	
Data Set Page Size	73728
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	82
Obs in First Data Page	41
Number of Data Set Repairs	0
Filename	/opt/sas/viya/config/var/tmp/compsrv/default/c4f057e0-c534-4802-967d-5bf07f9a0901/SAS_work923C000002A6_launcher-b966616f-b389-4b4e-8f22-3c578bc9789c-bxwxt/class.sas7bdat
Release Created	V.0400M0
Host Created	Linux
Inode Number	222692708
Access Permission	rw-r--r--
Owner Name	UNKNOWN
File Size	144KB
File Size (bytes)	147456

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
32	CHARSET	Char	8		Host Character Set
33	COLLATE	Char	8		Collating Sequence
28	COMPRESS	Char	8		Compression Routine
20	CRDATE	Num	8	DATETIME16.	Create Date
22	DELOBS	Num	8		Deleted Observations in Data Set
36	ENCRYPT	Char	8		Encryption Routine
19	ENGINE	Char	8		Engine Name
27	FLAGS	Char	3		Update Flags (Protect Contribute Add)
10	FORMAT	Char	32		Variable Format
12	FORMATD	Num	8		Number of Format Decimals
11	FORMATL	Num	8		Format Length
38	GENMAX	Num	8		Maximum Number of Generations
40	GE				umber

例 2: DIRECTORY オプションの使用

要素:

PROC CONTENTS ステートメントオプション
- DATA=
- DIRECTORY
- OUT=

OPTIONS ステートメント

TITLE ステートメント

詳細

この例は、Class3 データセットに対する CONTENTS プロシジャからの出力を示しています。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME health 'SAS-library';

proc contents data=health.group (read=green) directory;
title 'Contents Using the DIRECTORY Option';
run;
```

プログラムの説明

システムオプションを設定します。 PAGESIZE=オプションで、SAS ログと SAS 出力を構成する行の数が指定されます。LINESIZE=オプションでは、SAS ログと SAS プロシジャ出力の行サイズを指定します。NODATE オプションは、日時を印刷しないことを指定します。PAGENO=オプションでは、出力の次ページの開始ページ番号を指定します。

```
options pagesize=40 linesize=80 nodate pageno=1;
```

ライブラリ参照名を設定します。

```
LIBNAME health 'SAS-library';
```

記述するデータセットとして Class3 を指定します。 DIRECTORY オプションを使用して、HEALTH ライブラリ内にあるすべてのデータセットのリストを出力します。

```
proc contents data=health.group (read=green) directory;
title 'Contents Using the DIRECTORY Option';
run;
```

出力例

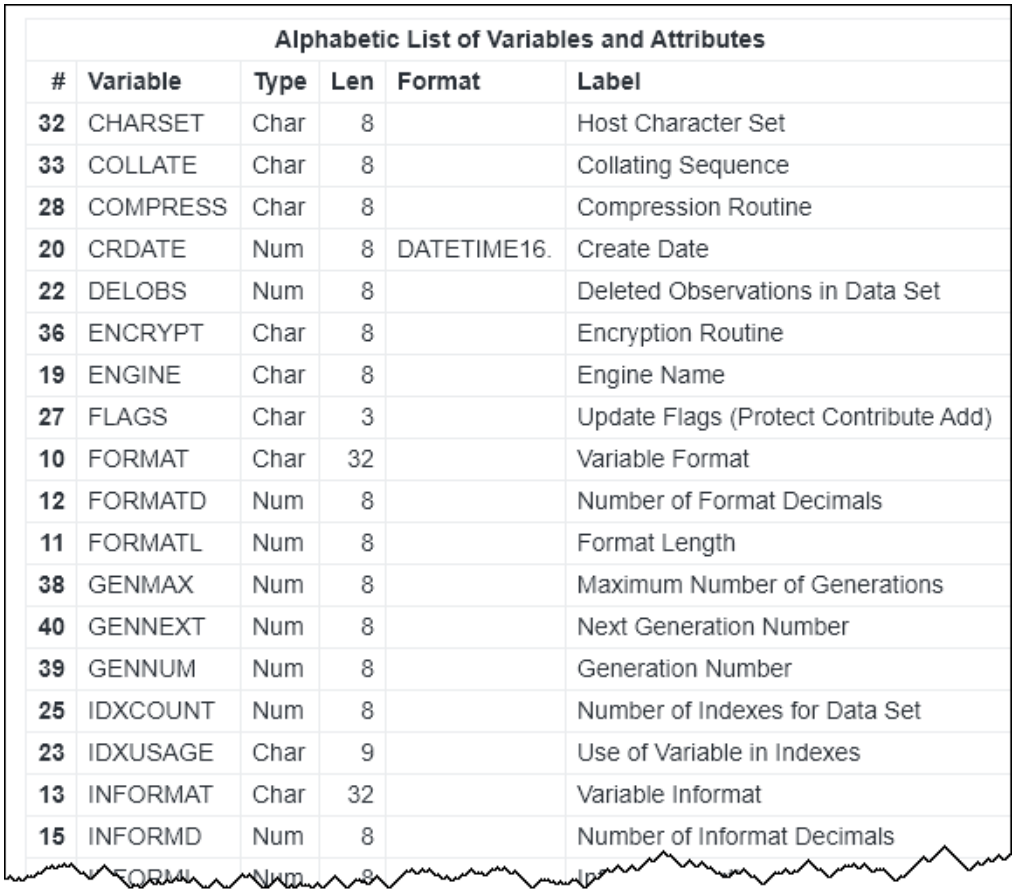
アウトプット 12.4　DIRECTORY オプションの使用 - セクション 1

The Contents Using Directory

The CONTENTS Procedure

Directory	
Libref	WORK
Engine	V9
Physical Name	/opt/sas/viya/config/var/tmp/compsrv/default/c4f057e0-c534-4802-967d-5bf07f9a0901/SAS_work923C000002A6_launcher-b966616f-b389-4b4e-8f22-3c578bc9789c-bxwxt
Filename	/opt/sas/viya/config/var/tmp/compsrv/default/c4f057e0-c534-4802-967d-5bf07f9a0901/SAS_work923C000002A6_launcher-b966616f-b389-4b4e-8f22-3c578bc9789c-bxwxt
Inode Number	222690576
Access Permission	rwx------
Owner Name	UNKNOWN
File Size	4KB
File Size (bytes)	4096

Engine/Host Dependent Information	
Data Set Page Size	73728
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	82
Obs in First Data Page	41
Number of Data Set Repairs	0
Filename	/opt/sas/viya/config/var/tmp/compsrv/default/c4f057e0-c534-4802-967d-5bf07f9a0901/SAS_work923C000002A6_launcher-b966616f-b389-4b4e-8f22-3c578bc9789c-bxwxt/class3.sas7bdat
Release Created	V.0400M0
Host Created	Linux
Inode Number	222692710
Access Permission	rw-r--r--
Owner Name	UNKNOWN
File Size	144KB
File Size (bytes)	147456



Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
32	CHARSET	Char	8		Host Character Set
33	COLLATE	Char	8		Collating Sequence
28	COMPRESS	Char	8		Compression Routine
20	CRDATE	Num	8	DATETIME16.	Create Date
22	DELOBS	Num	8		Deleted Observations in Data Set
36	ENCRYPT	Char	8		Encryption Routine
19	ENGINE	Char	8		Engine Name
27	FLAGS	Char	3		Update Flags (Protect Contribute Add)
10	FORMAT	Char	32		Variable Format
12	FORMATD	Num	8		Number of Format Decimals
11	FORMATL	Num	8		Format Length
38	GENMAX	Num	8		Maximum Number of Generations
40	GENNEXT	Num	8		Next Generation Number
39	GENNUM	Num	8		Generation Number
25	IDXCOUNT	Num	8		Number of Indexes for Data Set
23	IDXUSAGE	Char	9		Use of Variable in Indexes
13	INFORMAT	Char	32		Variable Informat
15	INFORMD	Num	8		Number of Informat Decimals
	FORML	Num	8		In...

例 3: ORDER=オプションの使用

要素:

PROC CONTENTS ステートメントオプション
- DATA=
- ORDER=
- OUT=

OPTIONS ステートメント

TITLE ステートメント

詳細

この例は、ORDER=オプションを使用して Grpout データセットの CONTENTS プロシジャからの出力を示しています。これは、異なる順序で変数のリストを出力します。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME health 'SAS-library';

proc contents data=health.grpout order=collate;
title 'Contents Using the ORDER= Option';
run;

proc contents data=health.grpout order=varnum;
title 'Contents Using the ORDER= Option';
run;
```

プログラムの説明

システムオプションを設定します。 PAGESIZE=オプションで、SAS ログと SAS 出力を構成する行の数が指定されます。LINESIZE=オプションでは、SAS ログと SAS プロシジャ出力の行サイズを指定します。NODATE オプションは、日時を印刷しないことを指定します。PAGENO=オプションでは、出力の次ページの開始ページ番号を指定します。

```
options pagesize=40 linesize=80 nodate pageno=1;
```

ライブラリ参照名を設定します。

```
LIBNAME health 'SAS-library';
```

Group データセットを指定します。 すべての変数のリストをアルファベット順に出力するには、ORDER=COLLATE オプションを使用します。

```
proc contents data=health.grpout order=collate;
title 'Contents Using the ORDER= Option';
run;
```

Group データセットを指定します。 すべての変数のリストを番号順に出力するには、ORDER=VARNUM オプションを使用します。

```
proc contents data=health.grpout order=varnum;
title 'Contents Using the ORDER= Option';
run;
```

出力例

アウトプット 12.5　ORDER=COLLATE オプションの使用

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
32	CHARSET	Char	8		Host Character Set
33	COLLATE	Char	8		Collating Sequence
28	COMPRESS	Char	8		Compression Routine
20	CRDATE	Num	8	DATETIME16.	Create Date
22	DELOBS	Num	8		Deleted Observations in Data Set
36	ENCRYPT	Char	8		Encryption Routine
19	ENGINE	Char	8		Engine Name
27	FLAGS	Char	3		Update Flags (Protect Contribute Add)
10	FORMAT	Char	32		Variable Format
12	FORMATD	Num	8		Number of Format Decimals
11	FORMATL	Num	8		Format Length
38	GENMAX	Num	8		Maximum Number of Generations
40	GENNEXT	Num	8		Next Generation Number
39	GENNUM	Num	8		Generation Number
25	IDXCOUNT	Num	8		Number of Indexes for Data Set
23	IDXUSAGE	Char	9		Use of Variable in Indexes
13	INFORMAT	Char	32		Variable Informat
15	INFORMD	Num	8		Number of Informat Decimals
14	INFORML	Num	8		Informat Length
16	JUST	Num	8		Justification

9	LABEL	Char	256		Variable Label
7	LENGTH	Num	8		Variable Length
1	LIBNAME	Char	8		Library Name
3	MEMLABEL	Char	256		Data Set Label
2	MEMNAME	Char	32		Library Member Name
24	MEMTYPE	Char	8		Library Member Type
21	MODATE	Num	8	DATETIME16.	Last Modified Date
5	NAME	Char	32		Variable Name
18	NOBS	Num	8		Observations in Data Set
34	NODUPKEY	Char	3		Sort Option: No Duplicate Keys
35	NODUPREC	Char	3		Sort Option: No Duplicate Records
17	NPOS	Num	8		Position in Buffer
37	POINTOBS	Char	3		Point to Observations
26	PROTECT	Char	3		Password Protection (Read Write Alter)
29	REUSE	Char	3		Reuse Space
30	SORTED	Num	8		Sorted and/or Validated
31	SORTEDBY	Num	8		Position of Variable in Sortedby Clause
41	TRANSCOD	Char	3		Character Variables Transcoded
6	TYPE	Num	8		Variable Type
4	TYPEMEM	Char	8		Special Data Set Type (From TYPE=)
8	VARNUM	Num	8		Variable Number

アウトプット 12.6　ORDER=VARNUM オプションの使用

Variables in Creation Order					
#	Variable	Type	Len	Format	Label
1	LIBNAME	Char	8		Library Name
2	MEMNAME	Char	32		Library Member Name
3	MEMLABEL	Char	256		Data Set Label
4	TYPEMEM	Char	8		Special Data Set Type (From TYPE=)
5	NAME	Char	32		Variable Name
6	TYPE	Num	8		Variable Type
7	LENGTH	Num	8		Variable Length
8	VARNUM	Num	8		Variable Number
9	LABEL	Char	256		Variable Label
10	FORMAT	Char	32		Variable Format
11	FORMATL	Num	8		Format Length
12	FORMATD	Num	8		Number of Format Decimals
13	INFORMAT	Char	32		Variable Informat
14	INFORML	Num	8		Informat Length
15	INFORMD	Num	8		Number of Informat Decimals
16	JUST	Num	8		Justification
17	NPOS	Num	8		Position in Buffer
18	NOBS	Num	8		Observations in Data Set
19	ENGINE	Char	8		Engine Name
20	CRDATE	Num	8	DATETIME16.	Create Date

21	MODATE	Num	8	DATETIME16.	Last Modified Date
22	DELOBS	Num	8		Deleted Observations in Data Set
23	IDXUSAGE	Char	9		Use of Variable in Indexes
24	MEMTYPE	Char	8		Library Member Type
25	IDXCOUNT	Num	8		Number of Indexes for Data Set
26	PROTECT	Char	3		Password Protection (Read Write Alter)
27	FLAGS	Char	3		Update Flags (Protect Contribute Add)
28	COMPRESS	Char	8		Compression Routine
29	REUSE	Char	3		Reuse Space
30	SORTED	Num	8		Sorted and/or Validated
31	SORTEDBY	Num	8		Position of Variable in Sortedby Clause
32	CHARSET	Char	8		Host Character Set
33	COLLATE	Char	8		Collating Sequence
34	NODUPKEY	Char	3		Sort Option: No Duplicate Keys
35	NODUPREC	Char	3		Sort Option: No Duplicate Records
36	ENCRYPT	Char	8		Encryption Routine
37	POINTOBS	Char	3		Point to Observations
38	GENMAX	Num	8		Maximum Number of Generations
39	GENNUM	Num	8		Generation Number
40	GENNEXT	Num	8		Next Generation Number
41	TRANSCOD	Char	3		Character Variables Transcoded

13

COPY プロシジャ

概要: COPY プロシジャ

COPY プロシジャの機能

COPY プロシジャは、1 つ以上のテーブルを、1 つの SAS ライブラリから別の SAS ライブラリにコピーします。

大まかに言えば、COPY プロシジャは DETASETS プロシジャの COPY ステートメントと同様に機能します。その違いは、次のとおりです。

- PROC COPY には IN=引数が必須です。COPY ステートメントでは IN=はオプションです。省略された場合、デフォルト値はプロシジャの入力ライブラリのライブラリ参照名です。
- PROC DATASETS は、順次データアクセスのみを許可するライブラリには使用できません。
- COPY ステートメントは NOWARN オプションを順守しますが、PROC COPY は違います。

注: MIGRATE プロシジャは、特に SAS ライブラリを前のリリースから最新のリリースに移行するためのものです。移行に関して、PROC MIGRATE は PROC COPY にはない利点を提供できます。詳細については、MIGRATE プロシジャ (1335 ページ)を参照してください。

ACCEL オプションを使用すると、PROC COPY は CAS アクションを実行して、同じ CAS セッション内の caslib 間で CAS テーブルをコピーします。

PROC COPY の In-Database 処理

In-Database 処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。機密データをデータソースから抽出する必要がないため、セキュリティを強化できます。データが、比較的低速なネットワーク接続を介して移送されるかわりに、高速の二次記憶装置を使用して、データソース上でローカル操作されます。データソースが使用されるのは、データソースに自由に使えるより多くの処理リソースがあり、クエリを最適化して高度に並列かつスケーラブルな方法で実行できるためです。

DATA=入力テーブルがデータソースでテーブルまたはビューとして 保存される場合、PROC COPY は In-Database 処理を使用して、データソース内の作業のほとんどを実行できます。In-Database 処理は、より迅速な処理と、データソースと SAS 間のデータ転送の減少という利点を提供できます。

PROC COPY の In-Database 処理では、次のデータソースがサポートされています。

- Greenplum
- PostgreSQL
- Vertica

COPY シジャはデータの動的な変換を実行します。この変換では、出力テーブルの特性、特に変数の数と名前およびそれらの型が、変数値および入力テーブルの特性から判断されます。この動的な動作は、2 パス処理で実現されます。最初のパスで入力テーブルの行を調べて出力テーブルの特性を確認し、2 回目のパスでデータを転置する作業を実行します。大量並列処理(MPP)データベース内での並列処理により、最初のパスと 2 回目のパスの両方が高速化されます。MPP では、多数のプロセッサまたは別のコンピューターを使用して一連の協調型計算を並列で実行します。最初のパスでは、クラスターのノードにすでに分割されている行のとおりに、並列

かつインプレースで調べられます。2 回目のパスで、行は再分割されて BY グループが作成され、それらのグループを個別に並列で処理します。

In-Database 処理の最初のパスと 2 回目のパスはどちらも、クラスターのノード内にある SAS Embedded Process 内の DS2 プログラムを実行して行われます。DBMS により、SAS Embedded Process ではテーブルからデータを読み込み、データをテーブルに書き込むことができます。SAS Embedded Process は、DS2 プログラムの実行コンテキストを提供します。作業の 2 つのパスは DS2 言語で表されるため、テーブルの列は、DS2 データ型を持つ変数にキャストされます。DS2 内のデータ型サポートは、従来の SAS システムで提供されているサポートよりも範囲が広いため、データベース内で実行すると、転置された出力データ内で入力データのデータ型と値を維持する COPY プロシジャの機能が拡張されます。

SQLGENERATION システムオプションまたは LIBNAME ステートメントオプションは、In-Database プロシジャがデータベース内で実行されるかどうか、およびその実行方法を制御します。In-Database 処理の妨げとなる可能性のあるプログラミングの考慮事項が多数あります。完全なリストについては、“Procedure Considerations and Limitations” (*SAS In-Database Products: User's Guide*)を参照してください。

PROC COPY で In-Database 処理を使用する例については、次のリソースを参照してください。

- “Running PROC COPY In-Database for Greenplum” (*SAS/ACCESS for Relational Databases: Reference*)
- “Running PROC COPY In-Database for PostgreSQL” (*SAS/ACCESS for Relational Databases: Reference*)
- “Running PROC COPY In-Database for Vertica” (*SAS/ACCESS for Relational Databases: Reference*)

構文: COPY プロシジャ

該当要素: 計算サーバー, CAS

制限事項: PROC COPY は、カタログとの連結を無視します。連結カタログをコピーする場合は、PROC CATALOG COPY を使用します。

PROC COPY では、データセットオプションはサポートされていません。

PROC COPY では、グラフィックカタログはバックアップされません。グラフィックカタログでバックアップするには、PROC CPORT または PROC CIMPORT を使用してください。

操作: 言語照合順序があるデータセットを並べ替えるには、ICU (International Components for Unicode)バージョンを使用します。言語的に並べ替えられたデータセットの ICU バージョン番号が現在のセッションの ICU バージョン番号と異なる場合、PROC COPY は、OUT=出力先ライブラリにデータセットの並べ替え順序を保持します。ただし、そのデータセットから並べ替え済みのマーキングがなくなり、SAS ログに警告メッセージが書き込まれます。言語的な並べ替えの詳細については、55 章, “SORT プロシジャ,” (2027 ページ)を参照してください。

注: 長整数よりも大きい大きな数値で終わるデータセット名および変数名は、番号付き範囲リストでは使用できません。詳細については、“SAS 変数の定義” (*SAS プログラマガイド: エッセンシャル*)を参照してください。.

PROC COPY は CAS 動作を使用して、IN=値と OUT=値の両方が CAS libname エンジンを使用し、両方の libnames が同じ CAS セッションを使用する場合、CAS サーバーでコピー操作を提供します。“CAS テーブルを別の CAS テーブルにコピー” (417 ページ)と“PROC COPY の CAS 処理” (322 ページ)を参照してください。

ヒント: COPY プロシジャの詳細なドキュメントは、PROC DATASETS の COPY ステートメント (407 ページ)にあります。

ステートメント ATTRIB、FORMAT、LABEL、WHERE を PROC COPY プロシジャと使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (23 ページ)を参照してください。

PROC COPY

<ACCEL | NOACCEL >
OUT=*libref-1*
<CLONE | NOCLONE>
< CONSTRAINT=YES | NO>
< DATECOPY>
< ENCRYPTKEY=*key-value*>
< FORCE>
IN=*libref-2*
< INDEX=YES | NO>
< MEMTYPE=(*member-type(s)*)>
<MOVE < ALTER=*alter-password*>>
< OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2 ...>*) >;

ATTRIB *variable-list(s) attribute-list(s)*;

EXCLUDE *SAS-file(s)*< / MEMTYPE=*member-type*>;

FORMAT *variable-1*<...*variable-n*> <*format*> <DEFAULT=*default-format*>;

LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;

SELECT *SAS-file(s)*
< / < ENCRYPTKEY=*key-value* > < ALTER=*alter-password* > <
MEMTYPE=*member-type* >>;

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク	例
“COPY ステートメント”	1 つ以上のファイルをコピーします	Ex. 1
“EXCLUDE ステートメント”	ファイルや種類を除外します	Ex.
“SELECT ステートメント”	ファイルや種類を選択します	Ex. 1, Ex. 2

PROC COPY ステートメント

SAS ライブラリのすべての、または一部の SAS ファイルをコピーします。

制限事項: COPY ステートメントでは、データセットオプションはサポートしていません。

注: SAS/ACCESS LIBNAME エンジンで COPY ステートメントを使用する場合は、“SAS/ACCESSLIBNAME エンジンを使用する場合の DATASETS プロシジャ出力の違い” (369 ページ)を参照してください。

CAS エンジンの詳細については、“PROC COPY の CAS 処理” (322 ページ)を参照してください。

ヒント: COPY ステートメントは、SAS/SHARE、SAS/CONNECT などのリモートライブラリサービス(RLS)の使用時は出力ライブラリのエンコーディングおよびデータ表現をデフォルトとして使用します。RLS を使用していない場合、PROC COPY オプション NOCLONE を出力ファイルに使用して、出力ライブラリのエンコーディングとデータ表現を求める必要があります。NOCLONE オプションを使用すると、データライブラリのデータ表現(OUTREP= LIBNAME オプションで指定されている場合)、または動作環境のネイティブなデータ表現を含むコピーになります。

例: “例 2: SAS ファイルの操作” (494 ページ)

構文

PROC COPY<ACCEL | NOACCEL >OUT=*libref-1*
<CLONE | NOCLONE>
< CONSTRAINT=YES | NO>
< DATECOPY>
< ENCRYPTKEY=*key-value*>
< FORCE>
IN=*libref-2*
< INDEX=YES | NO>
< MEMTYPE=(*member-type(s)*)>
<MOVE < ALTER=*alter-password*>>
< OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2 ...>*) >;

必須引数

OUT=*libref-1*
SAS ファイルのコピー先となる SAS ライブラリを指定します。

別名 OUTLIB=

OUTDD=

例 “例 2: SAS ファイルの操作” (494 ページ)

オプション引数

ACCEL | NOACCEL
CAS でコピー操作を実行するかどうかを指定します。IN=ライブラリと OUT=ライブラリの両方が CAS エンジンライブラリでなければならず、かつ同じ CAS セッションを使用する必要があります。

デフォルト ACCEL

参照項目: “CAS テーブルを別の CAS テーブルにコピー” (417 ページ)

CLONE | NOCLONE
次のデータセット属性をコピーするかどうかを指定します。

- 入力/出力バッファーのサイズ
- データセットの圧縮
- 空きスペースの再利用
- 入力データセットのデータ表現、ライブラリ、動作環境
- 値のエンコーディング
- 圧縮されているデータセットにオブザベーション番号で無作為にアクセス可能かどうか

これらの属性は、データセットオプション、SAS システムオプション、LIBNAME ステートメントオプションで指定されます。

- BUFSIZE= (入力/出力バッファーのサイズの値)
- COMPRESS= (データセットを圧縮するかどうかの値)
- REUSE= (空きスペースを再利用するかどうかの値)
- OUTREP= (データ表現の値)
- ENCODING=または INENCODING= (値のエンコーディング)
- POINTOBS= (圧縮されたデータセットにオブザベーション番号で無作為にアクセスできるかどうかの値)

次のテーブルに、BUFSIZE=属性に関する COPY ステートメントの動作を要約します。

表 13.1 CLONE とバッファーページサイズ属性

オプション	COPY ステートメント
CLONE	入力データセットからの BUFSIZE=値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで BUFSIZE=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	SAS システムオプション BUFSIZE=の現在の設定を出力データセットに使用します。
いずれも使用しない	入力データセット用エンジンと出力データセット用エンジンによって使用されるアクセス方法の種類(順次または無

オプション	COPY ステートメント
	作為)を指定します。両方のエンジンで同じ種類のアクセスが使用されている場合、COPY ステートメントは入力データセットの BUFSIZE=値を出力データセットに使用します。同じ種類のアクセスが使用されていない場合、COPY ステートメントは SAS システムオプション BUFSIZE=の設定を出力データセットに使用します。

次のテーブルに、COMPRESS=属性に関する COPY ステートメントの動作を要約します。

表 13.2　CLONE と圧縮属性

オプション	COPY ステートメント
CLONE	入力データセットの値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで COMPRESS=値を指定すると、コピーで指定エンコードが使用されることになります。
NOCLONE	動作環境の圧縮を含むコピー、または指定されている場合は、ライブラリに対する LIBNAME ステートメントの COMPRESS=オプションの値となります。
いずれも使用しない	デフォルトは CLONE となります。

次のテーブルに、REUSE=属性に関する COPY ステートメントの動作を要約します。

表 13.3　CLONE と空きスペースの再利用属性

オプション	COPY ステートメント
CLONE	入力データセットの値を出力データセットに使用します。入力データセット用エンジンで空きスペースの再利用属性がサポートされていない場合、COPY ステートメントは対応する SAS システムオプションの現在の設定を使用します。ただし、OVERRIDE=オプションリストで REUSE=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	SAS システムオプション COMPRESS=と REUSE=の現在の設定を出力データセットに使用します。
いずれも使用しない	デフォルトは CLONE となります。

次のテーブルに、OUTREP=属性に関する COPY ステートメントの動作を要約します。

表 13.4　CLONE とデータ表現属性

オプション	COPY ステートメント
CLONE	入力データセットの同一データ表現を含むコピーとなります。ただし、OVERRIDE=オプションリストに OUTREP=値を指定すると、データ表現が変更され、そのエンコーディングが、現在のセッションのデータ表現およびロケールとの互換性があるエンコーディングに変更されます。エンコーディングは、OVERRIDE=オプションリストに指定されている場合にも変更されます。
NOCLONE	動作環境のデータ表現を含むコピー、または指定されている場合は OUT=ライブラリの LIBNAME ステートメントの OUTREP=オプションの値となります。
いずれも使用しない	デフォルトは CLONE となります。

データ表現とは、特定の動作環境でデータを保存する形式です。さまざまな動作環境では、次のような目的のさまざまな規格または規則が使用されます。

- 浮動小数点数を保存(IEEE、IBM 390 など)
- 文字エンコーディング(ASCII または EBCDIC)
- メモリのバイトオーダリング(ビッグエンディアンまたはリトルエンディアン)
- ワード配置(4 バイト境界または 8 バイト境界)
- データ型の長さ(16 ビット、32 ビット、または 64 ビット)

ネイティブなデータ表現とは、ファイルのデータ表現が CPU 動作環境と同じ場合です。

次のテーブルに、ENCODING=属性に関する COPY ステートメントの動作を要約します。

表 13.5　CLONE とエンコーディング属性

オプション	COPY ステートメント
CLONE	入力データセットのエンコーディングを使用するコピー、または指定されている場合は、入力ライブラリの LIBNAME ステートメントの INENCODING=オプションの値となります。ただし、OVERRIDE=オプションリストで ENCODING=値を指定すると、コピーで指定エンコードが使用されることになります。
NOCLONE	現在のセッションエンコーディングのエンコーディングを使用するコピー、または指定されている場合は、出力ライブラリの LIBNAME ステートメントの OUTENCODING=オプションの値となります。

オプション	COPY ステートメント
いずれも使用しない	デフォルトは CLONE となります。

コンピューターによって保存、送信、処理されるテキスト(文字)データはすべてエンコーディングされます。エンコーディングは各文字を一意の数値表現にマップします。エンコーディングは、文字セットとエンコーディング方法の組み合わせです。文字セットは言語または言語グループによって 使用される文字と記号のレパートリーです。エンコーディング方法は、数字をエンコーディングで使用される文字セットに割り当てるために使用される一連のルールです。

次のテーブルに、POINTOBS=属性に関する COPY ステートメントの動作を要約します。POINTOBS=を使用するには、出力データセットを圧縮する必要があります。

表 13.6 CLONE と POINTOBS=属性

オプション	COPY ステートメント
CLONE	入力データセットの POINTOBS=値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで POINTOBS=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	LIBNAME ステートメントは、出力データセットが圧縮され、POINTOBS=オプションが指定されて出力エンジンでサポートされている場合に使用します。LIBNAME ステートメントが指定されず、データセットが圧縮されている場合、出力エンジンでサポートされているときデフォルトは POINTOBS=YES となります。
いずれも使用しない	デフォルトは CLONE となります。

CONSTRAINT=YES | NO
データセットのコピー時にすべての一貫性制約をコピーするかどうかを指定します。

デフォルト NO

ヒント 外部キーを持つ一貫性制約を含むデータセットの場合、COPY ステートメントは CONSTRAINT=YES が指定され、ライブラリ全体がコピーされる場合に一般制約と参照制約をコピーします。SELECT ステートメントまたは EXCLUDE ステートメントを使用してデータセットをコピーする場合、参照一貫性制約はコピーされません。

DATECOPY
結果として生じるファイルのコピーに、SAS ファイルの作成時および最終変更時の SAS 内部日時をコピーします。動作環境の日時は保持されません。

制限事項 DATECOPY は、暗号化されたファイルまたはカタログと使用できません。

DATECOPY は、結果の SAS ファイルで V8 エンジンまたは V9 エンジンが使用される場合にのみ使用できます。

ヒント MODIFY ステートメントの DTC=オプションを使用して、ファイル作成日時を変更できます。MODIFY ステートメント (441 ページ)を参照してください。

コピー中のファイルに追加処理が必要な属性が含まれている場合、最終変更日は現在の日付に変更されます。たとえば、インデックスを含むデータセットをコピーし、インデックスを再度作成する必要がある場合、最終変更日は現在の日付に変更されます。追加処理が必要で、最終変更日に影響する可能性のあるその他の属性には、一貫性制約とソートインジケーターが含まれます。

ENCRYPTKEY= *key-value*
IN=ライブラリ内で AES 暗号化を含むデータセットのコピーに必要なキー値を指定します。

注 出力ライブラリで AES 暗号化がサポートされておらず、入力データセットが AES 暗号化されている場合、COPY プロセスによってエラーが生成されます。

参照項目: “参照一貫性制約を含む AES 暗号化データファイルのコピー ” (421 ページ)

FORCE
MOVE オプションを監査証跡が存在する SAS データセットに使用できます。

注 AUDIT ファイルは、監査データセットと移動されません。

IN=*libref-2*
コピーする SAS ファイルを含む SAS ライブラリを指定します。

別名 INLIB=

INDD=

デフォルト プロシジャ入力ライブラリのライブラリ参照名

操作 選択したメンバーのみコピーするには、SELECT ステートメントまたは EXCLUDE ステートメントを使用します。

INDEX=YES | NO
データセットを SAS ライブラリ間でコピーする場合にデータセットのすべてのインデックスをコピーするかどうかを指定します。

デフォルト YES

MEMTYPE=(*member-type(s)*)
処理を 1 つ以上のメンバーの種類に制限します。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメントの MEMTYPE=を省略すると、デフォルトは MEMTYPE=ALL となります。

注 PROC COPY がテープ上の SAS ライブラリを処理し、MEMTYPE=オプションが指定されていない場合、ファイルの最後までエントリの順次ライブラリ全体がスキャンされます。順次ライブラリがマルチボリュームテープである場合、すべてのテープボリュームはマウントされます。この動作は、単一のボリュームテープライブラリの場合も同様です。

参照項目: “SAS ファイルのコピー/移動時のメンバーの種類の指定” (417 ページ)と “メンバーの種類” (466 ページ)

例 “例 2: SAS ファイルの操作” (494 ページ)

MOVE
SAS ファイルを入力データライブラリ(IN=オプションによって指定)から出力データライブラリ(OUT=オプションによって指定)に移動します。そして、元のファイルを入力データライブラリから削除します。

制限事項 MOVE オプションは、IN=エンジンでテーブルの削除がサポートされている場合にのみ、SAS ライブラリのメンバーの削除に使用できます。テープ形式エンジンでは、テーブルの削除はサポートされていません。テープ形式エンジンを使用する場合、MOVE 操作は実行されず、警告が発行されます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

例 “例 2: SAS ファイルの操作” (494 ページ)

ALTER=*alter-password*
ライブラリ間で移動中の変更保護されている SAS ファイルに対する変更パスワードを与えます。MOVE オプションは元のデータライブラリから SAS ファイルを削除するため、SAS ファイルの移動には変更アクセス権限が必要となります。

OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2> ...*)
入力データセットからコピーされた指定出力データセットオプションを上書きします。COPY の出力データセットコンテキストでは、一部のデータセットオプションは適切でない場合があります。

制限事項 NOCLONE オプションが指定される場合、OVERRIDE オプションは無視されます。ただし、NOCLONE オプションによって調整される属性以外のデータセット属性の変更には使用できます。

操作 OVERRIDE=オプションで OUTREP=値を指定すると、デフォルトのエンコーディングは、OUTREP=値と現在のセッションのロケールで表される動作環境に基づいています。UTF-8 などのデフォルト以外のエンコーディングを割り当てるには、OVERRIDE=オプションで ENCODING=値も指定する必

要があります。ロケールとエンコーディングの詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

ヒント デフォルト(または CLONE)の動作は、入力データセットからのデータ表現やエンコーディングを含む多くの属性を保持することです。代わりに、プロシジャを実行しているセッションのデータ表現とエンコーディングが必要な場合は、PROC COPY または COPY ステートメントで OVERRIDE=(OUTREP=SESSION ENCODING=SESSION)を指定します。入力データセットの属性を保持したくない場合は、代わりに NOCLONE を指定してください。

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

"ATTRIB ステートメント" (*SAS DATA ステップステートメント: リファレンス*).

EXCLUDE ステートメント

SAS ファイルをコピーから除外します。

制限事項: EXCLUDE ステートメントは、COPY ステートメントの後に指定する必要があります
EXCLUDE ステートメントは、SELECT ステートメントで同じ COPY ステップに記述できません

例: "例 2: SAS ファイルの操作" (494 ページ)

構文

EXCLUDE *SAS-file(s)* </ MEMTYPE=*member-type*>;

必須引数

SAS-file(s)
コピー操作から除外する SAS ファイルを 1 つ以上指定します。EXCLUDE ステートメントで指定するすべての SAS ファイルが、COPY ステートメントの IN=オプションで指定されるライブラリ内にある必要があります。SAS ファイルが世代グループである場合、EXCLUDE ステートメントにより基本バージョンの選択のみ可能になります。

次のショートカットを使用して、EXCLUDE ステートメントに複数の SAS ファイルをリストできます。

表 13.7 EXCLUDE ステートメントの使用

表記	意味
x1–xn	ファイル X1 から Xn を指定します。番号は連続している必要があります。
x:	文字 X で始まるすべてのファイルを指定します。

オプション引数

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

別名　MTYPE=

MT=

デフォルト　PROC DATASETS ステートメント、COPY ステートメント、または EXCLUDE ステートメントで MEMTYPE=を指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目:　“SAS ファイルのコピー/移動時のメンバーの種類の指定” (417 ページ) および“処理するメンバーの種類の制限” (465 ページ)

詳細

複数の同じ名前のファイルを除外

EXCLUDE ステートメントで複数の SAS ファイルをリストするためのショートカットを使用できます。

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント　変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。“出力形式の取り消し” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント　FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、“PUT

ステートメント: リスト" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=' '>; | *variable-1* = <...*variable-n*= >;

引数

variable

ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string

最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項

ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: “文字定数と文字変数” (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

SELECT ステートメント

コピーする SAS ファイルを選択します。

制限事項: SELECT ステートメントは、COPY ステートメントの後に指定する必要があります
SELECT ステートメントは、EXCLUDE ステートメントで同じ COPY ステップに記述できません

例: “例 2: SAS ファイルの操作” (494 ページ)

構文

SELECT *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*> <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

必須引数

SAS-file(s)
コピーする SAS ファイルを 1 つ以上指定します。指定するすべての SAS ファイルが、COPY ステートメントの IN=オプションで指定されるライブラリ参照名によって参照されるデータライブラリ内にある必要があります。SAS ファイルに世代グループが含まれる場合、SELECT ステートメントでは特定のバージョンを選択できないため、すべての世代がコピーされます。

オプション引数

ALTER=*alter-password*
ライブラリ間で移動中の変更保護されている SAS ファイルに対する変更パスワードを与えます。SAS ファイルを SAS ライブラリから移動し削除しているため、変更アクセス権限が必要になります。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

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

参照項目: “AES 暗号化データセットの追加” (389 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメント、COPY ステートメント、または SELECT ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目: “SAS ファイルのコピー/移動時のメンバーの種類の指定” (417 ページ) および“処理するメンバーの種類の制限” (465 ページ)

例 “例 2: SAS ファイルの操作” (494 ページ)

詳細

複数の同じ名前のファイルの選択

SELECT ステートメントで複数の SAS ファイルをリストするためのショートカットを使用できます。

表 13.8 SELECT ステートメントの使用

表記	意味
x1–xn	ファイル X1 から Xn を指定します。番号は連続している必要があります。
x:	文字 X で始まるすべてのファイルを 指定します。

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

使用法: COPY プロシジャ

PROC COPY の CAS 処理

IN=オプションと OUT=オプションの両方が CAS エンジンライブラリを参照し、両方のライブラリが同じ CAS セッションを使用する場合、COPY プロシジャは CAS アクションを使用してサーバー内のテーブルをコピーできます。

PROC COPY オプションは、以下を除く CAS LIBNAME エンジンを使用する場合に有効です。

- ENCRYPTKEY=

- OVERRIDE=
- PW=

CAS サーバー上でコピーが発生すると、(VALIDMEMNAME や VALIDVARNAME などの)MVA セッションシステムオプションは使用されません。

SAS Cloud Analysis Services (CAS)は、SAS Viya の分析サーバーおよび関連するクラウドサービスです。CAS LIBNAME エンジンは、CAS セッション名または CAS セッション UUID を使用して、計算サーバーセッションを既存の SAS Cloud Analytic Services セッションに接続できます。ライブラリ参照名は、SAS から特定のセッションと通信するためのハンドルになります。次の例では、CAS 処理で PROC COPY を使用する方法を示します。

次に、CAS で PROC COPY を実行する方法の例を示します。

```
/* Connect to a CAS server */
cas casauto host="cloud.example.com" port=5570;

/* Specify the LIBNAME statements. */
libname public cas caslib=public;
libname mycas cas caslib=casuser;


/* Loading data to Casuser. Note that you can specify OUTCASLIB=
on the PROC CASUTIL statement and it will apply to all statements
that follow in the procedure step (up to the quit). */
proc casutil;
load data=sashelp.class outcaslib=casuser;
load data=sashelp.class compress casout="class_comp" outcaslib=casuser;
load data=sashelp.class promote casout="class_prom" outcaslib=casuser;
quit;

/* Execute the PROC COPY statement. */
title 'Copy with CLONE;
proc copy in=casuser out=public;
run;
```

CAS LIBNAME ステートメントの使用方法については、*SAS Cloud Analytic Services: ユーザーガイド*を参照してください。

CAS テーブルでの CLONE | NOCLONE の使用

CLONE | NOCLONE オプションは、データセット属性をコピーするかどうかを指定します。CAS Engine で使用できる唯一の属性は COMPRESS です。

属性は、データセットオプション、システムオプション、または LIBNAME ステートメントオプションで指定されます。CAS Engine でサポートされるのは COMPRESS=YES | NO オプションのみです。その他の属性は CAS Engine ではサポートされません。

次のテーブルに、COPY ステートメントの動作を要約します。

表 13.9　CLONE と属性の相互作用

属性	終了	CLONE/ NOCLONE	説明
BUFSIZE= CAS Engine ではサポートされません。	SAS データセットから CAS テーブルへ		
	CAS テーブルから SAS データセットへ	CLONE	OVERRIDE=(BUFSIZE=*デフォルト以外の値*)。
		NOCLONE	BUFSIZE=システムオプションの設定を使用します。
	CAS テーブルから CAS テーブルへ		
COMPRESS= SAS データセット - COMPRESS=BINARY | CHAR | NO CAS テーブル - COMPRESS=NO | YES	SAS データセットから CAS テーブルへ	CLONE	OVERRIDE=が使用されない限り、圧縮された SAS データセットは圧縮された CAS テーブルになります。OVERRIDE=が使用されない限り、圧縮されていない SAS データセットは圧縮されていない CAS テーブルになります。
		NOCLONE	CAS LIBNAME の設定に従います。
	CAS テーブルから SAS データセットへ	CLONE	OVERRIDE=が使用されない限り、圧縮された CAS テーブルは SAS データセット CHAR になります。
		NOCLONE	COMPRESS=システムオプションまたは LIBNAME オプションの値が使用されます。
	CAS テーブルから CAS テーブルへ	CLONE	現在の設定を保持します。
		NOCLONE	OUT=ライブラリ参照名に対する CAS LIBNAME 設定を使用して設定します。

属性	終了	CLONE/ NOCLONE	説明
REUSE= CAS Engine ではサポートされません。	SAS データセットから CAS テーブルへ		
	CAS テーブルから SAS データセットへ	CLONE	OVERRIDE=または REUSE=YES システムオプションが使用されない限り、REUSE=NO です。
		NOCLONE	REUSE=システムオプション値を使用します。
	CAS テーブルから CAS テーブルへ		
POINTOBS= CAS Engine ではサポートされません。	SAS データセットから CAS テーブルへ		
	CAS テーブルから SAS データセットへ	CLONE	OVERRIDE=が使用されない限り、POINTOBS=NO です。
		NOCLONE	CAS テーブルが圧縮され、LIBNAME ステートメントで POINTOBS=NO が設定されている場合は、POINTOBS=NO です。 CAS テーブルが圧縮され、LIBNAME オプションがない場合は、POINTOBS=YES です。
	CAS テーブルから CAS テーブルへ		
OUTREP= CAS Engine ではサポートされません。	SAS データセットから CAS テーブルへ	CLONE	必要に応じて LINUX_86_64 に変換します。(OVERRIDE=オプションが使用されている場合は、警告がログに送信されます。)

属性	終了	CLONE/ NOCLONE	説明
		NOCLONE	必要に応じて LINUX_86_64 に変換します。
	CAS テーブルから SAS データセットへ		
		CLONE	データ表現を保持します。(OVERRIDE=オプションが使用されている場合は、警告がログに送信されます。)
	CAS テーブルから CAS テーブルへ		
ENCODING= CAS Engine では UTF-8 のみがサポートされています。 エンコーディングの変更は、CAS Engine では実装されていません。	SAS データセットから CAS テーブルへ	CLONE	必要に応じて UTF-8 に変換します。(OVERRIDE=オプションが使用されている場合は、警告がログに送信されます。)
		NOCLONE	必要に応じて UTF-8 に変換します。
	CAS テーブルから SAS データセットへ	CLONE	OVERRIDE=が使用されない限り、UTF-8 エンコーディングが保持されます。
		NOCLONE	LIBNAME が使用される出力データセットで OUTENCODING=が使用されない限り、UTF-8 エンコーディングが保持されます。
	CAS テーブルから CAS テーブルへ		

出力 CAS テーブルの圧縮

以前に圧縮されたテーブルをコピーする場合、次の処理が行われます。

- SAS データセットが圧縮されている場合、CAS テーブルで COMPRESS=YES 値が保持されます。
- CAS テーブルは、圧縮されている場合、COMPRESS=CHAR 値を持つ SAS データセットに変換されます。
- CAS テーブルを別の CAS テーブルにコピーすると、COMPRESS=YES 属性が維持されます。

COPY プロシジャの使用

ファイルの大規模ディレクトリからの選択ファイルのコピー

COPY プロシジャを使用すると、ライブラリのインメモリディレクトリが得られます。ライブラリに数千ものメンバーが含まれ、コピーされるのはその中のいくつかのメンバーのみである場合、パフォーマンスの問題が生じることがあります。こうしたパフォーマンスの問題を解決するには、COPY ステートメントの MEMTYPE=オプションと SELECT ステートメントの組み合わせを使用します。次にこのプロセスの例を示します。

```
proc copy in=work out=mylib memtype=(data catalog);
  select mydata x1-x10 data2;
run;
```

注: MEMTYPE=ALL またはワイルドカード指定(":")のいずれかを使用する場合、パフォーマンスコードは使用できません。

ヒント　MSGLEVEL=I オプションを設定し、SELECT パフォーマンスコードを使用できる場合、次のメッセージが SAS ログに送信されます。

```
INFO: COPY with SELECT performance is in use.
```

ホスト間での SAS データセットの移送

XPORT エンジンおよび XML エンジンとともに使用すると、COPY プロシジャで、ホスト間で移動可能な移送ファイルを作成し、読み込むことができます。PROC COPY では、SAS データセットでのみ移送ファイルを作成できます。カタログや他の種類の SAS ファイルでは作成できません。

移送は次の 3 段階のプロセスです。

1 PROC COPY を使用して 1 つ以上の SAS データセットを、トランスポート (XPORT)エンジンまたは XML エンジンで作成されたファイルにコピーします。このファイルは移送ファイルと呼ばれ、常に順次ファイルになります。

2 ファイルを作成したら、FTP などの通信ソフトやテープを介して、別の動作環境にそれを移動することもできます。通信ソフトを使用する場合には、変換を防ぐためにバイナリー形式で移動してください。ファイルをメインフレームに移動する場合は、そのファイルに特定の属性が必要です。詳細については、動作環境に応じた SAS ドキュメントと、SAS テクニカルサポートのウェブページを参照してください。

3 受信ホストにファイルが正常に移動したら、PROC COPY を使用して移送ファイルのデータセットを SAS ライブラリにコピーします。

例については、“例 1: ホスト間での SAS データセットのコピー” (329 ページ)を参照してください。

ファイルの移送の詳細については、*SAS ファイルの移動とアクセス*を参照してください。

CPORT および CIMPORT プロシジャを使用しても、SAS ファイルを移送できます。詳細については、14 章, “CPORT プロシジャ,” (335 ページ)および 10 章, “CIMPORT プロシジャ,” (173 ページ)を参照してください。

SAS ライブラリを前のリリースの SAS から移行する必要がある場合は、http://support.sas.com/migration で移行フォーカスエリアを参照してください。

詳細については、PROC DATASETS の CONTENT ステートメントの詳細 (415 ページ) セクションを参照してください。

AES で暗号化されたデータファイルのコピー

AES で暗号化されたデータファイルをコピーするときは、ENCRYPTKEY=データセットオプションを使用する必要があります。AES 暗号化をサポートしていないライブラリに、AES で暗号化されたファイルをコピーするとエラーが発生します。AES 暗号化の詳細については、“AES 暗号化” (*SAS プログラマガイド: エッセンシャル*)および“AES で暗号化されたデータファイルのコピー” (421 ページ)を参照してください。

次に、ENCRYPTKEY=データセットオプションを使用する例を示します。

```
proc copy in=Lib1 out=Lib2;
select My-Data1 (encryptkey=key-value1) <My-Data2 (encryptkey=key-value2) ...>;
```

```
run;
```

参照一貫性制約を含む、AES で暗号化されたデータファイルをコピーするには、“参照一貫性制約を含む AES 暗号化データファイルのコピー ” (421 ページ)を参照してください。

出力データファイルの圧縮

COPY プロシジャではデータセットオプションはサポートされません。したがって、PROC COPY かまたは PROC DATASETS の COPY ステートメントで COMPRESS=データセットオプションを使用することはできません。PROC COPY で生成された OUTPUT データセットを圧縮するには、COMPRESS=YES システムオプションを設定してから、NOCLONE オプションを設定して PROC COPY ステートメントを使用できます。

```
options compress=yes;
proc copy in=work out=new noclone;
select x;
run;
```

例: COPY プロシジャ

例 1: ホスト間での SAS データセットのコピー

要素:

- PROC COPY ステートメントオプション
  - IN=
  - MEMTYPE=
  - OUT=
  - SELECT ステートメント
- XPORT エンジン

詳細

この例は、ホストで移送ファイルを作成し、別のホストでそれを読み取る方法が示されています。

この例が正しく動作するには、使用している動作環境に対応する SAS ドキュメントに記載されているように、移送ファイルが特定の特性を持っている必要があります。また、移送ファイルは受信動作環境にバイナリー形式で移動する必要があります。

プログラム

```
libname source 'SAS-library-on-sending-host';

libname xptout xport 'filename-on-sending-host';

proc copy in=source out=xptout memtype=data;
  select bonus budget salary;
run;


libname insource xport 'filename-on-receiving-host';

proc copy in=insource out=work;
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 SOURCE などのライブラリ参照名を、移送する SAS データセットを含む SAS ライブラリに割り当てます。また、ライブラリ参照名を移送ファイルに割り当て、XPORT キーワードを使用して XPORT エンジンを指定します。

```
libname source 'SAS-library-on-sending-host';

libname xptout xport 'filename-on-sending-host';
```

SAS データセットを移送ファイルにコピーします。

```
proc copy in=source out=xptout memtype=data;
  select bonus budget salary;
run;
```

プロシジャで移送ファイルからのデータを読み取れるようにします。 LIBNAME ステートメントの XPORT エンジンによって、プロシジャは移送ファイルからデータを読み取れるようになります。

```
libname insource xport 'filename-on-receiving-host';
```

SAS データセットを受信ホストにコピーします。 ファイルをコピーしたら、PROC COPY を使用して SAS データセットを受信ホストの作業データライブラリにコピーします。

```
proc copy in=insource out=work;
run;
```

ログの例

例のコード 13.1 ソースライブラリログ

```
1    LIBNAME source 'SAS-library-on-sending-host ';
NOTE: Libref SOURCE was successfully assigned as follows:
      Engine:       V9
      Physical Name: SAS-library-on-sending-host
2    LIBNAME xptout xport 'filename-on-sending-host';
NOTE: Libref XPTOUT was successfully assigned as follows:
      Engine:       XPORT
      Physical Name: filename-on-sending-host
3    proc copy in=source out=xptout memtype=data;
4    select bonus budget salary;
5    run;

NOTE: Copying SOURCE.BONUS to XPTOUT.BONUS (memtype=DATA).
NOTE: The data set XPTOUT.BONUS has 1 observations and 3 variables.
NOTE: Copying SOURCE.BUDGET to XPTOUT.BUDGET (memtype=DATA).
NOTE: The data set XPTOUT.BUDGET has 1 observations and 3 variables.
NOTE: Copying SOURCE.SALARY to XPTOUT.SALARY (memtype=DATA).
NOTE: The data set XPTOUT.SALARY has 1 observations
```

例のコード 13.2 インソースライブラリログ

```
1     LIBNAME insource xport 'filename-on-receiving-host';
NOTE: Libref INSOURCE was successfully assigned as follows:
      Engine:      XPORT
      Physical Name: filename-on-receiving-host
2     proc copy in=insource out=work;
3     run;
NOTE: Input library INSOURCE is sequential.
NOTE: Copying INSOURCE.BUDGET to WORK.BUDGET (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set WORK.BUDGET has 1 observations and 3 variables.
NOTE: Copying INSOURCE.BONUS to WORK.BONUS (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set WORK.BONUS has 1 observations and 3 variables.
NOTE: Copying INSOURCE.SALARY to WORK.SALARY (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set WORK.SALARY has 1 observations and 3 variables.
```

例 2: Converting SAS Data Sets Encodings

要素:

PROC COPY statement options
- IN=
- NOCLONE
- OUT=
- SELECT ステートメント

CVP engine

詳細

この例は、エンコーディングの種類を別の種類に変換する方法を示しています。この例が正しく動作するには、２つのエンコーティングに互換性がある必要があります。詳細については、*“互換性があるエンコーディングと互換性がないエンコーディング” (SAS 各国語サポート(NLS): リファレンスガイド)*を参照してください。

プログラム

ライブラリ参照を割り当てます。 ２つのエンコーディングに互換性がある必要があります。

```
LIBNAME inlib cvp 'SAS-library';
LIBNAME outlib 'SAS-library' outencoding="encoding value for output";
proc copy noclone in=inlib out=outlib;
  select car;
run;
```

ログの例

例のコード 13.3　InLib ライブラリログ

```
1    LIBNAME inlib cvp  'SAS-library';
NOTE: Libref INLIB was successfully assigned as follows:
      Engine:       V9
      Physical Name: SAS-library
2    LIBNAME outlib 'SAS-library' outencoding="encoding value for output";
NOTE: Libref OUTLIB was successfully assigned as follows:
      Engine:       V9
      Physical Name: SAS-library
3    proc copy noclone in=inlib out=outlib;
4    select car;
5    run;

NOTE: Copying INLIB.CAR to OUTLIB.CAR (memtype=DATA).
NOTE: System Options for BUFSIZE and REUSE were used at user's request.

NOTE: Libname and/or system options for compress, pointobs, data representation and encoding attributes
were used at user's request.
NOTE: Data file OUTLIB.CAR.DATA is in a format that is native to another host, or the file encoding does not
match the session encoding.Cross Environment Data Access will be used, which might require additional
CPU resources and might reduce performance.

NOTE: There were 25 observations read from the data set INLIB.CAR.

NOTE: The data set OUTLIB.CAR has 25 observations and 2 variables.
```

例 3: SAS データセットの CAS テーブルへのコピー

要素:

PROC COPY ステートメントオプション
- IN=
- OUT=
- SELECT ステートメント

詳細

この例では、SAS データセットを CAS テーブルにコピーする方法を示します。

プログラム

```
libname mycas cas;
libname mylib 'BASE-engine-library';
```

```
proc copy in=mylib out=mycas;
  select monthly;
run;
quit;
```

プログラムの説明

ライブラリ参照を割り当てます。 CAS テーブルにコピーするデータセットを選択します。

```
libname mycas cas;
libname mylib 'BASE-engine-library';
```

PROC COPY および SELECT ステートメントを使用します。 SAS データセットを CAS テーブルにコピーします。

```
proc copy in=mylib out=mycas;
  select monthly;
run;
quit;
```

ログの例

例のコード 13.4　MyLib ライブラリログ

```
57       libname mycas cas;
NOTE: Libref MYCAS was successfully assigned as follows:
      Engine:       CAS
      Physical Name: 1f436ced
58       libname mylib 'BASE-engine-library';
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:       V9
      Physical Name: BASE-engine-library
59
60       proc copy in=mylib out=mycas;
61         select monthly;
62       run;

NOTE: Copying MYLIB.MONTHLY to MYCAS.MONTHLY (memtype=DATA).
NOTE: There were 12012 observations read from the data set MYLIB.MONTHLY.
NOTE: The data set MYCAS.MONTHLY has 12012 observations and 7 variables.
NOTE: PROCEDURE COPY used (Total process time):
      real time          0.06 seconds
      cpu time           0.02 seconds
```

14

CPORT プロシジャ

概要: CPORT プロシジャ

CPORT プロシジャの機能

CPORT プロシジャは、SAS データセット、SAS カタログ、または SAS ライブラリを順編成ファイル出力形式(移送ファイル)に書き出します。PROC CPORT を CIMPORT プロシジャと一緒に使うと、ある環境から他の環境にファイルを移動できます。*移送ファイル*は SAS ライブラリ、SAS カタログ、または SAS データセットを移送出力形式で保持している順編成ファイルです。PROC CPORT が書き出す移送

出力形式は、すべての環境およびすべての SAS リリースを通して同じです。PROC CPORT では、*エクスポート*は SAS ライブラリ、SAS カタログまたは SAS データセットを移送出力形式にすることを意味します。PROC CPORT はカタログやデータセットを単一または SAS ライブラリをしてエクスポートできます。PROC CIMPORT は、移送ファイルを SAS カタログ、SAS データセットまたは SAS ライブラリの元の形式に戻します(*imports*)。

PROC CPORT は、SAS ファイルの*変換*も行います。つまり、SAS ファイルの出力形式を SAS のバージョン間で適切な出力形式に変更します。たとえば、PROC CPORT と PROC CIMPORT を使って、前のリリースの SAS からもっと新しいリリースの SAS にファイルを移動できます。PROC CIMPORT は自動で移送ファイルを変換して、それをインポートします。

注: PROC CIMPORT と PROC CPORT を使用して、グラフィックカタログをバックアップできます。PROC COPY はグラフィックカタログのバックアップには使用できません。

PROC CPORT は出力を作成しませんが(移送ファイル以外)、SAS ログに NOTES を書き出します。

移送ファイルの作成処理と読み込み処理

次に、ソースコンピューターで移送ファイルを作成して、それをターゲットコンピューターで読み込むプロセスを示します。

1. 移送ファイルは、ソースコンピューターで PROC CPORT を使って作成されます。
2. 移送ファイルは、ソースコンピューターから移送されます。
3. 移送ファイルは、ターゲットコンピューターで PROC CIMPORT を使って読み込まれます。

   注: 移送ファイルは、PROC CPORT を使って作成されたものと、XPORT エンジンを使って作成されたものとで、互換性はありません。

移送ファイルの作成(PROC CPORT)、移送ファイルの転送、移送ファイルの復元(PROC CIMPORT)のステップの詳細については、*SAS ファイルの移動とアクセス*を参照してください。

構文: CPORT プロシジャ

ヒント: グラフィックカタログをバックアップするには、PROC CIMPORT または PROC CPORT を使用してください。PROC COPY はグラフィックカタログのバックアップには使用できません。

参照項目: 移送ファイルの作成(PROC CPORT)、移送ファイルの転送、移送ファイルの復元(PROC CIMPORT)のステップの詳細については、*SAS ファイルの移動とアクセス*を参照してください。

PROC CPORT *source-type=libref | <libref.>member-name*<*options*>;

EXCLUDE*SAS file(s) | catalog entry(s)*< / MEMTYPE=*mtype*>
< / ENTRYTYPE=*entry-type*>;

SELECT*SAS file(s) | catalog entry(s)*< / MEMTYPE=*mtype*>
< / ENTRYTYPE=*entry-type*> ;

TRANTABNAME=*translation-table-name*
<*options*>;

ステートメント	タスク	例
“PROC CPORT ステートメント”	移送ファイルを作成します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
“EXCLUDE ステートメント”	移送ファイルから 1 つ以上の指定ファイルを除外します	
“SELECT ステートメント”	移送ファイルに含める 1 つ以上のファイルやエントリを指定します	Ex. 2
“TRANTAB ステートメント”	エクスポートするカタログエントリの文字用の変換テーブル(複数可)を指定します。	Ex. 4

PROC CPORT ステートメント

移送ファイルを作成します。

例:
“例 1: 複数のカタログのエクスポート” (352 ページ)
“例 2: 個々のカタログエントリのエクスポート” (353 ページ)
“例 3: 単一 SAS データセットのエクスポート” (354 ページ)
“例 4: 変換テーブルの適用” (355 ページ)
“例 5: 変更日に基づくエントリのエクスポート” (357 ページ)

構文

PROC CPORT *source-type=libref | <libref.>member-name*<*options*>;

オプション引数の要約

NOEDIT

インポートした場合は編集不可で SAS/AF PROGRAM と SCL エントリをエクスポートします。

NOSRC
エクスポートされたカタログエントリはコンパイルされた SCL コードは含むがソースコードは含まないと指定します。

OUTLIB=*libref*
SAS ライブラリに関連付けされているライブラリ参照名を指定します。

Control the contents of the transport file

ASIS
表示される文字データの移送形式への変換を行いません。

CONSTRAINT=YES | NO
整合性の制約のエクスポートをコントロールします。

DATECOPY
作成日時と変更日時を移送ファイルにコピーします。

INDEX=YES | NO
インデックスとインデックス付きの SAS データセットのエクスポートをコントロールします。

INTYPE=IBM | HITAC | FACOM | DEC | SJIS | PCIBM
エクスポート対象の SAS ファイルに保存される DBCS データの種類を指定します。

NOCOMPRESS
移送ファイルでバイナリゼロとブランクの圧縮を抑制します。

OUTTYPE=UPCASE
すべてのアルファベット文字を大文字で移送ファイルに書き込みます。

TRANSLATE=(*translation-list*)
特定の文字を ASCII の一種または EBCDIC 値から他種へ変換します。

Identify the transport file

FILE=*fileref* | '*filename*'
書き込む移送ファイルを指定します。

TAPE
PROC CPORT の出力をテープに指定します。

Select files to export

AFTER=*date*
更新日が指定した日付以降の全データセットまたはカタログエントリのコピーをエクスポートします。

EET=(*etype(s)*)
移送ファイルから特定のエントリの種類を除外します。

ET=(*etype(s)*)
移送ファイルに特定のエントリの種類を含めます。

GENERATION=YES | NO
データセットのすべての世代をエクスポートするかどうか指定します。

MEMTYPE=ALL | CATALOG | DATA
ライブラリのエクスポート時にデータセットのみ、カタログのみ、または両方を移動させるかを指定します。

必須引数

source-type=libref | <libref.>member-name
エクスポートするファイルの種類を特定し、エクスポートするカタログ、SAS データセット、または SAS ライブラリを指定します。

source-type
単一カタログ、単一 SAS データセット、または SAS ライブラリのメンバーとしてエクスポートする 1 つ以上のファイルを特定します。*source-type* 引数は、次のうちのどれかになります。

CATALOG=
ファイルの種類が SAS カタログであることを指定します。

別名 C=

CAT=

DATA=
ファイルの種類が SAS データセットであることを指定します。

別名 D=

DS=

LIBRARY=
ファイルの種類は SAS ライブラリと指定します。

別名 L=

LIB=

注 パスワード保護がかかっているデータセットをソースの種類として指定した場合、移送ファイル作成時にはパスワードも含まなくてはなりません。詳細については、“PROC CPORT ステートメントの READ=データセットオプション ” (348 ページ)を参照してください。

libref
<libref.>member-name
エクスポートする特定のカタログ、SAS データセット、または SAS ライブラリを指定します。*source-type* が CATALOG または DATA の場合、ライブラリ参照名とメンバー名の両方を指定できます。*libref* が省略された場合、PROC CPORT はデフォルトのライブラリ(通常は WORK ライブラリ)を *libref* として使用します。*source-type* 引数が LIBRARY の場合、*libref* のみを指定します。ライブラリを指定した場合、PROC CPORT はライブラリからデータセットとカタログのみをエクスポートします。他の種類のファイルをエクスポートすることはできません。

参照項目: 名前とメンバー名に使用できる命名規則については、“SAS 名” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

オプション引数

AFTER=*date*
更新日が指定した日付以降の全データセットまたはカタログエントリのコピーをエクスポートします。変更日時は、一番最近でデータセットやカタログのコンテンツが変更になった日付です。日付を SAS 日付リテラルまたは SAS 日付値(数値)で指定します。

ヒント CATALOG プロシジャを使うと、カタログエントリの変更日時を特定できます。

例 “例 5: 変更日に基づくエントリのエクスポート” (357 ページ)

ASIS
表示される文字データの移送形式への変換を行いません。このオプションは、DBCS(ダブルバイト文字セット)データを含むファイルを、1 つのオペレーション環境から同じ種類の DBCS データを使う別の環境に移動する時に使います。

操作 ASIS オプションは NOCOMPRESS オプションを起動します。

同じ PROC CPORT ステップで ASIS オプションと OUTTYPE=オプションを同時に使用できません。

CONSTRAINT=YES | NO
データセットに定義されている整合性の制約のエクスポートをコントロールします。CONSTRAINT=YES を指定すると、ライブラリではすべての種類の整合性の制約がエクスポートされ、単一データセットでは一般整合性の制約のみエクスポートされます。CONSTRAINT=NO を指定した場合、一貫性制約なしで作成されたインデックスはポートされますが、一貫性制約も一貫性制約ありで作成されたインデックスもポートされません。一貫性制約の詳細については、“Integrity Constraints” (*SAS V9 LIBNAME Engine: Reference*)を参照してください。

別名 CON=

デフォルト YES

操作 同じ PROC CPORT で CONSTRAINT=と INDEX=の両方を設定することはできません。

INDEX=NO を指定した場合、一貫性制約はエクスポートされません。

DATECOPY
SAS ファイルが最初に作成された時と最後に変更された時のそれぞれの SAS 内部日時を、結果として生じる移送ファイルにコピーします。動作環境の日時は保持されません。

制限事項 DATECOPY は出力先のファイルが V8 または V9 エンジンを使用する場合のみ使用できます。

注 移送中のファイルに追加処理が必要な属性が含まれている場合、最終変更日は現在の日時に変更される可能性があります。

ヒント タイムゾーンオフセットを使用してデータセットを移送するには DATECOPY オプションを指定する必要があります。

ファイル作成日時は、PROC DATASETS ステップの MODIFY ステートメントの DTC=オプションで変更できます。詳細については、“MODIFY ステートメント” (441 ページ)を参照してください。

EET=(*etype(s)*)
移送ファイルから特定のエントリの種類を除外します。*etype* が単一エントリの種類の場合は、かっこは省略できません。複数の値はスペースで区切ります。

操作 同じ PROC CPORT ステップで、EET=オプションと ET=オプションを同時に使用できません。

ET=(*etype(s)*)
移送ファイルに特定のエントリの種類を含めます。*etype* が単一エントリの種類の場合は、かっこは省略できません。複数の値はスペースで区切ります。

操作 同じ PROC CPORT ステップで、EET=オプションと ET=オプションを同時に使用できません。

FILE=*fileref* | '*filename*'
書き込む移送ファイルの事前に定義されているファイル参照名またはファイル名を指定します。FILE=オプションを省略した場合、定義されていれば、PROC CPORT はファイル参照名 SASCAT に書き込みます。ファイル参照名 SASCAT が定義されていない場合、PROC CPORT は現在のディレクトリの SASCAT.DAT に書き込みます。

注 SASCAT が定義されていない場合の PROC CPORT の動作は、動作環境によって違います。詳細については、動作環境に関する SAS のドキュメントを参照してください。

例 すべての例。

GENERATION=YES | NO
SAS データセットのすべての世代をエクスポートするかどうか指定します。データセットの基本生成のみエクスポートするには、GENERATION=NO を PROC CPORT ステートメントで指定します。特定の世代番号をエクスポートするには、PROC CPORT ステートメントでデータセットを指定する際に GENNUM=データセットオプションを使います。詳細については、“Generation Data Sets” (*SAS V9 LIBNAME Engine: Reference*)を参照してください。

別名 GEN=

デフォルト ライブラリは YES で単一データセットは NO

注 PROC CIMPORT は移送ファイルにあるデータセットのすべての世代をインポートします。同じ名前の前の世代セットを削除し、インポートされた世代セットと世代数が同じでなくても置き換えます。

INDEX=YES | NO
インデックスとインデックス付きのデータセットをエクスポートするか指定します。

デフォルト YES

操作　同じ PROC CPORT で CONSTRAINT=と INDEX=の両方を設定することはできません。

INDEX=NO を指定した場合、一貫性制約はエクスポートされません。

INTYPE=IBM | HITAC | FACOM | DEC | SJIS | PCIBM
エクスポート対象の SAS ファイルに保存される DBCS データの種類を指定します。ダブルバイト文字セット(DBCS)データはセットの各文字に最大 2 バイト使用します。*DBCS-type* は次の値のうちの 1 つになります。

IBM
z/OS または VSE の場合

HITAC
z/OS の場合

FACOM
z/OS の場合

DEC
OpenVMS の場合

SJIS
OpenVMS または OS/2 の場合

PCIBM
OS/2 の場合

デフォルト　INTYPE=オプションを使用しない場合、DBCS は SAS システムオプションの DBCSTYPE=の値になります。

制限事項　INTYPE=オプションは、ダブルバイト文字セット(DBCS)拡張が SAS に付随している場合のみ使用できます。これらの拡張は膨大な計算リソースを使用するため、それを必要とするサイト用の特別なディストリビューションがあります。このオプションが DBCS 拡張が有効になっていないサイトで使われた場合、エラーがでます。

操作　INTYPE=オプションを OUTTYPE=オプションと同時に使用して、DBCS データの種類を他の種類に変更します。

INTYPE=オプションは NOCOMPRESS オプションを起動します。

同じ PROC CPORT ステップで INTYPE=オプションと ASIS オプションを同時に使用できません。

ヒント　構成ファイルの SAS システムオプション DBCSTYPE=の値を設定できます。

MEMTYPE=ALL | CATALOG | DATA
PROC CPORT が移送ファイルに書き込む SAS ファイルの種類を限定します。MEMTYPE=は処理を 1 つのメンバーの種類に限定します。*mtype* の値は、次のどれかになります。

ALL
カタログとデータセットの両方

CATALOG
カタログ

別名 CAT

DATA
SAS データセット

別名 DS

別名 MT=

デフォルト ALL

例 “例 1: 複数のカタログのエクスポート” (352 ページ)

NOCOMPRESS
移送ファイルでバイナリゼロとブランクの圧縮を抑制します。

別名 NOCOMP

デフォルト PROC CPORT はスペースを節約するために、デフォルトでバイナリゼロとブランクを圧縮します。

操作 ASIS と INTYPE=と OUTTYPE=オプションは NOCOMPRESS オプションを起動します。

注 移送ファイルを圧縮しても、カタログやデータセットにある元のファイルが圧縮されていたかどうかのフラッグは変更されません。

NOEDIT
インポートした場合は編集不可で SAS/AF PROGRAM と SCL エントリをエクスポートします。

NOEDIT オプションでは、SAS/AF ソフトウェアの BUILD プロシジャで MERGE ステートメントで NOEDIT オプションを使用して SCL コードを含む新規カタログを作成するのと同じ結果が得られます。

別名 NEDIT

注 NOEDIT オプションは SAS/AF PROGRAM と SCL エントリにのみ影響します。FSEDIT SCREEN または FSVIEW FORMULA エントリには影響しません。

NOSRC
エクスポートされたカタログエントリはコンパイルされた SCL コードは含むがソースコードは含まないと指定します。

NOSRC オプションでは、SAS/AF ソフトウェアの BUILD プロシジャで MERGE ステートメントで NOSRC オプションを使用して SCL コードを含む新規カタログを作成するのと同じ結果が得られます。

別名 NSRC

OUTLIB=*libref*
SAS ライブラリに関連付けされているライブラリ参照名を指定します。
OUTLIB= option を指定した場合、PROC CIMPORT は自動的に起動され、指定ライブラリ内の入力ライブラリ、データセット、またはカタログを再作成します。

別名 OUT=

ヒント 元のデータを完全なまま保持したい場合は、OUTLIB=オプションを使用して同じ動作環境内で SAS ファイルの DBCS の種類を別の種類に変更します。

OUTTYPE=UPCASE
すべての表示されている文字を大文字で移送ファイルと OUTLIB=ファイルに書き込みます。

操作 OUTTYPE=オプションは NOCOMPRESS オプションを起動します。

TAPE
PROC CPORT の出力をテープに指定します。

デフォルト PROC CPORT からの出力がディスクに送信されます。

TRANSLATE=(*translation-list*)
特定の文字を ASCII の一種または EBCDIC 値から他種へ変換します。
translation-list の各要素は次の形式になります。

- *ASCII-value-1* TO *ASCII-value-2*
- *EBCDIC-value-1* TO *EBCDIC-value-2*

ASCII の値には 16 進数表現も 10 進数表現も使用できます。16 進数表現を使用する場合、値は数字で始まり、x で終わらなければなりません。16 進数の値がアルファベット文字で始まる場合は、先頭にゼロを追加します。

たとえば、すべての左かっこを左中かっこに変換するには、TRANSLATE=オプションを次のように指定します(ASCII 文字の場合)。

```
translate=(5bx to 7bx)
```

次の例はすべての左かっこを左中かっこに、そしてすべての右かっこを右中かっこに変換します。

```
translate=(5bx to 7bx 5dx to 7dx)
```

EXCLUDE ステートメント

指定ファイルまたはエントリを移送ファイルから除外します。

操作: PROC CPORT ステップでは EXCLUDE ステートメントまたは SELECT ステートメントを使用できますが、両方を同時には使用できません。

ヒント: PROC CPORT 一回の起動で、使用できる EXCLUDE ステートメントの数に制限はありません。

構文

EXCLUDE*SAS file(s)* | *catalog entry(s)*< / MEMTYPE=*mtype*>
< / ENTRYTYPE=*entry-type*>;

必須引数

SAS file(s) | *catalog entry(s)*
移送ファイルから除外する 1 つ以上の SAS ファイルを指定します。SAS ライブラリのエクスポート時は SAS ファイル名を指定し、個別 SAS カタログのエクスポート時はカタログエントリ名を指定します。複数のファイル名またはエントリ名はスペースで区切ります。EXCLUDE ステートメントでは、ショートカットを使って、多数の似ている名前をリストすることができます。詳細については、*"SAS データセット" (SAS プログラマガイド: エッセンシャル)*を参照してください。

移送ファイルから除外する 1 つ以上のカタログエントリを指定します。個々の SAS カタログをエクスポートするときに、カタログエントリ名を指定します。複数のエントリ名はスペースで区切ります。EXCLUDE ステートメントでは、ショートカットを使って、多数の似ている名前をリストすることができます。詳細については、*"SAS データセット" (SAS プログラマガイド: エッセンシャル)*を参照してください。

オプション引数

ENTRYTYPE=*entry-type*
EXCLUDE ステートメントにリストされているカタログエントリには、単一のエントリの種類を指定します。参照: *"Catalog Names and Catalog Entries" (SAS V9 LIBNAME Engine: Reference)*.

別名 ETYPE=

ET=

制限事項 ENTRYTYPE=は個別の SAS カタログをエクスポートする時にのみ有効です。

MEMTYPE=*mtype*
EXCLUDE ステートメントにリストされている 1 つ以上の SAS ファイルに対して、単一のメンバーの種類を指定します。有効な値は CATALOG、DATA、または ALL です。EXCLUDE ステートメントで MEMTYPE=オプションを指定しない場合、処理は PROC CPORT ステートメントの MEMTYPE=オプションで指定したメンバーの種類に限定されます。

ファイル名の後に、MEMTYPE= option をかっこで囲んで指定することもできます。かっこの中で、MEMTYPE=はそのすぐ前にあるファイル名の種類を識別します。オプションのこの形式を使用する場合は、EXCLUDE ステートメントでスラッシュの後に続く MEMTYPE=オプションを無効にします。

別名 MTYPE=

MT=

デフォルト　PROC CPORT ステートメントまたは EXCLUDE ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

制限事項　MEMTYPE=は SAS ライブラリをエクスポートする時にのみ有効です。

PROC CPORT ステートメントの MEMTYPE=でメンバーの種類を指定する場合、EXCLUDE ステートメントの MEMTYPE=で指定するメンバーの種類と一致しなければなりません。

参照項目:　名前とメンバー名に使用できる命名規則については、"SAS 名" (*SAS プログラマガイド: エッセンシャル*)を参照してください。

SELECT ステートメント

指定ファイルまたはエントリを移送ファイルに含めます。

操作:　PROC CPORT ステップでは EXCLUDE ステートメントまたは SELECT ステートメントを使用できますが、両方を同時には使用できません。

ヒント:　PROC CPORT 一回の起動で、使用できる SELECT ステートメントの数に制限はありません。

例:　"例 2: 個々のカタログエントリのエクスポート" (353 ページ)

"例 4: PROC CIMPORT を使用してフランス語のデータセットを UTF-8SAS セッションにインポートする" (200 ページ)

構文

SELECT*SAS file(s)* | *catalog entry(s)*</ MEMTYPE=*mtype*>
< / ENTRYTYPE=*entry-type*> ;

必須引数

SAS file(s) | *catalog entry(s)*
移送ファイルに含む SAS ファイルまたはカタログを指定します。SAS ライブラリのエクスポート時は SAS ファイル名を指定し、個別 SAS カタログのエクスポート時はカタログエントリ名を指定します。複数のファイル名またはエントリ名はスペースで区切ります。SELECT ステートメントでは、ショートカットを使って、多数の似ている名前をリストすることができます。詳細については、"SAS データセット" (*SAS プログラマガイド: エッセンシャル*)を参照してください。

オプション引数

ENTRYTYPE=*entry-type*
SELECT ステートメントにリストされているカタログエントリには、単一のエントリの種類を指定します。参照先カタログエントリの種類の完全リストについ

ては、“Catalog Names and Catalog Entries” (*SAS V9 LIBNAME Engine: Reference*)を参照してください。

別名 ETYPE=

ET=

制限事項 ENTRYTYPE=は個別の SAS カタログをエクスポートする時にのみ有効です。

MEMTYPE=*mtype*
SELECT ステートメントにリストされている 1 つ以上の SAS ファイルに対して、単一のメンバーの種類を指定します。有効な値は CATALOG、DATA、または ALL です。SELECT ステートメントで MEMTYPE=オプションを指定しない場合、処理は PROC CPORT ステートメントの MEMTYPE=オプションで指定したメンバーの種類に限定されます。

メンバーの名前の後に、MEMTYPE= option をかっこで囲んで指定することもできます。かっこの中で、MEMTYPE=はそのすぐ前にあるメンバーの名前の種類を識別します。オプションのこの形式を使用する場合は、SELECT ステートメントでスラッシュの後に続く MEMTYPE=オプションを無効にします。

別名 MTYPE=

MT=

デフォルト PROC CPORT ステートメントまたは SELECT ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

制限事項 MEMTYPE=は SAS ライブラリをエクスポートする時にのみ有効です。

PROC CPORT ステートメントの MEMTYPE=でメンバーの種類を指定する場合、SELECT ステートメントの MEMTYPE=で指定するメンバーの種類と一致しなければなりません。

参照項目: 名前とメンバー名に使用できる命名規則については、“SAS 名” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

TRANTAB ステートメント

エクスポートするカタログエントリの文字に対し翻訳テーブルを指定します。

制限事項: TRANTAB ステートメントは、DBCS または UTF-8 SAS セッションをサポートしていません。

ヒント: 各 TRANTAB ステートメントでは 1 つの変換テーブルしか指定できません。ただし、PROC CPORT の一回の起動で複数の変換テーブルを使用することができます。

参照項目: CPORT プロシジャと、UPLOAD プロシジャおよび DOWNLOAD プロシジャの TRANTAB ステートメントに関しては、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

例: “例 4: 変換テーブルの適用” (355 ページ)

構文

TRANTABNAME=*translation-table-name*<*options*>;

必須引数

NAME=
使用する変換テーブルの名前を指定します。

使用法: CPORT プロシジャ

PROC CPORT ステートメントの READ=データセットオプション

読み取り保護されているデータセットの移送ファイルを作成するには、パスワード(クリアテキストまたはエンコードされているもの)を提供しなくてはなりません。パスワードが含まれていない場合、移送ファイルは作成できません。

パスワード保護付きのデータセットを扱っている場合は、READ=オプションでパスワードを提供できます。READ=オプションで読み取り保護されたデータセットのパスワードを提供しない場合、パスワードの入力を求めるメッセージが表示されます。

読み取り保護のデータセットの移送ファイルを作成するには、READ=データセットオプションを使って、適切なパスワードを提供します。例 1 では、PROC CPORT は SOURCE.GRADES という名前の入力ファイルを コピーし、データセットとパスワード ADMIN を提供し、GRADESOUT という名前の移送ファイルを作成します。

例 1: クリアテキストパスワード:

```
proc cport data=source.grades(read=admin) file=gradesout;
```

例 2 では、エンコードされたパスワードは READ=オプションで指定されます。エンコードされたパスワードは、PWENCODE プロシジャによって 生成されます。詳細については、45 章, “PWENCODE プロシジャ,” (1561 ページ)を参照してください。

例 2: エンコードされたパスワード

```
proc cport
data=source.grades(read={SAS005}ADD8AB7108595A7D1A69190D78CDFE6145C1EB849CC
7A43D)
  file=gradesout;
```

パスワード保護さらたデータセットを参照するケースでパスワードが省略されている場合、 SAS はパスワード入力を促す画面を表示します。有効でないパスワードが入力された場合、ログにエラーメッセージが出力されます。エラーの例は次になります。

```
ERROR: Invalid or missing READ password on member WORK.XYZ.DATA
```

データセットがライブラリの一部として移送される場合や SELECT ステートメントで指定されている場合は、パスワードは必要ありません。ただし、パスワード保護のデータセットが移送されてもターゲット SAS エンジンがパスワードをサポートしていない場合、移送ファイルのインポートはできません。

READ=データセットオプションの詳細については *SAS データセットオプション: リファレンス*を、そしてパスワード保護されたデータセットの詳細については *SAS プログラマガイド: エッセンシャル*を参照してください。

注: PROC CIMPORT は移送先の環境で移送ファイルの復元でパスワードを必要としません。ただし、パスワード保護のデータを扱う他の SAS プロシジャではパスワードが必要になります。

CPORT の問題: トランスポートファイルの作成

移送ファイルとエンコーディングについて

SAS Viya プラットフォームでは、UTF-8 がデフォルトのエンコーディングです。ただし、SAS Viya プラットフォームは他のレガシーエンコーディングもサポートしています。

移送ファイルの文字データは、次の種類のエンコーディングで作成されます。

- 移送ファイルが作成される SAS セッションの UTF-8 エンコーディング。
- PROC CPORT を使用して移送ファイル(ASCII プラットフォームでは US-ASCII でエンコードされる)を作成すると、セッションエンコーディングに関係なく、その移送ファイルに対する US-ASCII エンコーディングが保持されます。その後、PROC CIMPORT を使用して、そのデータセットを ASCII プラットフォームに移送した場合、その移送ファイルに対する US-ASCII エンコーディングが保持され、トランスコードされません。作成されたデータセットには、セッションエンコーディングではなく、US-ASCII エンコーディングが含まれます。たとえば、セッションエンコーディングが WLATIN1 の場合に、PROC CPORT を使用して、US-ASCII のエンコーディングを含むデータセットを作成したとします。移送ファイルでは、WLATIN1 エンコーディングではなく、US-ASCII エンコーディングが保持されます。このデータセットに PROC CIMPORT を使用した場合もこの保持が行われます。PROC CIMPORT を使用してデータセットを ASCII プラットフォームに移送する場合、US-ASCII エンコーディングが保持されトランスコードされません。

注: US-ASCII エンコーディングの保持は、ASCII プラットフォームでのみ行われます。

PROC CIMPORT によって生成される警告については、“SAS Viya プラットフォームで PROC CIMPORT を使用すると生成される警告メッセージ” (190 ページ)を参照してください。

次の例に、エンコーディングの移送ファイルへの適用方法を示します。SAS Studio でエンコーディングを設定する方法については、“Opening a Program” (*SAS Studio: User's Guide*)を参照してください。

表 14.1 移送ファイルへのエンコーディングの割り当て

移送ファイルのエンコーディング値	SAS 起動	説明
utf-8 または us-ascii	エンコーディングは utf8	SAS セッションは、UTF-8 エンコーディングを使用して起動されます。セッションエンコーディングは、移送ファイルに適用されます。データセットが US-ASCII の場合、セッションエンコーディングではなく、US-ASCII エンコーディングが保持されます。
wlatin2	ロケールは pl_PL	SAS セッションは、Polish Poland ロケールに関連付けられているエンコーディング、LATIN2 を使用して起動されます。

各ロケールに関連付けられているエンコーディングの完全なリストについては、ロケールテーブル(*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

移送ファイルをインポートするためには、移送元と移送先の SAS セッションのエンコーディングに互換性がなければなりません。次に、互換性のあるソースおよびターゲット SAS セッションの例を示します。

表 14.2 互換性のあるエンコーディング s

ソース SAS セッション			ターゲット SAS セッション	
ロケール	Linux の SAS セッションエンコーディング	移送ファイルエンコーディング	ロケール	SAS セッションエンコーディング
es_MX (Spanish Mexico)	latin1	wlatin1	it_IT (Italian Italy)	wlatin1

ソースおよびターゲット SAS セッションのエンコーディングは、互換性があります。es_MX ロケールに対するデフォルトエンコーディングが WLATIN1 で、ターゲット SAS セッションのエンコーディングが WLATIN1 であるためです。

ただし、ソースおよびターゲット SAS セッションのエンコーディングに互換性がない場合、移送ファイルが正常にインポートされないことがあります。(このセクションの導入部を参照してください。)次に、互換性のないエンコーディングの例を示します。

表 14.3 互換性のないエンコーディング

ソース SAS セッション			ターゲット SAS セッション	
ロケール	Linux の SAS セッションエンコーディング	移送ファイルエンコーディング	ロケール	エンコーディング
cs_CZ (Czech Czechoslovakia)	latin2	wlatin2	de_DE (German Germany)	latin1

ソースおよびターゲット SAS セッションのエンコーディングは、互換性がありません。cs_CZ ロケールのデフォルトエンコーディングが WLATIN2 で、ターゲット SAS セッションのエンコーディングが latin1 であるためです。移送ファイルは、これらのロケール間でインポートできません。

ターゲットセッションエンコーディングは latin1 です。CPORT 移送ファイルは WLATIN2 であり、このセッションでインポートできます。ただし、一部の文字は不適切に変更される場合があります。次の警告メッセージが表示されます。

```
WARNING: The transport file was created in the encoding wlatin2.  このセッションエンコーディング wlatin1
で読み込み、作成できる移送ファイルのエンコーディングは wlatin1 です。 データが正しくインポートされない可
能性があります。
```

例: CPORT プロシジャ

例 1: 複数のカタログのエクスポート

要素: PROC CPORT ステートメントオプション
FILE=
MEMTYPE=

詳細

この例では、PROC CPORT を使って指定する SAS ライブラリのすべての SAS カタログにあるエントリーをエクスポートする方法を示します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;

proc cport library=source file=tranfile memtype=catalog;
run;
```

プログラムの説明

エクスポート対象のソースファイルを含む SAS ライブラリのライブラリ参照と、出力移送ファイルが書き込まれるファイル参照を指定します。 LIBNAME ステートメントは、SAS ライブラリのライブラリ参照名を割り当てます。FILENAME ステートメントは、PROC CPORT が作成する移送ファイルのファイル特性に対するファイル参照名と動作環境オプションを割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;
```

移送ファイルを作成します。 PROC CPORT ステップは、ソースライブラリがある動作環境で実行します。MEMTYPE=CATALOG は、ソースライブラリのすべての SAS カタログを移送ファイルに書き込みます。

```
proc cport library=source file=tranfile memtype=catalog;
run;
```

ログの例

例のコード 14.1　複数のカタログのエクスポート

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FINANCE
NOTE: The catalog has 5 entries and its maximum logical record length is 866.
NOTE: Entry LOAN.FRAME has been transported.
NOTE: Entry LOAN.HELP has been transported.
NOTE: Entry LOAN.KEYS has been transported.
NOTE: Entry LOAN.PMENU has been transported.
NOTE: Entry LOAN.SCL has been transported.

NOTE: Proc CPORT begins to transport catalog SOURCE.FORMATS
NOTE: The catalog has 2 entries and its maximum logical record length is 104.
NOTE: Entry REVENUE.FORMAT has been transported.
NOTE: Entry DEPT.FORMATC has been transported.
```

例 2: 個々のカタログエントリのエクスポート

要素:

PROC CPORT ステートメントオプション
　FILE=
SELECT ステートメント

詳細

この例では、PROC CPORT を使って、カタログにあるすべてのエントリではなく、個々のカタログエントリをエクスポートする方法を示します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;

proc cport catalog=source.finance file=tranfile;
```

```
select loan.scl;
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME と FILENAME ステートメントは、ソースライブラリのライブラリ参照名と、移送ファイルのファイル参照名をそれぞれ割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;
```

エントリを移送ファイルに書き込みます。 SELECT は、LOAN.SCL エントリのみをエクスポート対象の移送ファイルに書き込みます。

```
proc cport catalog=source.finance file=tranfile;
select loan.scl;
run;
```

ログの例

例のコード 14.2　個々のカタログエントリのエクスポート

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FINANCE
NOTE: The catalog has 5 entries and its maximum logical record length is 866.
NOTE: Entry LOAN.SCL has been transported.
```

例 3: 単一 SAS データセットのエクスポート

要素:　PROC CPORT ステートメントオプション
　FILE=

詳細

この例では、PROC CPORT を使って単一 SAS データセットをエキスポートする方法を示します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;

proc cport data=source.times file=tranfile;
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME と FILENAME ステートメントは、ソースライブラリのライブラリ参照名と、移送ファイルのファイル参照名をそれぞれ割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;
```

エクスポート中のファイルの種類を指定します。 PROC CPORT ステートメントの DATA=指定により、プロシジャはライブラリやカタログではなく、SAS データセットがエクスポート中であることを認識します。

```
proc cport data=source.times file=tranfile;
run;
```

ログの例

例のコード 14.3　単一 SAS データセットのエクスポート

```
NOTE: Proc CPORT begins to transport data set SOURCE.TIMES
NOTE: The data set contains 2 variables and 2 observations.
      Logical record length is 16.
NOTE: Transporting data set index information.
```

例 4: 変換テーブルの適用

要素:

PROC CPORT ステートメントオプション
　FILE=
TRANTAB ステートメントオプション
　TYPE=

詳細

この例では、カスタマイズした変換テーブルを、PROC CPORT がエクスポートする前にトランスポートファイルに適用する方法を示します。この例では、カスタマイズした変換テーブル TTABLE1 はすでに作成されたと想定します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;

proc cport catalog=source.formats file=tranfile;
  trantab name=ttable1 type=(format);
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME と FILENAME ステートメントは、ソースライブラリのライブラリ参照名と、移送ファイルのファイル参照名をそれぞれ割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;
```

翻訳固有を適用します。 TRANTAB ステートメントは、カスタマイズした翻訳テーブル TTABLE1 で指定する翻訳を適用します。TYPE=によって、変換の適用対象が FORMAT エントリのみに制限されます。

```
proc cport catalog=source.formats file=tranfile;
  trantab name=ttable1 type=(format);
run;
```

ログの例

例のコード 14.4　変換テーブルの適用

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FORMATS
NOTE: The catalog has 2 entries and its maximum logical record length is 104.
NOTE: Entry REVENUE.FORMAT has been transported.
NOTE: Entry DEPT.FORMATC has been transported.
```

例 5: 変更日に基づくエントリのエクスポート

要素: PROC CPORT ステートメントオプション
AFTER=
FILE=

詳細

この例は、PROC CPORT で AFTER=オプションで指定する日付以降に変更されたカタログエントリのみを移送する方法を示します。

プログラム

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;

proc cport catalog=source.finance file=tranfile
      after='09sep1996'd;
run;
```

プログラムの説明

ライブラリ参照を割り当てます。 LIBNAME と FILENAME ステートメントは、ソースライブラリのライブラリ参照名と、移送ファイルのファイル参照名をそれぞれ割り当てます。

```
libname source 'sas-library';
filename tranfile 'transport-file';

host-options-for-file-characteristics;
```

移送ファイルに書き込まれるカタログエントリを指定します。 AFTER=は、変更日が 1996 年 9 月 9 日以降のカタログエントリのみが移送ファイルに書き込まれるように指定します。

```
proc cport catalog=source.finance file=tranfile
      after='09sep1996'd;
run;
```

ログの例

PROC CPORT は指定カタログのすべてのエントリのエクスポートプロセスが開始したと知らせるメッセージを SAS ログに出力します。しかし、PROC CPORT は FINANCE カタログの LOAN.FRAME と LOAN.HELP エントリのみを移送ファイルに書き出しています。なぜなら、それら 2 つのエントリのみの変更日時が 1996 年 9 月 9 日以降だったからです。すなわち、指定カタログ内のすべてのエントリで、その 2 つのみが AFTER=オプションの要件を満たしていたからです。

例のコード 14.5　変更日に基づくエントリのエクスポート

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FINANCE
NOTE: The catalog has 5 entries and its maximum logical record length is 866.
NOTE: Entry LOAN.FRAME has been transported.
NOTE: Entry LOAN.HELP has been transported.
```

15

DATASETS プロシジャ

概要: DATASETS プロシジャ

DATASETS プロシジャを使用したデータセットの管理

DATASETS プロシジャは SAS ファイルを管理するユーティリティプロシジャです。PROC DATASETS を使用して、次のことができます。

- SAS ファイルを 1 つの SAS ライブラリから他のライブラリにコピーする
- SAS ファイルの名前を変更する
- SAS ファイルを修復する
- SAS ファイルを削除する
- SAS ライブラリにある SAS ファイルをリストする
- SAS データセットの属性をリストする
  - データが最後に変更された日付
  - データが圧縮されているかどうか
  - データがインデックス付きかどうか
- SAS ファイルのパスワードを操作する
- SAS データセットに追加する
- SAS データセットの属性やデータセット内の変数を変更する
- SAS データセットのインデックスを作成および削除する
- SAS データセットの監査ファイルを作成および管理する
- SAS データセットの一貫性制約を作成および削除する
- データセットの拡張属性を作成および管理する

注

- DATASETS プロシジャでもカタログに対していくつかの操作はできますが、通常はカタログ管理には CATALOG プロシジャが最適なユーティリティです。
- *member* という用語は、しばしば *SAS ファイル*と同意語で使われます。
- PROC DATASETS は順次データライブラリとは使用できません。
- LENGTH ステートメントや、ATTRIB ステートメントの LENGTH=オプションを使用して、変数の長さを変更することはできません。
- PROC DATASETS や PROC CONTENTS および SAS エクスプローラーのような他の SAS のコンポーネントの間で、変更日時に差がでることがあります。2 つの変更日時には明らかな違いがあります。
  - オペレーション環境の変更日時は SAS エクスプローラーと PROC DATASETS LIST オプションで表示されます。
  - CONTENTS ステートメントで使われる変更日付はデータセット内のデータが実際に変更された日時です。
- 大量のメンバーを含むライブラリの場合、DATASETS プロシジャの処理時間が長くなる場合があります。パフォーマンスを向上するには、ライブラリを小さいライブラリに再編成するとよいでしょう。
- 拡張属性は計算サーバーでサポートされていますが、CAS ではサポートされていません。

- DATASETS プロシジャの KILL 機能と DBMS=Redshift を使用したい場合は、LIBNAME ステートメントに SCHEMA=オプションを指定する必要があります。

概念: DATASETS プロシジャ

プロシジャの実行

ステートメントの実行

DATASETS プロシジャを開始する際、PROC DATASETS ステートメントでプロシジャ入力ライブラリを指定します。プロシジャ入力ライブラリを省略すると、プロシジャは現在のデフォルトの SAS ライブラリ(通常は WORK ライブラリ)を使います。新しいプロシジャ入力ライブラリを指定するには、DATASETS プロシジャを再度発行してください。

ステートメントは記述された順番で実行されます。データセットのコンテンツを確認し、データセットをコピーして、最初のデータセットのコンテンツと 2 番目のものを視覚的に比較する場合、CONTENTS、COPY、CONTENTS を使用します。

RUN グループ処理

PROC DATASETS は RUN グループ処理をサポートしています。RUN グループ処理では、プロシジャを終了しなくても RUN グループをサブミットできます。

DATASETS プロシジャは 4 種類の RUN グループをサポートしています。各 RUN グループはそれを構成および実行するステートメントで定義されます。

PROC DATASETS のいくつかのステートメントは、インプライド RUN ステートメントとしての役割を果たし、その前にある RUN グループを実行させます。

次のリストは、どのステートメントが RUN グループを構成して、何が各 RUN グループを実行させるかについて説明します。

- PROC DATASETS ステートメントは、いつでも即座に実行されます。PROC DATASETS ステートメントの実行に、他のステートメントは必要ありません。したがって、PROC DATASETS ステートメントは、それ単体で RUN グループになります。
- MODIFY ステートメントとそれに従属するステートメントは RUN グループを形成します。これらの RUN グループは、いつでも即座に実行されます。MODIFY RUN グループの実行に、他のステートメントは必要ありません。

MODIFY ステートメントは計算サーバーで実行されていますが、CAS ではサポートされていません。

- APPEND、CONTENTS、および COPY ステートメント(存在する場合は、EXCLUDE と SELECT も含む)は、それぞれに別の RUN グループを形成します。すべての APPEND ステートメントは単一ステートメント RUN グループ、すべての CONTENTS ステートメントも単一ステートメント RUN グループ、そしてすべての COPY ステップは RUN グループを形成します。プロシジャのその他のステートメント(COPY ステートメントまたは NODIFY ステートメントのいずれかに従属しているステートメントを除く)により、RUN グループが実行されます。
- RUN グループは、次のうち 1 つ以上のステートメントから形成されます。
  - AGE
  - CHANGE
  - DELETE
  - EXCHANGE
  - REPAIR
  - SAVE

  これらのステートメントのうちいずれかが PROC ステップのシーケンスで記述されている場合、そのシーケンスは RUN グループを形成しています。たとえば、REPAIR ステートメントが SAVE ステートメントの直後に記述されている場合、REPAIR ステートメントは強制的に SAVE ステートメントを実行しません。同じ RUN グループの一部となります。RUN グループを実行するには、次のステートメントのうちいずれかをサブミットします。
  - PROC DATASETS
  - APPEND
  - CONTENTS
  - COPY
  - MODIFY
  - QUIT
  - RUN
  - 別の DATA ステップまたは PROC ステップ

1 つのタスクと関連付けられているプログラムステートメントが、RUN ステートメントまたはインプライド RUN ステートメントまで読み込まれます。その前のすべてのステートメントが即座に実行され、別の RUN ステートメントまたはインプライド RUN ステートメントまで読み込みが続行されます。最後のタスクを実行するには、RUN ステートメント、またはプロシジャを停止するステートメントを使用する必要があります。

次の PROC DATASETS ステップには、5 つの RUN グループが含まれています。

```
LIBNAME dest 'SAS-library';
  /* RUN group */

proc datasets;
    /* RUN group */
  change nutr=fatg;
```

```
  delete bldtest;
  exchange xray=chest;
    /* RUN group */
  copy out=dest;
    select report;
    /* RUN group */
  modify bp;
    label dias='Taken at Noon';
    rename weight=bodyfat;
    /* RUN group */
  append base=tissue data=newtiss;
quit;
```

注: 対話型ラインモードで実行している場合、RUN ステートメントをサブミットする前にステートメントがすでに実行されたというメッセージを受信できます。この環境を PROC DATASETS の実行に使用している場合、タスクの計画には注意してください。

エラー処理

通常、ステートメントでエラーが発生した場合、エラーを含む RUN グループは実行されません。エラーを含む RUN グループの前後の RUN グループは正常に実行されます。MODIFY RUN グループは例外です。MODIFY ステートメントに従属しているステートメントで構文エラーが発生した場合、エラーを含むステートメントのみ実行されません。RUN グループのその他のステートメントは実行されます。

注: ステートメント(ステートメント名)の最初の文字にエラーがあり、プロシジャでそれを認識できない場合、プロシジャはそのステートメントを前の RUN グループの一部として処理します。

パスワードエラー

ステートメントに正しくないパスワードまたは省略されたパスワードに関連するエラーがある場合、エラーはエラーを含むステートメントにのみ影響します。RUN グループのその他のステートメントは実行されます。

エラーのある RUN グループの強制実行

PROC DATASETS ステートメントの FORCE オプションは、1 つ以上のステートメントにエラーが含まれている場合でも RUN グループの実行を強制します。エラーのないステートメントのみ実行されます。

プロシジャの終了

DATASETS プロシジャを停止するには、QUIT ステートメント、RUN CANCEL ステートメント、新しい PROC ステートメントまたは DATA ステートメントのいずれかを発行する必要があります。QUIT ステートメントをサブミットすると、実行されなかったステートメントが実行されます。RUN CANCEL ステートメントをサブミットすると、実行されなかったステートメントがキャンセルされます。

拡張属性

計算サーバーでは拡張属性を使用できますが、CAS では使用できません。

拡張属性とは、SAS ファイル用にカスタマイズされたメタデータのことです。ユーザー定義の特性で、SAS データセットまたは変数と関連付けられます。データセットの変数長などの SAS 属性は事前に定義された SAS システム属性ですが、拡張属性は自分で定義する属性です。拡張属性は(名前, 値)ペアから構成されます。

MODIFY ステートメントを使用すると、拡張属性を追加、削除、設定、更新できます。COPY ステートメントを使用すると、OUT=ライブラリエンジンが拡張属性をサポートする場合、拡張属性がコピーされます。APPEND ステートメントを使用すると、BASE=データセットが存在しない場合に拡張属性は付加されません。

拡張属性は、カスタム属性を変数またはデータセットに関連付けるために必要なタスクの自動化に使用されます。

拡張属性には、数値または文字値を指定できます。各著属性の文字値に最大長はありません。デフォルトでは、各値は 256 バイトのセグメントに保存されます。XATTR OPTIONS ステートメントの SEGLEN=オプションを使用すると、セグメント長を変更できます。このオプションは、文字属性値を保持するストレージ要素長を指定します。特定の文字属性値の中に大きさが足りないセグメントサイズがある場合、別のセグメントが割り当てられます。処理時間を最小限に抑えるため、データセットの大部分の属性値を調整する長さを選択します。

次の出力は、拡張属性を含むデータセットと変数を示します。

アウトプット 15.1　拡張属性を持つデータセットのコンテンツ

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
2	age	Num	8
5	cars	Num	8
3	income	Num	8
4	kids	Num	8
1	purchase	Char	3

Alphabetic List of Data Set Extended Attributes

Extended Attribute	Numeric Value	Character Value
attrib	.	table
role	.	train

Alphabetic List of Extended Attributes on Variables

Extended Attribute	Attribute Variable	Numeric Value	Character Value
level	income	.	interval
level	purchase	.	nominal
role	age	.	reject
role	income	.	input
role	purchase	.	target

拡張属性の使用については、“XATTR ADD ステートメント” (457 ページ)、“XATTR DELETE ステートメント” (458 ページ)、“XATTR REMOVE ステートメント” (459 ページ)、“XATTR SET ステートメント” (460 ページ)、および“XATTR UPDATE ステートメント” (461 ページ)を参照してください。

ディレクトリレベルでのライブラリのコンテンツ I

連結ライブラリでは、ライブラリの各物理的なピースはレベルです。レベル 1 は、ライブラリ参照名定義の最初の場所であり、レベル 2 は 2 番目の場所です。PROC DATASETS は、ライブラリのレベル数を LEVEL 指定の値として報告します。

VARCHAR を含む CONTENTS ステートメント

テーブルに VARCHAR データ型がある場合、PROC CONTENTS または PROC DATASETS CONTENTS ステートメントは、バイトと文字での長さ、および使用される最大バイトを示します。PROC CONTENTS は、テーブルに関するメタデータと変数に関するメタデータをレポートします。CAS エンジンは、VARCHAR をサポートする唯一のエンジンです。

PROC CONTENTS(変数メタデータに関連する)の下部は、計算サーバーセッションで表されるメタデータです。Viya で使用される可変バイト長は、セッションのエンコーディングに依存します。セッションで使用される可変バイト長は、バイト長と異なる場合があります。

アウトプット 15.2　Mycas.French2 テーブルのコンテンツ

Data Set Name	MYCAS.FRENCH2	Observations	11
Member Type	DATA	Variables	2
Engine	CAS	Indexes	0
Created	08/16/2017 18:05:10	Observation Length	40
Last Modified	08/16/2017 18:05:10	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Limit	100MB
Caslib	CASUSER,
Scope	Session

Alphabetic List of Variables and Attributes

#	Variable	Type	Len		Max Bytes Used
			Bytes	Chars	
1	nterm	Varchar	80	20	19
2	term	Char	19		

Mycas ライブラリのコンテンツを表示するには、次のコードを使用します。

```
proc datasets lib=mycas;
contents data=cars;
run;
```

アウトプット 15.3　Mycas ライブラリと Mycas.Cars テーブル

Data Set Name	MYCAS.CARS	Observations	428
Member Type	DATA	Variables	15
Engine	CAS	Indexes	0
Created	08/16/2017 17:55:43	Observation Length	160
Last Modified	08/16/2017 17:55:43	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

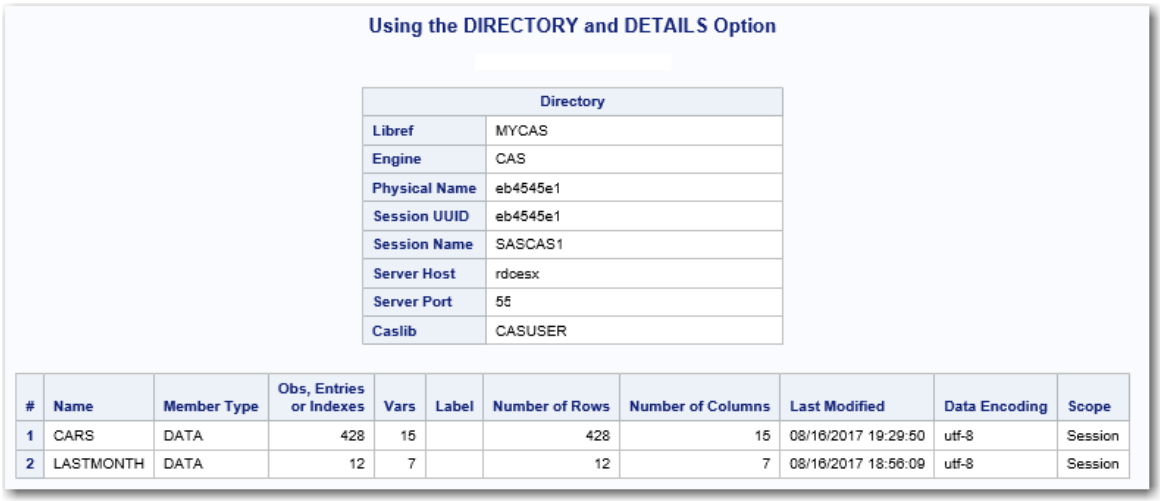
Engine/Host Dependent Information	
Data Limit	100MB
Caslib	CASUSER
Scope	Session

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
9	Cylinders	Num	8		
5	DriveTrain	Char	5		
8	EngineSize	Num	8		Engine Size (L)
10	Horsepower	Num	8		
7	Invoice	Num	8	DOLLAR8.	
15	Length	Num	8		Length (IN)
11	MPG_City	Num	8		MPG (City)
12	MPG_Highway	Num	8		MPG (Highway)
6	MSRP	Num	8	DOLLAR8.	
1	Make	Char	13		
2	Model	Char	40		
4	Origin	Char	6		
3	Type	Char	8		
13	Weight	Num	8		Weight (LBS)
14	Wheelbase	Num	8		Wheelbase (IN)

Mycas ライブラリのコンテンツを表示するには、次のコードを使用します。

```
proc datasets lib=mycas;
contents data=mycs.cars directory details;
run;
```



Using the DIRECTORY and DETAILS Option

Directory	
Libref	MYCAS
Engine	CAS
Physical Name	eb4545e1
Session UUID	eb4545e1
Session Name	SASCAS1
Server Host	rdoesx
Server Port	55
Caslib	CASUSER

#	Name	Member Type	Obs, Entries or Indexes	Vars	Label	Number of Rows	Number of Columns	Last Modified	Data Encoding	Scope
1	CARS	DATA	428	15		428	15	08/16/2017 19:29:50	utf-8	Session
2	LASTMONTH	DATA	12	7		12	7	08/16/2017 18:56:09	utf-8	Session

SAS/ACCESSLIBNAME エンジンを使用する場合の DATASETS プロシジャ出力の違い

SAS/ACCESS LIBNAME エンジンを使用してデータベースにアクセスする 場合、SAS データセットのヘッダーで使用できる情報の一部は、プロシジャで使用できません。したがって、子のプロシジャには表示する情報がありません。CONTENTS プロシジャ、DATASETS COPY ステートメント、および DATASETS CONTENTS ステートメントには、索引、一貫性制約、およびオブザベーション数に関する情報はありません。たとえば、索引と一貫性制約を持つテーブルをコピーすると、テーブルにアクセスするために使用されたエンジンが SAS/ACCESS LIBNAME エンジンの場合は、索引や一貫性制約のないテーブルが作成されます。次のコードでは、CONTENTS ステートメントからの出力に次のように表示されます。データベースにインデックスがあってもインデックスフィールドには 0(ゼロ)が表示され、オブザベーション数が不明な場合には . (期間)が表示されます。

```
proc datasets lib=mylib details;
  contents data=table-name;
run;
quit;
```

オブザベーション数を取得するには、LIBNAME エンジンを使用してカウントを取得する SQL クエリを実行するか、DBMS テーブルを明示的にパススルーします。

索引および一貫性制約に関する情報を入手するには、データベース固有の構文の明示的なパススルーを使用して、システムテーブルを照会します。明示的なパススルーに関するドキュメントについては、“パススルー機能を使用した DBMS への接続” (*SAS SQL プロシジャユーザーガイド*)を参照してください。

DATASETS プロシジャの CAS の処理

DATASETS プロシジャを使用して CAS LIBNAME エンジンを介して CAS テーブルにアクセスすると、他の LIBNAME エンジンとは異なる動作が発生します。これらの動作の違いを次に示します。

- インデックス、一貫性制約、および拡張属性は、CAS テーブルではサポートされていませが、計算サーバーではサポートされています。
- UPDATE アクセスがサポートされていないため、MODIFY ステートメントはエラーになります。
- テーブルに VARCHAR データ型がある場合、CONTENTS ステートメントには、バイト単位の長さ、文字単位の長さ、および使用された最大バイト数が表示されます。
- CONTENTS ステートメントで示されるエンコーディングは、CAS テーブルのエンコーディングです。
- CAS テーブルを SAS データセットに追加する場合、ブロック I/O メソッドは使用できません。
- UPDATE アクセスはサポートされていないため、APPEND ステートメントを使用して CAS テーブルにデータを追加することはできません。CAS テーブルのデータを SAS データセットに追加できます。
- CAS LIBNAME エンジンが COPY ステートメントで入力ライブラリと出力ライブラリの両方に使用されている場合、COPY タスクは CAS サーバー上で実行されます。NOACCEL オプションはこの動作を無効にし、計算サーバーセッションで COPY タスクが実行されるようにします。

PROC DATASETS と ODS (Output Delivery System)

ほとんどのプロシジャは、メッセージをログに送信し、プロシジャ結果を出力に送信します。PROC DATASETS は、プロシジャ結果をログとプロシジャ出力テーブルの両方に送信するため、固有のものです。ODS へのインターフェイスが作成された際に、すべてのプロシジャ 結果(ログとプロシジャ出力テーブルの両方からの)が ODS で利用可能になる必要があるとされました。この機能を実装し、前のリリースとの互換性を保持するため、ODS へのインターフェイスは通常のインターフェイスと多少異なっている必要がありました。

デフォルトで、PROC DATASETS ステートメントはそれ自体で 2 つの出力オブジェクト Members と Directory を生成します。これらのオブジェクトはログに送られます。CONTENTS ステートメントは、デフォルトで 3 つの出力オブジェクト Attributes、EngineHost、および Variables を生成します。(さまざまなオプションの使用により、その他の出力オブジェクトが追加されます)。これらのオブジェクトは、プロシジャ出力テーブルに送られます。ODS 出力先(HTML、RTF、PRINTER な

ど)を開いている場合、これらのオブジェクトはすべてデフォルトでその出力先に送られます。

その他のプロシジャに対して実行するのと同じように、ODS SELECT ステートメントと ODS EXCLUDE ステートメントを使用して、オブジェクトとオブジェクトの出力先を制御できます。

PROC DATASETS と PROC CONTENTS は、作成するテーブルにそれぞれ名前を割り当てます。これらの名前を使用して、Output Delivery System (ODS)を使用してテーブルを選択し、出力テーブルを作成する際にそのテーブルを参照できます。

PROC CONTENTS は、CONTENTS ステートメントを使用した PROC DATASETS と同じ ODS テーブルを生成します。

表 15.1 CONTENTS ステートメントなしで DATASETS プロシジャによって生成される ODS テーブル

ODS テーブル	説明	テーブルの生成
Directory	一般ライブラリ情報	NOLIST オプションを指定しない場合
Members	ライブラリメンバー情報	NOLIST オプションを指定しない場合

表 15.2 CONTENTS ステートメントで PROC CONTENTS と PROC DATASETS によって生成される ODS テーブル名

ODS テーブル	説明	テーブルの生成
属性	データセット属性	SHORT オプションを指定しない場合
Directory	一般ライブラリ情報	DATA=<*libref*.>_ALL_ または DIRECTORY オプションを指定する場合
Members	ライブラリメンバー情報	DATA=<*libref*.>_ALL_ または DIRECTORY オプションを指定する場合
位置	テーブルの論理位置別の変数の詳細リスト	VARNUM オプションを指定し、SHORT オプションを指定しない場合
PositionShort	テーブルの論理位置別の変数の簡潔リスト	VARNUM オプションと SHORT オプションを指定する場合
変数	アルファベット順の変数の詳細リスト	SHORT オプションを指定しない場合
VariablesShort	アルファベット順の変数の簡潔リスト	SHORT オプションを指定する場合

構文: DATASETS プロシジャ

注: 長整数よりも大きい大きな数値で終わるデータセット名および変数名は、番号付き範囲リストでは使用できません。詳細については、“SAS 変数の定義” (*SAS プログラマガイド: エッセンシャル*)を参照してください。.

注: _LAST_は、セッションで作成された最後のデータセットの名前です。BASE=データセットが存在せず、PROC APPEND がそれを作成する場合、PROC APPEND は _LAST_を BASE=データセットの名前に設定します。

ヒント: RUN グループ処理をサポートします。

Output Delivery System をサポートします。詳細については、“Output Delivery System: 基本概念” (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

参照項目: 詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (23 ページ)を参照してください。

PROC DATASETS <*options*>;

AGE *current-name related-SAS-file(s)*
</ <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

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

AUDIT *SAS-file* <(*SAS-password* <ENCRYPTKEY=*key-value*>
<GENNUM=*integer*>)>;

INITIATE <AUDIT_ALL=NO | YES>;

LOG <ADMIN_IMAGE=YES | NO>
<BEFORE_IMAGE=YES | NO>
<DATA_IMAGE=YES | NO>
<ERROR_IMAGE=YES | NO>;

<SUSPEND | RESUME | TERMINATE;>

<USER_VAR> *variable-name-1* <$> <*length*> <LABEL='*variable-label*' >
<*variable-name-2* <$> <*length*> <LABEL='*variable-label*' > ...>;

CHANGE *old-name-1=new-name-1*
<*old-name-2=new-name-2* ...>
</ <ENCRYPTKEY=*key-value*>
<ALTER=*alter-password*> <GENNUM=ALL | *integer*> <MEMTYPE=*member-type*>>;

CONTENTS <*options*>;

COPY <ACCEL | NOACCEL>OUT=*libref-1*

<CLONE | NOCLONE>
<CONSTRAINT=YES | NO>
<DATECOPY>
<ENCRYPTKEY=*key-value*>
<FORCE>
<IN=*libref-2*>
<INDEX=YES | NO>
<MEMTYPE=(*member-type(s)*)>
<MOVE <ALTER=*alter-password*>>
<OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2 ...>*) >;

SELECT *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*> <ALTER=*alter-password*>
<MEMTYPE=*member-type*>>;

DELETE *SAS-file(s)*
</ <ALTER=*alter-password*>
<ENCRYPTKEY=*key-value*>
<GENNUM=ALL | HIST | REVERT | *integer*>
<MEMTYPE=*member-type*>>;

EXCHANGE *name-1=other-name-1 <name-2=other-name-2 ...>*
</ <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

MODIFY *SAS-file*<(*options*)>
</ <CORRECTENCODING=*encoding-value*> <DTC=*SAS-date-time*>
<GENNUM=*integer*> <MEMTYPE=*member-type*>>;

ATTRIB *variable-list(s) attribute-list(s)*;

FORMAT *variable-1*<*format-1*>
<*variable-2* <*format-2*> ...>;

IC CREATE<*constraint-name=*>*constraint*<MESSAGE='*message-string*'
<MSGTYPE=USER>>;

IC DELETE*constraint-name(s)* | _ALL_;

IC REACTIVATE*foreign-key-name* REFERENCES *libref*;

INDEX CENTILES *index(s)*
</ <REFRESH> <UPDATECENTILES=ALWAYS | NEVER | *integer*>>;

INDEX CREATE *index-specification(s)*
</ <NOMISS> <UNIQUE> <UPDATECENTILES=ALWAYS | NEVER |
integer>>;

INDEX DELETE*index(s)* | _ALL_;

INFORMAT *variable-1*<*informat-1*>
<*variable-2* <*informat-2*> ...>;

LABEL*variable-1=<'label-1'* | ' '>
<*variable-2=<'label-2'* | ' '> ...>;

RENAME *old-name-1=new-name-1*
<*old-name-2=new-name-2* ...>;

XATTR ADD DS *attribute-name-1=attribute-value-1*
<*attribute-name-2=attribute-value-2* ...>;

or

XATTR ADD VAR *variable-name-1* (*attribute-name-1=attribute-value-1*
< *attribute-name-2=attribute-value-2* ...>)

<variable-name-2 (*attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 ...>)>;

XATTR DELETE;

XATTR OPTIONS<SEGLEN=*number-of-bytes*>;

XATTR REMOVE DS *attribute-name(s)* ;

or

XATTR REMOVE VAR *variable-name-1* (*attribute-name(s)*)
<variable-name-2 (*attribute-name(s)* ...)>;

XATTR SET DS *attribute-name-1=attribute-value-1*
<attribute-name-2="attribute-value-2" ...>;

or

XATTR SET VAR *variable-name-1* (*attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 ... >)
<variable-name-2 (*attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2> ...)>;

XATTR UPDATE DS *attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 ...>;

or

XATTR UPDATE VAR *variable-name-1* (*attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 ...>)
<variable-name-2 (*attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 ...>)>;

REBUILD *SAS-file* </ <ENCRYPTKEY=*key-value*>
<ALTER=*password*> <GENNUM=*n*> <MEMTYPE=*member-type*>
<NOINDEX>>;

REPAIR </ <ENCRYPTKEY=*key-value*>
<ALTER=*alter-password*> <GENNUM=*integer*> <MEMTYPE=*member-type*>>;

SAVE *SAS-file(s)* </ MEMTYPE=*member-type*>;

ステートメント	タスク	例
“PROC DATASETS ステートメント”	SAS ファイルを管理する	
“AGE ステートメント”	関連 SAS ファイルグループの名前を変更する	Ex. 7
“APPEND ステートメント”	ある SAS データセットのオブザベーションを別の SAS データセットの末尾に追加する	Ex. 6
“ATTRIB ステートメント”	出力形式、入力形式、またはラベルを MODIFY ステートメントで指定した SAS データセットの変数と関連付ける	Ex. 1
“AUDIT ステートメント”	監査ファイルへのイベントの記録を開始、制御、中断、再開、終了する	
“CHANGE ステートメント”	1 つ以上の SAS ファイルの名前を変更する	Ex. 2

ステートメント	タスク	例
“CONTENTS ステートメント”	1 つ以上の SAS データセットのコンテンツを記述し、SAS ライブラリのディレクトリを出力する	Ex. 5
“COPY ステートメント”	SAS ファイルのすべて、または一部をコピーする	Ex. 2
“DELETE ステートメント”	SAS ファイルを削除する	Ex. 2
“EXCHANGE ステートメント”	2 つの SAS ファイル名を交換する	Ex. 2
“EXCLUDE ステートメント”	SAS ファイルをコピーから除外する	Ex. 2
“FORMAT ステートメント”	変数の出力形式を常に割り当て、変更、削除する	Ex. 4
“IC CREATE ステートメント”	一貫性制約を作成する	
“IC DELETE ステートメント”	一貫性制約を削除する	
“IC REACTIVATE ステートメント”	外部キー一貫性制約を再アクティブ化する	
“INDEX CENTILES ステートメント”	インデックス付き変数のパーセント点数統計量を更新する	
“INDEX CREATE ステートメント”	単一インデックスまたは複合インデックスを作成する	Ex. 4
“INDEX DELETE ステートメント”	1 つ以上のインデックスを削除する	
“INFORMAT ステートメント”	変数の入力形式を常に割り当て、変更、削除する	Ex. 4
“INITIATE ステートメント”	SAS データファイルと同じ名前で、データセットの種類が AUDIT の監査ファイルを作成する	Ex. 8
“LABEL ステートメント”	変数ラベルを割り当て、変更、削除する	Ex. 4
“LOG ステートメント”	監査ファイルの設定を指定する	Ex. 8
“MODIFY ステートメント”	SAS ファイルの属性と変数の属性を変更する	Ex. 4
“REBUILD ステートメント”	無効化されたインデックスと一貫性制約を元に戻すか、削除するかどうかを指定する	
“RENAME ステートメント”	SAS データセットの変数の名前を変更する	Ex. 4

ステートメント	タスク	例
“REPAIR ステートメント”	破損した SAS データセットまたはカタログを元に戻す	
“RESUME ステートメント”	監査ファイルが中断された場合に、監査ファイルへのイベント記録を再開する	Ex. 8
“SAVE ステートメント”	SAVE ステートメントにリストされている以外のすべての SAS ファイルを削除する	Ex. 3
“SELECT ステートメント”	コピーする SAS ファイルを選択する	Ex. 2
“SUSPEND ステートメント”	監査ファイルへのイベントの記録を中断する	Ex. 8
“TERMINATE ステートメント”	イベントの記録を終了し、監査ファイルを削除する	Ex. 8
“USER_VAR ステートメント”	オブザベーションへの更新のたびに任意変数が監査ファイルに記録されるように定義する	Ex. 8
“XATTR ADD ステートメント”	拡張属性を変数またはデータセットに追加する	Ex. 9
“XATTR DELETE ステートメント”	すべての拡張属性をデータセットから削除する	
“XATTR OPTIONS ステートメント”	拡張属性に必要なオプションを指定する	Ex. 9
“XATTR REMOVE ステートメント”	変数またはデータセットから拡張属性を削除する	
“XATTR SET ステートメント”	拡張属性を変数またはデータセットを更新または追加する	
“XATTR UPDATE ステートメント”	変数またはデータセットの拡張属性を更新する	Ex. 9

PROC DATASETS ステートメント

SAS ファイルを管理します。

構文

PROC DATASETS <*options*>;

オプション引数の要約

ALTER=*alter-password*
: SAS ライブラリ内の変更保護された SAS ファイルへの変更アクセス権限を与えます。

DETAILS

NODETAILS
: オブザベーションの数、変数の数、インデックスの数、およびデータセットラベルに関する情報をログに含めます。

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

FORCE
: エラーがある場合でも、RUN グループの実行または追加操作を強制的に行います。

GENNUM=ALL | HIST | REVERT | *integer*
: 世代データセットの処理を制限します。

KILL
: SAS ファイルを削除します。

LIBRARY=*libref*
: プロシジャ入力/出力ライブラリを指定します。

MEMTYPE=(*member-type(s)*)
: 処理を特定の種類の SAS ファイルに制限します。

NOLIST
: ディレクトリを出力しません。

NOPRINT
: ログおよびリストへの出力を抑制します。

NOWARN
: エラー処理を行いません。

PW= *password*
: 読み取り、書き込み、変更のアクセス権限を与えます。

READ=*read-password*
: 読み取りアクセス権限を与えます。

オプション引数

ALTER=*alter-password*
SAS ライブラリ内の変更保護された SAS ファイルに変更パスワードを与えます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

DETAILS | NODETAILS
次の列がログに書き込まれるかどうかを決定します。

Obs、Entries、または Indexes	種類 AUDIT、DATA、VIEW の SAS ファイルのオブザベーション数、種類 CATALOG のエントリ数、およびデータファイルと関連付けられている種類 INDEX のファイル数(ある場合)を提供します。SAS データセットのオブザベーション数を決定できない場合、この列の値は欠損値に設定されます。たとえば、非常に大きなデータセットで、オブザベーションまたは削除されたオブザベ

ーションの数が倍精度整数で保存可能な数を超えている場合、カウントは欠損値として表示されます。タイプ CATALOG の値は、エントリの合計数です。その他のタイプの場合、この列はブランクです。

Vars　種類 AUDIT、DATA、VIEW の変数の数を提供します。SAS データセットの変数の数を決定できない場合、この列の値は欠損値に設定されます。その他のタイプの場合、この列はブランクです。

Label　SAS データセットと関連付けられているラベルが含まれます。この列は、種類 DATA に対してのみラベルを出力します。

> ヒント　タイプ INDEX のファイルの値は、ユーザー定義のインデックスと、一貫性制約によって作成されるインデックスを含みます。インデックス所有権と属性情報を表示するには、PROC DATASETS を CONTENTS ステートメントと OUT2 オプションと使用します。

注: DETAILS オプションが出力に影響するのは、ディレクトリが指定され、そのディレクトリに SAS ライブラリ内の読み取り保護されているすべての SAS ファイルへの読み取りアクセス権限が必要な場合だけです。読み取りパスワードを入力しない場合、ディレクトリリストには DETAILS オプションによって作成された列に対する欠損値が含まれます。

デフォルト　DETAILS も NODETAILS も指定されない場合、デフォルトはシステムオプション設定となります。デフォルトシステムオプション設定は、NODETAILS です。

ヒント　SAS ウィンドウ環境を使用し、読み取り保護されている SAS ファイルを含むライブラリに対し DETAILS オプションを指定する場合、ダイアログボックスで PROC DATASETS ステートメントで指定しない各読み取りパスワードの入力を求められます。そのため、同一の読み取りパスワードを同一の SAS ライブラリ内のすべてのファイルに割り当てます。

例　“例 2: SAS ファイルの操作” (494 ページ)

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

参照項目:　“AES 暗号化データセットの追加” (389 ページ)

FORCE
2 つの別のアクションを実行します。

- RUN グループの 1 つ以上のステートメントにエラーがある場合でも、強制的に RUN グループを実行します。RUN グループ処理とエラー処理の詳細については、“RUN グループ処理” (362 ページ) を参照してください。
- データセット内の変数が同一でない場合でも、すべての APPEND ステートメントによる 2 つのデータセットの連結を強制的に行います。APPEND ステートメントは、NOWARN オプションが指定(APPEND ステートメントまたは PROC DATASETS によって)されていない限り余剰の変数をドロップし、警告

メッセージを SAS ログに発行します。FORCE オプションの詳細については、APPEND ステートメント (383 ページ)を参照してください。

GENNUM=ALL | HIST | REVERT | *integer*
世代データセットの処理を制限します。有効な値は次のとおりです。

ALL
下位の CHANGE ステートメントと DELETE ステートメントについては、世代グループの基本バージョンおよびすべての履歴バージョンを参照します。

HIST
下位の DELETE ステートメントについては、すべての履歴バージョンを参照します。ただし、世代グループの基本バージョンは除きます。

REVERT | 0
下位の DELETE ステートメントについては、世代グループの基本バージョンを参照し、(ある場合は)最新の履歴バージョンを基本バージョンに変更します。

integer
下位のステートメント AUDIT、CHANGE、MODIFY、DELETE、REPAIR については、世代グループの特定なバージョンを参照します。正の数の指定は、データセット名に追加される特定の世代番号の絶対参照です(つまり、**gennum=2** は MYDATA#002 を指定します)。負の数を指定すると、最新から最古への基本バージョンに相対する履歴バージョンの相対参照になります(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。

参照項目: "世代データセットの処理制限" (463 ページ)

KILL
SAS ライブラリ内の処理可能なすべての SAS ファイルを削除します。MEMTYPE= オプションは、ステートメントが削除するメンバーの種類をサブセットします。次の例では、WORK ライブラリ内のすべてのデータファイルを削除します。

```
proc datasets lib=work kill memtype=data; run; quit;
```

注意
KILL オプションは、ステートメントのサブミット直後に SAS ファイルを削除します。 SAS ファイルに ALTER=パスワードが割り当てられている場合、そのパスワードを SAS ファイルを削除するために指定する必要があります。

LIBRARY=*libref*
プロシジャが処理するライブラリを指定します。このライブラリは、*プロシジャ入力/出力ライブラリ*です。

別名 DDNAME=
DD=
LIB=

デフォルト Work または User

注 順次エンジン(テープ形式エンジンなど)を介してアクセスされる SAS ライブラリは、LIBRARY=オプションの値として指定できません。

例 “例 2: SAS ファイルの操作” (494 ページ)

MEMTYPE=(*member-type(s)*)
処理を 1 つ以上のメンバーの種類に制限し、データライブラリディレクトリのリストを指定されているメンバーの種類の SAS ファイルに制限します。たとえば、次の PROC DATASETS ステートメントは処理をデフォルトデータライブラリ内の SAS データセットに制限し、SAS ログ内のディレクトリリストをメンバーの種類 DATA の SAS ファイルに制限します。

```
proc datasets memtype=data;
```

別名 MTYPE=

MT=

デフォルト ALL

参照項目: “処理するメンバーの種類の制限” (465 ページ)

NOLIST
SAS ログおよび開いている LISTING 以外の出力先で SAS ファイルのディレクトリの出力を抑制します。

注 ODS LISTING を有効化して LISTING ODS 以外の出力先を開く場合、PROC DATASETS 出力は SAS ログおよび ODS 出力先の両方に送信されます。NOLIST オプションはこのいずれにも出力しません。SAS ログのみで出力を確認するには、メンバーおよびディレクトリ出力オブジェクトを指定し、ODS EXCLUDE ステートメントを使用します。たとえば、RTF および LISTING 出力先が両方開いており、ディレクトリとメンバー情報が **LOG** ウィンドウのみで必要な場合、次を使用します。

```
ods rtf file="listing_nolist.rtf";
ods trace on;
ods rtf exclude directory members;

proc datasets lib=work;
run;
quit;

ods rtf close;
```

例 “例 4: SAS データセットの変更” (502 ページ)

NOPRINT
SAS ログおよび開いている LISTING 以外の出力先で SAS ファイルのディレクトリの出力を抑制します。NOPRINT オプションは、CONTENTS ステートメントにおける NOLIST オプションと NOPRINT オプションの組み合わせです。

NOWARN
ステートメント SAVE、CHANGE、EXCHANGE、REPAIR、DELETE または COPY で指定される SAS ファイル、または AGE ステートメントで最初の SAS ファイル

としてリストされる SAS ファイルがプロシジャ入力ライブラリにない場合に発生するエラー処理を行いません。エラーが発生し、NOWARN オプションが有効な場合、PROC DATASETS はその RUN グループの処理を続行します。NOWARN が無効な場合、PROC DATASETS は RUN グループの処理を停止し、処理を停止しない DELETE 以外のすべての操作に対し警告を発行します。

PW= *password*
SAS ライブラリ内の保護されている SAS ファイルに対するパスワードを与えます。PW=は、READ=、WRITE=または ALTER=に対する別名として機能します。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

READ=*read-password*
SAS ライブラリ内の読み取り保護されている SAS ファイルに対する読み取りパスワードを与えます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

AGE ステートメント

ライブラリ内の関連 SAS ファイルグループの名前を変更します。

例: “例 7: SAS データセットのエージング” (511 ページ)

構文

AGE *current-name related-SAS-file(s)*
</ <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

必須引数

current-name
プロシジャが名前を変更する SAS ファイルです。*current-name* は、*related-SAS-file(s)*の最初の名前を受け取ります。

related-SAS-file(s)
SAS ライブラリ内の 1 つ以上の SAS ファイルです。

オプション引数

ALTER=*alter-password*
AGE ステートメントで指定した、変更保護されている SAS ファイルに対する変更パスワードを与えます。AGE ステートメントは SAS ファイルの名前変更および削除を行うため、AGE ステートメントを使用するには変更アクセス権限が必要です。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。AGE ステートメントで指定するすべての SAS ファイルは、同一のメンバーの種類である必要があります。このオプションは、各 SAS ファイル名の後に、スラッシュの後にかっこで囲んで使用します。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメントで MEMTYPE=を指定しない場合、デフォルトは DATA となります。

参照項目: “処理するメンバーの種類の制限” (465 ページ)

詳細

AGE ステートメントは、*current-name* の名前を *related-SAS-files* の最初の名前に変更し、*related-SAS-files* の最初の名前を *related-SAS-files* の 2 番目の名前に変更し、以下、*related-SAS-files* の最後から 2 番目の SAS ファイルの名前が *related-SAS-files* の最後の名前に変更されるまで続きます。次に、AGE ステートメントは *related-SAS-files* の最後のファイルを削除します。

AGE ステートメントで指定した最初の SAS ファイルが SAS ライブラリ内にない場合、PROC DATASETS は AGE ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。AGE ステートメントは、*related-SAS-files* のいずれもエージングしません。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを使用します。

related-SAS-files のうちいずれも存在しない場合、プロシジャは SAS ログに警告メッセージを発行しますが、可能な SAS ファイルのエージングは続行します。

インデックスを含むデータセットをエージングする場合、インデックスは続行してそのデータセットに対応します。

エージングできるのは、世代グループ全体のみです。たとえば、データセット A と B に世代グループがある場合、次のステートメントが世代グループ B を削除し、世代グループ A を名前 B にエージング(名前変更)します。

```
age a b;
```

たとえば、データセット A の世代グループに 3 つの履歴バージョンがあり、データセット B の世代グループに 2 つの履歴バージョンがあるとします。この場合、A から B へのエージングがこの影響を受けます。

表 15.3 世代グループ

処理前の名前	バージョン	処理後の名前	バージョン
A	base	B	base
A	1	B	1

処理前の名前	バージョン	処理後の名前	バージョン
A	2	B	2
A	3	B	3
B	base	削除される	
B	1	削除される	
B	2	削除される	

APPEND ステートメント

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

デフォルト: BASE=データセットが SAS サーバー経由でアクセスされる場合や、APPEND ステートメントが処理を開始する時にその他のユーザーがデータセットを開いていない場合、BASE=データセットのデフォルトは CNTLLEV=MEMBER (メンバーレベルロック)となります。この動作が発生した場合、その他のユーザーはそのデータセットの処理時にファイルを更新できません。

制限事項: BASE=オプションに既存の CAS テーブルは指定できません。

要件 BASE=データセットは、更新処理をサポートする SAS ライブラリのメンバーである必要があります。

注: BASE=データセットでオプション DROP=、KEEP=、または RENAME=を使用する場合、オプション ONLY が APPEND 処理に影響し、追加された BASE=データセットの変数を変更しません。DROP=オプションと KEEP=オプションを使用してドロップされる、または保持されない変数は、追加された BASE=データセットに存在しています。RENAME=オプションを使用して名前が変更される変数は、追加された BASE=データセットにその元の名前で存在しています。

CAS エンジンは更新処理をサポートしていないため、BASE=に既存の CAS テーブルを指定することはできません。

ヒント: BASE=引数と DATA=オプションに対し、ほとんどのデータセットオプションを指定できます。ただし、データオプション DROP=、KEEP=または RENAME= を BASE=データセットに対して指定する場合、そのオプションは無視されます。グローバルステートメントも使用できます。

処理中にエラーが発生した場合、データセットは損傷としてマーク付けされ、次の REPAIR ステートメントで追加前の状態にリセットされます。データセットにインデックスがある場合、そのインデックスはオブザベーションごとに更新されませんが、最後に一度更新されます。(この動作は、APPENDVER=V6 が設定されていない限りバージョン 7 以降のものです。)

各パスワードと暗号化キーオプションは別個の行でコード化し、ログに適切に書き込まれるようにしてください。

例: “例 6: 2 つの SAS データセットを連結する” (508 ページ)

構文

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

必須引数

BASE=<*libref.*> *SAS-data-set*
オブザベーションの追加先となるデータセットを指定します。

libref
SAS データセットを含むライブラリを指定します。ライブラリ参照名を省略すると、デフォルトはプロシジャ入力ライブラリのライブラリ参照名となります。PROCAPPEND を使用している場合、ライブラリ参照名のデフォルトは WORK または USER のいずれかになります。

SAS-data-set
SAS データセットを指定します。APPEND ステートメントは、この名前の既存データセットが見つからない場合、ライブラリに新しいデータセットを作成します。つまり、APPEND ステートメントを使用して、BASE=引数で新しいデータセット名を指定することによりデータセットを作成できます。

新しいデータセットを作成する、既存データセットに追加するにかかわらず、BASE=データセットはすべての追加操作の後の最新の SAS データセットとなります。

別名 OUT=

例 “例 6: 2 つの SAS データセットを連結する” (508 ページ)

オプション引数

APPENDVER=V6
オブザベーションを BASE=データセットに追加するためのバージョン 6 の動作を使用します。一度に 1 つのオブザベーションが追加されます。バージョン 7 からは、パフォーマンスを向上させるため、データセットの処理後にすべてのオブザベーションが追加されるようにデフォルトの動作が変わりました。

参照項目: “インデックス付きデータセットへの追加 — 高速追加メソッド” (390 ページ)

DATA=<*libref.*> *SAS-data-set*
BASE=引数で指定される SAS データセットの最後に追加するオブザベーションを含む SAS データセットを指定します。

libref
SAS データセットを含むライブラリを指定します。ライブラリ参照名を省略すると、デフォルトはプロシジャ入力ライブラリのライブラリ参照名となり

ます。DATA=データセットは、SAS ライブラリのものとなります。データセットがプロシジャ入力ライブラリ以外のライブラリに存在する場合、2 レベル名を使用する必要があります。

SAS-data-set
SAS データセットを指定します。APPEND ステートメントはこの名前の既存データセットが見つからなかった場合、処理を停止します。

別名 NEW=

デフォルト SAS ライブラリからの、最も新しく作成された SAS データセット

参照項目: “世代グループを使用した追加” (394 ページ)

例 “例 6: 2 つの SAS データセットを連結する” (508 ページ)

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

参照項目: “AES 暗号化データセットの追加” (389 ページ)

FORCE
DATA=データセットに次の基準のうちいずれかを満たす変数が含まれている場合、APPEND ステートメントによるデータセットの連結を強制的に行います。

- BASE=データセットにありません。
- BASE=データセットの変数と同じ種類がありません。
- BASE=データセットの変数より長いです。

ヒント GENNUM=データセットオプションを使用して、世代グループの特定のバージョン間での追加が可能です。次に、いくつかの例を示します。

```
/* appends historical version to base A */
proc datasets;
  append base=a
    data=a (gennum=2);

/* appends current version of A to historical version */
proc datasets;
  append base=a (gennum=1)
    data=a;
```

参照項目: “変数が異なるデータセットへの追加” (391 ページ)および“属性が異なる変数を含むデータセットへの追加” (392 ページ)

例 “例 6: 2 つの SAS データセットを連結する” (508 ページ)

GETSORT
ソートインジケーターを DATA=データセットから BASE=データセットにコピーします。ソートインジケーターは、次の基準が満たされる場合に PROC SQL の PROC SORT 句または ORDERBY 句によって作成されます。

- BASE=データセットは、次の基準を満たしていなければなりません。
  - SAS バージョン 7 以降

- □ オブザベーションが含まれていない
- □ ソートインジケーターを受け入れる

注意
DATA=データセットが並べ替えられない場合でも、BASE=データセット上の既存するソートインジケーターが警告なしで上書きされます。

- ■ DATA=データセットは、次の基準を満たしていなければなりません。
- □ PROC SORT によって作成されたソートインジケーターを含む
- □ BASE=データセットと同じデータ表現

制限事項 BASE=データセットが監査証跡と関連付けられている場合、GETSORT オプションはそのデータセットに影響しません。この制約は、APPEND プロセスが続行される間、出力における WARNING の原因となります。

DATA=データファイルにドロップ、保持、または名前変更された変数がある場合、GETSORT オプションはそのデータセットに影響しません。

NOWARN
FORCE オプションと使用した場合、異なる変数を持つ 2 つのデータセットを連結する時に、警告を非表示にします。

詳細

APPEND ステートメントの代わりに APPEND プロシジャを使用

APPEND プロシジャと PROC DATASETS の APPEND ステートメントの違いは、BASE=引数と DATA=引数のライブラリ参照名のデフォルト値のみです。PROC APPEND のデフォルトは WORK または USER です。APPEND ステートメントのデフォルトは、プロシジャの入力ライブラリのライブラリ参照名です。

WHERE=データセットオプションと DATA=データセットを使用して、追加されるオブザベーションを制限することができます。同様に、WHERE ステートメントを使用すると、DATA=データセットからのオブザベーションを制限することができます。WHERE ステートメントは、BASE=データセットに影響しません。

注意
既存の BASE=データセットに WHERE オプションが使用されている場合、WHEREUP=オプションが YES に設定されている場合にのみ使用されます。 BASE=データセットが存在しない場合、WHERE オプションを有効にするのに WHEREUP=オプションは必要ありません。

並べ替えたデータセットの追加

次のガイドラインを使用して、並べ替え済みデータセットを追加し、並べ替えを保持できます。

- ■ DATA=データセットと BASE=データセットには、SORT プロシジャからのソートインジケーターが含まれています。

- DATA=データセットと BASE=データセットは、同一の変数を使用して並べ替えられます。
- DATA=データセットから追加されたオブザベーションは、BASE=データセットの並べ替え順序に準拠しています。

BASE=データセットからのソートインジケーターは保持されます。

ブロック I/O メソッドを使用して追加

注: CAS テーブルを SAS データセットに追加する場合、ブロック I/O メソッドは使用できません。

ブロック I/O メソッドは、一度に 1 オブザベーションずつ追加するのではなく、データブロックを追加するために使用されます。大きなデータセットを追加する場合、このメソッドによりパフォーマンスが向上します。SAS によって、ブロック I/O メソッドを使用するかどうかが決定されます。すべてのデータセットでブロック I/O メソッドが使用できるわけではありません。APPEND ステートメントと Base SAS エンジンによって設定された制約があります。

使用中の追加メソッドに関する情報を SAS ログに表示するには、MSGLEVEL=システムオプションを次のように指定できます。

```
options msglevel=i;
```

ブロック I/O メソッドが使用されていない場合、SAS ログに次のメッセージが出力されます。

INFO: データセットブロック I/O は、下記の理由により使用できません。

APPEND ステートメントがブロック I/O を使用しないと判断した場合、次の説明のうちの 1 つが SAS ログに出力されます。

INFO: - データセットは異なるエンジンを使用し、異なる変数を持っているか、または異なる可能性がある属性を持っています。

INFO: - WHERE 句があります。

INFO: - メンバーレベルロックがありません。

INFO: - OBS オプションがアクティブです。

INFO: - FIRSTOBS オプションがアクティブです。

ブロック I/O メソッドを使用されないと Base SAS エンジンが判断した場合、次のいずれかの説明が SAS ログに書き込まれます。

INFO: - 参照一貫性制約が存在します。

INFO: - クロス環境データアクセスが使用されています。

INFO: - ファイルが圧縮されています。

INFO: - ファイルに、中断されない監査ファイルが含まれています。

追加されるオブザベーションの制限

追加されるオブザベーションを制限するため、WHERE=データセットオプションと DATA=データセットを使用できます。同様に、DATA=データセットからのオブザベ

ーションを制限するため、WHERE ステートメントを使用できます。WHERE ステートメントは、BASE=データセットに影響しません。WHERE=データセットオプションと BASE=データセットを使用する場合、WHERE=は影響しません。

注意
既存する BASE=データセットの場合: BASE=データセットに WHERE ステートメントがある場合、WHEREUP=オプションが YES に設定されている場合にのみ有効です。

注意
存在しない BASE=データセットの場合: 存在しない BASE=データセットに WHERE ステートメントがある場合、WHEREUP オプション設定に関係なく、WHERE ステートメントを使用します。

注: WHERE=データセットオプションを使用して、データセットをそれ自体に追加することはできません。

SET ステートメントと APPEND ステートメント間の選択

DATA ステップで SET ステートメントを使用し、2 つのデータセットを連結する場合、両方のデータセットのすべてのオブザベーションを 処理して新しく作成する必要があります。APPEND ステートメントは元のデータセットのデータ処理を行わずに、新しいオブザベーションを元のデータセットの最後に直接追加します。次のうちいずれかが発生した場合は、SET ステートメントよりも APPEND ステートメントを使用したほうが効率的です。

- BASE=データセットが大きい。
- BASE=データセットのすべての 変数の長さと種類が DATA=データセットの変数と同じで、すべての変数が両方のデータセットに存在する場合。

  注: CONTENTS ステートメントを使用して、変数の長さと種類を確認できます。

オブザベーションを(データをジャーナル型のデータセットに継続的に追加している本稼働プログラムなどで)SAS データセットに頻繁に追加する場合、APPEND ステートメントは非常に便利です。

パスワード保護されている SAS データセットの追加

APPEND ステートメントを使用するため、DATA=データセットへの読み取りアクセス権限と BASE=データセットへの書き込みアクセス権限が必要です。アクセス権を得るには、APPEND ステートメント内でデータセット名の直後にかっこで囲み READ=および WRITE=データセットを使用します。パスワード保護されているデータセットを追加する場合、次のガイドラインを使用します。

- APPEND ステートメントで DATA=データセットに対する読み取りパスワードを与えない場合、デフォルトでプロシジャが PROC DATASETS ステートメントの DATA=データセットに対する読み取りパスワードを検索します。ただし、プロシジャは PROC DATASETS ステートメントの BASE=データセットに対する書き込みパスワードは検索しません。そのため、APPEND ステートメントの BASE=データセットに対する書き込みパスワードを指定する必要があります。

- BASE=データセットが読み取り保護だけされている場合、APPEND ステートメントでその読み取りパスワードを指定する必要があります。

AES 暗号化データセットの追加

2 つの AES 暗号化データセットを使用する場合、データセットにアクセスするために ENCRYPTKEY=データセットオプションを使用する必要があります。次は、ENCRYPTKEY=オプションを両方のデータセットに使用する例です。

```
proc datasets;
  append base=a (encryptkey=secret)
    data=a (encryptkey=jlgh56);
run;
```

暗号化されていないデータセットに追加する場合、DATA=データセットで ENCRYPTKEY=を指定する必要があります。なお、BASE=データセットエンジンで AES 暗号化がサポートされていない場合でも、データセットに追加できます。追加されたデータは暗号化されません。

```
proc datasets;
  append base=a
    data=a (encryptkey=key-value);
run;
```

AES 暗号化の詳細については、*"AES or AES2" (SAS データセットオプション: リファレンス)*を参照してください。ENCRYPTKEY=データセットオプションの詳細については、*"ENCRYPTKEY= データセットオプション" (SAS データセットオプション: リファレンス)*を参照してください。

圧縮データセットへの追加

圧縮されている SAS データセットを連結できます。BASE=データセット、DATA=データセットのいずれか、またはその両方を圧縮できます。BASE=データセットで削除されたオブザベーションからのスペースの 再利用が可能な場合、APPEND ステートメントはオブザベーションを BASE=データセットの真ん中に挿入し、利用可能なスペースを 利用します。

BASE= SAS データセットと DATA=データセットまたはテーブルのどちらか一方、あるいは両方を圧縮できます。BASE=データセットで削除された行からのスペースの再利用が可能な場合、APPEND ステートメントは BASE=データセットの真ん中に行を挿入します。

COMPRESS=データセットと REUSE=データセットおよびシステムオプションの詳細については、*SAS データセットオプション: リファレンス*および *SAS システムオプション: リファレンス*を参照してください。

PROC APPEND は、すべての新しいオブザベーションを BASE=データセットの最後に配置し、スペースの再利用を試みます。これは、SAS7 で追加された高速追加動作の一部です。BASE データセットへの書き込みとインデックスの更新のパフォーマンスを向上させるために、高速追加動作が追加されました。スペースを 再利用する場合は、PROC DATASETS APPEND ステートメントに APPENDVER=v6 オプションを追加する必要があります。PROC DATASETS APPEND は、新しいオブザベーションが、削除されたオブザベーションによって 以前に占有されていたスペースに 収まる場合にのみ、スペースを 再利用します。

```
proc append base=libref.data-set data=librefdata-set appendver=v6;
run;
```

属性が異なる変数を含むデータセットへの追加

BASE= SAS データセットの変数の属性が DATA=データセットまたはテーブルの変数と異なる場合は、BASE=データセットの属性が優先されます。

DATA= SAS データセットまたはテーブルの出力形式が BASE= SAS データセットのものと異なる場合、BASE=データセットの出力形式が使用されます。ただし、BASE=データセットの出力形式と一貫性を取るために、DATA=データセットまたはテーブルのデータは変換されません。結果は、正しくないように見えるデータになります。警告メッセージがログに表示されます。

次のいずれかが発生した場合、FORCE オプションを使用します。

- DATA= SAS データセットまたはテーブルの変数が BASE=データセットの変数よりも長い場合
- 同じ変数が一方のデータセットでは文字変数で他方では数値変数の場合

FORCE を使用した結果は、次のとおりです。

- BASE= SAS データセットの変数の長さが優先します。DATA=データセットまたはテーブルの値は切り捨てられ、BASE=データセットで指定されている長さに合うように調整されることがあります。
- BASE=データセットの変数の種類が優先します。APPEND ステートメントは、間違った種類の値(DATA=データセットまたはテーブルの変数に対するすべての値)を欠損値と置き換えます。

インデックス付きデータセットへの追加 — 高速追加メソッド

バージョン 7 から、パフォーマンスを向上させるためにインデックス付きデータセットへの追加の動作が変わりました。

- バージョン 6 では、インデックス付きデータセットに追加する際に、インデックスは追加された各オブザベーションに対して更新されました。インデックス更新は無作為になる傾向にあります。そのため、ディスク I/O が高くなった可能性があります。
- 現在では、インデックスはすべてのオブザベーションがデータセットに追加されるまで更新されません。追加後、オブザベーションが内部的に並べ替えられ、データがインデックスに逐次挿入されます。この動作によりほとんどのディスク I/O が減少し、追加メソッドがさらに早くなります。

次の要件を満たしている場合、高速追加メソッドはデフォルトで使用されます。その他の場合、バージョン 6 メソッドが使用されます。

- BASE=データセットは、メンバーレベルロックが可能です。CNTLLEV=が記録のために設定されている場合、高速追加メソッドは使用されていません。
- BASE=データセットには、参照一貫性制約が含まれていません。
- BASE=データセットにはクロス環境データアクセス(CEDA)機能を使用してアクセスしません。
- BASE=データセットは、WHERE=データセットオプションを使用していません。

使用中の追加メソッドに関する情報を SAS ログに表示するには、MSGLEVEL=システムオプションを次のように指定できます。

```
options msglevel=i;
```

高速追加メソッドが使用されている場合にメッセージが表示されるか、高速追加メソッドが使用されていない理由に関してメッセージが表示されます。

現在の追加メソッドは、インデックスによって決定される制約に関係なく、オブザベーションを BASE=データセットに内部的に追加します。たとえば、UNIQUEF オプションによって作成されたインデックスを含む変数の場合、その値はインデックスが更新されるまでその一意性について検証されません。一意でない値が検出されると、問題のあるオブザベーションはデータセットから削除されます。オブザベーションが追加されると、そのうちのいくつかが後で削除される場合があります。

簡単な例として、BASE=データセットに変数 ID に対し UNIQUE インデックスを含む、1 から 10 までの番号の付いた 10 のオブザベーションが含まれているとします。1 から 5 までの 5 つのオブザベーションを含むデータセットを追加し、オブザベーション 3 と 4 に ID に対する同一の値が含まれているとします。次のアクションが発生します。

1 オブザベーションが追加されると、BASE=データセットには 1 から 15 までの番号の付いた 15 のオブザベーションが含まれます。

2 ID に対するインデックスが更新され、値が検証され、オブザベーション 13 と 14 に ID に対する同一の値が含まれるよう指定されます。

3 BASE=データセットからオブザベーションのいずれかが削除され、その結果、1 から 15 までの番号の付いた 14 のオブザベーションとなります。たとえば、オブザベーション 13 が削除されます。どのオブザベーションが削除されるかは予測できません。内部並べ替えではどちらかのオブザベーションが最初に識別される場合があるためです。(バージョン 6 では、オブザベーション 13 が追加され、オブザベーション 14 が削除されるということが予測可能でした。)

現在の動作(オブザベーションが削除される)を希望しない場合、または追加されるオブザベーションの予測を希望する場合は、APPENDVER=V6 オプションを指定して、バージョン 6 追加メソッドを要求してください。

```
proc datasets;
  append base=a data=b appendver=v6;
run;
```

注: バージョン 6 では、インデックスを削除してから追加後に再度作成することにより、パフォーマンスの向上が可能でした。現在のメソッドではその必要がありません。ただし、パフォーマンスはデータの性質によって異なります。

変数が異なるデータセットへの追加

DATA=データセットに BASE=データセットにない変数が含まれている場合、APPEND ステートメントで FORCE オプションを使用し、2 つのデータセットの連結を強制的に行います。APPEND ステートメントは余剰の変数をドロップし、警告メッセージを発行します。NOWARN オプションを使用して、警告メッセージを非表示にできます。

BASE=データセットに DATA=データセットにない変数が含まれている場合、APPEND ステートメントはデータセットを連結しますが、DATA=データセットからのオブザベーションには、DATA=データセットになかった変数に対する欠損値が含まれます。この場合、FORCE オプションは不要です。

BASE=データセットでオプション DROP=、KEEP=、または RENAME=を使用する場合、オプション ONLY が APPEND 処理に影響し、追加された BASE=データセットの変数を変更しません。DROP=オプションと KEEP=オプションを使用してドロップされる、または保持されない変数は、追加された BASE=データセットに存在しています。RENAME=オプションを使用して名前が変更される変数は、追加された BASE=データセットにその元の名前で存在しています。

属性が異なる変数を含むデータセットへの追加

BASE=データセットの変数の属性が DATA=データセットの変数と異なる場合、BASE=データセットの属性が優先されます。

DATA=データセットの SAS 出力形式が BASE=データセットのものと異なる場合、BASE=データセットの SAS 出力形式が使用されます。ただし、BASE=データセットの SAS 出力形式と一貫性を取るために DATA=データセットのデータは変換されません。結果は、正しくないように見えるデータになります。警告が SAS ログに表示されます。次の例に、異なる SAS 出力形式を使用したデータの追加について示します。

```
data format1;
  input Date date9.;
  format Date date9.;
datalines;
24sep1975
22may1952
;

data format2;
  input Date datetime20.;
  format Date datetime20.;
datalines;
25aug1952:11:23:07.4
;

proc append base=format1 data=format2;
run;
```

次のメッセージが SAS ログに表示されます。

```
NOTE: Appending WORK.FORMAT2 to WORK.FORMAT1.
WARNING: Variable Date has format DATE9. on the BASE data set
   and format DATETIME20. on the DATA data set. DATE9. used.
NOTE: There were 1 observations read from the data set WORK.FORMAT2.
NOTE: 1 observations added.
NOTE: The data set WORK.FORMAT1 has 3 observations and 1 variables.
```

DATA=データセットの変数の長さが BASE=データセットのものより長い場合、または同じ変数が 1 つのデータセットでは文字変数で、もう 1 つのデーセットでは数値変数の場合、FORCE オプションを使用します。FORCE を使用した結果は、次のとおりです。

- BASE=データセットの変数の長さが優先します。DATA=データセットの値は切り捨てられ、BASE=データセットで指定されている長さに合うように調整されます。
- BASE=データセットの変数の種類が優先します。APPEND ステートメントは、正しくない種類の値(DATA=データセットの変数に対するすべての値) を欠損値と置き換えます。

注: 文字変数のトランスコーディング属性が BASE=データセットと DATA=データセットで逆の場合(一方が YES、もう一方が NO など)、警告が発行されます。トランスコーディング属性を決定するには、各データセットに対し CONTENTS プロシジャを使用します。トランスコーディング属性は、ATTRIB ステートメントの TRANSCODE= オプション、または PROC SQL の TRANSCODE=列修飾子によって設定します。

一貫性制約を含むデータセットの追加

DATA=データセットに一貫性制約が含まれ、BASE=データセットが存在しない場合、APPEND ステートメントは一般制約をコピーします。参照制約はコピーされません。BASE=データセットが存在する場合、APPEND アクションはオブザベーションのみコピーします。

一貫性制約またはインデックスを含むザベーションデータセットへの追加

次は、既存データセットからインデックスと一貫性制約を取得し、別のデータセットに適用する際の簡単な方法です。Work.Model がすでに存在し、オブザベーションがないことにご注意ください。

```
Proc contents data=sashelp.zipcode out2=work.ConstraintsIndexes;
run;

proc sql noprint;
  select recreate into :macroVarCIs separated by ' '
       from work.ConstraintsIndexes;
quit;

data work.test_zipcode;
  set sashelp.zipcode;
run;

proc datasets lib=work nolist;
  modify test_zipcode;
  &macroVarCIs
quit;

proc contents data=work.test_zipcode;
run;
```

世代グループを使用した追加

GENNUM=データセットオプションを使用して、世代グループの特定のバージョンに追加します。例を次に示します。

表 15.4　世代グループを使用した追加

SAS ステートメント	結果
proc datasets; append base=a data=b(gennum=2);	履歴バージョン B#002 を base A に追加
proc datasets; append base=a(gennum=2) data=b(gennum=2);	履歴バージョン B#002 を履歴バージョン A#002 に追加

システム障害

プロシジャの実行時にシステム障害またはその他の種類の障害が発生した場合、追加操作が正常に行われないことがあります。オブザベーションの一部、またはすべてが BASE=に追加されない可能性があります。また、BASE=データセットが破損することがあります。また、BASE=データセットが破損することがあります。
APPEND 操作はかわりに更新を実行します。つまり、オブザベーションの追加を開始する前に元のデータセットのコピーを作成しません。元のオブザベーションを復元する場合、基本データファイルの監査証跡を開始し、更新前のオブザベーションの保存を選択できます。次に、DATA ステップを書き込み、元のオブザベーションをデータファイルに抽出して再度適用できます。監査証跡の開始の詳細については、“AUDIT ステートメント” (395 ページ)を参照してください。

ATTRIB ステートメント

出力形式、入力形式またはラベルを MODIFY ステートメントで指定した SAS データセットの変数と関連付けます。

制限事項: ATTRIB ステートメントは MODIFY RUN グループに記述する必要があります。

注: ATTRIB ステートメントは、CONTENTS ステートメント出力に影響を与えません。CONTENTS は、実際のメンバーに関するラベル、入力形式、および出力形式を報告します。

例: AUDIT ステートメント (488 ページ)

構文

ATTRIB *variable-list(s) attribute-list(s)*;

必須引数

variable-list(s)
属性と関連付ける変数を指定します。SAS で許可される形式の変数をリストできます。

attribute-list(s)
variable-list に割り当てる属性を 1 つ以上指定します。ATTRIB ステートメントで次の属性のうち 1 つ以上を指定します。

FORMAT=*format*
出力形式を *variable-list の変数と関連付けます。*

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
入力形式を *variable-list の変数と関連付けます。*

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
ラベルを *variable-list の変数と関連付けます。*

詳細

DATASETS プロシジャ内で、ATTRIB ステートメントは MODIFY RUN グループで使用する必要があり、オプション FORMAT、INFORMAT、LABEL のみ使用できます。ATTRIB ステートメントは、キーワード_ALL_を使用した、データセット内のすべての変数ラベル、出力形式または入力形式を削除、変更するための最も簡単な方法です。例については、“例 1: データセットのすべてのラベルと 出力形式の削除” (488 ページ)を参照してください。

一部の属性を削除または変更している場合、LABEL ステートメント (439 ページ)、FORMAT ステートメント (429 ページ)、および INFORMAT ステートメント (437 ページ)を使用するとさらに簡単です。

AUDIT ステートメント

監査ファイルへのイベントの記録を開始、制御し、監査ファイルのイベントの記録を中断、再開、終了します。

ヒント: AUDIT ステートメントは、監査証跡を開始するかどうか、監査ファイルのイベントの記録を中断、再開、終了するかどうかによって、2 種類あります。

PROC DATASETS MODIFY ステートメントを使用して、出力形式、入力形式などの属性をデータファイルのユーザー変数に対し定義できます。

構文

AUDIT *SAS-file* <(*SAS-password* <ENCRYPTKEY=*key-value*> <GENNUM=*integer*>)>;

INITIATE <AUDIT_ALL=NO | YES>;

LOG<ADMIN_IMAGE=YES | NO>
<BEFORE_IMAGE=YES | NO>
<DATA_IMAGE=YES | NO>
<ERROR_IMAGE=YES | NO>;

<SUSPEND | RESUME | TERMINATE;>

<USER_VAR*variable(s)*>;

必須引数

SAS-file
監査するプロシジャ入力ライブラリの SAS データファイルを指定します。

オプション引数

SAS-password
SAS データファイルのパスワードを指定します(存在する場合)。かっこは必須です。

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

GENNUM=*integer*
アクション SUSPEND、RESUME または TERMINATE が世代ファイルの監査証跡で実行されるように指定します。世代ファイルで監査証跡を開始できません。GENNUM=に有効な値は *integers* で、世代ファイルの特定バージョンを参照する番号です。正の数を指定すると、データセット名に追加される特定の世代番号の絶対参照になります(つまり、**gennum=2** は MYDATA#002 を指定することになります)。負の数を指定すると、最新から最古への基本バージョンに相対する履歴バージョンの相対参照になります(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。デフォルトの 0 を指定すると、基本バージョンが参照されます。かっこは必須です。

制限事項 GENNUM=オプションは、INITIATE ステートメントまたは USER_VAR ステートメントの前に指定できません。

詳細

監査ファイルの作成

注: 監査証跡の開始は、BaseSAS エンジンでのみ可能です。

次の例では、監査ファイル MyLib.MyFile.audit を作成し、更新をデータファイル MyLib.MyFile.data に書き込み、すべての利用可能なレコードイメージを保存します。

```
proc datasets library=mylib;
  audit myfile (alter=password);
  initiate;
run;
```

次の例では、同じ監査ファイルを作成しますが、エラーレコードイメージのみを保存します。

```
proc datasets library=mylib;
  audit myfile (alter=password);
  initiate;
    log data_image=no
      before_image=no
      data_image=no;
run;
```

次の例では、AUDIT_ALL=YES を使用して監査ファイルを開始します。

```
proc datasets lib=mylib;  /* all audit image types will be logged
                     and the file cannot be suspended */
   audit myfile (alter=password);
   initiate audit_all=yes;
quit;
```

次の例では、監査ファイルを終了します。

```
proc datasets lib=mylib;
   audit myfile (alter=password);
   terminate;
quit;
```

AUDIT ステートメントは、ファイルの*監査実行グループ*を開始します。ファイルの複数の監査実行グループは、次の方法でサブミット可能です。

- 同一の PROC DATASETS ステップで
- 別の PROC DATASETS ステップで
- 別の計算サーバーセッションで

すべての監査ファイル関連ステートメント(INITIATE、USER_VAR、LOG、SUSPEND、RESUME、TERMINATE)の前に、それらが適用されるファイルを識別する AUDIT ステートメントを配置する必要があります。

INITIATE ステートメントは監査ファイルを作成し、最初の AUDIT ステートメントでサブミットされる必要があります。USER_VAR、LOG、SUSPEND、RESUME、TERMINIATE などのその他の監査関連ステートメントは、INITIATE ステートメントがサブミットされるまで監査ファイルに有効になりません。世代データセットで監査ファイルを開始できますが、これは世代グループの最新世代である必要があります。GENNUM を指定して最新世代を識別できません。最新世代はデフォルトで取得されます。

次は、指定ファイルに対する最初の AUDIT RUN グループの AUDIT ステートメントの例です。

AUDIT file <(*SAS-password*)>;

監査ファイルが一度開始されると、そのファイルの INITIATE ステートメントの後に続く AUDIT ステートメントは GENNUM を指定できます。

AUDIT file <(<*SAS-password*><GENNUM=*integer*>)>;

USER_VAR ステートメントは、同じ AUDIT 実行グループの INITIATE ステートメントの直後に指定する必要があります。

CHANGE ステートメント

同一の SAS ライブラリの 1 つ以上の SAS ファイルの名前を変更します。

例: “例 2: SAS ファイルの操作” (494 ページ)

構文

CHANGE *old-name-1=new-name-1*
<*old-name-2=new-name-2* ...>
</ <ENCRYPTKEY=*key-value*>
<ALTER=*alter-password*> <GENNUM=ALL | *integer*> <MEMTYPE=*member-type*>>;

必須引数

old-name=new-name
入力データライブラリ内の SAS ファイルの名前を変更します。 *old-name* は、入力データライブラリにある既存の SAS ファイルの名前である必要があります。

例 “例 2: SAS ファイルの操作” (494 ページ)

オプション引数

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。このオプションが必要なのは、RELATIVE GENNUM が指定されている場合に限られます。詳細については、“ライブラリコンテンツと AES 暗号化” (404 ページ)を参照してください。

ALTER=*alter-password*
CHANGE ステートメントで指定した、変更保護されている SAS ファイルに対する変更パスワードを与えます。CHANGE ステートメントは SAS ファイル名を変更するため、*new-name* に CHANGE ステートメントを使用するために変更アクセス権限が必要です。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

GENNUM=ALL | *integer*
世代データセットの処理を制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。次のリストに、有効値を示します。

ALL
0
世代グループの基本バージョンとすべての 履歴バージョンを参照します。

integer
世代グループの特定バージョンを参照します。正の数を指定すると、データセット名に追加される特定の世代番号の絶対参照になります(つまり、**gennum=2** は MYDATA#002 を指定することになります)。負の数を指定すると、最新から最古への基本バージョンに相対する履歴バージョンの相対参照になります(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。

たとえば、次のステートメントはバージョン名 A#003 を base B に変更します。

```
proc datasets;
  change A=B / gennum=3;

proc datasets;
  change A(gennum=3)=B;
```

次の CHANGE ステートメントではエラーが発生します。

```
proc datasets;
  change A(gennum=3)=B(gennum=3);
```

参照項目: “世代データセットの処理制限” (463 ページ).

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

別名 MTYPE=
MT=

デフォルト PROC DATASETS ステートメントで MEMTYPE=を指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目: “処理するメンバーの種類の制限” (465 ページ)

詳細

CHANGE ステートメントは、CHANGE ステートメントで変更をリストする順序ではなく、*old-names* がディレクトリリストで発生する順序によって名前を変更します。

old-name の SAS ファイルが SAS ライブラリに存在しない場合、PROC DATASETS は CHANGE ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを使用します。

インデックスを含むデータセットの名前を変更する場合、インデックスは続行してそのデータセットに対応します。

CONTENTS ステートメント

1 つ以上の SAS データセットのコンテンツを記述し、SAS ライブラリのディレクトリを出力します。

制限事項: PROC CONTENTS はオブザベーションを処理しないため、出力に作用する WHERE オプションは使用できません。

注: ATTRIB ステートメントは、CONTENTS ステートメント出力に影響を与えません。CONTENTS は、実際のメンバーに関するラベル、入力形式、および出力形式を報告します。

SAS/ACCESS LIBNAME エンジンで CONTENTS ステートメントを使用する場合は、“SAS/ACCESSLIBNAME エンジンを使用する場合の DATASETS プロシジャ出力の違い” (369 ページ)を参照してください。

ヒント: DATA=、OUT=および OUT2=オプションではデータセットオプションが使用できます。グローバルステートメントも使用できます。

例: “例 5: SAS データセットを記述” (505 ページ)

構文

CONTENTS <*options*>;

オプション引数

CENTILES
インデックス付き変数に関するパーセント点情報を出力します。

CENTILES オプションが選択され、インデックスがデータセット上に存在する場合、次の追加フィールドが PROC CONTENTS のデフォルトレポートに出力されます。追加フィールドは、インデックスが単一または複合かどうかによって異なります。

#	データセット上のインデックスの数。
インデックス	インデックス名。
Update Centiles	インデックス付き変数に対し CENTILES の前に変更が必要なデータ値のパーセントが自動的に更新されます。
現在の更新率	CENTILES が更新されてから更新されたインデックスのパーセント。
一意の値の数	一意のインデックス付き値の数です。
Variables	インデックスの作成に使用される変数の名前。パーセント点情報は、変数の下にリストされます。

DATA=*SAS-file-specification*
ライブラリ全体またはライブラリ内の特定の SAS データセットを指定します。*SAS-file-specification* は、次の形式のうちいずれかが可能です。

<libref.>SAS-data-set
処理する SAS データセットを 1 つ指定します。ライブラリ参照名のデフォルトは、プロシジャ入力ライブラリのライブラリ参照名です。たとえば、プロシジャ入力ライブラリから SAS データセット HtWt のコンテンツを取得するには、次の CONTENTS ステートメントを使用します。

```
contents data=HtWt;
```

世代グループから特定バージョンのコンテンツを取得するには、次の CONTENTS ステートメントにあるように GENNUM=データセットオプションを使用します。

```
contents data=HtWt(gennum=3);
```

<libref.>_ALL_
MEMTYPE==オプションに指定した種類(複数可)のすべての SAS データセットに関する情報を提供します。*libref* は、SAS ライブラリを参照します。ライブラリ参照名のデフォルトは、プロシジャ入力ライブラリのライブラリ参照名です。

- _ALL_キーワードを使用している場合、SAS ライブラリの読み取り保護されているすべての SAS データセットへの読み取りアクセス権限が必要です。
- DATA=_ALL_は、SAS ライブラリに含まれる SAS ファイルのリストを自動的に出力します。SAS ビューの場合、そのビューと関連付けられているすべてのライブラリ参照名は、リストに対して処理されるように現在のセッションで割り当てられる必要があります。

デフォルト　SAS ライブラリからの、ジョブまたはセッションで最も新しく作成されたデータセット。

ヒント　DATA=オプションで読み取り保護されているデータセットを指定し、読み取りパスワードを与えない場合、デフォルトでプロシジャが PROC DATASETS ステートメントで読み取りパスワードを検索します。ただし、DATA=オプションを指定せず、デフォルトのデータセット(セッションで最後に作成されたもの)が読み取り保護されている場合、プロシジャは PROC DATASETS ステートメントで読み取りパスワードを検索しません。

例　“例 5: SAS データセットを記述” (505 ページ)

DETAILS | NODETAILS
オブザベーションの数、変数の数、インデックスの数、およびデータセットラベルに関する情報を出力に含めます。DETAILS には出力に情報の追加列が含まれますが、DIRECTORY も指定されている場合だけです。

デフォルト　DETAILS も NODETAILS も指定されていない場合、デフォルトは次のようになります。CONTENTS プロシジャの場合、デフォルトはシステムオプション設定の NODETAILS です。CONTENTS ステートメントの場合、デフォルトは PROC DATASETS ステートメントで指定されている値で、これもシステムオプション設定となります。

参照項目: PROC DATASETS ステートメント (376 ページ) の Optional Argument セクションの追加列の説明

DIRECTORY
指定した SAS ライブラリのすべての SAS ファイルのリストを出力します。DETAILS も指定されている場合、DIRECTORY を使用すると DETAILS | NODETAILS (401 ページ)に記述されている追加列が出力されます。

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。詳細については、"ライブラリコンテンツと AES 暗号化" (404 ページ)を参照してください。

FMTLEN
入力形式または出力形式の長さを出力します。入力形式または出力形式の長さを変数と関連付けるときに指定しない場合、長さは FMTLEN オプションを使用しない限り CONTENTS ステートメントの出力に表示されません。長さは出力データセットの FORMATL 変数または INFORML 変数にも表示されます。

MEMTYPE=(*member-type(s)*)
処理を 1 つ以上のメンバーの種類に制限します。CONTENTS ステートメントは、メンバーの種類 DATA、VIEW および ALL(DATA と VIEW が含まれます)に対してのみ出力を生成します。

CONTENTS ステートメントの MEMTYPE=は、DATASETS プロシジャのその他の大部分のステートメントの MEMTYPE=と次の点が異なります。

- スラッシュはオプションの前に指定できません。
- MEMTYPE=オプションをかっこで囲み、その影響をその直前の SAS ファイルにのみ制限することはできません。

MEMTYPE=は結果として、DATA=メンバーが置かれているライブラリのディレクトリとなります。ただし、MEMTYPE=は、_ALL_キーワードが DATA=オプションで使用されない限り、コンテンツが表示されるメンバーの種類を制限しません。たとえば、次のステートメントは、メンバーの種類が DATA の SAS データセットのみのコンテンツを生成します。

```
proc datasets memtype=data;
  contents data=_all_;
run;
```

別名 MTYPE=
MT=

デフォルト DATA

NODS
DATA=オプションで_ALL_を指定する際に、個々のファイルの内容を出力しません。CONTENTS ステートメントは、SAS ライブラリディレクトリのみを出力します。DATA=オプションで SAS データセットを 1 つだけ指定する際に、NODS オプションは使用できません。

NOPRINT
CONTENTS ステートメントの出力を行いません。

ORDER=COLLATE | CASECOLLATE | IGNORECASE | VARNUM

COLLATE 変数のリストを大文字名、次に小文字名で始まるアルファベット順で出力します。

CASECOLLATE 変数の一覧を、大文字と小文字が混在する名前と数値が含まれている場合でも、アルファベット順で出力します。

IGNORECASE 変数のリストを大文字と小文字に関係なくアルファベット順で出力します。

VARNUM VARNUM オプションと同じです。

注 ORDER=オプションは、OUT=と OUT2=データセットの順序には影響しません。

参照項目: “VARNUM” (404 ページ)

例 ORDER=のデフォルトと 4 つのオプションを比較するには、“例 3: ORDER=オプションの使用” (296 ページ)を参照してください。

OUT=*SAS-data-set*

出力 SAS データセットを指定します。

ヒント OUT=は、ステートメントからの出力を非表示にしません。出力を非表示にする場合、NOPRINT オプションを使用する必要があります。

参照項目: “OUT=データセット” (480 ページ)には、OUT=データセットの変数の説明が記載されています。

OUT2=*SAS-data-set*

インデックスと一貫性制約に関する情報を含む出力データセットを指定します。

注 OUT2=*PermanentLibrary*_ALL_オプションを PROC CONTENTS または PROC DATASETS 内で CONTENTS ステートメントとともに使用する場合、REPLACE=YES データセットオプションまたは REPLACE システムオプションも設定する必要があります。

ヒント UPDATECENTILES がインデックス定義で指定されなかった場合、デフォルト値 5 が OUT2 データセットの RECREATE 変数で使用されます。

OUT2=は、ステートメントからの出力を非表示にしません。出力を非表示にするには、NOPRINT オプションを使用する必要があります。

参照項目: “OUT2=データセット” (486 ページ)には、OUT2=データセットの変数の説明が記載されています。

SHORT

SAS データセットの変数名のリスト、インデックス情報、並べ替え情報のみ出力します。

制限事項 変数のリストが 32,767 文字を超える場合、リストは切り捨てられ、WARNING が SAS ログに書き込まれます。変数の完全リストを取得するには、変数のアルファベット順リストを要求します。

VARNUM
変数名のリストをデータセットの論理位置の順序で出力します。デフォルトでは、CONTENTS ステートメントは変数をアルファベット順でリストします。データセットの変数の物理位置は、エンジン依存です。

詳細

CONTENTS ステートメントのかわりに CONTENTS プロシジャを使用

CONTENTS プロシジャと PROC DATASETS の CONTENTS ステートメントの唯一の違いは、DATA=オプションのライブラリ参照名のデフォルトです。PROC CONTENTS の場合、デフォルトは WORK です。CONTENTS ステートメントの場合、デフォルトはプロシジャ入力ライブラリのライブラリ参照名です。

変数の出力

CONTENTS ステートメントは、デフォルトで変数のアルファベット順リストを出力します。x1–x100 などの番号範囲リストは、増分順で x1–x100 と出力されます。詳細については、“変数と属性のアルファベット順リスト” (475 ページ)を参照してください。

注: ビューが変数ラベルを含むデータセットから作成された後にラベルが変更される場合、CONTENTS プロシジャ出力または DATASETS プロシジャ出力に元のラベルが表示されます。CONTENTS プロシジャ出力または DATASETS プロシジャ出力が新しい変数ラベルを反映するために、ビューを再コンパイルする必要があります。

ICU 改訂番号の表示

CONTENTS ステートメントは、データセットの言語的に並べ替えに SORT プロシジャを使用する場合、ICU (nternational Components for Unicode)の改訂番号を出力します。言語的な並べ替えの詳細については、55 章, “SORT プロシジャ,” (2027 ページ)を参照してください。

ライブラリコンテンツと AES 暗号化

ライブラリ内の全データのコンテンツを要求する場合は、_ALL_オプションを使用します。ライブラリに AES 暗号化されたデータファイルが含まれる場合、データファイルへのアクセスには ENCRYPTKEY=データセットオプションを使用する必要があります。次は、ENCRYPTKEY=オプションを使用する例です。

```
proc contents data=MyLib._all_ (encryptkey=key-value);
run;
```

キー値がライブラリ内の特定のデータファイルのキー値に一致しない場合、正しいキー値を入力するよう求められます。

AES 暗号化の詳細については、“AES 暗号化” (*SAS プログラマガイド: エッセンシャル*)を参照してください。ENCRYPTKEY=データセットオプションの詳細について

は、“ENCRYPTKEY= データセットオプション” (*SAS データセットオプション: リファレンス*)を参照してください。

SAS データセットのオブザベーションの長さ、配置、埋め込み

配置には 3 つの異なるケースがあります。

- SAS データセット内のオブザベーションは、可能な場合は 2 バイト境界上で配置されます。結果として、8 バイト数値変数と 4 バイト数値変数がデータセットの前の 8 バイト境界に配置されます。その後に、文字変数が発生順に続きます。データセットに 4 バイト数値データのみ含まれている場合、配置は 4 バイト境界に基づいて行われます。数値倍精度は比較または増分の前に移動し配置するよりも直接操作可能なため、境界によってパフォーマンスがさらに向上します。

  指定されているディスクデータページバッファー 内に多くのオブザベーションが含まれているため、オブザベーション間に埋め込みがある場合があります。この埋め込みにより、各オブザベーションをダブルバイト境界で配置できるようになります。次の例を参照してください。

```
data a;
  length aa 7 bb 6 cc $10 dd 8 ee 3;
  aa = 1;
  bb = 2;
  cc = 'abc';
  dd = 3;
  ee = 4;
  ff = 5;
  output;
run;

proc contents data=a out=a1;
run;

proc print data=a1(keep=name length varnum npos);
run;
```

図 15.1　オブザベーション長

The CONTENTS Procedure

Data Set Name	WORK.A	Observations	1
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	08/16/2016 12:08:25	Observation Length	48
Last Modified	08/16/2016 12:08:25	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

PROC CONTENTS はオブザベーション長 48 を表示します。PROC PRINT は、NPOS が各変数に対しゼロ基準オフセットであるオブザベーション内の変数の内部レイアウトを表示します。

図 15.2　オブザベーションおよび変数の境界観測と可変境

Obs	NAME	LENGTH	VARNUM	NPOS
1	aa	7	1	16
2	bb	6	2	23
3	cc	10	3	32
4	dd	8	4	0
5	ee	3	5	29
6	ff	8	6	8

変数 DD と FF は、実際の数値倍精度であり、それぞれオフセット 0 と 8 のため、自動的に配置されます。その他のオブザベーションには、残りの数値変数と文字変数が含まれます。

このレイアウトの最後の物理変数は、オフセット 32、長さ 10 の CC です。これにより、PROC CONTENTS がオブザベーション長を 48 とレポートする場合でも内部長 42 が指定されます。差異は埋め込みの 6 バイトのため、次のオブザベーションはディスクページバッファー内のダブルバイト境界で配置されます。

- 次の例にあるように、オブザベーションに 8 バイトの数値変数が含まれていない場合は配置は行われません。この場合、オブザベーション長 7 が指定され、ディスクページバッファー内のオブザベーション間に埋め込みはありません。

```
data b;
  length aa 6 cc $1;
  aa = 1;
  cc = 'x';
  output;
run;

proc contents data=b out=b1;
run;

proc print data=b1(keep=name length varnum npos);
run;
```

図 15.3 変数と属性

Obs	NAME	LENGTH	VARNUM	NPOS
1	aa	6	1	0
2	cc	1	2	6

- 圧縮データセットのオブザベーションはディスクページバッファー内で配置されませんが、同じアルゴリズムがオブザベーション内の変数の配置に使用されます。圧縮オブザベーションは解凍し、作業バッファーに移動する必要があります。8 バイトの数値は配置され、解凍後すぐに使用可能です。PROC CONTENTS 出力のオブザベーション長は、オペレーティングシステム固有のオーバーヘッドのため、長くなることがあります。

COPY ステートメント

SAS ライブラリのすべての、または一部の SAS ファイルをコピーします。

制限事項: COPY ステートメントでは、データセットオプションはサポートしていません。

注: SAS/ACCESS LIBNAME エンジンで COPY ステートメントを使用する場合は、“SAS/ACCESSLIBNAME エンジンを使用する場合の DATASETS プロシジャ出力の違い” (369 ページ)を参照してください。

DATA=入力テーブルがデータソースでテーブルまたはビューとして 保存される場合、PROC DATASETS の COPY ステートメントは In-Database 処理を使用して、データソース内の作業のほとんどを実行できます。PROC DATASETS は、Greenplum、PostgresSQL、および Vertica をサポートしています。“PROC COPY の In-Database 処理” (304 ページ)。

CAS エンジンの詳細については、“PROC COPY の CAS 処理” (322 ページ)を参照してください。

ヒント: COPY ステートメントは、SAS/SHARE、SAS/CONNECT などのリモートライブラリサービス(RLS)の使用時は出力ライブラリのエンコーディングおよびデータ表現をデフォルトとして使用します。RLS を使用していない場合、PROC COPY オプション

NOCLONE を出力ファイルに使用して、出力ライブラリのエンコーディングとデータ表現を求める必要があります。NOCLONE オプションを使用すると、データライブラリのデータ表現(OUTREP= LIBNAME オプションで指定されている場合)、または動作環境のネイティブなデータ表現を含むコピーになります。

例: “例 2: SAS ファイルの操作” (494 ページ)

構文

COPY <ACCEL | NOACCEL>OUT=*libref-1*
<CLONE | NOCLONE>
<CONSTRAINT=YES | NO>
<DATECOPY>
<ENCRYPTKEY=*key-value*>
<FORCE>
<IN=*libref-2*>
<INDEX=YES | NO>
<MEMTYPE=(*member-type(s)*)>
<MOVE <ALTER=*alter-password*>>
<OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2 ...>*) >;

必須引数

OUT=*libref-1*
SAS ファイルのコピー先となる SAS ライブラリを指定します。

別名 OUTLIB=

OUTDD=

例 “例 2: SAS ファイルの操作” (494 ページ)

オプション引数

ACCEL
CAS でコピー操作を実行することを指定します。IN=ライブラリと OUT=ライブラリの両方が CAS エンジンライブラリでなければならず、かつ同じ CAS セッションを使用する必要があります。

デフォルト ACCEL

参照項目: “NOACCEL” (408 ページ)

“CAS テーブルを別の CAS テーブルにコピー” (417 ページ)

NOACCEL
SAS Compute Server でコピー操作を実行することを指定します。IN=ライブラリと OUT=ライブラリの両方が CAS エンジンライブラリでなければならず、かつ同じ CAS セッションを使用する必要があります。 SAS Compute Server でコピー操作を実行することを指定します。IN=ライブラリと OUT=ライブラリの両方

が CAS エンジンライブラリでなければならず、かつ同じ CAS セッションを使用する必要があります。

参照項目: “ACCEL” (408 ページ)

“CAS テーブルを別の CAS テーブルにコピー” (417 ページ)

CLONE
次のデータセット属性をコピーするかどうかを指定します。

- 入力/出力バッファーのサイズ
- データセットの圧縮
- 空きスペースの再利用
- 入力データセットのデータ表現、ライブラリ、動作環境
- 値のエンコーディング
- 圧縮されているデータセットにオブザベーション番号で無作為にアクセス可能かどうか

これらの属性は、データセットオプション、SAS システムオプション、LIBNAME ステートメントオプションで指定されます。

- BUFSIZE= (入力/出力バッファーのサイズの値)
- COMPRESS= (データセットを圧縮するかどうかの値)
- REUSE= (空きスペースを再利用するかどうかの値)
- OUTREP= (データ表現の値)
- ENCODING=または INENCODING= (値のエンコーディング)
- POINTOBS= (圧縮されたデータセットにオブザベーション番号で無作為にアクセスできるかどうかの値)

次のテーブルに、BUFSIZE=属性に関する COPY ステートメントの動作を要約します。

表 15.5 CLONE とバッファーページサイズ属性

オプション	COPY ステートメント
CLONE	入力データセットからの BUFSIZE=値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで BUFSIZE=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	SAS システムオプション BUFSIZE=の現在の設定を出力データセットに使用します。
いずれも使用しない	入力データセット用エンジンと出力データセット用エンジンによって使用されるアクセス方法の種類(順次または無作為)を指定します。両方のエンジンで同じ種類のアクセスが使用されている場合、COPY ステートメントは入力データセットの BUFSIZE=値を出力データセットに使用します。同じ種類のアクセスが使用されていない場合、COPY

オプション	COPY ステートメント
	ステートメントは SAS システムオプション BUFSIZE=の設定を出力データセットに使用します。

次のテーブルに、COMPRESS=属性に関する COPY ステートメントの動作を要約します。

表 15.6　CLONE と圧縮属性

オプション	COPY ステートメント
CLONE	入力データセットの値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで COMPRESS=値を指定すると、コピーで指定エンコードが使用されることになります。
NOCLONE	動作環境の圧縮を含むコピー、または指定されている場合は、ライブラリに対する LIBNAME ステートメントの COMPRESS=オプションの値となります。
いずれも使用しない	デフォルトは CLONE となります。

次のテーブルに、REUSE=属性に関する COPY ステートメントの動作を要約します。

表 15.7　CLONE と空きスペースの再利用属性

オプション	COPY ステートメント
CLONE	入力データセットの値を出力データセットに使用します。入力データセット用エンジンで空きスペースの再利用属性がサポートされていない場合、COPY ステートメントは対応する SAS システムオプションの現在の設定を使用します。ただし、OVERRIDE=オプションリストで REUSE=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	SAS システムオプション COMPRESS=と REUSE=の現在の設定を出力データセットに使用します。
いずれも使用しない	デフォルトは CLONE となります。

次のテーブルに、OUTREP=属性に関する COPY ステートメントの動作を要約します。

表 15.8 CLONE とデータ表現属性

オプション	COPY ステートメント
CLONE	入力データセットの同一データ表現を含むコピーとなります。ただし、OVERRIDE=オプションリストに OUTREP=値を指定すると、データ表現が変更され、そのエンコーディングが、現在のセッションのデータ表現およびロケールとの互換性があるエンコーディングに変更されます。エンコーディングは、OVERRIDE=オプションリストに指定されている場合にも変更されます。
NOCLONE	動作環境のデータ表現を含むコピー、または指定されている場合は OUT=ライブラリの LIBNAME ステートメントの OUTREP=オプションの値となります。
いずれも使用しない	デフォルトは CLONE となります。

データ表現とは、特定の動作環境でデータを保存する形式です。さまざまな動作環境では、次のような目的のさまざまな規格または規則が使用されます。

- 浮動小数点数の格納
- 文字エンコーディング(ASCII または EBCDIC)
- メモリのバイトオーダリング(ビッグエンディアンまたはリトルエンディアン)
- ワード配置(4 バイト境界または 8 バイト境界)
- データ型の長さ(16 ビット、32 ビット、または 64 ビット)

ネイティブなデータ表現とは、ファイルのデータ表現が CPU 動作環境と同じであることです。

次のテーブルに、ENCODING=属性に関する COPY ステートメントの動作を要約します。

表 15.9 CLONE とエンコーディング属性

オプション	COPY ステートメント
CLONE	入力データセットのエンコーディングを使用するコピー、または指定されている場合は、入力ライブラリの LIBNAME ステートメントの INENCODING=オプションの値となります。ただし、OVERRIDE=オプションリストで ENCODING=値を指定すると、コピーで指定エンコードが使用されることになります。
NOCLONE	現在のセッションエンコーディングのエンコーディングを使用するコピー、または指定されている場合は、出力ライブラリの LIBNAME ステートメントの OUTENCODING=オプションの値となります。

オプション	COPY ステートメント
いずれも使用しない	デフォルトは CLONE となります。

コンピューターによって保存、送信、処理されるテキスト(文字)データはすべてエンコーディングされます。エンコーディングは各文字を一意の数値表現にマップします。エンコーディングは、文字セットとエンコーディング方法の組み合わせです。文字セットは言語または言語グループによって 使用される文字と記号のレパートリーです。エンコーディング方法は、数字をエンコーディングで使用される文字セットに割り当てるために使用される一連のルールです。

次のテーブルに、POINTOBS=属性に関する COPY ステートメントの動作を要約します。POINTOBS=を使用するには、出力データセットを圧縮する必要があります。

表 15.10　CLONE と POINTOBS=属性

オプション	COPY ステートメント
CLONE	入力データセットの POINTOBS=値を出力データセットに使用します。ただし、OVERRIDE=オプションリストで POINTOBS=値を指定すると、コピーで指定値が使用されることになります。
NOCLONE	LIBNAME ステートメントは、出力データセットが圧縮され、POINTOBS=オプションが指定されて出力エンジンでサポートされている場合に使用します。LIBNAME ステートメントが指定されず、データセットが圧縮されている場合、出力エンジンでサポートされているときデフォルトは POINTOBS=YES となります。
いずれも使用しない	デフォルトは CLONE となります。

CONSTRAINT=YES | NO
データセットのコピー時にすべての一貫性制約をコピーするかどうかを指定します。

デフォルト　NO

ヒント　外部キーを持つ一貫性制約を含むデータセットの場合、COPY ステートメントは CONSTRAINT=YES が指定され、ライブラリ全体がコピーされる場合に一般制約と参照制約をコピーします。SELECT ステートメントまたは EXCLUDE ステートメントを使用してデータセットをコピーする場合、参照一貫性制約はコピーされません。

DATECOPY
結果として生じるファイルのコピーに、SAS ファイルの作成時および最終変更時の SAS 内部日時をコピーします。動作環境の日時は保持されません。

制限事項 DATECOPY は、暗号化されたファイルまたはカタログと使用できません。

ヒント MODIFY ステートメントの DTC=オプションを使用して、ファイル作成日時を変更できます。MODIFY ステートメント (441 ページ)を参照してください。

コピー中のファイルに追加処理が必要な属性が含まれている場合、最終変更日は現在の日付に変更されます。たとえば、インデックスを含むデータセットをコピーし、インデックスを再度作成する必要がある場合、最終変更日は現在の日付に変更されます。追加処理が必要で、最終変更日に影響する可能性のあるその他の属性には、一貫性制約とソートインジケーターが含まれます。

ENCRYPTKEY= *key-value*
IN=ライブラリ内で AES 暗号化を含むデータセットのコピーに必要なキー値を指定します。

注 出力ライブラリで AES 暗号化がサポートされておらず、入力データセットが AES 暗号化されている場合、COPY プロセスによってエラーが生成されます。

参照項目: “参照一貫性制約を含む AES 暗号化データファイルのコピー ” (421 ページ)

FORCE
MOVE オプションを監査証跡が存在する SAS データセットに使用できます。

注 AUDIT ファイルは、監査データセットと移動されません。

IN=*libref-2*
コピーする SAS ファイルを含む SAS ライブラリを指定します。

別名 INLIB=

INDD=

デフォルト プロシジャ入力ライブラリのライブラリ参照名

操作 選択したメンバーのみコピーするには、SELECT ステートメントまたは EXCLUDE ステートメントを使用します。

INDEX=YES | NO
データセットを SAS ライブラリ間でコピーする場合にデータセットのすべてのインデックスをコピーするかどうかを指定します。

デフォルト YES

MEMTYPE=(*member-type(s)*)
処理を 1 つ以上のメンバーの種類に制限します。

別名 MTYPE=

MT=

デフォルト
PROC DATASETS ステートメントの MEMTYPE=を省略すると、デフォルトは MEMTYPE=ALL となります。

注
PROC COPY がテープ上の SAS ライブラリを処理し、MEMTYPE=オプションが指定されていない場合、ファイルの最後までエントリの順次ライブラリ全体がスキャンされます。順次ライブラリがマルチボリュームテープである場合、すべてのテープボリュームはマウントされます。この動作は、単一のボリュームテープライブラリの場合も同様です。

参照項目:
“SAS ファイルのコピー/移動時のメンバーの種類の指定” (417 ページ)および“メンバーの種類” (466 ページ)

例
“例 2: SAS ファイルの操作” (494 ページ)

MOVE
SAS ファイルを入力データライブラリ(IN=オプションによって指定)から出力データライブラリ(OUT=オプションによって指定)に移動します。そして、元のファイルを入力データライブラリから削除します。

制限事項
MOVE オプションは、IN=エンジンでテーブルの削除がサポートされている場合にのみ、SAS ライブラリのメンバーの削除に使用できます。テープ形式エンジンでは、テーブルの削除はサポートされていません。テープ形式エンジンを使用する場合、MOVE 操作は実行されず、警告が発行されます。

参照項目:
“DATASETS プロシジャでパスワードを使用する” (462 ページ)

例
“例 2: SAS ファイルの操作” (494 ページ)

ALTER=*alter-password*
ライブラリ間で移動中の変更保護されている SAS ファイルに対する変更パスワードを与えます。MOVE オプションは元のデータライブラリから SAS ファイルを削除するため、SAS ファイルの移動には変更アクセス権限が必要となります。

OVERRIDE=(*ds-option-1=value-1 <ds-option-2=value-2> ...*)
入力データセットからコピーされた指定出力データセットオプションを上書きします。COPY の出力データセットコンテキストでは、一部のデータセットオプションは適切でない場合があります。

制限事項
NOCLONE オプションが指定される場合、OVERRIDE オプションは無視されます。ただし、NOCLONE オプションによって調整される属性以外のデータセット属性の変更には使用できます。

操作
OVERRIDE=オプションで OUTREP=値を指定すると、デフォルトのエンコーディングは、OUTREP=値と現在の計算サーバーセッションのロケールで表されるオペレーティング環境に基づいています。UTF-8 などのデフォルト以外のエンコーディングを割り当てるには、OVERRIDE=オプションで ENCODING=値も指定する必要があります。ロケールとエンコーディング

の詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

ヒント デフォルト(または CLONE)の動作は、入力データセットからのデータ表現やエンコーディングを含む多くの属性を保持することです。代わりに、プロシジャを実行している計算サーバーセッションのデータ表現とエンコーディングが必要な場合は、PROC COPY または COPY ステートメントで OVERRIDE=(OUTREP=SESSION ENCODING=SESSION)を THEN 指定します。入力データセットの属性を保持したくない場合は、代わりに NOCLONE を指定してください。

NOCLONE
詳細については、"CLONE" (409 ページ)を参照してください。

詳細

SELECT と MEMTYPE を使用したパフォーマンスの向上

COPY ステートメントを使用する際、ライブラリのインメモリディレクトリが取得されます。ライブラリに数千ものメンバーが含まれ、コピーされるのはその中のいくつかのメンバーのみである場合、パフォーマンスの問題が生じることがあります。こうしたパフォーマンスの問題を解決するには、COPY ステートメントの MEMTYPE=オプションと SELECT ステートメントの組み合わせを使用します。次にこのプロセスの例を示します。

```
proc datasets lib=work;
   copy out=mylib memtype=(data catalog);
   select mydata x1-x10 data2;
run;
```

注: MEMTYPE=ALL またはワイルドカード指定(":")のいずれかを使用する場合、パフォーマンスコードは使用できません。

ブロック I/O メソッドを使用してコピー

ブロック I/O メソッドは、一度に 1 オブザベーションずつではなくデータブロックをコピーするために使用されます。大きなデータセットをコピーする場合、このメソッドによりパフォーマンスが向上します。このメソッドを使用するかどうかが決定されます。すべてのデータセットでブロック I/O メソッドが使用できるわけではありません。COPY ステートメントと Base SAS エンジンによって設定された制約があります。

使用されているコピーメソッドに関する情報を SAS ログに表示するには、MSGLEVEL=システムオプションを次のように指定できます。

```
options msglevel=i;
```

ブロック I/O メソッドが使用されていない場合、SAS ログに次のメッセージが出力されます。

INFO: データセットブロック I/O は、下記の理由により使用できません。

COPY ステートメントがブロック I/O を使用しないと判断した場合、次の説明のうちの 1 つが SAS ログに出力されます。

INFO: - データセットは異なるエンジンを使用し、異なる変数を持っているか、または異なる可能性がある属性を持っています。

INFO: - メンバーレベルロックがありません。

INFO: - OBS オプションがアクティブです。

INFO: - FIRSTOBS オプションがアクティブです。

ブロック I/O メソッドを使用されないと Base SAS エンジンが判断した場合、次のいずれかの説明が SAS ログに書き込まれます。

INFO: - 参照一貫性制約が存在します。

INFO: - クロス環境データアクセスが使用されています。

INFO: - ファイルが圧縮されています。

INFO: - ファイルに、中断されない監査ファイルが含まれています。

パフォーマンスに問題があり、テスト用に大きなデータセットのサブセットを作成する場合、OBS=0 オプションを使用できます。この場合、ブロック I/O メソッドを無効化して、システムリソースの使用を減らします。

次の例では、OBS=0 オプションを使用して、システムリソースの使用を減らします。

```
options obs=0 msglevel=i;
proc copy in=old out=lib;
select a;
run;
```

SET ステートメントを使用した場合も、同じ結果になります。

```
data lib.new;
   if 0 then set old.a;
   stop;
run;
```

ライブラリ全体のコピー

SAS ライブラリ全体をコピーするには、COPY ステートメントに続いて入力ライブラリと出力ライブラリを指定します。たとえば、次のステートメントは SOURCE データライブラリのすべての SAS ファイルを DEST データライブラリにコピーします。

```
proc datasets library=source;
  copy out=dest;
run;
```

選択した SAS ファイルのコピー

選択した SAS ファイルをコピするには、SELECT または EXCLUDE ステートメントを使用します。SELECT ステートメントまたは EXCLUDE ステートメントと COPY ステートメントの併用の詳細については、“SAS ファイルのコピー/移動時のメンバーの種類の指定” (417 ページ)を参照してください。例は SAS ファイルの操作 (494 ペー

ジ) にあります。また、 EXCLUDE ステートメント (428 ページ) と SELECT ステートメント (453 ページ)も参照してください。

省略したメンバーリストを選択または除外することもできます。たとえば、次のステートメントはメンバー Test1、Test2、および Test3 を選択します。

```
select tabs test1-test3;
```

名前が同じ文字で始まるメンバーのグループは、共通の文字に続きコロン(:)を入力すると、選択できます。たとえば、次のステートメントを指定すれば、直前の例の４メンバーとそれ以外の T で始まる名前のすべてのメンバーを 選択できます。

```
select t:;
```

選択と同じように除外するメンバーも指定できます。つまり、個々のメンバー名をリストすることもできますし、省略したリストも使用でき、共通の文字とコロン(:)の指定もできるということです。たとえば、次のステートメントは、メンバー Stats、Teams1、Teams2、Teams3、Teams4、および RBI という文字で始まるすべてのメンバーを、コピー操作から除外します。

```
exclude stats teams1-teams4 rbi:;
```

MEMTYPE=オプションは、どの種類のメンバーが選択または除外可能になるかどうかに影響を与えます。

SELECT ステートメントまたは EXCLUDE ステートメントが CONSTRAINT=YES と使用される場合、データセットに関する一般一貫性制約だけがコピーされます。参照一貫性制約はコピーされません。

CAS テーブルを別の CAS テーブルにコピー

PROC COPY IN=と OUT=オプションが CAS ライブラリ参照名に設定され、同じ CAS セッションが両方のライブラリ参照名でデフォルトで使用されている場合、コピーは CAS で行われます。

これは、デフォルトの動作への変更です。MSGLEVEL=I システムオプションが設定されている場合、COPY が CAS で発生すると、INFO メッセージがログに送信されます。 INFO: クラウド分析サービスで COPY を実行します。

PROC COPY オプション ACCEL | NOACCEL は、COPY プロシジャの実行場所を決定します。ACCEL がデフォルトです。つまり、COPY 操作は CAS サーバー上で実行されます。COPY プロシジャが CAS サーバー上で失敗した場合、ファイルのコピーは試行されません。

PROC COPY 呼び出しで OVERRIDE オプションが指定されている場合、このオプションは無視され、メモがログに送られます。

PROC COPY を使用してセッション間でコピーすることはできません。CAS ライブラリが 2 つの異なるセッションで定義されている場合、次のエラーが表示されます。 ERROR: このアクションに使用される CAS セッション SESS1 と SESS2 が一致しません。

SAS ファイルのコピー/移動時のメンバーの種類の指定

COPY ステートメントの MEMTYPE=オプションは、プロシジャのその他のステートメントの MEMTYPE=オプションといくつかの点が異なります。

- スラッシュはオプションの前に指定できません。

- MEMTYPE=オプションをかっこで囲んで、その影響をその直前のメンバーに制限することはできません。
- SELECT ステートメント、EXCLUDE ステートメントおよび IN=オプション(COPY ステートメントの)は、次のルールに従って、COPY ステートメントの MEMTYPE=オプションの動作に影響します。

  1 SELECT ステートメントまたは EXCLUDE ステートメントの MEMTYPE=は、COPY ステートメントの MEMTYPE=オプションに優先します。次のステートメントは、Vision.catalog と Nutr.data だけをデフォルトのデータライブラリから Dest データライブラリにコピーします。最初の SELECT ステートメントの MEMTYPE=値は COPY ステートメントの MEMTYPE=値に優先します。

```
proc datasets;
  copy out=dest memtype=data;
    select vision(memtype=catalog) nutr;
run;
```

  2 IN=オプションを使用しない、またはそれをプロシジャ入力ライブラリとなるライブラリを指定するために使用する場合、PROC DATASETS ステートメントの MEMTYPE=オプションの値は、処理可能な SAS ファイルの種類を制限します。プロシジャは、ルール 1 で説明されている優先順位を使用し、コピー可能な種類をさらにサブセット化します。次のステートメントはメンバーをデフォルトのデータライブラリから DEST データライブラリにコピーしません。代わりに、SELECT ステートメントで指定した MEMTYPE=値が PROC DATASETS ステートメントの MEMTYPE=オプションの値の一つでないため、プロシジャはエラーメッセージを 発行します。

```
   /* This step fails! */
proc datasets memtype=(data program);
  copy out=dest;
    select apples / memtype=catalog;
run;
```

  3 IN=オプションでプロシジャ入力ライブラリ以外の入力データライブラリを指定する場合、PROC DATASETS ステートメントの MEMTYPE=オプションはコピー操作に影響しません。サブセット化が発生していないため、プロシジャはルール 1 で説明されている優先順序を使用して、コピー可能な種類をサブセット化します。次のステートメントは、Bodyfat.data を Dest データライブラリに正常にコピーします。COPY ステートメントの IN=オプションで指定した Source ライブラリが PROC DATASETS ステートメントの MEMTYPE=オプションによって影響を受けないためです。

```
proc datasets library=work memtype=catalog;
  copy in=source out=dest;
    select bodyfat / memtype=data;
run;
```

COPY ステートメントを使用する際、ライブラリのインメモリディレクトリが取得されます。ライブラリに数千ものメンバーが含まれ、コピーされるのはその中のいくつかのメンバーのみである場合、パフォーマンスの問題が生じることがあります。こうしたパフォーマンスの問題を解決するには、COPY ステートメントの MEMTYPE=オプションと SELECT ステートメントの組み合わせを使用します。次にこのプロセスの例を示します。

```
proc datasets lib=work;
```

```
  copy out=mylib memtype=(data catalog);
  select mydata x1-x10 data2;
run;
```

注: MEMTYPE=ALL またはワイルドカード指定(":")のいずれかを使用する場合、パフォーマンスコードは使用できません。

ビューのコピー

NOCLONE が指定されている COPY ステートメントでは、SQL ビュー、DATA ステップビュー、および一部のビュー(Oracle、Sybase)に対し、OUTREP=オプションと ENCODING= LIBNAME オプションをサポートしています。COPY ステートメントを SAS/SHARE、SAS/CONNECT などのリモートライブラリサービス(RLS)と使用する際は、COPY ステートメントは出力ライブラリのエンコーディングおよびデータ表現をデフォルトとして使用します。

注意
DATA ステップビューの作成時に DATA ステートメントの SOURCE=NOSAVE オプションを使用する場合、ビューを SAS のバージョン間でコピーすることはできません。

パスワード保護されている SAS ファイルのコピー

パスワード保護されている SAS ファイルをパスワードを指定せずにコピーできます。また、パスワードは続行して SAS ファイルに対応しているため、コピー後に SAS ファイルにアクセスし操作するためにパスワードを知っている必要があります。

長い変数名を持つデータセットのコピー

VALIDVARNAME=V6 システムオプションが設定されていて、かつデータセットが長い変数名を持つ場合、その長い変数名は切り捨てられ、一意の変数名が生成され、コピーが実行されます。インデックス名についても同様です。
VALIDVARNAME=ANY で OUT=エンジンが長い変数名をサポートしていない場合、コピーは失敗します。

変数名が切り捨てられる場合、変数名は 8 バイトに短くされます。もしもこの名前がデータセットですでに定義されているものであった場合、名前は切り捨てられ、それに数字の 2 から始まる数字が追加されます。切り捨てと数字の追加は、名前が重複しないものになるまで続きます。たとえば、LONGVARNAME という変数名は、データセットで重複する名前が存在しないとすると、LONGVARN となります。すでにその名前が存在する場合は、LONGVAR2 となります。

注意
切り捨てられた変数名は入力データセット内で定義されている名前と競合することがあります。 この動作は、すでに定義されている変数名が 8 バイトの長さで最後が数字で終わっている場合に起こります。次の例では、出力データセットで切り捨てられた名前が定義されており、入力データセットからの名前が変更されています。

```
options validvarname=any;
data test;
  longvar10='aLongVariableName';
```

```
  retain longvar1-longvar5 0;
run;

options validvarname=v6;
proc copy in=work out=sasuser;
  select test;
run;
```

この例では、LONGVAR10 は LONGVAR1 に切り捨てられて、出力データセット内に配置されます。次に、元の LONGVAR1 がコピーされます。その名前はもう一意ではありません。そのため、LONGVAR2 という名前に変更されます。入力データセットの他の変数も、名称変更のアルゴリズムに従って名前が変更されます。次の例は、SAS ログからのものです。

```
1   options validvarname=any;
2   data test;
3     longvar10='aLongVariableName';
4     retain longvar1-longvar5 0;
5   run;

NOTE: The data set WORK.TEST has 1 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time          2.60 seconds
      cpu time           0.07 seconds



6
7   options validvarname=v6;
8   proc copy in=work out=sasuser;
9     select test;
10  run;

NOTE: Copying WORK.TEST to SASUSER.TEST (memtype=DATA).
NOTE: The variable name longvar10 has been truncated to longvar1.
NOTE: The variable longvar1 now has a label set to longvar10.
NOTE: Variable LONGVAR1 already exists on file SASUSER.TEST, using LONGVAR2
instead.
NOTE: The variable LONGVAR2 now has a label set to LONGVAR1.
NOTE: Variable LONGVAR2 already exists on file SASUSER.TEST, using LONGVAR3
instead.
NOTE: The variable LONGVAR3 now has a label set to LONGVAR2.
NOTE: Variable LONGVAR3 already exists on file SASUSER.TEST, using LONGVAR4
instead.
NOTE: The variable LONGVAR4 now has a label set to LONGVAR3.
NOTE: Variable LONGVAR4 already exists on file SASUSER.TEST, using LONGVAR5
instead.
NOTE: The variable LONGVAR5 now has a label set to LONGVAR4.
NOTE: Variable LONGVAR5 already exists on file SASUSER.TEST, using LONGVAR6
instead.
NOTE: The variable LONGVAR6 now has a label set to LONGVAR5.
NOTE: There were 1 observations read from the data set WORK.TEST.
NOTE: The data set SASUSER.TEST has 1 observations and 6 variables.
NOTE: PROCEDURE COPY used (Total process time):
      real time          13.18 seconds
      cpu time           0.31 seconds
```

```
11
12  proc print data=test;
13  run;

ERROR: The value LONGVAR10 is not a valid SAS name.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE PRINT used (Total process time):
    real time          0.15 seconds
    cpu time           0.01 seconds
```

AES で暗号化されたデータファイルのコピー

AES 暗号化でデータファイルをコピーするには、次の 2 つの方法のいずれかで ENCRYPTKEY=*key-value* を指定する必要があります。

- ライブラリ内のすべての AES 暗号化データセットに同じ ENCRYPTKEY=*key-value* が含まれる場合、次の例を使用してください。

```
proc copy in=lib1 out=lib2 encryptkey=key-value;
run;
```

- ENCRYPTKEY=*key-value* が各データセットで異なる場合、次の例に示すように SELECT ステートメントを使用してください。

```
proc copy in=lib1 out=lib2;
   select mydata1 (encryptkey=key-value1) mydata2 (encryptkey=key-value2);
run;
```

AES 暗号化の詳細については、"AES 暗号化" (*SAS プログラマガイド: エッセンシャル*)を参照してください。ENCRYPTKEY=データセットオプションの詳細については、"ENCRYPTKEY= データセットオプション" (*SAS データセットオプション: リファレンス*)を参照してください。

参照一貫性制約を含む AES 暗号化データファイルのコピー

参照用一貫性制約を含む AES 暗号化でデータファイルをコピーするには、次を実行する必要があります。

- IN=ライブラリのコンテンツ全体をコピーする
- CONSTRAINT=YES ステートメントオプションを指定する
- ENCRYPTKEY=*key-value* オプションを指定する。指定しない場合は、ライブラリがメタデータバインド型ライブラリであり、ライブラリの暗号化キーが登録されている必要があります。

```
proc copy in=Lib1 out=Lib2 constraint=yes encryptkey=key-value;
run;
```

ただし、IN=LIB 内に複数のデータプライマリキーと参照用データセットのペアがあり、各ペアに異なる ENCYPTKEY=値が含まれる場合、上記のコードは機能しなくなります。たとえば、次のような 2 つのペアがある場合、

```
primarydset1/foreigndset1 having ENCRYPTKEY=secret1
primarydset2/foreigndset2 having ENCRYPTKEY=secret2
```

上記のスキームは 1 つのペアに対してのみ機能します。すべてのペアには同じキー値が含まれる必要があります。

詳細については、“ENCRYPTKEY= データセットオプション” (*SAS データセットオプション: リファレンス*)を参照してください。

世代グループのコピー

COPY ステートメントを使用して、世代グループ全体をコピーできます。ただし、世代グループの特定のバージョンをコピーすることはできません。

ホスト間での SAS データセットの移送

XPORT エンジンまたは REMOTE エンジンと一緒に COPY プロシジャを使うと、ホスト間で SAS データセットを移動することができます。詳細については、*SAS ファイルの移動とアクセス*にある“Strategies for Moving and Accessing SAS Files”を参照してください。

COPY ステートメントの代わりに COPY プロシジャを使用

大まかに言えば、COPY プロシジャは DETASETS プロシジャの COPY ステートメントと同様に機能します。両者の違いを次に示します。

- PROC COPY には IN=引数が必須です。COPY ステートメントでは IN=はオプションです。省略された場合、デフォルト値はプロシジャの入力ライブラリのライブラリ参照名です。
- PROC DATASETS は、順次データアクセスのみを許可するライブラリには使用できません。
- COPY ステートメントは NOWARN オプションを順守しますが、PROC COPY は違います。

DELETE ステートメント

SAS ファイルを SAS ライブラリから削除します。

例: “例 2: SAS ファイルの操作” (494 ページ)

構文

DELETE*SAS-file(s)*
</ <ALTER=*alter-password*>
<ENCRYPTKEY=*key-value*>
<GENNUM=ALL | HIST | REVERT | *integer*>
<MEMTYPE=*member-type*>>;

必須引数

SAS-file(s)
削除する SAS ファイルを 1 つ以上指定します。SAS ファイルに ALTER=パスワードが割り当てられている場合、そのパスワードを SAS ファイルを削除するために指定する必要があります。番号範囲リストまたはコロンリストを使用することもできます。詳細については、“データセット名リスト” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

オプション引数

ALTER=*alter-password*
削除する変更保護された SAS ファイルに対する変更パスワードを与えます。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

ENCRYPTKEY=*key-value*
AES 暗号化のキー値を指定します。このオプションは、GENNUM=REVERT (GENNUM=0 と同じ)または GENNUM=*relative-generation-number* を指定する場合は必須です。ENCRYPTKEY=オプションのキー値は、基本バージョンのキー値である必要があります。詳細については、“ライブラリコンテンツと AES 暗号化” (404 ページ)を参照してください。

GENNUM=ALL | HIST | REVERT | *integer*
世代データセットの処理を制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。有効な値は次のとおりです。

ALL	世代グループの基本バージョンとすべての 履歴バージョンを参照します。
HIST	世代グループの基本バージョンを除く、すべての履歴バージョンを参照します。
REVERT | 0	基本バージョンを削除し、存在する場合は最新履歴バージョンを基本バージョンに変更します。
integer	世代グループ内の特定のバージョンを参照する番号です。正の数を指定すると、データセット名に追加される特定の世代番号の絶対参照になります(つまり、**gennum=2** は MYDATA#002 を指定することになります)。負の数を指定すると、最新から最古への基本バージョンに相対する履歴バージョンの相対参照になります(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。

参照項目: “世代データセットの処理制限” (463 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

別名 MTYPE=

MT=

デフォルト DATA

参照項目: “処理するメンバーの種類の制限” (465 ページ)

例 “例 2: SAS ファイルの操作” (494 ページ)

詳細

基本

RUN グループの実行時に、SAS ファイルが即時に削除されます。削除操作は、開始前に確認できません。

SAS ファイルに ALTER=パスワードが割り当てられている場合、そのパスワードを SAS ファイルを削除するために指定する必要があります。

プロシジャ入力ライブラリに存在しない SAS ファイルを削除しようとすると、PROC DATASETS はメッセージを発行し、処理は続行されます。NOWARN が使用されている場合、メッセージは発行されません。

DELETE ステートメントを使用して、インデックスが関連付けられているデータセットを削除する場合、ステートメントはそのインデックスも削除します。

DELETE ステートメントを使用して、外部キー一貫性制約または外部キー参照を含むプライマリキーを持つデータファイルを削除することはできません。外部キーを持つデータファイルについては、データファイルを削除する前に外部キーを削除する必要があります。外部キー参照を含むプライマリキーを持つデータファイルについては、データファイルを削除する前にそのプライマリキーを参照する外部キーを削除する必要があります。

入力 CAS エンジンライブラリに存在しない CAS テーブルを削除しようとすると、メッセージがログに書き込まれ、処理が続行されます。NOWARN が使用されている場合、メッセージは発行されません。

世代グループの操作

世代グループの操作の概要

世代グループを使用している場合、DELETE ステートメントを使用して、次のバージョンを削除できます。

- 基本バージョンとすべての 履歴バージョンを削除します
- 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更します
- 絶対バージョンを削除します
- 関連バージョンを削除します
- すべての履歴バージョンを削除し、基本バージョンを残します

基本バージョンとすべての 履歴バージョンの削除

次のステートメントは、基本バージョンとすべての 履歴バージョンを削除します。データセット名は A です。

```
proc datasets;
  delete A(gennum=all);

proc datasets;
  delete A / gennum=all;

proc datasets gennum=all;
  delete A;
```

次のステートメントは、基本バージョンとすべての 履歴バージョンを削除します。データセット名は文字 A で始まります。

```
proc datasets;
  delete A:(gennum=all);

proc datasets;
  delete A: / gennum=all;

proc datasets gennum=all;
  delete A:;
```

基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更

次のステートメントは、基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更します。データセット名は A です。

```
proc datasets;
  delete A(gennum=revert);

proc datasets;
  delete A / gennum=revert;

proc datasets gennum=revert;
  delete A;
```

次のステートメントは、基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更します。データセット名は文字 A で始まります。

```
proc datasets;
  delete A:(gennum=revert);

proc datasets;
  delete A: / gennum=revert;

proc datasets gennum=revert;
  delete A:;
```

絶対数によってバージョンを削除

次のステートメントは絶対数を使用して、最初の履歴バージョンを削除します。

```
proc datasets;
```

```
  delete A(gennum=1);

proc datasets;
  delete A / gennum=1;

proc datasets gennum=1;
  delete A;
```

次のステートメントは、特定の履歴バージョンを削除します。データセット名は文字 A で始まります。

```
proc datasets;
  delete A:(gennum=1);

proc datasets;
  delete A: / gennum=1;

proc datasets gennum=1;
  delete A:;
```

相対値によってバージョンを削除

次のステートメントは、相対値を使用して、最新履歴バージョンを削除します。データセット名は A です。

```
proc datasets;
  delete A(gennum=-1);

proc datasets;
  delete A / gennum=-1;

proc datasets gennum=-1;
  delete A;
```

次のステートメントは、相対値を使用して、最新履歴バージョンを削除します。データセット名は A で始まります。

```
proc datasets;
  delete A:(gennum=-1);

proc datasets;
  delete A: / gennum=-1;

proc datasets gennum=-1;
  delete A:;
```

すべての履歴バージョンの削除および基本バージョンの保留

次のステートメントは、すべての履歴バージョンを削除し、基本バージョンを残します。データセット名は A です。

```
proc datasets;
  delete A(gennum=hist);

proc datasets;
  delete A / gennum=hist;

proc datasets gennum=hist;
```

```
delete A;
```

次のステートメントは、すべての履歴バージョンを削除し、基本バージョンを残します。データセット名は文字 A で始まります。

```
proc datasets;
  delete A:(gennum=hist);

proc datasets;
  delete A: / gennum=hist;

proc datasets gennum=hist;
  delete A:;
```

EXCHANGE ステートメント

SAS ライブラリ内の 2 つの SAS ファイルの名前を入れ換えます。

例: “例 2: SAS ファイルの操作” (494 ページ)

構文

EXCHANGE *name-1=other-name-1* <*name-2=other-name-2* ...>
</ <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

必須引数

name=other-name
プロシジャ入力ライブラリ内の SAS ファイルの名前を入れ換えます。*name* と *other-name* がプロシジャ入力ライブラリに存在する必要があります。

オプション引数

ALTER=*alter-password*
入れ換える名前の変更保護されている SAS ファイルに対する変更パスワードを与えます。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。同じ種類の SAS ファイルの名前のみ入れ換えることができます。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

デフォルト PROC DATASETS ステートメントで MEMTYPE=を指定しない場合、デフォルトは ALL となります。

参照項目: “処理するメンバーの種類の制限” (465 ページ)

詳細

1 つの EXCHANGE ステートメントで複数の名前ペアを入れ換える場合、PROC DATASETS は EXCHANGE ステートメントで入れ換えをリストする順序ではなく、SAS ファイルの名前がディレクトリリストで発生する順序で入れ換えを実行します。

name の SAS ファイルが SAS ライブラリに存在しない場合、PROC DATASETS は EXCHANGE ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを指定します。

また、EXCHANGE ステートメントは関連付けられているインデックスを新しい名前と対応するように入れ換えます。

EXCHANGE ステートメントにより、2 つの既存の世代グループのみ名前の入れ換えが可能になります。特定の世代番号を既存する基本バージョンまたは別の世代番号と入れ換えることはできません。

EXCLUDE ステートメント

SAS ファイルをコピーから除外します。

制限事項: EXCLUDE ステートメントは、COPY ステートメントの後に指定する必要があります
EXCLUDE ステートメントは、SELECT ステートメントで同じ COPY ステップに記述できません

例: “例 2: SAS ファイルの操作” (494 ページ)

構文

EXCLUDE *SAS-file(s)* </ MEMTYPE=*member-type*>;

必須引数

SAS-file(s)
コピー操作から除外する SAS ファイルを 1 つ以上指定します。EXCLUDE ステートメントで指定するすべての SAS ファイルが、COPY ステートメントの IN=オプションで指定されるライブラリ内にある必要があります。SAS ファイルが世代グループである場合、EXCLUDE ステートメントにより基本バージョンの選択のみ可能になります。

次のショートカットを使用して、EXCLUDE ステートメントに複数の SAS ファイルをリストできます。

表 15.11 EXCLUDE ステートメントの使用

表記	意味

x1–xn	ファイル X1 から Xn を指定します。番号は連続している必要があります。
x:	文字 X で始まるすべてのファイルを指定します。

オプション引数

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメント、COPY ステートメント、または EXCLUDE ステートメントで MEMTYPE=を指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目: “SAS ファイルのコピー/移動時のメンバーの種類の指定” (417 ページ) および“処理するメンバーの種類の制限” (465 ページ)

詳細

複数の同じ名前のファイルを除外

EXCLUDE ステートメントで複数の SAS ファイルをリストするためのショートカットを使用できます。

FORMAT ステートメント

MODIFY ステートメントで常に指定される SAS データセットの変数の出力形式を割り当て、変更、削除します。

制限事項: FORMAT ステートメントは、MODIFY RUN グループで指定する必要があります

例: “例 4: SAS データセットの変更” (502 ページ)

構文

FORMAT *variable-1* <*format-1*>
<*variable-2* <*format-2*> ...>;

必須引数

variable
割り当て、変更、削除する出力形式の変数を 1 つ以上指定します。出力形式と変数の関連付けを解除する場合、リストの最後に変数をリストします。出力形式はその後に指定しません。

```
format x1-x3 4.1 time hhmm2.2 age;
```

オプション引数

format
その前にリストされている変数に適用する出力形式を指定します。出力形式を指定しない場合、FORMAT ステートメントは *variable-list* の変数と関連付けられている出力形式を削除します。

ヒント すべての出力形式をデータセットから削除するには、ATTRIB ステートメント (394 ページ) と_ALL_キーワードを使用します。

IC CREATE ステートメント

一貫性制約を作成します。

制限事項: IC CREATE ステートメントは MODIFY RUN グループに記述する必要があります。

注: 作成する参照制約について、外部キーでプライマリキーと同じ変数の数を同じ順序で指定する必要があります。変数は同じ種類(文字/数値)、同じ長さである必要があります。

構文

IC CREATE <*constraint-name*=> *constraint* <MESSAGE='*message-string*' <MSGTYPE=USER>>;

必須引数

constraint
制約の種類です。有効な値は次のとおりです。

NOT NULL (*variable*)	*variable* に特殊欠損値を含む SAS 欠損値が含まれないように指定します。
UNIQUE (*variables*)	*variables* の値が一意であるように指定します。この制約は DISTINCT と同じです。
DISTINCT (*variables*)	*variables* の値が一意であるように指定します。この制約は UNIQUE と同じです。

CHECK (WHERE-*expression*)	変数のデータ値を特定のセット、範囲、値リストに制限します。この動作は、WHERE 式によって実行されます。
PRIMARY KEY (*variables*)	プライマリキー、つまり欠損値が含まれず、その値が一意である一連の変数を指定します。

重複するプライマリキー制約と外部キー制約を定義するとき、つまりデータファイルの変数がプライマリキー定義と外部キー定義の一部であるとき、まったく同じ変数を使用している場合、変数を異なる順序で定義する必要があります。

プライマリキーは外部キーによって参照されるまで個別のデータファイルの値に影響します。

注: null 以外の制約が、新しいプライマリキー制約の定義に使用されている変数に存在する場合、そのプライマリキー制約によって既存の null 以外の制約は置換されます。

FOREIGN KEY (*variables*) REFERENCES *table-name* <ON DELETE *referential-action*> <ON UPDATE *referential-action*>	外部キー、つまりその値が別のデータファイルのプライマリキー変数の値にリンクされている一連の変数を指定します。参照アクションは、外部キーによって参照されるプライマリキー変数の値が更新されるときに強制されます。参照アクションには、RESTRICT、SET NULL、CASCADE の 3 つの種類があります。

次の操作は、RESTRICT 参照アクションによって 実行できます。

削除操作	外部キー値が削除された値に一致しない場合にのみプライマリキー行を削除します。
更新操作	外部キー値が更新対象となる現在の値と一致しない場合にのみプライマリキー値を更新します。

次の操作は、SET NULL 参照アクションによって 実行できます。

削除操作	プライマリキー行を削除し、対応する外部キー値を NULL に設定します。
更新操作	プライマリキー値を変更し、一致するすべての外部キー値を NULL に設定します。

次の操作は、CASCADE 参照アクションによって 実行できます。

更新操作	プライマリキー値を変更し、さらに一致する外部キー値を同じ値に変更します。CASCADE は、削除操作にはサポートされていません。

デフォルト　RESTRICT は、参照アクションが指定されていない場合、デフォルトのアクションです。

要件 重複するプライマリキー制約と外部キー制約を定義するとき、つまりデータファイルの変数がプライマリキー定義と外部キー定義の一部であるとき、次のアクションが必要です。

まったく同じ変数を使用する場合、その変数は異なる順序で定義する必要があります。

外部キーの更新および削除の参照アクションは両方とも RESTRICT でなければなりません。

操作 SET NULL 参照アクションまたは CASCADE 参照アクションを強制する前に、プライマリキーを参照し、意図した操作に RESTRICT を指定するその他の外部キーがあるかどうかを確認します。RESTRICT が指定されている、または制約がデフォルト値に戻る場合、外部キー値が更新対象または削除対象となる値に一致しない場合を除き、RESTRICT はすべての外部キーに対して強制されます。

オプション引数

constraint-name
制約のオプションの名前です。名前は有効な SAS 名である必要があります。制約名を入力しない場合、デフォルトの名前が生成されます。このデフォルトの制約名には、次の形式があります。

表 15.12 RESTRICT=の使用

デフォルト名	制約の種類
NM*xxxx*	Not Null
UN*xxxx*	Unique
CK*xxxx*	Check
PK*xxxx*	Primary key
FK*xxxx*	Foreign key

xxxx は、0001 から始まるカウンターです。

注: 名前 PRIMARY、FOREIGN、MESSAGE、UNIQUE、DISTINCT、CHECK および NOT は、*constraint-name* の値として使用することはできません。

MESSAGE='*message-string*'
message-string は、データが制約に違反した場合にログに書き込まれるエラーメッセージのテキストです。

```
ic create not null(socsec)
  message='Invalid Social Security number';
```

長さ　メッセージの最大長は 250 文字です。

例　次の例に、一貫性制約の作成方法を示します。

```
ic create a = not null(x);
ic create Unique_D = unique(d);
ic create Distinct_DE = distinct(d e);
ic create E_less_D = check(where=(e < d or d = 99));
ic create primkey = primary key(a b);
ic create forkey = foreign key (a b) references table-name
  on update cascade on delete set null;
ic create not null (x);
```

MSGTYPE=USER
一貫性制約エラーメッセージの形式を制御します。デフォルトで MESSAGE= オプションが指定されると、定義したメッセージが制約に関する SAS エラーメッセージにスペースで 区切られて挿入されます。MSGTYPE=USER は、メッセージの SAS 部分を非表示にします。

IC DELETE ステートメント

一貫性制約を削除します。

制限事項:　IC DELETE は、MODIFY RUN グループ内にある必要があります

構文

IC DELETE *constraint-name(s)* | _ALL_;

必須引数

constraint-name(s)
削除する制約を 1 つ以上指定します。たとえば、制約 Unique_D と Unique_E を削除するには、次のステートメントを使用します。ic delete Unique_D Unique_E;

ALL
前の MODIFY ステートメントで指定した SAS データファイルに対するすべての制約を削除します。

IC REACTIVATE ステートメント

非アクティブな外部キー一貫性制約を再アクティブ化します。

制限事項:　IC REACTIVATE は、MODIFY RUN グループ内にある必要があります

構文

IC REACTIVATE *foreign-key-name* REFERENCES *libref*;

必須引数

foreign-key-name
再アクティブ化する外部キーの名前です。

libref
外部キーによって参照されるプライマリキーを含むデータセットを含む SAS ライブラリを参照します。

例

外部キー FKEY をデータセット MYLIB.MYOWN で定義し、FKEY がデータセット MAINLIB.MAIN のプライマリキーにリンクされているとします。一貫性制約がコピー操作または移動操作によって非アクティブ化される場合、次のコードを使用して一貫性制約を再アクティブ化することができます。

```
proc datasets library=mylib;
  modify myown;
  ic reactivate fkey references mainlib;
run;
```

INDEX CENTILES ステートメント

インデックス付き変数のパーセント点統計量を更新します。

制限事項: INDEX CENTILES は、MODIFY RUN グループ内にある必要があります

構文

INDEX CENTILES *index(s)*
</ <REFRESH> <UPDATECENTILES=ALWAYS | NEVER | *integer*>>;

必須引数

index(s)
インデックスを 1 つ以上指定します。

オプション引数

REFRESH
UPDATECENTILES の値に関係なく、パーセント点を即時更新します。

UPDATECENTILES=ALWAYS | NEVER | *integer*
パーセント点がいつ更新されるかを指定します。データセット更新のたびにパーセント点を更新することは実用的ではありません。そのため、インデックス付き変数のパーセント点が更新される前に変更可能なデータ値のパーセントを UPDATECENTILES の値として指定できます。

有効な値は次のとおりです。

ALWAYS | 0	データセットインデックスが変更された場合、データセットのクローズ時にパーセント点を更新します。ALWAYS または 0 を指定しても、結果は同じです。
NEVER | 101	パーセント点を更新しません。NEVER または 101 を指定しても、結果は同じです。
integer	パーセント点が更新される前に更新可能なインデックス付き変数の値のパーセントです。別名は UPDCEN です。デフォルトは 5(パーセント)です。

INDEX CREATE ステートメント

MODIFY ステートメントで指定される SAS データセットの単一インデックスまたは複合インデックスを作成します。

制限事項: INDEX CREATE は、MODIFY RUN グループ内にある必要があります

例: “例 4: SAS データセットの変更” (502 ページ)

構文

INDEX CREATE *index-specification(s)*
</ <NOMISS> <UNIQUE> <UPDATECENTILES=ALWAYS | NEVER | *integer*>>;

必須引数

index-specification(s)
次の形式のうちいずれか、またはその両方になります。

variable
指定した変数上に単一インデックスを作成します。

index=(variables)
複合インデックスを作成します。*index* に指定する名前は、複合インデックスの名前です。有効な SAS 名である必要があります。いずれの変数名、またはその他の複合インデックス名と同じにはできません。

注 変数を少なくとも 2 つ指定する必要があります。インデックス名は、_N_ などの自動変数の予約名、_ALL_などの特殊変数リスト名の使用を避けるなど、SAS 変数名と同じルールに従う必要があります。

オプション引数

NOMISS
すべてのインデックス変数に対する欠損値を含むすべてのオブザベーションをインデックスから除外します。

NOMISS オプションによってインデックスを作成する際、インデックスは WHERE 処理にのみ、および欠損値が WHERE 式を満たしていない場合にのみ使用されます。たとえば、次の WHERE ステートメントを使用すると、欠損値が WHERE 式を満たしているためインデックスは使用されません。

```
where dept ne '01';
```

BY グループ処理は、NOMISS オプションによって作成されるインデックスを無視します。

例 “例 4: SAS データセットの変更” (502 ページ)

UNIQUE
インデックス変数の値の組み合わせが一意になるように指定します。UNIQUE を指定し、複数のオブザベーションのインデックス変数の値が同じ場合、インデックスは作成されません。

例 “例 4: SAS データセットの変更” (502 ページ)

UPDATECENTILES=ALWAYS | NEVER | *integer*
パーセント点がいつ更新されるかを指定します。データセット更新のたびにパーセント点を更新することは実用的ではありません。そのため、インデックス付き変数に対するパーセント点が更新される前に変更可能なデータ値のパーセントを指定できます。有効な値は次のとおりです。

ALWAYS | 0	データセットインデックスが変更された場合、データセットのクローズ時にパーセント点を更新します。ALWAYS または 0 を指定しても、結果は同じです。
NEVER | 101	パーセント点を更新しません。NEVER または 101 を指定しても、結果は同じです。
integer	パーセント点が更新される前に更新可能なインデックス付き変数の値のパーセントを指定します。別名は UPDCEN で、デフォルトは 5(パーセント)です。

INDEX DELETE ステートメント

MODIFY ステートメントで指定した SAS データセットと関連付けられている 1 つ以上のインデックスを削除します。

制限事項: INDEX DELETE ステートメントは、MODIFY RUN グループ内で指定する必要があります

注: CONTENTS ステートメントを使用すると、データセットのすべてのインデックスリストを生成できます。

構文

INDEX DELETE *index(s)* | _ALL_;

必須引数

index(s)
削除するインデックスを 1 つ以上指定します。インデックスは、前の MODIFY ステートメントで指定される SAS データセットの変数に対応している必要があります。単一インデックスと複合インデックスの両方を削除できます。

ALL
一貫性制約によって所有されるインデックスを除く、すべてのインデックスを削除します。インデックスは作成時にユーザー、一貫性制約、またはその両方によって所有されていることがマーク付けされます。インデックスがユーザーと一貫性制約の両方によって所有されている場合、インデックスは IC DELETE ステートメントと INDEX DELETE ステートメントが処理されるまで削除されません。

INFORMAT ステートメント

MODIFY ステートメントで常に指定されるデータセットの変数の入力形式を割り当て、変更、削除します。

制限事項: INFORMAT ステートメントは、MODIFY RUN グループで指定する必要があります

例: “例 4: SAS データセットの変更” (502 ページ)

構文

INFORMAT *variable-1* <*informat-1*>
<*variable-2* <*informat-2*> ...>;

必須引数

variable
割り当て、変更、削除する入力形式の変数を 1 つ以上指定します。入力形式と変数の関連付けを解除する場合、リストの最後に変数をリストします。入力形式はその後に指定しません。

```
informat a b 2. x1-x3 4.1 c;
```

オプション引数

informat
変数の入力形式をステートメントの入力形式の直前に指定します。入力形式を指定しない場合、INFORMAT ステートメントは *variable-list* の変数の既存する入力形式を削除します。

ヒント すべての入力形式をデータセットから削除するには、 ATTRIB ステートメント (394 ページ) と_ALL_キーワードを使用します。

INITIATE ステートメント

SAS データファイルと同じ名前で、データセットの種類が AUDIT の監査ファイルを作成します。

制限事項: INITIATE ステートメントは、AUDIT RUN グループ内で指定する必要があります。

INITIATE ステートメントは監査ファイルごとに一度実行可能な必須のステートメントで、そのファイルに対する最初の AUDIT 実行グループの AUDIT ステートメントの直後におく必要があります。

例: “例 8: 監査ファイルの初期化” (513 ページ)

構文

INITIATE <AUDIT_ALL=NO | YES>;

オプション引数

AUDIT_ALL=NO | YES
記録が中断可能かどうか、監査設定が変更可能かどうかを指定します。AUDIT_ALL=YES は、すべてのイメージが記録され、中断できないように指定します。つまり、LOG ステートメントを使用して、特定のイメージの記録を無効化できません。また、SUSPEND ステートメントを使用してイベントの記録を中断できません。記録を無効化するには、イベントの記録を終了し、監査ファイルを削除する TERMINATE ステートメントを使用する必要があります。

デフォルト NO

例 “例 8: 監査ファイルの初期化” (513 ページ)

詳細

監査ファイルは、SAS データファイルへの追加、削除、更新を記録します。監査証跡は、中断、再開、終了できるようにするには、開始する必要があります。INITIATE ステートメント直前の AUDIT ステートメントは GENNUM=オプションを指定できませんが、指定したファイルが世代データセットグループを識別する場合、INITIATE ステートメントによって作成された監査ファイルは、世代グループで最も新しく作成された世代に追加されます。

次の例では、監査ファイル MyLib.MyFile.audit を作成し、更新をデータファイル MyLib.MyFile.data に書き込み、すべての利用可能なレコードイメージを保存します。

```
proc datasets library=mylib;
```

```
  audit myfile (alter=password);
  initiate;
run;
```

次の例では、AUDIT_ALL=YES を使用して監査ファイルを開始します。

```
proc datasets lib=mylib;  /* all audit image types will be logged
                             and the file cannot be suspended */
   audit myfile (alter=password);
   initiate audit_all=yes;
quit;
```

LABEL ステートメント

MODIFY ステートメントで指定した SAS データセットの変数ラベルを割り当て、変更、削除します。

制限事項: LABEL ステートメントは、MODIFY RUN グループで指定する必要があります

例: “例 4: SAS データセットの変更” (502 ページ)

構文

LABEL *variable-1='label-1'* | ' '
<variable-2=<'label-2' | ' '> ...>;

必須引数

variable=<'label'>
256 文字までのテキスト文字列を指定します。ラベルテキストに単一引用符が含まれている場合、ラベルの前後に二重引用符を使用するか、ラベルテキストで単一引用符を 2 つ使用して文字列を単一引用符で囲みます。ラベルをデータセットから削除するには、引用符に囲まれているブランクに相当するラベルを割り当てます。

範囲 1 - 256 文字

ヒント データセットのすべての変数ラベルを削除するには、 ATTRIB ステートメント (394 ページ) と_ALL_キーワードを使用します。

LOG ステートメント

監査ファイル設定を指定します。

制限事項: LOG ステートメントは、AUDIT RUN グループ内の INITIATE ステートメントの直後に指定する必要があります。

例: “例 8: 監査ファイルの初期化” (513 ページ)

構文

LOG <ADMIN_IMAGE=YES | NO>
<BEFORE_IMAGE=YES | NO>
<DATA_IMAGE=YES | NO>
<ERROR_IMAGE=YES | NO>;

オプション引数

ADMIN_IMAGE=YES | NO
管理イベントが監査ファイルに書き込まれるかどうか(SUSPEND アクションと RESUME アクション)を指定します。

デフォルト YES

ヒント 特定のイメージを書き込まない場合、そのイメージの種類に対して NO を指定します。たとえば、次のコードはエラーイメージの書き込みをオフにしますが、管理イベント、更新前のレコードイメージ、データイメージの書き込みは継続されます。log error_image=no;

BEFORE_IMAGE=YES | NO
更新前のレコードイメージが監査ファイルに書き込まれるかどうかを指定します。

デフォルト YES

DATA_IMAGE=YES | NO
追加、削除、更新された後のレコードイメージが監査ファイルに書き込まれるかどうかを指定します。

デフォルト YES

ERROR_IMAGE=YES | NO
更新後のレコードイメージが監査ファイルに書き込まれるかどうかを指定します。

デフォルト YES

詳細

次の例では、同じ監査ファイルを作成しますが、エラーレコードイメージのみを保存します。

```
proc datasets library=mylib;
  audit myfile (alter=password);
  initiate;
    log admin_image=no
      before_image=no
      data_image=no;
run;
```

MODIFY ステートメント

SAS ファイルの属性、および従属ステートメントの使用により SAS ファイルの変数の属性を変更します。

制限事項: Compute Server で使用しますが、CAS では使用しません。
ATTRIB ステートメントで LENGTH=オプションを使用して変数の長さを変更することはできません。

例: “例 4: SAS データセットの変更” (502 ページ)

構文

MODIFY *SAS-file* <(*options*)>
</ <CORRECTENCODING=*encoding-value*> <DTC=*SAS-date-time*>
<GENNUM=*integer*> <MEMTYPE=*member-type*>>;

必須引数

SAS-file
プロシジャ入力ライブラリに存在する SAS ファイルを指定します。

オプション引数

ALTER=*password-modification*
MODIFY ステートメントで指定される SAS ファイルに対する変更パスワードを割り当て、変更、削除します。*password-modification* は、次のうちいずれかになります。

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

参照項目: “パスワードの操作” (445 ページ)

CORRECTENCODING=*encoding-value*
ファイル内のデータの実際のエンコーディングに一致させるため、ファイルのディスクリプター情報に記録されるエンコーディングインジケーターの変更を可能にします。

参照項目: “CORRECTENCODING=オプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)

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

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

DTC=*SAS-date-time*
作成時に SAS ファイルに挿入される日時スタンプのかわりとなる日時を指定します。このオプションは、各 SAS ファイル名の後にかっこで囲んで使用することはできません。スラッシュの後に DTC=を指定する必要があります。

```
modify mydata / dtc='03MAR00:12:01:00'dt;
```

制限事項 SAS ファイルの作成日時は、ファイルが実際に作成された日時よりも後に設定できません。

DTC=は、暗号化されたファイルまたは順編成ファイルと使用できません。

ヒント COPY プロシジャ、CPORT プロシジャ、SORT プロシジャで DATECOPY オプションを、DATASETS プロシジャで COPY ステートメントを使用する前に、DTC=を使用して SAS ファイルの作成日時を変更します。

GENMAX=*number-of-generations*
バージョンの最大数を指定します。このオプションは SAS ファイル名の後にかっこで囲んで使用します。

デフォルト 0

範囲 0 から 1,000

GENNUM=*integer*
世代データセットの処理を制限します。GENNUM=は、各 SAS ファイル名の後にかっこで囲むか、スラッシュの後に指定します。有効な値は *integer* です。これは、世代グループの特定バージョンを参照する数字です。正の数を指定すると、データセット名に追加される特定の世代番号の絶対参照になります(つまり、**gennum=2** は MYDATA#002 を指定することになります)。負の数を指定すると、最新から最古への基本バージョンに相対する履歴バージョンの相対参照になります(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。デフォルトの 0 を指定すると、基本バージョンが参照されます。

LABEL='*data-set-label*' | ' '
MODIFY ステートメントで指定される SAS データセットのデータセットラベルを割り当て、変更、削除します。単一引用符がラベルで使用されている場合、2 つの単一引用符として記述します。LABEL=または LABEL=' 'が現在のラベルを削除します。

範囲 1 - 256 文字

制限事項 ビューラベルは、ラベルの作成後は更新できません。

ヒント データセットのすべての変数ラベルを削除するには、ATTRIB ステートメント (394 ページ)を使用します。

参照項目: “例 4: SAS データセットの変更” (502 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。MEMTYPE=は、各 SAS ファイル名の後にかっこで囲んで指定できません。スラッシュの後に指定する必要があります。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメントまたは MODIFY ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=DATA となります。

PW=*password-modification*
MODIFY ステートメントで指定された SAS ファイルに対する読み取り、書き込み、変更パスワードを、割り当て、変更、削除します。*password-modification* は、次のうちいずれかになります。

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

READ=*password-modification*
MODIFY ステートメントで指定された SAS ファイルに対する読み取りパスワードを、割り当て、変更、削除します。*password-modification* は、次のうちいずれかになります。

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

参照項目: “パスワードの操作” (445 ページ)

例 “例 4: SAS データセットの変更” (502 ページ)

SORTEDBY=*sort-information*
データの現在の並べ替え方法を指定します。ファイルとともに並べ替え情報が保存されますが、データが指定した方法で並べ替えられていることは確認されません。*sort-information* は、次のうちいずれかになります。

by-clause </ *collate-name*>
データの現在の並べ替え方法を示します。*by-clause* の値は、PROC SORT ステップの BY ステートメントで使用できる変数とオプションです。*collate-name* は、並べ替えに使用される照合シーケンスです。デフォルトで、照合順序はホスト動作環境の照合順序です。

NULL
既存のソートインジケーターを削除します。

制限事項 データは指定した順序で並べ替える必要があります。データが指定した順序になっていない場合、並べ替えは実行されません。

ヒント MODIFY SORTEDBY オプションの使用時は、番号範囲リストまたはコロンリストを使用することもできます。詳細については、"データセット名リスト" (*SAS プログラマガイド: エッセンシャル*)を参照してください。

例 "例 4: SAS データセットの変更" (502 ページ)

TYPE=*special-type*
SAS データセットの特殊データセットの種類を割り当て、変更します。SAS では、次の点について確認しません。

- TYPE=オプションで指定する SAS データセットの種類(長さが 8 文字以下であるかどうかのチェックは除く)
- SAS データセットの構造が指定した種類に適している

注 TYPE=オプションと MEMTYPE=オプションを混同しないようにしてください。TYPE=オプションは、特殊な SAS データセットの種類を指定します。MEMTYPE= オプションは、SAS ライブラリ内の 1 つ以上の SAS ファイルの種類を指定します。

ヒント ほとんどの SAS データセットには、特殊な種類はありません。ただし、特定の SAS プロシジャは、CORR プロシジャのように、多くの特殊な SAS データセットを作成できます。また、SAS/STAT ソフトウェアと SAS/EIS ソフトウェアでは、特殊データセットの種類がサポートされています。

WRITE=*password-modification*
MODIFY ステートメントで指定された SAS ファイルに対する書き込みパスワードを、割り当て、変更、削除します。*password-modification* は、次のうちいずれかになります。

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

参照項目: "パスワードの操作" (445 ページ)

詳細

データセットラベルと変数ラベルの変更

LABEL オプションは、データセット内のデータセットラベルまたは変数ラベルのいずれかを変更できます。データセットラベルを変更するには、次の構文を使用します。

```
modify datasetname(label='Label for Data Set');
run;
```

データセット内の 1 つ以上の変数ラベルを変更できます。データセット内の変数ラベルを変更するには、次の構文を使用します。

```
modify datasetname;
   label  variablename='Label for Variable';
run;
```

同じ PROC DATASETS でデータセットラベルと変数ラベルの両方を変更する例については、“例 4: SAS データセットの変更” (502 ページ)を参照してください。

パスワードの操作

パスワードの割り当て、変更、削除には、ファイル上に現在存在する最上位レベルの保護に対してパスワードを指定する必要があります。

パスワードの割り当て

```
/* assigns a password to an unprotected file */
modify colors (pw=green);

/* assigns an Alter password to an already read-protected SAS data set */
modify colors (read=green alter=red);
```

パスワードの変更

```
/* changes the Write password from YELLOW to BROWN */
modify cars (write=yellow/brown);

/* uses Alter access to change unknown Read password to BLUE */
modify colors (read=/blue alter=red);
```

パスワードの削除

```
/* removes the Alter password RED from STATES */
modify states (alter=red/);

/* uses Alter access to remove the Read password */
modify zoology (read=green/ alter=red);

/* uses PW= as an alias for either WRITE= or ALTER= to remove unknown
  Read password */
modify biology (read=/ pw=red);
```

世代グループの操作

世代数の変更

```
/* changes the number of generations on data set A to 99 */
modify A (genmax=99);
```

パスワードの削除

```
/* removes the Alter password RED from STATES#002 */
modify states (alter=red/) / gennum=2;
```

QUIT ステートメント

実行されなかったステートメントを実行し、プロシジャを終了します。

構文

QUIT;

REBUILD ステートメント

無効化されたインデックスと一貫性制約を元に戻す、または削除するかどうかを指定します。

デフォルト: インデックスと一貫性制約の再作成

制限事項: REBUILD ステートメントが適用されるのは、バージョン 7 以降で作成されたデータセットのみです

構文

REBUILD *SAS-file* </ <ENCRYPTKEY=*key-value*>
<ALTER=*password*> <GENNUM=*n*> <MEMTYPE=*member-type*> <NOINDEX>>;

必須引数

SAS-file
無効化されたインデックスと一貫性制約を含む SAS データファイルを指定します。番号範囲リストまたはコロンリストを使用することもできます。

参照項目: *"データセット名リスト" (SAS プログラマガイド: エッセンシャル)*.

オプション引数

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

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

ALTER=*alter-password*
REBUILD ステートメントで指定される変更保護された SAS ファイルに対する変更パスワードを与えます。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

GENNUM=*integer*
世代データセットの処理を制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。有効な値は *integer* です。これは、世代グループの特定バージョンを参照する数字です。正の数を指定すると、データセット名に追加される特定の世代番号の絶対参照になります(つまり、**gennum=2** は MYDATA#002 を指定することになります)。負の数を指定すると、最新から最古への基本バージョンに相対する履歴バージョンの相対参照になります(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。デフォルトの 0 を指定すると、基本バージョンが参照されます。

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメントまたは REBUILD ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

NOINDEX
無効化されたインデックスと一貫性制約の削除を指定します。

制限事項 NOINDEX オプションは、参照一貫性制約を 1 つ以上含むデータファイルに使用できません。

詳細

DLDMGACTION=NOINDEX データセットまたはシステムオプションが指定され、損傷したデータファイルが見つかると、次の項目が実行されます。

- インデックスと一貫性制約を含まないデータファイルの修復
- インデックスファイルの削除
- 無効化されたインデックスと一貫性制約を反映するためのデータファイルの更新
- INPUT モードでのみ開くデータファイルの制限
- 次の警告の SAS ログへの書き込み

```
WARNING: SAS data file MYLIB.MYFILE.DATA was damaged and
      has been partially repaired. To complete the repair,
      execute the DATASETS procedure REBUILD statement.
```

REBUILD ステートメントは、すべてのデータファイルの無効化されたインデックスと一貫性制約を再作成または削除することにより、損傷した SAS データファイルの修復を完了します。REBUILD ステートメントは、新しいインデックスファイルを処理し、インデックスと一貫性制約を復元するために、メンバーレベルロックを作成し、使用します。

インデックスファイルを再作成し、インデックスと一貫性制約を復元するには、次のコードを使用します。

```
proc datasets library=mylib;
rebuild myfile
/alter=password
 gennum=n
 memtype=mytype;
```

無効化されたインデックスと一貫性制約を削除するには、次のコードを使用します。

```
proc datasets library=mylib;
rebuild myfile /noindex;
```

REBUILD ステートメントを実行すると、データファイルは INPUT モードに制限されなくなります。

REBUILD ステートメントのデフォルトは、インデックス、一貫性制約、インデックスファイルを再作成することです。

データファイルに 1 つ以上の参照一貫性制約が含まれ、NOINDEX オプションを REBUILD ステートメントで使用する場合、次のエラーメッセージが SAS ログに書き込まれます。

```
Error: Unable to rebuild data file MYLIB.MYFILE.DATA using the
              NOINDEX option because the data file contains referential
              constraints. Resubmit the REBUILD statement without the
              NOINDEX option to restore the data file.
```

RENAME ステートメント

MODIFY ステートメントで指定した SAS データセット内の変数の名前を変更します。

制限事項: RENAME ステートメントは、MODIFY RUN グループで指定する必要があります

例: “例 4: SAS データセットの変更” (502 ページ)

構文

RENAME *old-name-1=new-name-1*
<*old-name-2=new-name-2* ...>;

必須引数

old-name=new-name
MODIFY ステートメントで指定されたデータセット内の変数の名前を変更します。*old-name* は、データセットにすでに存在する変数である必要があります。*new-name* は、データセットにすでに存在する変数の名前またはインデックスの名前にすることはできません。また、新しい名前は有効な SAS 名である必要があります。

たとえば、Oxygen データセットに OXYGEN という名前の変数が含まれているとします。次のコードは、OXYGEN 変数の名前を intake に変更します。

```
modify oxygen;
   rename oxygen=intake;
```

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Label
1	AGE	Num	8	
6	MAXPULSE	Num	8	
4	RSTPULSE	Num	8	
5	RUNPULSE	Num	8	
3	RUNTIME	Num	8	
2	WEIGHT	Num	8	
7	intake	Num	8	Intake Measurement

詳細

old-name が SAS データセット内に存在しない、または *new-name* がすでに存在する場合、PROC DATASETS は RENAME ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。

RENAME ステートメントを使用して単一インデックスがある変数の名前を変更する場合、ステートメントはインデックスの名前も変更します。

名前を変更している変数が複合インデックスで使用されると、その複合インデックスは自動的に新しい変数名を参照します。ただし、変数の名前をすでに複合インデックスに使用されている名前に変更しようとすると、エラーメッセージが表示されます。

REPAIR ステートメント

永久ライブラリ内の破損した SAS データセットまたはカタログを、使用に適した状態に復元します。

注: REPAIR ステートメントは、現在のバックアップを保持することの代替になるものではありません。

構文

REPAIR *SAS-file(s)*
</ <ENCRYPTKEY=*key-value*>
<ALTER=*alter-password*> <GENNUM=*integer*> <MEMTYPE=*member-type*>>;

必須引数

SAS-file(s)
プロシジャ入力ライブラリ内の 1 つ以上の SAS データセットまたはカタログを指定します。番号範囲リストまたはコロンリストを使用することもできます。

参照項目: “データセット名リスト” (*SAS プログラマガイド: エッセンシャル*)

オプション引数

ALTER=*alter-password*
REPAIR ステートメントで指定される変更保護された SAS ファイルに対する変更パスワードを与えます。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

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

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

GENNUM=*integer*
世代データセットの処理を制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。有効な値は *integer* です。これは、世代グループの特定バージョンを参照する数字です。正の数を指定すると、データセット名に追加される特定の世代番号の絶対参照になります(つまり、**gennum=2** は MYDATA#002 を指定することになります)。負の数を指定すると、最新から最古への基本バージョンに相対する履歴バージョンの相対参照になります(つまり、**gennum=-1** は最新の履歴バージョンを参照します)。デフォルトの 0 を指定すると、基本バージョンが参照されます。

参照項目: “世代データセットの処理制限” (463 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメントまたは REPAIR ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目: “処理するメンバーの種類の制限” (465 ページ)

詳細

注意
データセットに大規模な破損がある場合、REPAIR ステートメントで修復することはできません。 SAS データセットまたは補助ファイル(インデックス、監査、拡張属性のファイル)が常駐しているデバイスが破損した場合は、破損したデータセットおよび補助ファイルをバックアップデバイスから復元する必要があります。

REPAIR ステートメントが役立つ最も一般的な状況は、次のとおりです。

- SAS データセットまたはカタログの更新中にシステムエラーが発生した場合。

REPAIR ステートメントを SAS データセットに使用する場合、データセットのすべてのインデックスが再作成されます。また、データセットが使用に適した状態に復元されますが、復元されたデータセットにシステム障害が発生する前に行った最終更新が含まれていない場合があります。REPAIR ステートメントを使用して、PROC SORT ステップで FORCE オプションを使用することにより破壊されたインデックスを再作成することはできません。

- SAS データセットまたはカタログエントリの書き込み中に I/O エラーが発生した場合。

  SAS データセットを保存するディスクがファイルの書き込みが完了する前にいっぱいになった場合、データセットの書き込みを行うステップは失敗し、SAS ログにエラーが書き込まれます。REPAIR ステートメントはデータセットヘッダーを修復して、使用に適した状態にデータセットを復元します。データセットの完全性と整合性に問題がある可能性があります。最も信頼性が高いのは、データセットをバックアップから再作成する方法です。

修復プロセス中にオブザベーションが失われることがあります。

REPAIR ステートメントをカタログに使用する場合、REPAIR ステートメントがエントリを復元したかどうかというメッセージが表示されます。カタログ全体が壊されている可能性がある場合、REPAIR ステートメントはカタログのすべてのエントリを復元しようとします。単一のエントリだけが破壊されている可能性のある場合、たとえば単一のエントリが更新中で、ディスクフルになると、ほとんどのシステムでは問題の発生時に開いていたエントリだけが破壊されている可能性があります。この場合、REPAIR ステートメントはそのエントリだけを修復しようとします。復元されたカタログ内のエントリには、システム障害または I/O エラーの前に行われた最終更新が含まれていないことがあります。REPAIR ステートメントは、切り捨てられたデータを含むエントリに対し警告メッセージを発行します。

破損したカタログを修復するには、カタログを更新できるバージョンの SAS を使用する必要があります。

指定したライブラリに存在しない SAS ファイルに対し REPAIR ステートメントを発行すると、PROC DATASETS は REPAIR ステートメントを含む実行グループの処理を停止し、エラーメッセージを発行します。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを使用します。

クロス環境データアクセス(CEDA)を使用して破損した外部 SAS データセットを処理する場合は、PROC DATASETS REPAIR ステートメントを使用して修復を行う前に、データセットをネイティブ環境に戻す必要があります。CEDA では、損傷したデータセットの修復に必要な更新処理がサポートされていません。

CEDA の詳細については、“クロス環境データアクセス(CEDA)の定義” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

RESUME ステートメント

監査ファイルへのイベントの記録を、中断されている場合は再開します。

制限事項: RESUME ステートメントは、AUDIT RUN グループ内で指定する必要があります。

例: “例 8: 監査ファイルの初期化” (513 ページ)

構文

RESUME;

詳細

RESUME、SUSPEND、TERMINIATE、USER_VAR、LOG などのその他の監査関連ステートメントは、INITIATE ステートメントがサブミットされるまで監査ファイルに対して有効になりません。詳細については、“INITIATE ステートメント” (438 ページ)を参照してください。

RESUME ステートメントには、LOG ステートメントの ADMIN_IMAGE=YES オプションが必要です。このオプションは、管理イベントが監査ファイルに書き込まれるかどうか(SUSPEND アクションと RESUME アクション)を指定します。詳細については、“LOG ステートメント” (439 ページ)を参照してください。

SAVE ステートメント

ライブラリ内の、SAVE ステートメントにリストされている以外のすべての SAS ファイルを削除します。

例: “例 3: SAS ファイルを削除せずに保存する” (499 ページ)

構文

SAVE *SAS-file(s)* </ MEMTYPE=*member-type*>;

必須引数

SAS-file(s)
SAS ライブラリから削除しない SAS ファイルを 1 つ以上指定します。

注: SAS ファイルに ALTER=パスワードが割り当てられている場合、そのパスワードを SAS ファイルを削除するために指定する必要があります。

オプション引数

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメントまたは SAVE ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目: “処理するメンバーの種類の制限” (465 ページ)

例 “例 3: SAS ファイルを削除せずに保存する” (499 ページ)

詳細

SAS-file の SAS ファイルがプロシジャ入力ライブラリに存在しない場合、PROC DATASETS は SAVE ステートメントを含む RUN グループの処理を停止し、エラーメッセージを発行します。この動作を無効にするには、PROC DATASETS ステートメントで NOWARN オプションを指定します。

SAVE ステートメントは SAS データセットを削除する際に、これらのデータセットと関連付けられているインデックスも削除します。(削除対象の SAS データセットに ALTER=パスワードが割り当てられている場合、その ALTER=パスワードを SAS データセットを削除するために指定する必要があります。

注意
RUN グループのサブミット時に、ライブラリとライブラリメンバーは即座に削除されます。 削除操作は、開始前に確認を求められません。SAVE ステートメントは、1 つの操作で多数の SAS ファイルを削除します。どの種類の SAS ファイルが保存され、どの種類が削除されるかについて、MEMTYPE=オプションはどのように影響を与えるかを理解していることを確認してください。

SAVE ステートメントを世代グループと使用する場合、SAVE ステートメントは基本バージョンとすべての 履歴バージョンを単位として扱います。特定のバージョンを保存することはできません。

SELECT ステートメント

コピーする SAS ファイルを選択します。

制限事項: SELECT ステートメントは、COPY ステートメントの後に指定する必要があります
SELECT ステートメントは、EXCLUDE ステートメントで同じ COPY ステップに記述できません

例: “例 2: SAS ファイルの操作” (494 ページ)

構文

SELECT *SAS-file(s)*

</ <ENCRYPTKEY=*key-value*> <ALTER=*alter-password*> <MEMTYPE=*member-type*>>;

必須引数

SAS-file(s)
コピーする SAS ファイルを 1 つ以上指定します。指定するすべての SAS ファイルが、COPY ステートメントの IN=オプションで指定されるライブラリ参照名によって参照されるデータライブラリ内にある必要があります。SAS ファイルに世代グループが含まれる場合、SELECT ステートメントでは特定のバージョンを選択できないため、すべての世代がコピーされます。

オプション引数

ALTER=*alter-password*
ライブラリ間で移動中の変更保護されている SAS ファイルに対する変更パスワードを与えます。SAS ファイルを SAS ライブラリから移動し削除しているため、変更アクセス権限が必要になります。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

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

参照項目: “AES 暗号化データセットの追加” (389 ページ)

MEMTYPE=*member-type*
処理を 1 つのメンバーの種類に制限します。このオプションは各 SAS ファイル名の後にかっこで囲んで指定することも、スラッシュの後に指定することもできます。

別名 MTYPE=

MT=

デフォルト PROC DATASETS ステートメント、COPY ステートメント、または SELECT ステートメントで MEMTYPE=オプションを指定しない場合、デフォルトは MEMTYPE=ALL となります。

参照項目: “SAS ファイルのコピー/移動時のメンバーの種類の指定” (417 ページ) および“処理するメンバーの種類の制限” (465 ページ)

例 “例 2: SAS ファイルの操作” (494 ページ)

詳細

複数の同じ名前のファイルの選択

SELECT ステートメントで複数の SAS ファイルをリストするためのショートカットを使用できます。

表 15.13 SELECT ステートメントの使用

表記	意味
x1–xn	ファイル X1 から Xn を指定します。番号は連続している必要があります。
x:	文字 X で始まるすべてのファイルを指定します。

SUSPEND ステートメント

監査ファイルへのイベントの記録を中断しますが、監査ファイルは削除しません。

制限事項: SUSPEND ステートメントは、AUDIT RUN グループ内で指定する必要があります。

例: “例 8: 監査ファイルの初期化” (513 ページ)

構文

SUSPEND;

詳細

SUSPEND、RESUME、TERMINIATE、USER_VAR、LOG などのその他の監査関連ステートメントは、INITIATE ステートメントがサブミットされるまで監査ファイルに対して有効になりません。詳細については、“INITIATE ステートメント” (438 ページ)を参照してください。

SUSPEND ステートメントには、LOG ステートメントの ADMIN_IMAGE=YES オプションが必要です。このオプションは、管理イベントが監査ファイルに書き込まれるかどうか(SUSPEND アクションと RESUME アクション)を指定します。詳細については、“LOG ステートメント” (439 ページ)を参照してください。

TERMINATE ステートメント

イベントの記録を終了し、監査ファイルを削除します。

制限事項: TERMINATE ステートメントは、AUDIT RUN グループ内で指定する必要があります。

例: “例 8: 監査ファイルの初期化” (513 ページ)

構文

TERMINATE;

詳細

TERMINIATE、SUSPEND、RESUME、USER_VAR、LOG などのその他の監査関連ステートメントは、INITIATE ステートメントがサブミットされるまで監査ファイルに対して有効になりません。詳細については、“INITIATE ステートメント” (438 ページ)を参照してください。

USER_VAR ステートメント

オブザベーションへの更新のたびに任意変数が監査ファイルに書き込まれるように定義します。USER_VAR を使用する場合、INITIATE ステートメントの後に指定してください。

制限事項: USER_VAR ステートメントは、AUDIT RUN グループ内で指定する必要があります。

USER_VAR ステートメントはオプションです。USER_VAR ステートメントを指定する場合は、適切な監査ファイルのために INITIATE ステートメントの直後に指定する必要があります。

例: “例 8: 監査ファイルの初期化” (513 ページ)

構文

USER_VAR *variable-name-1*<$> <*length*> <LABEL='*variable-label*' > <*variable-name-2* <$> <*length*> <LABEL='*variable-label*' > ...>;

必須引数

variable-name
変数の名前です。

オプション引数

$
変数が文字変数であることを示します。

length
変数の長さを指定します。

デフォルト 8

LABEL='*variable-label*'
変数のラベルを指定します。

XATTR ADD ステートメント

拡張属性を変数またはデータセットに追加します。

制限事項: Compute Server で使用しますが、CAS では使用しません。
XATTR ADD ステートメントは、MODIFY RUN グループ内で指定する必要があります。
世代データセットは拡張属性をサポートしません。

注: 拡張属性には、数値または文字値を指定できます。
文字値の中に空白スペースがある場合、欠損値とみなされます。欠損数値も使用できます。
拡張属性名は、SAS の命名規則に従っている必要があります。SAS 名の最大文字数は 32 文字です。

例: “例 9: 拡張属性” (519 ページ)

構文

XATTR ADD DS *attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 ...>;

or

XATTR ADD VAR *variable-name-1 (attribute-name-1=attribute-value-1*
< attribute-name-2=attribute-value-2 ...>)
<variable-name-2 (attribute-name-1=attribute-value-1
<attribute-name-2=attribute-value-2 ...>)>;

必須引数

文字値の場合、*attribute-value* を引用符で囲む必要があります(*"attribute-value"*)。

XATTR ADD DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*
拡張属性をデータセットに追加します。拡張属性がすでに存在する場合、エラーが返されます。

XATTR ADD VAR *variable-name-1 (attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>) <variable-name-2 (attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>)>*
拡張属性を変数に追加します。拡張属性がすでに存在する場合、エラーが返されます。

詳細

拡張属性は(名前, 値)ペアから構成されます。新しい属性を追加し、その属性がすでに存在する場合、エラーが SAS ログに書き込まれます。

XATTR DELETE ステートメント

すべての拡張属性を SAS ファイルから削除します。

制限事項: Compute Server で使用しますが、CAS では使用しません。
XATTR DELETE ステートメントは、MODIFY RUN グループ内で指定する必要があります

例: “例 9: 拡張属性” (519 ページ)

構文

XATTR DELETE;

必須引数

XATTR DELETE
　すべての拡張属性をデータセットから削除します。

詳細

XATTR DELETE ステートメントを使用すると、すべての拡張属性をデータセットから削除できます。このコマンドを使用した後は、どの拡張属性も存在しなくなります。次の例では、すべての拡張属性をデータセットから削除します。

```
proc datasets lib=library_name nolist;
  modify dataset_name;
    xattr delete;
  run;
quit;
```

XATTR OPTIONS ステートメント

拡張属性に必要なオプションを指定します。現在、SEGLEN=のみが有効なオプションです。

制限事項: Compute Server で使用しますが、CAS では使用しません。
XATTR OPTIONS ステートメントは、MODIFY RUN グループ内で指定する必要があります

例: “例 9: 拡張属性” (519 ページ)

構文

XATTR OPTIONS <SEGLEN=*number-of-bytes*>;

必須引数

SEGLEN=*number-of-bytes*
文字属性値を保持するストレージ要素長を指定します。値は 1〜32,760 バイトまでです。

デフォルト 256

XATTR REMOVE ステートメント

変数またはデータセットから拡張属性を削除します。

制限事項: Compute Server で使用しますが、CAS では使用しません。
XATTR REMOVE ステートメントは、MODIFY RUN グループ内で指定する必要があります

例: “例 9: 拡張属性” (519 ページ)

構文

XATTR REMOVE DS *attribute-name(s)* ;

or

XATTR REMOVE VAR *variable-name-1* (*attribute-name(s)*)
<*variable-name-2* (*attribute-name(s)* ...)>;

必須引数

XATTR REMOVE DS *attribute-name(s)*
拡張属性をデータセットから削除します。

XATTR REMOVE VAR *variable-name-1* (*attribute-name(s)*) <*variable-name-2* (*attribute-name(s)*)>
拡張属性を変数から削除します。

詳細

作成した拡張属性をもう必要としない場合、XATTR REMOVE ステートメントを使用するとその属性を変数またはデータセットから削除できます。XATTR REMOVE ステートメントによって削除されるのは、指定する拡張属性に限られます。

XATTR SET ステートメント

拡張属性を変数またはデータセットに更新または追加します。

制限事項: Compute Server で使用しますが、CAS では使用しません。
XATTR SET ステートメントは、MODIFY RUN グループ内で指定する必要があります
世代データセットは拡張属性をサポートしません。

注: 拡張属性には、数値または文字値を指定できます。
文字値の中に空白スペースがある場合、欠損値とみなされます。欠損数値も使用できます。
拡張属性名は、SAS の命名規則に従っている必要があります。SAS 名の最大文字数は 32 文字です。

例: “例 9: 拡張属性” (519 ページ)

構文

XATTR SET DS *attribute-name-1=attribute-value-1*
<attribute-name-2="attribute-value-2" ...>;

or

XATTR SET VAR *variable-name-1 (attribute-name-1=attribute-value-1*
<attribute-name-2=attribute-value-2 ... >)
<variable-name-2 (attribute-name-1=attribute-value-1
<attribute-name-2=attribute-value-2> ...)>;

必須引数

文字値の場合、*attribute-value* を引用符で囲む必要があります(*"attribute-value"*)。

XATTR SET DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*
拡張属性をデータセットに更新または追加します。データセットに拡張属性が存在しない場合、拡張属性が追加されます。拡張属性が存在する場合、指定値によって更新されます。

XATTR SET VAR *variable-name-1* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*) <*variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*)>
拡張属性を変数に更新または追加します。変数と拡張属性の組み合わせが存在しない場合、拡張属性が追加されます。組み合わせが存在する場合、拡張属性は指定値によって更新されます。

詳細

拡張属性が存在するかどうかわからない場合は、XATTR SET ステートメントを使用してください。拡張属性が存在する場合、その属性は更新されます。拡張属性が存在しない場合、拡張属性が追加されます。XATTR SET ステートメントは、変数またはデータセットに拡張属性が存在しない場合でも、その属性を定義します。XATTR ADD を使用するとき、その値を使用する既存の拡張属性がある場合、エラーが発生します。まだ存在しない拡張属性に XATTR UPDATE を使おうとしても、エラーが発生します。XATTR SET を使用すると、変数またはデータセットの拡張属性を定義できます。拡張属性が存在していなかった場合、これからは存在します。拡張属性が存在していた場合、新しい値が指定されます。

XATTR UPDATE ステートメント

拡張属性を変数またはデータセットに更新します。

制限事項: Compute Server で使用しますが、CAS では使用しません。
XATTR UPDATE ステートメントは、MODIFY RUN グループ内で指定する必要があります

注: 文字値の中に空白スペースがある場合、欠損値とみなされます。欠損数値も使用できます。
拡張属性名は、SAS の命名規則に従っている必要があります。SAS 名の最大文字数は 32 文字です。

例: “例 9: 拡張属性” (519 ページ)

構文

XATTR UPDATE DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*;

or

XATTR UPDATE VAR *variable-name-1 (attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>)*
<*variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*)>;

必須引数

文字値の場合、*attribute-value* を引用符で囲む必要があります(*"attribute-value"*)。

XATTR UPDATE DS *attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*
データセット内の拡張属性を更新します。拡張属性が存在しない場合、エラーが SAS ログに書き込まれます。

XATTR UPDATE VAR *variable-name-1* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*) *<variable-name-2* (*attribute-name-1=attribute-value-1 <attribute-name-2=attribute-value-2 ...>*)>
変数の拡張属性を更新します。変数と拡張属性の組み合わせが見つからない場合、エラーが SAS ログに書き込まれます。

詳細

既存の拡張属性に変更を加えるには、XATTR UPDATE ステートメントを使用します。存在しない拡張属性を更新しようと試みると、エラーが SAS ログに書き込まれます。

使用法: DATASETS プロシジャ

DATASETS プロシジャでパスワードを使用する

DATASETS プロシジャのいくつかのステートメントでは、SAS ファイルのパスワードを操作するオプションがサポートされています。これらのオプション ALTER=、PW=、READ=、WRITE=もデータセットオプションです。[1] SAS ファイルへのパスワードの影響については、*"パスワード" (SAS プログラマガイド: エッセンシャル)*を参照してください。

ステートメント AGE、CHANGE、DELETE、EXCHANGE、REPAIR、SELECT でパスワード保護されている SAS ファイルを使用している場合、PROC DATASETS ステートメントまたはそれに従属するステートメントで ALTER=および PW=パスワードオプションを指定できます。

注: ALTER=オプションは、COPY(ファイルの移動時)ステートメントおよび MODIFY ステートメントについては多少結果が異なります。詳細については、 COPY ステートメント (407 ページ)および MODIFY ステートメント (441 ページ)を参照してください。

1. APPEND ステートメントと CONTENTS ステートメントでは、これらのオプションは、SAS データセットオプションと同じように、SAS データセット名の後にかっこで囲んで使用します。

パスワードの検索は、次の順序で実行されます。

1 従属ステートメントの SAS ファイル名の後のかっこ内。かっこ内で使用される場合、オプションはそのオプションの直前の名前のみ参照します。データライブラリ内の複数の SAS ファイルを使用しており、SAS ファイルにそれぞれ異なるパスワードがある場合、個別の名前の後にパスワードオプションをかっこで囲んで指定する必要があります。

次のステートメントでは、ALTER=オプションは、SAS ファイル Bones にのみパスワード red を付与します。

```
delete xplant bones(alter=red);
```

2 従属ステートメントのスラッシュ(/)の後。スラッシュの後にパスワードオプションを使用する場合、同じオプションが SAS ファイル名の後のかっこ内に記述されない限り、オプションはステートメントで指定したすべての SAS ファイルを参照します。この方法は、使用している複数の SAS ファイルすべてのパスワードが同一である場合に使用します。

次のステートメントでは、かっこ内の ALTER=オプションが SAS ファイル Chest にパスワード red を、スラッシュの後の ALTER=オプションが SAS ファイル Virus にパスワード blue を付与します。

```
delete chest(alter=red) virus / alter=blue;
```

3 PROC DATASETS ステートメント内。PROC DATASETS ステートメントでのパスワード指定は、ライブラリで使用しているすべての SAS ファイルのパスワードが同一である場合に便利です。この場合、オプションはかっこ内に指定しないでください。

次の PROC DATASETS ステップで、PW=オプションは SAS ファイル Insulin と Abneg に対してパスワード red を賦与します。

```
proc datasets pw=red;
  delete insulin;
  contents data=abneg;
run;
```

注: SELECT ステートメントの SAS ファイルに対するパスワードの場合、PROC DATASETS ステートメントの前に COPY ステートメントが検索されます。

世代データセットの処理制限

DATASETS プロシジャの複数のステートメントでは、世代データセットの処理を制限するための GENNUM=オプションがサポートされています。GENNUM=はデータセットオプションでもあります。[1] 世代データセットの要求および使用方法については、"Generation Data Sets"を参照してください。世代データセットを修復する方法については、“Generation Data Sets” (*SAS V9 LIBNAME Engine: Reference*)を参照してください。

1. APPEND ステートメントと CONTENTS ステートメントでは、GENNUM=は、SAS データセットオプションと同じように、SAS データセット名の後にかっこで囲んで使用します。

ステートメント AUDIT、CHANGE、DELETE、MODIFY、REPAIR の世代グループを使用している場合、PROC DATASETS ステートメントまたは下位ステートメントの処理を特定のバージョンに制限できます。

注: GENNUM=オプションは、MODIFY ステートメントについては多少結果が異なります。MODIFY ステートメント (441 ページ)を参照してください。

注: ステートメント AGE、COPY、EXCHANGE、SAVE については、処理を特定のバージョンに制限できません。これらのステートメントは、世代グループ全体に適用されます。

世代指定が次の順序で検索されます。

1 従属ステートメントの SAS データセット名の後のかっこ内。かっこ内で使用される場合、オプションはそのオプションの直前の名前のみ参照します。使用しているデータライブラリ内の複数の SAS データセットのバージョンにそれぞれ異なる世代バージョンを指定する場合、個別の名前の後に GENNUM=をかっこで囲んで指定する必要があります。

   次のステートメントで、GENNUM=オプションは SAS データセット Bones に対してのみ世代グループのバージョンを指定します。

   ```
   delete xplant bones (gennum=2);
   ```

2 従属ステートメントのスラッシュ(/)の後。スラッシュの後に GENNUM=オプションを使用する場合、同じオプションが SAS データセット名の後のかっこ内に記述されない限り、オプションはステートメントで指定したすべての SAS データセットを参照します。この方法は、使用している複数のファイルすべてのバージョンを同一にする場合に使用します。

   次のステートメントでは、かっこ内の GENNUM=オプションにより SAS データセット Chest の世代バージョンが指定され、スラッシュの後の GENNUM=オプションにより SAS データセット Virus の世代バージョンが指定されます。

   ```
   delete chest (gennum=2) virus / gennum=1;
   ```

3 PROC DATASETS ステートメント内。ライブラリで使用しているすべての SAS データセットのバージョンを同一にする場合、PROC DATASETS ステートメントで世代バージョンを指定すると便利です。この場合、オプションはかっこ内に指定しないでください。

   次の PROC DATASETS ステップでは、GENNUM=オプションは SAS ファイル Insulin と Abneg に対して世代バージョンを指定します。

   ```
   proc datasets gennum=2;
     delete insulin;
     contents data=abneg;
   run;
   ```

   注: SELECT ステートメントの SAS ファイルに対する世代バージョンの場合、PROC DATASETS ステートメントの前に COPY ステートメントが検索されます。

処理するメンバーの種類の制限

PROC DATASETS ステートメント内

下位ステートメントの複数のメンバーの種類を参照し、メンバーの種類を PROC DATASETS ステートメントで指定した場合、そのメンバーの種類をすべて PROC DATASETS ステートメントに含めます。元の PROC DATASETS ステートメントのメンバーの種類のみ有効です。次の例は、複数のメンバーの種類をリストしたものです。

```
proc datasets lib=library memtype=(data view);
```

従属ステートメント内

次の従属ステートメントで MEMTYPE=オプションを使用し、処理可能なメンバーの種類に制限します。

AGE CHANGE DELETE EXCHANGE

EXCLUDE REPAIR SAVE SELECT

注: MEMTYPE=オプションは、ステートメント CONTENTS、COPY、MODIFY については多少結果が異なります。詳細については、CONTENTS ステートメント (400 ページ)、COPY ステートメント (407 ページ)、および MODIFY ステートメント (441 ページ)を参照してください。

プロシジャでは、MEMTYPE=が次の順序で検索されます。

1 SAS ファイル名の直後のかっこ内。かっこ内で使用される場合、MEMTYPE=オプションはそのオプションの直前の SAS ファイルのみ参照します。たとえば、次のステートメントは、HOUSE.DATA、LOT.CATALOG、SALES.DATA を削除します。DELETE ステートメントのデフォルトのメンバーの種類が DATA であるためです。(各ステートメントのデフォルトの種類については表 15.47 (467 ページ)を参照してください。)

```
delete house lot(memtype=catalog) sales;
```

2 ステートメントの最後のスラッシュ(/)の後。スラッシュの後に使用される場合、MEMTYPE=オプションは SAS ファイル名の後のかっこ内に記述されない限り、ステートメントで指定したすべての SAS ファイルを参照します。たとえば、次のステートメントは Lotpix.catalog、Regions.data、および Appl.catalog を削除します。

```
delete lotpix regions(memtype=data) appl / memtype=catalog;
```

3 PROC DATASETS ステートメント内。たとえば、この DATASETS プロシジャは Appl.catalog を削除します。

```
proc datasets memtype=catalog;
  delete appl;
run;
```

注: EXCLUDE ステートメントと SELECT ステートメントを使用する際は、プロシジャは MEMTYPE=オプションを PROC DATASETS ステートメントの前に COPY ステートメントで検索します。詳細については、“SAS ファイルのコピー/移動時のメンバーの種類の指定” (417 ページ) 参照してください。

4 (デフォルト値について)MEMTYPE=オプションを下位ステートメントまたは PROC DATASETS ステートメントで指定しない場合、下位ステートメントのデフォルト値により、処理可能なメンバーの種類が決定されます。

メンバーの種類

MEMTYPE=オプションに指定できる値は、次のとおりです。

ACCESS
: アクセスディスクリプターファイル (SAS/ACCESS ソフトウェアによって作成)

ALL
: すべてのメンバーの 種類

CATALOG
: SAS カタログ

DATA
: SAS データファイル

FDB
: 財務データベース

MDDB
: 多次元データベース

PROGRAM
: 保存されたコンパイル済み SAS プログラム

VIEW
: SAS ビュー

次のテーブルに、各ステートメントで使用できるメンバーの種類を示します。

表 15.14 従属ステートメントと適切なメンバーの種類

ステートメント	適切なメンバーの種類	デフォルトのメンバーの種類
AGE	ACCESS、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	DATA
CHANGE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
CONTENTS	ALL、DATA、VIEW	DATA[1]
COPY	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
DELETE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	DATA
EXCHANGE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
EXCLUDE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
MODIFY	ACCESS、DATA、VIEW	DATA
REPAIR	ALL、CATALOG、DATA	ALL[2]
SAVE	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL
SELECT	ACCESS、ALL、CATALOG、DATA、FDB、MDDB、PROGRAM、VIEW	ALL

1 CONTENTS ステートメントで DATA=_ALL_の場合、デフォルトは ALL です。ALL には、DATA と VIEW のみ含まれます。
2 ALL には、DATA と CATALOG のみ含まれます。

PROC DATASETS のサンプル出力

DATASETS プロシジャの処理内容は次のとおりです。

- CONTROL ライブラリのすべてのデータセットを HEALTH ライブラリにコピー
- HEALTH ライブラリのコンテンツをリスト表示
- HEALTH ライブラリから SYNDROME データセットを削除

- PRENAT データセットの名前を INFANT に変更

SAS ログは次のように出力されます。

```
LIBNAME control 'SAS-library-1';
LIBNAME health 'SAS-library-2';

proc datasets memtype=data;
  copy in=control out=health;
run;

proc datasets library=health memtype=data details;
  delete syndrome;
  change prenat=infant;
run;
quit;
```

例のコード 15.1 PROC DATASETS のログ

```
744  proc datasets library=health memtype=data details;
                          Directory

Libref         HEALTH
Engine         V9
Physical Name  /home/userid/Documents and Settings/myfil/My Documents/procdatasets/health
Filename       /home/userid/Documents and Settings/myfile/My Documents/procdatasets/health


            Member  Obs, Entries                          File
 #  Name     Type    or Indexes   Vars  Label                Size  Last Modified

 1  ALL      DATA       23       17                     13312  12Sep07:10:57:50
 2  BODYFAT  DATA       83       13  California Results      13312  12Sep07:10:57:54
 3  CONFOUND DATA        8        4                      5120  12Sep07:10:57:50
 4  CORONARY DATA       39        4                      5120  12Sep07:10:57:50
 5  DRUG1    DATA        6        2  JAN2005 DATA          5120  12Sep07:10:57:50
 6  DRUG2    DATA       13        2  MAY2005 DATA          5120  12Sep07:10:57:50
 7  DRUG3    DATA       11        2  JUL2005 DATA          5120  12Sep07:10:57:50
 8  DRUG4    DATA        7        2  JAN2002 DATA          5120  12Sep07:10:57:50
 9  DRUG5    DATA        1        2  JUL2002 DATA          5120  12Sep07:10:57:50
10  GROUP    DATA      148       11  Test Subjects         33792  12Sep07:13:01:16
    GROUP    INDEX       1                                9216  12Sep07:13:01:16
11  GRPOUT   DATA       11       40                     17408  13Dec10:11:40:24
12  GRPOUT1  DATA       11       40                     17408  13Dec10:10:28:32
13  INFANT   DATA      149        6                     17408  12Sep07:10:57:52
14  MLSCL    DATA       32        4  Multiple Sclerosis Data   5120  12Sep07:10:57:52
15  NAMES    DATA        7        4                      5120  12Sep07:10:57:52
16  OXYGEN   DATA       31        7                     17408  12Sep07:13:01:16
17  PERSONL  DATA      148       11                     25600  12Sep07:10:57:52
18  PHARM    DATA        6        3  Sugar Study           5120  12Sep07:10:57:52
19  POINTS   DATA        6        6                      5120  12Sep07:10:57:52
20  RESULTS  DATA       10        5                      5120  12Sep07:10:57:54
21  SLEEP    DATA      108        6                      9216  12Sep07:10:57:54
22  TEST2    DATA       15        5                      5120  12Sep07:10:57:54
23  TRAIN    DATA        7        2                      5120  12Sep07:10:57:54
24  VISION   DATA       16        3                      5120  12Sep07:10:57:54
25  WEIGHT   DATA        1        2                      5120  12Sep07:10:57:50
26  WGHT     DATA       83       13                     13312  12Sep07:10:57:54
745    delete syndrome;
746    change prenat=infant;
747  run;
ERROR: The file HEALTH.PRENAT (memtype=DATA) was not found, but appears on a CHANGE statement.
748  quit;
749
750  proc datasets memtype=data;
```

結果: DATASETS プロシジャ

SAS ログとしてのディレクトリリスト

PROC DATASETS ステートメント NOLIST を指定しない限り、プロシジャ入力ライブラリの SAS ファイルをリストする オプションが指定されています。 NOLIST オプションは、ログに移動するプロシジャ結果の作成を防止します。MEMTYPE=オプションを指定すると、指定された種類のみリストされます。 DETAILS オプションを指定すると、PROC DATASETS は **Obs, Entries or Indexes**、**Vars**、**Label** という追加情報列を出力します。

SAS 出力としてのディレクトリリスト

CONTENTS ステートメントは、DIRECTORY オプションを使用する場合、または DATA=_ALL_を指定する場合、プロシジャ入力ライブラリのディレクトリをリストします。

ディレクトリのみ必要な場合、NODS オプションと、DATA=オプションの_ALL_キーワードを使用します。 NODS オプションは、SAS データセットの記述を抑制します。ディレクトリのみ出力されます。

注: CONTENTS ステートメントは、ディレクトリを出力データセットに配置しません。NODS オプションを使用して出力データセットを作成しようとすると、空の出力データセットを受け取ります。 SQL プロシジャを使用して、SAS ライブラリに関する情報を含む SAS データセットを作成します。

注: ODS RTF 出力先を指定する場合、PROC DATASETS 出力は SAS ログと ODS 出力領域の両方に移動します。NOLIST オプションはこのいずれにも出力しません。SAS ログのみの出力を確認するには、メンバーディレクトリを除外として指定し、ODS EXCLUDE ステートメントを使用します。

プロシジャ出力

CONTENTS ステートメント

プロシジャ出力を生成する PROC DATASETS の唯一のステートメントは、CONTENTS ステートメントです。 このセクションでは、Health ライブラリと Group データセットの CONTENTS ステートメントからの出力を示します。出力は次のとおりです。

説明が必要な出力内の項目についてのみ説明します。

HEALTH ライブラリの内容

Health ライブラリ内のすべてのファイルとカタログのリストを以下に示します。

Directory	
Libref	HEALTH
Engine	V9
Physical Name	c:\procdatasets1\health
Filename	c:\procdatasets1\health
Owner Name	BUILTIN\Administrators
File Size	12KB
File Size (bytes)	12288

#	Name	Member Type	File Size	Last Modified
1	BODYFAT	DATA	8KB	09/03/2014 10:35:01
2	CONFOUND	DATA	8KB	09/03/2014 10:35:01
3	CORONARY	DATA	8KB	09/03/2014 10:35:01
4	FORMATS	CATALOG	17KB	11/16/2011 13:53:09
5	GROUP	DATA	32KB	09/03/2014 10:35:02
6	GRPOUT	DATA	144KB	01/15/2016 15:37:10
7	GRPOUT1	DATA	144KB	08/06/2015 09:54:32
8	INFANT	DATA	17KB	09/12/2007 10:57:52
9	MLSCL	DATA	8KB	09/03/2014 10:35:02
10	NAMES	DATA	8KB	09/03/2014 10:35:02
11	OXYGEN	DATA	16KB	09/03/2014 10:35:02
12	PERSONL	DATA	32KB	09/03/2014 10:35:02
13	PHARM	DATA	8KB	09/03/2014 10:35:02
14	POINTS	DATA	8KB	09/03/2014 10:35:02
15	POSTDRUG	CATALOG	61KB	11/16/2011 13:53:08
16	PRENAT	DATA	24KB	09/03/2014 10:35:02
17	RESULTS	DATA	8KB	09/03/2014 10:35:03
18	SLEEP	DATA	12KB	09/03/2014 10:35:03
19	SPDATA	VIEW	5KB	03/24/2005 13:12:22
20	SYNDROME	DATA	16KB	09/03/2014 10:35:03
21	TENSION	DATA	8KB	09/03/2014 10:35:03
22	TRAIN	DATA	8KB	09/03/2014 10:35:03
23	VISION	DATA	8KB	09/03/2014 10:35:03
24	WEIGHT	DATA	24KB	09/03/2014 10:35:03
25	WGHT	DATA	24KB	09/03/2014 10:35:03

データセット属性

次の出力で表示される選択フィールドの説明です。

Member Type
ライブラリメンバーの種類(DATA または VIEW)です。

Created
データセットが作成された日付と時刻を示します。この日付と時刻は、TIMEZONE=システムオプションの設定を反映します。TIMEZONE=システムオプションが設定されていない場合、計算サーバーセッションが実行されているローカルタイムゾーンが使用されます。

Last Modified
データセットが最後に変更された日付と時刻を示します。この日付と時刻は、TIMEZONE=システムオプションの設定を反映します。TIMEZONE=システムオプションが設定されていない場合、計算サーバーセッションが実行されているローカルタイムゾーンが使用されます。

Protection
SAS データセットの Read、Write、Alter のパスワード保護がされているかどうかを示します。

Data Set Type
特殊データセットの種類(CORR、COV、SSPC、EST、FACTOR など)を指定します(存在する場合)。

Observations
ファイル内の現在のオブザベーションの合計数です。たとえば、非常に大きなデータセットで、オブザベーションの数が倍精度浮動小数点で表現可能な最大整数値を超えている場合、カウントは欠損値として表示されます。

Deleted Observations
削除がマーク付けされているオブザベーションの数です。 これらのオブザベーションは、**Observations** フィールドに表示されているオブザベーションの合計数に含まれません。 非常に大きなデータセットで、削除されたオブザベーションの数が倍精度整数で保存可能な数を超えている場合、カウントは欠損値として表示されます。また、COMPRESS=YES オプションを REUSE=YES オプションと POINTOBS=NO オプションのいずれか、または両方と一緒に使用する場合、**Deleted Observations** のカウントには欠損値が表示されます。

Compressed
データセットが圧縮されるかどうかを示します。 データセットが圧縮される場合、出力には追加項目、**Reuse Space** が(値 YES または NO とともに)含まれます。この項目は、オブザベーションの削除時に利用可能になるスペースを 再利用するかどうかを示します。

Sorted
データセットが並べ替えられるかどうかを示します。 データセットを PROC SORT、PROC SQL で並べ替える場合、または並べ替え情報を SORTEDBY=データセットオプションで指定する場合、ここでは値 YES が表示され、出力に追加セクションがあります。 詳細については、“並べ替え情報” (477 ページ) を参照してください。

Data Representation
データがコンピューターアーキテクチャ上または動作環境内で表現される形式です。たとえば IBM PC では、文字データは ASCII エンコーディングとバイトスワップ整数によって表現されます。ネイティブなデータ表現は、データ表現がファイルにアクセスしている CPU と比較される環境を参照します。

Encoding
エンコーディング値です。 エンコーディングは、コンピューターが使用可能なコードポイントと呼ばれる数値にマッピングされた文字(通常の文字、表語文字、数字、句読点、記号、コントロール文字など)のセットです。コードポイントは、エンコーディングメソッドを適用する際に文字セットの文字に割り当てられます。

アウトプット 15.4　Class データセットの属性

Printing Class

The CONTENTS Procedure

Data Set Name	SASHELP.CLASS
Member Type	DATA
Engine	V9
Created	10/22/2019 09:38:47
Last Modified	10/22/2019 09:38:47
Protection	
Data Set Type	
Label	Student Data
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64, LINUX_POWER_6
Encoding	us-ascii ASCII (ANSI)

エンジンおよび動作環境依存情報

CONTENTS ステートメントは、動作環境固有情報とエンジン固有情報を生成します。 この情報は、動作環境によって異なります。

アウトプット 15.5　Group データセットのエンジン/ホスト依存情報

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1632
Obs in First Data Page	19
Number of Data Set Repairs	0
Filename	/opt/sas/viya/home/SASFoundation/sashelp/class.sas7bdat
Release Created	V.0305M0
Host Created	Linux
Inode Number	70297014
Access Permission	rw-r--r--
Owner Name	sas
File Size	128KB
File Size (bytes)	131072

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

変数と属性のアルファベット順リスト

次の出力で選択されている列の説明です。

#
オブザベーション内の各変数の論理位置です。この数字は、変数の定義時に変数に割り当てられます。

Variable
各変数の名前です。 デフォルトで、変数はアルファベット順に表示されます。

注: 変数名が X1、X10、X2 の実際の照合順序ではなく、X1、X2、X10 の順序で表示されるように並べ替えられます。アンダースコアと数字を含む変数名は、非標準の並べ替え順序で表示されることがあります。 たとえば、P25 と P75 は P2_5 の前に表示されます。

Type
変数の種類として文字または数値を指定します。

Len
変数の長さを指定します。これは、SAS データセット内の変数の値をそれぞれ保存するために使用されるバイト数です。

Transcode
文字変数がトランスコードされるかどうかを指定します。 属性が NO の場合、トランスコードは実行されません。デフォルトで、必要な場合に文字変数はトランスコードされます。トランスコードの詳細については、*SAS 各国語サポート (NLS): リファレンスガイド*を参照してください。

注: SAS データセット内の変数に出力形式、入力形式、ラベルが関連付けられていない場合、またはすべての変数が Transcode=YES に設定されている場合、属性の列は表示されません。

アウトプット 15.6　Group データセットの変数と属性のリスト

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat
9	BIRTH	Num	8		
4	CITY	Char	15		
3	FNAME	Char	15		
10	HIRED	Num	8	DATE7.	DATE7.
11	HPHONE	Char	12		
1	IDNUM	Char	4		
7	JOBCODE	Char	4		
2	LNAME	Char	15		
8	SALARY	Num	8	COMMA8.	
6	SEX	Char	2		
5	STATE	Char	3		

インデックスと属性のアルファベット順リスト

次の出力に表示されるセクションは、データセットにインデックスが関連付けられている場合にのみ表示されます。

#
各インデックスの数を示します。 インデックスは定義された順序で連続して番号が付けられます。

Index
各インデックスの名前を表示します。単一インデックスの場合、インデックスの名前はデータセットの変数と同じです。

Unique Option
インデックスに一意の値を含める必要があるかどうかを示します。 列に YES が含まれている場合、インデックス変数の値の組み合わせは各オブザベーションに対して一意です。

Nomiss Option
インデックスがすべてのインデックス 変数の欠損値を除外するかどうかを示します。 列に YES が含まれている場合、インデックスにはすべてのインデックス変数に対する欠損値を含むオブザベーションは含まれていません。

# of Unique Values
インデックス内の一意の値の数を示します。

Variables
複合インデックス内の各変数を指定します。

アウトプット 15.7　GROUP データセットのインデックスと属性のリスト

Alphabetic List of Indexes and Attributes

#	Index	Unique Option	NoMiss Option	# of Unique Values	Variables
1	vital	YES	YES	148	BIRTH SALARY

並べ替え情報

次の出力に表示されているセクションは、**Sorted** フィールドの値が YES の場合にのみ表示されます。

Sortedby
データの現在の並べ替え方法を示します。 このフィールドには、PROC SORT の BY ステートメントで使用する変数とオプション、PROC SQL の列名 SORTEDBY= オプションで指定する値のいずれかが含まれます。

Validated
データが PROC SORT または SORTEDBY を使用して並べ替えられたかどうかを示します。PROC SORT または PROC SQL がデータセットを並べ替えた場合、値は YES です。 SORTEDBY=データセットオプションでソートインジケーターを割り当てた場合、値は NO となります。

Character Set
データの並べ替えに使用される文字セットです。 このフィールドの値は、ASCII、EBCDIC、PASCII のうちのいずれかになります。

Collating Sequence
データセットの並べ替えに使用される照合順序です。変換テーブル名、エンコーディング値、またはデータセットを言語的に並べ替える場合は LINGUISTIC を使用できます。 文字セットと異なる照合順序を指定しない場合、このフィールドは表示されません。

データセットが言語的に並べ替えられると、ロケール、照合形式などの追加の言語照合順序情報が **Collating Sequence** の後に表示されます。言語照合に指定可能な照合ルールのリストです。

Sort Option
データセットの並べ替え時に PROC SORT が NODUPKEY オプションを使用したかどうかを示します。 このフィールドは、PROC SORT ステートメントでこのオプションを使用しなかった場合は表示されません(非表示)。

アウトプット 15.8　Group データセット並べ替え情報

Sort Information	
Sortedby	LNAME
Validated	NO
Character Set	ANSI

PROC DATASETS と ODS (Output Delivery System)

ほとんどの SAS プロシジャは、メッセージを SAS ログに送信し、プロシジャ結果を出力に送信します。PROC DATASETS は、プロシジャ結果を SAS ログとプロシジャ出力ファイルの両方に送信するため、固有のものです。ODS へのインターフェイスが作成された際に、すべてのプロシジャ 結果(ログとプロシジャ出力ファイルの両方からの)が ODS で利用可能にする必要があると判断されました。この機能を実装し、旧リリースとの互換性を保持するため、ODS へのインターフェイスは通常のインターフェイスと 多少異なっている必要がありました。

デフォルトで、PROC DATASETS ステートメントはそれ自体で 2 つの出力オブジェクト Members と Directory を生成します。これらのオブジェクトは､SAS ログに送られます。CONTENTS ステートメントは、デフォルトで 3 つの出力オブジェクト Attributes、EngineHost、および Variables を生成します。(さまざまなオプションの使用により､その他の出力オブジェクトが追加されます)。 これらのオブジェクトは、プロシジャ出力ファイルに送られます。ODS 出力先(HTML、RTF、PRINTER など)を開いている場合、これらのオブジェクトはすべてデフォルトでその 出力先に送られます。

その他のプロシジャに対して実行するのと同じように、ODS SELECT ステートメントと ODS EXCLUDE ステートメントを使用して、どのオブジェクトがどの出力先に送られるかを制御できます。ただし、PROC DATASETS と ODS 間の固有のインターフェイスのため、ODS SELECT ステートメントまたは ODS EXCLUDE ステートメントでキーワード LISTING を使用する際は、ログとリストの両方に影響します。

ODS テーブル名

PROC DATASETS と PROC コンテンツは、それらが作成するテーブル毎に名前を割り当てます。これらの名前を使用して、Output Delivery System (ODS)を使用してテーブルを選択し、出力データセットを作成する際にそのテーブルを参照できます。

PROC CONTENTS は、CONTENTS ステートメントを使用した PROC DATASETS と同じ ODS テーブルを生成します。

表 15.15 CONTENTS ステートメントなしで DATASETS プロシジャによって生成される ODS テーブル

ODS テーブル	説明	テーブルの生成
Directory	一般ライブラリ情報	NOLIST オプションを指定しない場合
Members	ライブラリメンバー情報	NOLIST オプションを指定しない場合

表 15.16 CONTENTS ステートメントで PROC CONTENTS と PROC DATASETS によって生成される ODS テーブル名

ODS テーブル	説明	テーブルの生成
属性	データセット属性	SHORT オプションを指定しない場合
Directory	一般ライブラリ情報	DATA=<*libref*.>_ALL_または DIRECTORY オプションを指定する場合 [1]
EngineHost	エンジンおよび動作環境情報	SHORT オプションを指定しない場合
IntegrityConstraints	一貫性制約の詳細リスト	データセットに一貫性制約があり、SHORT オプションを指定しない場合
IntegrityConstraints Short	一貫性制約の簡潔リスト	データセットに一貫性制約があり、SHORT オプションを指定する場合
Indexes	インデックスの詳細リスト	データセットがインデックス化され、SHORT オプションを指定しない場合
IndexesShort	インデックスの簡潔リスト	データセットがインデックス化され、SHORT オプションを指定する場合
Members	ライブラリメンバー情報	DATA=<*libref*.>_ALL_または DIRECTORY オプションを指定する場合 [1]

ODS テーブル	説明	テーブルの生成
Position	データセットの論理位置別の変数の詳細リスト	VARNUM オプションを指定し、SHORT オプションを指定しない場合
PositionShort	データセットの論理位置別の変数の簡潔リスト	VARNUM オプションと SHORT オプションを指定する場合
PositionVarchar	varchar タイプを含む位置	データセットに varchar があり、VARNUM を指定し、SHORT オプションを指定しない場合
Sortedby	詳細並べ替え情報	データセットが並べ替えられ、SHORT オプションを指定しない場合
SortedbyShort	簡潔並べ替え情報	データセットが並べ替えられ、SHORT オプションを指定する場合
Variables	アルファベット順の変数の詳細リスト	SHORT オプションを指定しない場合
VariablesShort	アルファベット順の変数の簡潔リスト	SHORT オプションを指定する場合
VariablesVarchar	varchar タイプを含む変数	データセットに varchar がある場合で、SHORT オプションを指定しない場合

1 PROC DATASETS の場合、NOLIST オプションと、DIRECTORY オプションまたは DATA=<*libref*.>_ALL_のいずれかが指定されると、NOLIST オプションは無視されます。

出力データセット

CONTENTS ステートメント

CONTENTS ステートメントは、出力データセットを生成する DATASETS プロシジャの唯一のステートメントです。

OUT=データセット

CONTENTS ステートメントの OUT=オプションは、出力データセットを作成します。各 DATA=データセットの各変数には、OUT=データセットにオブザベーションが 1 つあります。出力データセットの変数は次のとおりです。

CHARSET
データセットの並べ替えに使用される文字セット。 値は、ASCII、EBCDIC、PASCII のいずれかです。 データセットにソートインジケーターが保存されていない場合、ブランクが表示されます。

COLLATE
データセットの並べ替えに使用される照合順序。 入力データセットのソートインジケーターに照合順序が含まれていない場合、ブランクが表示されます。

COMPRESS
データセットが圧縮されるかどうかを示します。

CRDATE
データセットが作成された日付です。

DELOBS
データセットの削除がマーク付けられているオブザベーションの数です。(オブザベーションは削除のマーク付け設定が可能ですが、SAS/FSP ソフトウェアの FSEDIT プロシジャを使用する際は実際には削除されません)。

ENCRYPT
データセットが暗号化されるかどうかを示します。

ENGINE
データセット間の読み取りと書き込みに使用されるメソッドの名前です。

FLAGS
SQL ビューの変数が保護されているかどうか(**P**)、派生変数に影響(**C**)するかどうかを示します。

P
変数が保護されていることを示します。変数の値は表示できますが、更新できません。

C
変数が派生変数に影響するかどうかを示します。

P または **C** が SQL ビューに適用されない場合、またはデータセットビューである場合、FLAG の値はブランクとなります。

FORMAT
可変長形式。 変数に出力形式を関連付けない場合、FORMAT の値はブランクとなります。

FORMATD
変数に出力形式を関連付けるときに指定する小数点以下の桁数。 出力形式の小数点以下桁数を指定しない場合、FORMATD の値は 0 となります。

FORMATL
出力形式の長さ。 出力形式と変数を関連付ける際に出力形式の長さを指定する場合、指定する長さは FORMATL の値です。 出力形式と変数を関連付ける際に出力形式の長さを指定しない場合、FORMATL の値は、FMTLEN オプションを使用する場合は出力形式のデフォルト長、FMTLEN オプションを使用しない場合は 0 となります。

GENMAX
世代グループのバージョンの最大数。

GENNEXT
世代グループの次の世代番号。

GENNUM
バージョン番号。

IDXCOUNT
データセットのインデックスの数。

IDXUSAGE
インデックス内の変数の使用。 可能な値は次のとおりです。

NONE
変数はインデックスの一部ではありません。

SIMPLE
変数に単一インデックスがあります。 その他の変数はインデックスに含まれていません。

COMPOSITE
変数は複合インデックスの一部です。

BOTH
この変数には単一インデックスがあり、またこの変数は複合インデックスの一部です。

INFORMAT
変数の入力形式。 この変数に入力形式を関連付けない場合、値はブランクとなります。

INFORMD
入力形式と変数を関連付けるときに指定する小数点以下の桁数です。 入力形式と変数を関連付ける際に小数点以下の桁数を指定しない場合、値は 0 です。

INFORML
入力形式の長さ。 入力形式と変数を関連付ける際に入力形式の長さを指定する場合、指定する長さは INFORML の値です。 変数にこの入力形式を関連付ける際に入力形式の長さを指定しない場合、INFORML の値は、FMTLEN オプションを使用する場合は入力形式のデフォルト長、FMTLEN オプションを使用しない場合は 0 となります。

JUST
位置合わせ(0=左、1=右)。

LABEL
変数ラベル(指定されていない場合はブランク)。

LENGTH
変数長。

LIBNAME
データライブラリに使用されるライブラリ参照名。

MEMLABEL
この SAS データセットのラベル(ラベルがない場合はブランク)。

MEMNAME
変数を含む SAS データセット。

MEMTYPE
ライブラリメンバーの種類(DATA または VIEW)

MODATE
データセットが最後に変更された日付。

NAME
変数名。

NOBS
データセットのオブザベーションの数。

NODUPKEY
NODUPKEY オプションが PROC SORT ステートメントで入力データセットの並べ替えに使用されたかどうかを示します。

NPOS
データセットの変数の最初の文字の物理位置。

POINTOBS
データセットがオブザベーションによって処理可能かどうかを示します。

PROTECT
保護レベルの最初の文字。 PROTECT の値は、次のうち 1 つ以上となります。

A
データセットが変更保護されていることを示します。

R
データセットが読み取り保護されていることを示します。

W
データセットが書き込み保護されていることを示します。

REUSE
オブザベーションが圧縮データセットから削除されたときに利用可能になるスペースを再利用する必要があるかどうかを示します。 データセットが圧縮されていない場合、REUSE 変数の値は NO となります。

SORTED
値は、入力データセットの並べ替えの特性によって異なります。 取り得る値は次のとおりです。

.(ピリオド)
並べ替えられていない場合。

0
並べ替えられているが、検証されていない場合。

1
並べ替えられ、検証されている場合。

SORTEDBY
値は、並べ替えにおける変数の役割によって異なります。取り得る値は次のとおりです。

.(ピリオド)
変数が入力データセットの並べ替えに使用されなかった場合。

n
n は、並べ替えにおける変数の位置を示す整数である場合。 *n* の負の値は、データセットが変数の降順で並べ替えられることを示します。

TRANSCOD
変数がトランスコードされるかどうかを指定します。

TYPE
変数の種類(1=数値、2=文字)。

TYPEMEM
　特殊データセットの種類(TYPE=値が指定されていない場合はブランク)。

VARNUM
　データセットの変数番号。 変数は表示される順序で番号が付けられます。

出力データセットは、変数 LIBNAME と MEMNAME によって並べ替えられます。

注: 変数名は、値 X1、X2、X10 が X1、X10、X2 の実際の照合順序ではなく、X1、X2、X10 の順序でリストされるように並べ替えられます。 そのため、後続のステップで MEMNAME で BY ステートメントを使用する場合、出力データセットで最初に PROC SORT ステップを実行します。BY ステートメントでは、NOTSORTED オプションを使用することもできます。

次は、Group データセットから作成された出力データセットの例です。 “例 5: SAS データセットを記述” (505 ページ) および “プロシジャ出力” (471 ページ)に示されています。

Health.Grpout のサイズのため、次の出力は 5 つのセクションになります。

アウトプット 15.9　出力データセットの例 — セクション 1

Obs	LIBNAME	MEMNAME	MEMLABEL	TYPEMEM	NAME	TYPE	LENGTH	VARNUM
1	HEALTH	GROUP			BIRTH	1	8	9
2	HEALTH	GROUP			CITY	2	15	4
3	HEALTH	GROUP			FNAME	2	15	3
4	HEALTH	GROUP			HIRED	1	8	10
5	HEALTH	GROUP			HPHONE	2	12	11
6	HEALTH	GROUP			IDNUM	2	4	1
7	HEALTH	GROUP			JOBCODE	2	4	7
8	HEALTH	GROUP			LNAME	2	15	2
9	HEALTH	GROUP			SALARY	1	8	8
10	HEALTH	GROUP			SEX	2	2	6
11	HEALTH	GROUP			STATE	2	3	5

アウトプット 15.10　出力データセットの例 — セクション 2

LABEL	FORMAT	FORMATL	FORMATD	INFORMAT	INFORML	INFORMD
		0	0		0	0
		0	0		0	0
		0	0		0	0
	DATE	7	0	DATE	7	0
		0	0		0	0
		0	0		0	0
		0	0		0	0
		0	0		0	0
	COMMA	8	0		0	0
		0	0		0	0
		0	0		0	0

アウトプット 15.11　出力データセットの例 — セクション 3

JUST	NPOS	NOBS	ENGINE	CRDATE	MODATE	DELOBS
1	8	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
0	58	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
0	43	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
1	16	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
0	82	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
0	24	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
0	78	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
0	28	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
1	0	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
0	76	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0
0	73	148	V9	03SEP14:10:35:02	03SEP14:10:35:02	0

アウトプット 15.12　出力データセットの例 — セクション 4

IDXUSAGE	MEMTYPE	IDXCOUNT	PROTECT	FLAGS	COMPRESS	REUSE	SORTED	SORTEDBY
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.
NONE	DATA	0	---	---	NO	NO	.	.

アウトプット 15.13　出力データセットの例 — セクション 5

CHARSET	COLLATE	NODUPKEY	NODUPREC	ENCRYPT	POINTOBS	GENMAX	GENNUM	GENNEXT	TRANSCOD
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES
		NO	NO	NO	YES	0	.	.	YES

OUT2=データセット

CONTENTS ステートメントの OUT2=オプションは、インデックスと一貫性制約に関する情報を含む出力データセットを作成します。出力データセットの変数は次のとおりです。

IC_OWN
　インデックスが一貫性制約によって所有される場合、YES を含みます。

INACTIVE
　一貫性制約が非アクティブの 場合、YES を含みます。

LIBNAME
データライブラリに使用されるライブラリ参照名。

MEMNAME
変数を含む SAS データセット。

MG
IC CREATE ステートメントのメッセージ=の値(使用される場合)。

MSGTYPE
一貫性制約に違反せず、メッセージを指定しない限り、値はブランクです。

NAME
インデックスまたは一貫性制約の名前。

NOMISS
NOMISS オプションがインデックスに定義されている場合、YES を含みます。

NUMVALS
インデックス内の重複しない値の数(パーセント点に対して表示)。

NUMVARS
インデックスまたは一貫性制約に関連する変数の数。

ONDELETE
外部キー一貫性制約に対し、該当する場合(IC CREATE ステートメントの ON DELETE オプション)は RESTRICT または SET NULL を含みます。

ONUPDATE
外部キー一貫性制約に対し、該当する場合(IC CREATE ステートメントの ON UPDATE オプション)は RESTRICT または SET NULL を含みます。

RECREATE
インデックスまたは一貫性制約の再作成に必要な SAS ステートメント。

REFERENCE
外部キー一貫性制約に対し、参照されるデータセットの名前を含みます。

TYPE
種類。インデックスの場合、値は"Index"で、一貫性制約の場合、値は一貫性制約の種類(Not Null、Check、Primary Key など)です。

UNIQUE
UNIQUE オプションがインデックスに定義されている場合、YES を含みます。

UPERC
最終更新以降に更新されたインデックスのパーセント(パーセント点に対して表示)。

UPERCMX
更新をトリガーするインデックス更新のパーセント(パーセント点に対して表示)。

WHERE
一貫性制約のチェックに対し、WHERE ステートメントを含みます。

例: DATASETS プロシジャ

例 1: データセットのすべてのラベルと 出力形式の削除

要素:

PROC DATASETS ステートメントオプション
- LIB=
- MEMTYPE=
- CONTENTS ステートメント
- MODIFY ステートメント

CONTENTS プロシジャ

FORMAT プロシジャ

OPTIONS ステートメント

TITLE ステートメント

詳細

この例は、次のタスクを示しています。

- システムオプションを設定する
- ユーザー定義 FORMAT を作成する
- データセットを作成する
- データセットからラベルと形式を削除する
- PROC CONTENTS を使用して、ラベルと形式を含むまたは含まないデータを表示する

プログラム

```
options ls=79 nodate center;
title ;

libname mylib '/home/userid/mylib';

proc format;
```

```
  value clsfmt 1='Freshman' 2='Sophomore' 3='Junior' 4='Senior';
run;

data mylib.class;
  format z clsfmt.;
  label x='ID NUMBER'
    y='AGE'
    z='CLASS STATUS';
  input x y z;
datalines;
1 20 4
2 18 1
;

proc contents data=mylib.class;
run;

proc datasets lib=mylib memtype=data;
  modify class;
   attrib _all_ label=' ';
   attrib _all_ format=;
contents data=mylib.class;
run;
quit;
```

プログラムの説明

システムオプションと LIBNAME ステートメントを設定します。 この例では、LIBNAME は MyLib です。CENTER オプションの指定によって、SAS プロシジャ出力は中央揃えで配置されます。NODATE オプションは、日時が出力されないように指定します。LS=オプションでは、SAS ログと出力の行サイズを指定します。TITLE ステートメントと後ろの空白スペースによって 、計算サーバーセッションの既存のタイトルが削除されます。

```
options ls=79 nodate center;
title ;

libname mylib '/home/userid/mylib';
```

ユーザー定義の出力形式を値は CLSFMT で作成します。

```
proc format;
  value clsfmt 1='Freshman' 2='Sophomore' 3='Junior' 4='Senior';
run;
```

CLASS という名前のデータセットを作成します。 変数 Z に CLSFMT 出力形式を使用します。そして変数 X、Y と Z にラベルを作成します。

```
data mylib.class;
  format z clsfmt.;
  label x='ID NUMBER'
    y='AGE'
    z='CLASS STATUS';
  input x y z;
datalines;
1 20 4
2 18 1
```

```
;
```

ラベルと出力形式を削除する前に、PROC CONTENTS を使用してデータセットのコンテンツを確認します。

```
proc contents data=mylib.class;
run;
```

PROC DATASETS で、MODIFY ステートメントと ATTRIB オプションを使用してすべてのラベルと出力形式を削除します。PROC DATASETS で CONTENTS ステートメントを使ってラベルや出力形式なしでデータセットのコンテンツを表示します。

```
proc datasets lib=mylib memtype=data;
  modify class;
   attrib _all_ label=' ';
   attrib _all_ format=;
contents data=mylib.class;
run;
quit;
```

出力例

アウトプット 15.14　ラベルと出力形式を含むクラスデータセットの CONTENTS プロシジャ

The CONTENTS Procedure

Data Set Name	MYLIB.CLASS	Observations	2
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	08/16/2016 14:48:34	Observation Length	24
Last Modified	08/16/2016 14:48:34	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	2715
Obs in First Data Page	2
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\mylib\class.sas7bdat
Release Created	9.0401M4
Host Created	X64_7PRO
Owner Name	BUILTIN\Administrators
File Size	128KB
File Size (bytes)	131072

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
2	x	Num	8		ID NUMBER
3	y	Num	8		AGE
1	z	Num	8	CLSFMT.	CLASS STATUS

アウトプット 15.15 ラベルと出力形式を含まないクラスデータセットの CONTENTS ステートメント

The DATASETS Procedure

Data Set Name	MYLIB.CLASS	Observations	2
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	08/16/2016 14:48:34	Observation Length	24
Last Modified	08/16/2016 14:48:34	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	2715
Obs in First Data Page	2
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\mylib\class.sas7bdat
Release Created	9.0401M4
Host Created	X64_7PRO
Owner Name	BUILTIN\Administrators
File Size	128KB
File Size (bytes)	131072

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
2	x	Num	8
3	y	Num	8
1	z	Num	8

例 2: SAS ファイルの操作

要素:

PROC DATASETS ステートメントオプション
- DETAILS
- LIBRARY=
- COPY ステートメント
- CHANGE ステートメント
- DELETE ステートメント
- EXCHANGE ステートメント

COPY プロシジャ

EXCLUDE ステートメント

OPTIONS ステートメント

詳細

この例は、次のタスクを示しています。

- SAS ファイル名の変更
- SAS ライブラリ間の SAS ファイルのコピー
- SAS ファイルの削除
- コピーする SAS ファイルの選択
- SAS ファイル名の入れ換え
- コピー操作からの SAS ファイルの除外

プログラム

```
options pagesize=60 linesize=80 nodate pageno=1 source;

LIBNAME dest1 'SAS-library-1';
LIBNAME dest2 'SAS-library-2';
LIBNAME health 'SAS-library-3';

proc datasets library=health details;
```

```
  delete tension a2(mt=catalog);
  change a1=postdrug;
  exchange weight=bodyfat;

  copy out=dest1 move memtype=view;

    select spdata;

    select etest1-etest5 / memtype=catalog;

  copy out=dest2;
    exclude d: mlscl oxygen test2 vision weight;
quit;
```

プログラムの説明

システムオプションを設定します。 SOURCE システムオプションはプログラミングステートメントを SAS ログに書き出します。PAGESIZE=オプションは SAS ログと SAS 出力のページを構成する行数を指定します。LINESIZE=オプションは、SAS ログと SAS プロシジャ出力のページサイズを指定します。NODATE オプションは、日時を印刷しないことを指定します。PAGENO=オプションで、出力の次ページの開始ページ番号を指定します。

```
options pagesize=60 linesize=80 nodate pageno=1 source;

LIBNAME dest1 'SAS-library-1';
LIBNAME dest2 'SAS-library-2';
LIBNAME health 'SAS-library-3';
```

プロシジャの入力ライブラリを指定して、ディレクトリに詳細を追加します。 DETAILS は、**Obs**、**Entries** または **Indexes**、**Vars**、**Label** の列をディレクトリに追加出力します。MEMTYPE=オプションは PROC DATASETS ステートメントには表示されないので、処理にはすべてメンバーの 種類が使用できます。

```
proc datasets library=health details;
```

ライブラリの 2 つのファイルを削除して、SAS データセットとカタログの名前を変更します。 DELETE ステートメントは TENSION データセットとカタログ A2 を削除します。DELETE ステートメントのデフォルトのメンバーの種類は DATA であるため、MT=CATALOG は A2 にのみ適用となります。CHANGE ステートメントはカタログ A1 の名前を POSTDRUG に変更します。EXCHANGE ステートメントは WEIGHT データセットと BODYFAT データセットの名前を入れ替えます。CHANGE または EXCHANGE ステートメントのデフォルトは MEMTYPE=ALL であるため、MEMTYPE=オプションは必要ありません。

```
  delete tension a2(mt=catalog);
  change a1=postdrug;
  exchange weight=bodyfat;
```

処理を 1 つのメンバーの種類に制限し、データビューを削除し移動します。 MEMTYPE=VIEW は処理を SAS ビューに制限します。MOVE はこのステップの SELECT ステートメントに名前があるすべての SAS ビューを Health データライブラリから削除し、そして Dest1 データライブラリに移動すると指定しています。

```
  copy out=dest1 move memtype=view;
```

SAS ビュー Spdata を Health データライブラリから Dest1 データライブラリに移動します。

```
select spdata;
```

カタログを他のデータライブラリに移動します。 SELECT ステートメントでは Etest1 から Etest5 までのカタログを Health データライブラリから Dest1 データライブラリに移動すると指定しています。MEMTYPE=CATALOG は COPY ステートメントの MEMTYPE=VIEW オプションを無効にします。

```
select etest1-etest5 / memtype=catalog;
```

指定の条件に該当するすべてのファイルを 処理から除外します。 EXCLUDE ステートメントは文字 D で始まるすべての SAS ファイルと他にリストされているファイルをコピー操作から除外します。HEALTH データライブラリにある残りの SAS ファイルはすべて Dest2 データライブラリにコピーされます。

```
copy out=dest2;
  exclude d: mlscl oxygen test2 vision weight;
quit;
```

ログの例

例のコード 15.2 Dest1 の SAS ログ

```
117   options pagesize=60 linesize=80 nodate pageno=1 source;
118  LIBNAME dest1 'SAS-library-1';
NOTE: Libref DEST1 was successfully assigned as follows:
      Engine:       V9
      Physical Name: SAS-library-1\dest1
119  LIBNAME dest2 'SAS-library-2';
NOTE: Libref DEST2 was successfully assigned as follows:
      Engine:       V9
      Physical Name: SAS-library-2\dest2
120  LIBNAME health 'SAS-library-3';
NOTE: Libref HEALTH was successfully assigned as follows:
      Engine:       V9
      Physical Name: SAS-library-3\health


121  proc datasets library=health details;
                           Directory

Libref         HEALTH
Engine         V9
Physical Name  \myfiles\health
Filename       \myfiles\health



                 Member   Obs, Entries
      #  Name     Type     or Indexes   Vars  Label

      1  A1       CATALOG      23
      2  ALL      DATA         23       17
      3  BODYFAT  DATA          1        2
      4  CONFOUND DATA          8        4
      5  CORONARY DATA         39        4
      6  DRUG1    DATA          6        2  JAN2005 DATA
      7  DRUG2    DATA         13        2  MAY2005 DATA
      8  DRUG3    DATA         11        2  JUL2005 DATA
      9  DRUG4    DATA          7        2  JAN2002 DATA
     10  DRUG5    DATA          1        2  JUL2002 DATA
     11  ETEST1   CATALOG       1
     12  ETEST2   CATALOG       1
     13  ETEST3   CATALOG       1
     14  ETEST4   CATALOG       1
     15  ETEST5   CATALOG       1
     16  ETESTS   CATALOG       1
     17  FORMATS  CATALOG       6
     18  GROUP    DATA         148       11
     19  GRPOUT   DATA          11       40
     20  INFANT   DATA         149        6
     21  MLSCL    DATA          32        4  Multiple Sclerosis Data
     22  NAMES    DATA           7        4
     23  OXYGEN   DATA          31        7
     24  PERSONL  DATA         148       11
     25  PHARM    DATA           6        3  Sugar Study
     26  POINTS   DATA           6        6
     27  RESULTS  DATA          10        5
     28  SLEEP    DATA         108        6
     29  SPDATA   VIEW           .        2
     30  TEST2    DATA          15        5
     31  TRAIN    DATA           7        2
     32  VISION   DATA          16        3
     33  WEIGHT   DATA          83       13  California Results
     34  WGHT     DATA          83       13
```

```
                File
            #   Size  Last Modified

            1  62464  07Mar05:14:36:20
            2  13312  12Sep07:13:57:48
            3   5120  12Sep07:13:57:48
            4   5120  12Sep07:13:57:48
            5   5120  12Sep07:13:57:48
            6   5120  12Sep07:13:57:49
            7   5120  12Sep07:13:57:49
            8   5120  12Sep07:13:57:49
            9   5120  12Sep07:13:57:49
           10   5120  12Sep07:13:57:49
           11  17408  04Jan02:14:20:16
           12  17408  04Jan02:14:20:16
           13  17408  04Jan02:14:20:16
           14  17408  04Jan02:14:20:16
           15  17408  04Jan02:14:20:16
           16  17408  24Mar05:16:12:20
           17  17408  24Mar05:16:12:20
           18  25600  12Sep07:13:57:50
           19  17408  24Mar05:15:33:31
           20  17408  12Sep07:13:57:51
           21   5120  12Sep07:13:57:50
           22   5120  12Sep07:13:57:50
           23   9216  12Sep07:13:57:50
           24  25600  12Sep07:13:57:51
           25   5120  12Sep07:13:57:51
           26   5120  12Sep07:13:57:51
           27   5120  12Sep07:13:57:52
           28   9216  12Sep07:13:57:52
           29   5120  24Mar05:16:12:21
           30   5120  12Sep07:13:57:52
           31   5120  12Sep07:13:57:53
           32   5120  12Sep07:13:57:53
           33  13312  12Sep07:13:57:53
           34  13312  12Sep07:13:57:53122  delete tension
a2(mt=catalog);
123  change a1=postdrug;
124  exchange weight=bodyfat;
NOTE: Changing the name HEALTH.A1 to HEALTH.POSTDRUG (memtype=CATALOG).
NOTE: Exchanging the names HEALTH.WEIGHT and HEALTH.BODYFAT (memtype=DATA).
125  copy out=dest1 move memtype=view;
126  select spdata;
127
128  select etest1-etest5 / memtype=catalog;
NOTE: Moving HEALTH.SPDATA to DEST1.SPDATA (memtype=VIEW).
NOTE: Moving HEALTH.ETEST1 to DEST1.ETEST1 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST2 to DEST1.ETEST2 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST3 to DEST1.ETEST3 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST4 to DEST1.ETEST4 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST5 to DEST1.ETEST5 (memtype=CATALOG).
```

```
129  copy out=dest2;
130  exclude d: mlscl oxygen test2 vision weight;
131  quit;

NOTE: Copying HEALTH.ALL to DEST2.ALL (memtype=DATA).
NOTE: There were 23 observations read from the data set HEALTH.ALL.
NOTE: The data set DEST2.ALL has 23 observations and 17 variables.
NOTE: Copying HEALTH.BODYFAT to DEST2.BODYFAT (memtype=DATA).
NOTE: There were 83 observations read from the data set HEALTH.BODYFAT.
NOTE: The data set DEST2.BODYFAT has 83 observations and 13 variables.
NOTE: Copying HEALTH.CONFOUND to DEST2.CONFOUND (memtype=DATA).
NOTE: There were 8 observations read from the data set HEALTH.CONFOUND.
NOTE: The data set DEST2.CONFOUND has 8 observations and 4 variables.
NOTE: Copying HEALTH.CORONARY to DEST2.CORONARY (memtype=DATA).
NOTE: There were 39 observations read from the data set HEALTH.CORONARY.
NOTE: The data set DEST2.CORONARY has 39 observations and 4 variables.
NOTE: Copying HEALTH.ETESTS to DEST2.ETESTS (memtype=CATALOG).
NOTE: Copying HEALTH.FORMATS to DEST2.FORMATS (memtype=CATALOG).
NOTE: Copying HEALTH.GROUP to DEST2.GROUP (memtype=DATA).
NOTE: There were 148 observations read from the data set HEALTH.GROUP.
NOTE: The data set DEST2.GROUP has 148 observations and 11 variables.
NOTE: Copying HEALTH.GRPOUT to DEST2.GRPOUT (memtype=DATA).
NOTE: There were 11 observations read from the data set HEALTH.GRPOUT.
NOTE: The data set DEST2.GRPOUT has 11 observations and 40 variables.
NOTE: Copying HEALTH.INFANT to DEST2.INFANT (memtype=DATA).
NOTE: There were 149 observations read from the data set HEALTH.INFANT.
NOTE: The data set DEST2.INFANT has 149 observations and 6 variables.
NOTE: Copying HEALTH.NAMES to DEST2.NAMES (memtype=DATA).
NOTE: There were 7 observations read from the data set HEALTH.NAMES.
NOTE: The data set DEST2.NAMES has 7 observations and 4 variables.
NOTE: Copying HEALTH.PERSONL to DEST2.PERSONL (memtype=DATA).
NOTE: There were 148 observations read from the data set HEALTH.PERSONL.
NOTE: The data set DEST2.PERSONL has 148 observations and 11 variables.
NOTE: Copying HEALTH.PHARM to DEST2.PHARM (memtype=DATA).
NOTE: There were 6 observations read from the data set HEALTH.PHARM.
NOTE: The data set DEST2.PHARM has 6 observations and 3 variables.
NOTE: Copying HEALTH.POINTS to DEST2.POINTS (memtype=DATA).
NOTE: There were 6 observations read from the data set HEALTH.POINTS.
NOTE: The data set DEST2.POINTS has 6 observations and 6 variables.
NOTE: Copying HEALTH.POSTDRUG to DEST2.POSTDRUG (memtype=CATALOG).
NOTE: Copying HEALTH.RESULTS to DEST2.RESULTS (memtype=DATA).
NOTE: There were 10 observations read from the data set HEALTH.RESULTS.
NOTE: The data set DEST2.RESULTS has 10 observations and 5 variables.
NOTE: Copying HEALTH.SLEEP to DEST2.SLEEP (memtype=DATA).
NOTE: There were 108 observations read from the data set HEALTH.SLEEP.
NOTE: The data set DEST2.SLEEP has 108 observations and 6 variables.
NOTE: Copying HEALTH.TRAIN to DEST2.TRAIN (memtype=DATA).
NOTE: There were 7 observations read from the data set HEALTH.TRAIN.
NOTE: The data set DEST2.TRAIN has 7 observations and 2 variables.
NOTE: Copying HEALTH.WGHT to DEST2.WGHT (memtype=DATA).
NOTE: There were 83 observations read from the data set HEALTH.WGHT.
NOTE: The data set DEST2.WGHT has 83 observations and 13 variables.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           44.04 seconds
      cpu time            0.60 seconds
```

例 3: SAS ファイルを削除せずに保存する

要素: PROC DATASETS ステートメントオプション
LIB=

詳細

この例では、SAVE ステートメントを使用して、一部の SAS ファイルを削除から保存し、その他の SAS ファイルを削除する方法を示します。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1 source;
LIBNAME elder 'SAS-library';
proc datasets lib=elder;
   save chronic aging clinics / memtype=data;
run;
```

プログラムの説明

システムオプションと LIBNAME ステートメントを設定します。 SOURCE システムオプションはプログラミングステートメントを SAS ログに書き出します。LINESIZE=オプションは、SAS ログと SAS プロシジャ出力のページサイズを指定します。NODATE オプションは、日時を印刷しないことを指定します。

```
options pagesize=40 linesize=80 nodate pageno=1 source;
LIBNAME elder 'SAS-library';
```

処理するプロシジャ入力ライブラリを指定します。

```
proc datasets lib=elder;
```

データセット CHRONIC、AGING、と CLINICS を保存し、ELDER ライブラリのその他の SAS ファイル(すべての種類)をすべて削除します。 ELDER ライブラリには CLINICS という名前のカタログと同じ名前のデータセットが あるので、MEMTYPE=DATA が必要になります。

```
   save chronic aging clinics / memtype=data;
run;
```

ログの例

例のコード 15.3 Elder ライブラリの SAS ログ

```
6    options pagesize=40 linesize=80 nodate pageno=1 source;
7    LIBNAME elder '/home/userid/procdatasets/elder';
NOTE: Libref ELDER was successfully assigned as follows:
      Engine:       V9
      Physical Name: /home/userid/procdatasets/elder
8    proc datasets lib=elder;
9      save chronic aging clinics / memtype=data;
10   run;

NOTE: Saving ELDER.CHRONIC (memtype=DATA).
NOTE: Saving ELDER.AGING (memtype=DATA).
NOTE: Saving ELDER.CLINICS (memtype=DATA).
11   quit;

NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.06 seconds
      cpu time            0.01 seconds
```

アウトプット 15.16 SAVE ステートメントの使用前後の Elder ライブラリ

Directory	
Libref	ELDER
Engine	V9
Physical Name	c:\procdatasets\elder
Filename	c:\procdatasets\elder
Owner Name	BUILTIN\Administrators
File Size	4KB
File Size (bytes)	4096

#	Name	Member Type	File Size	Last Modified
1	AGING	DATA	8KB	04/17/2014 12:59:39
2	CHRONIC	DATA	8KB	04/17/2014 12:59:39
3	CLINICS	DATA	8KB	04/17/2014 12:59:39

Directory	
Libref	ELDER
Engine	V9
Physical Name	c:\procdatasets\elder
Filename	c:\procdatasets\elder
Owner Name	BUILTIN\Administrators
File Size	4KB
File Size (bytes)	4096

#	Name	Member Type	File Size	Last Modified
1	AGING	DATA	8KB	04/17/2014 12:59:39
2	CHRONIC	DATA	8KB	04/17/2014 12:59:39
3	CLINICS	DATA	8KB	04/17/2014 12:59:39

例 4: SAS データセットの変更

要素:

PROC DATASETS ステートメントオプション
- LIB=
- NOLIST
- MODIFY ステートメント

OPTIONS ステートメント

詳細

この例では、MODIFY ステートメントと、それに従属しているステートメントを使用して 2 つの SAS データセットを変更します。“例 5: SAS データセットを記述” (505 ページ) に Group データセットへの変更を示します。

この例は、次のタスクを示しています。

- SAS ファイルの変更
- SAS データセットのラベル付け
- SAS データセットへの READ パスワードの追加
- SAS データセットの現在の並べ替え方法の表示
- SAS データセットのインデックスの作成
- SAS データセット内の変数への入力形式と出力形式の割り当て
- SAS データセット内の変数の名前変更

- SAS データセット内の変数のラベル付け

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1 source;
LIBNAME health 'SAS-library';
proc datasets library=health nolist;
  modify group (label='Test Subjects' read=green sortedby=lname);
    index create vital=(birth salary) / nomiss unique;
    informat birth date7.;
    format birth date7.;
    label salary='current salary excluding bonus';
  modify oxygen;
    rename oxygen=intake;
    label intake='Intake Measurement';
quit;
```

プログラムの説明

システムオプションと LIBNAME ステートメントを設定します。 SOURCE システムオプションはプログラミングステートメントを SAS ログに書き出します。PAGESIZE=オプションは SAS ログと SAS 出力のページを構成する行数を指定します。LINESIZE=オプションは、SAS ログと SAS プロシジャ出力のページサイズを指定します。NODATE オプションは、日時を印刷しないことを指定します。PAGENO=オプションで、出力の次ページの開始ページ番号を指定します。

```
options pagesize=40 linesize=80 nodate pageno=1 source;
LIBNAME health 'SAS-library';
```

HEALTH を処理するプロシジャ入力ライブラリに指定します。 NOLIST は HEALTH データライブラリのディレクトリリストの表示を抑制します。

```
proc datasets library=health nolist;
```

データセットにラベルを追加し、読み取りパスワードを付与し、データの並べ替えを指定します。 LABEL=はデータセット GROUP にデータセットラベルを追加します。READ=は green を読み取りパスワードとして割り当てます。パスワードは SAS ログでは XXX...と表示されます。SAS はファイルの変更保護を含まないレベルのパスワード保護が指定された場合は、警告メッセージを出します。SORTEDBY=はデータの並べ替え方法を指定します。

```
  modify group (label='Test Subjects' read=green sortedby=lname);
```

GROUP データセットの変数 BIRTH と SALARY に複合インデックス VITAL を作成します。 NOMISS は BIRTH と SALARY に欠損値があるオブザベーションをインデックスから除外します。UNIQUE は BIRTH と SALARY の値が一意の組み合わせであるオブザベーションのみでインデックスを作成すると指定します。

```
index create vital=(birth salary) / nomiss unique;
```

変数 BIRTH に入力形式と出力形式をそれぞれ割り当てます。

```
informat birth date7.;
format birth date7.;
```

変数 SALARY にラベルを割り当てます。

```
label salary='current salary excluding bonus';
```

変数の名前を変更し、ラベルを割り当てます。 変数 OXYGEN を INTAKE に名称変更してデータセット OXYGEN を変更し、変数 INTAKE にラベルを割り当てます。

```
modify oxygen;
  rename oxygen=intake;
  label intake='Intake Measurement';
quit;
```

ログの例

例のコード 15.4　Health ライブラリの SAS ログ

```
169  options pagesize=40 linesize=80 nodate pageno=1 source;
170  LIBNAME health 'SAS-library';
NOTE: Libref HEALTH was successfully assigned as follows:
      Engine:       V9
      Physical Name: SAS-library\health

NOTE: PROCEDURE DATASETS used (Total process time):
      real time           8:06.11
      cpu time            0.54 seconds


171  proc datasets library=health nolist;
172   modify group (label='Test Subjects' read=XXXXX sortedby=lname);
WARNING: The file HEALTH.GROUP.DATA is not ALTER protected.  It could be
         deleted or replaced without knowing the password.
173   index create vital=(birth salary) / nomiss unique;
NOTE: Composite index vital has been defined.
NOTE: MODIFY was successful for HEALTH.GROUP.DATA.
174  informat birth date7.;
175  format birth date7.;
176  label salary='current salary excluding bonus';
177  modify oxygen;
178  rename oxygen=intake;
NOTE: Renaming variable oxygen to intake.
179  label intake='Intake Measurement';
180  quit;

NOTE: MODIFY was successful for HEALTH.OXYGEN.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           15.09 seconds
      cpu time            0.06 seconds
```

例 5: SAS データセットを記述

要素:

PROC DATASETS ステートメントオプション
- LIB=
- NOLIST
- CONTENTS ステートメント

OPTIONS ステートメント

詳細

この例に、GROUP データセットに対する CONTENTS ステートメントからの出力を示します。出力には、“例 4: SAS データセットの変更” (502 ページ)で Group データセットに行われた変更が表示されます。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME health 'SAS-library';

proc datasets library=health nolist;

  contents data=group (read=green) out=grpout;
  title  'The Contents of the GROUP Data Set';
run;
quit;
```

プログラムの説明

システムオプションと LIBNAME ステートメントを設定します。 PAGESIZE=オプションは SAS ログと SAS 出力のページを構成する行数を指定します。LINESIZE=オプションは、SAS ログと SAS プロシジャ出力のページサイズを指定します。NODATE オプションは、日時を印刷しないことを指定します。PAGENO=オプションで、出力の次ページの開始ページ番号を指定します。

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME health 'SAS-library';
```

Health をプロシジャの入力ライブラリに指定し、NOLIST オプションでディレクトリリストの表示を抑制します。

```
proc datasets library=health nolist;
```

データセット GROUP から出力データセット GRPOUT を作成します。 記述するデータセットとして GROUP を指定し、データセット GROUP に読み込みアクセス権を付与して、出力データセット GRPOUT を作成して OUT=データセットに表示します。

```
    contents data=group (read=green) out=grpout;
    title  'The Contents of the GROUP Data Set';
run;
quit;
```

出力例

アウトプット 15.17　GROUP データセットのコンテンツ

The Contents of the GROUP Data Set

The DATASETS Procedure

Data Set Name	HEALTH.GROUP	**Observations**	148
Member Type	DATA	**Variables**	11
Engine	V9	**Indexes**	0
Created	09/03/2014 10:35:02	**Observation Length**	96
Last Modified	09/03/2014 10:35:02	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

アウトプット 15.18　エンジン/ホスト依存情報

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	3
First Data Page	1
Max Obs per Page	84
Obs in First Data Page	63
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\procdatasets\health\group.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO
Owner Name	BUILTIN\Administrators
File Size	32KB
File Size (bytes)	32768

アウトプット 15.19　変数と属性のアルファベット順リスト

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Informat
9	BIRTH	Num	8		
4	CITY	Char	15		
3	FNAME	Char	15		
10	HIRED	Num	8	DATE7.	DATE7.
11	HPHONE	Char	12		
1	IDNUM	Char	4		
7	JOBCODE	Char	4		
2	LNAME	Char	15		
8	SALARY	Num	8	COMMA8.	
6	SEX	Char	2		
5	STATE	Char	3		

アウトプット 15.20　データセットと変数の拡張属性のアルファベット順リスト

Alphabetic List of Data Set Extended Attributes		
Extended Attribute	Numeric Value	Character Value
level	1	
region	.	NorthEast

Alphabetic List of Extended Attributes on Variables			
Extended Attribute	Attribute Variable	Numeric Value	Character Value
classification	jobcode	.	local
length	idnum	4	

例 6: 2 つの SAS データセットを連結する

要素:
PROC DATASETS ステートメントオプション
LIBRARY=
NOLIST
APPEND ステートメント
PRINT プロシジャ
OPTIONS ステートメント

データセット:
EXP ライブラリ

詳細

この例は、次のタスクを示しています。

- ライブラリの印刷の抑制
- 2 つのデータセットの追加
- 追加前のデータセットの印刷、および追加後の新しいデータセットの印刷

Exp.Results および Exp.Sur の各データセットを作成し、この例を使用して連結する前にそれらのデータセットを印刷するには、“EXP ライブラリ” (2489 ページ)を参照してください。

プログラム

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

```
LIBNAME exp 'SAS-library';

proc datasets library=exp nolist;

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

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

プログラムの説明

この例では、あるデータセットの末尾に別のデータセットを追加します。

データセット Exp.Sur には変数 Wt6Mos が含まれていますが、Exp.Results データセットには含まれていません。

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

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

LIBNAME ステートメントでライブラリを割り当てます。

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

Exp ライブラリの出力を抑制します。 LIBRARY=は、Exp をプロシジャ入力ライブラリとして指定します。NOLIST は Exp ライブラリのディレクトリリストの表示を抑制します。

```
proc datasets library=exp nolist;
```

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

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

データセットを出力します。

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

出力: 2 つのデータセットを連結する

アウトプット 15.21　Results データセット

The RESULTS Data Set

ID	TREAT	INITWT	WT3MOS	AGE
1	Other	166.28	146.98	35
2	Other	214.42	210.22	54
3	Other	172.46	159.42	33
5	Other	175.41	160.66	37
6	Other	173.13	169.40	20
7	Other	181.25	170.94	30
10	Other	239.83	214.48	48
11	Other	175.32	162.66	51
12	Other	227.01	211.06	29
13	Other	274.82	251.82	31

アウトプット 15.22　Sur データセット

The EXP.SUR Data Set

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

アウトプット 15.23　Results データセットと Sur データセットの連結

The RESULTS Data Set

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

例 7: SAS データセットのエージング

要素:

PROC DATASETS ステートメントオプション
- LIBRARY=
- NOLIST
- AGE ステートメント

OPTIONS ステートメント

詳細

この例では、AGE ステートメントが SAS ファイルをエージングする方法を示します。

プログラム

```
options pagesize=40 linesize=80 nodate pageno=1 source;
```

```
LIBNAME daily 'SAS-library';

proc datasets library=daily nolist;

  age today day1-day7;
run;
```

プログラムの説明

システムオプションを設定します。 SOURCE システムオプションはプログラミングステートメントを SAS ログに書き出します。PAGESIZE=オプションは SAS ログと SAS 出力のページを構成する行数を指定します。LINESIZE=オプションは、SAS ログと SAS プロシジャ出力のページサイズを指定します。NODATE オプションは、日時を印刷しないことを指定します。PAGENO=オプションで、出力の次ページの開始ページ番号を指定します。

```
options pagesize=40 linesize=80 nodate pageno=1 source;

LIBNAME daily 'SAS-library';
```

DAILY をプロシジャの入力ライブラリに指定し、ディレクトリリストの表示を抑制します。

```
proc datasets library=daily nolist;
```

ファイルを削除してエージングします。 リストの最後の SAS ファイルである Day7 を削除し、Day6 を Day7 にエージング(名称変更)します。同じように Day5 を Day6 というように、TODAY が Day1 になるまで続けます。

```
  age today day1-day7;
run;
```

ログの例

例のコード 15.5　SAS ログ

```
6  options pagesize=40 linesize=80
nodate pageno=1 source;
7
8    proc datasets library=daily nolist;
9
10       age today day1-day7;
11   run;
NOTE: Deleting DAILY.DAY7 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY6 to DAILY.DAY7 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY5 to DAILY.DAY6 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY4 to DAILY.DAY5 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY3 to DAILY.DAY4 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY2 to DAILY.DAY3 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY1 to DAILY.DAY2 (memtype=DATA).
NOTE: Ageing the name DAILY.TODAY to DAILY.DAY1 (memtype=DATA).
```

例 8: 監査ファイルの初期化

要素:

PROC DATASETS ステートメントオプション
- LIB=
- AUDIT ステートメント
- INITIATE ステートメント
- LOG ステートメント
- RESUME ステートメント
- SUSPEND ステートメント
- TERMINATE ステートメント
- USER_VAR ステートメント

SQL プロシジャ

詳細

この例は、次のタスクを示しています。

- 監査ファイルを作成する
- データセットを更新する
- 監査ファイルを中断する
- 監査ファイルを終了する

プログラム

```
libname mylib "SAS-library";

data mylib.inventory;
input vendor $10. +1 item $4. +1 description $11. +1 units 4.;
datalines;
SmithFarms F001 Apples      10
Tropicana  B002 OrangeJuice 45
UpperCrust C215 WheatBread  25
;
run;

proc datasets lib=mylib;
  audit inventory;
  initiate;
  user_var reason $ 30;
quit;


proc sql;
```

```
  Insert into mylib.inventory values ('Bordens','B132', 'Milk', 100,
                                      'increase on hand');
  Update mylib.inventory    set units=10, reason='recounted inventory'
                        where item='B002';
quit;


proc datasets lib=mylib;
audit inventory;
log admin_image=no;
suspend;
quit;


proc sql;
select * from mylib.inventory(type=audit);
quit;


proc datasets lib=mylib;
audit inventory;
resume;
quit;


proc datasets lib=mylib;
audit inventory;
terminate;
quit;
```

プログラムの説明

USER_VAR を使って監査ファイルを開始します。

```
libname mylib "SAS-library";

data mylib.inventory;
input vendor $10. +1 item $4. +1 description $11. +1 units 4.;
datalines;
SmithFarms F001 Apples      10
Tropicana  B002 OrangeJuice 45
UpperCrust C215 WheatBread  25
;
run;

proc datasets lib=mylib;
  audit inventory;
  initiate;
  user_var reason $ 30;
quit;
```

データセットを更新します。

```
proc sql;
  Insert into mylib.inventory values ('Bordens','B132', 'Milk', 100,
                                      'increase on hand');
  Update mylib.inventory    set units=10, reason='recounted inventory'
                        where item='B002';
quit;
```

ADMIN イメージの記録を中止し、監査ファイルを中断します。

```
proc datasets lib=mylib;
audit inventory;
log admin_image=no;
suspend;
quit;
```

監査ファイルを表示します。

```
proc sql;
select * from mylib.inventory(type=audit);
quit;
```

監査ファイルを再開します。

```
proc datasets lib=mylib;
audit inventory;
resume;
quit;
```

監査ファイルを終了します。

```
proc datasets lib=mylib;
audit inventory;
terminate;
quit;
```

ログの例

例のコード 15.6　監査ファイルの初期化

```
1    options nocenter;
2
3    libname mylib "SAS-library";
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:       V9
      Physical Name: /home/userod/mylib
4
5    data mylib.inventory;
6    input vendor $10.+1 item $4.+1 description $11.+1 units 4.;
7    datalines;

NOTE: The data set MYLIB.INVENTORY has 3 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           3.45 seconds
      cpu time            0.00 seconds


11   ;
12   run;
13
14   proc datasets lib=mylib;
NOTE: Writing HTML Body file: sashtml.htm
15       audit inventory;
16       initiate;
WARNING: The audited data file MYLIB.INVENTORY.DATA is not password protected.  Apply an Alter
password to prevent accidental
         deletion or replacement of it and any associated audit files.
17       user_var reason $ 30;
18   quit;

NOTE: The data set MYLIB.INVENTORY.AUDIT has 0 observations and 11 variables.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           18.17 seconds
      cpu time            0.79 seconds


19
20   proc sql;
21       Insert into mylib.inventory values ('Bordens','B132', 'Milk', 100,
22                                          'increase on hand');
NOTE: 1 row was inserted into MYLIB.INVENTORY.

23       Update mylib.inventory    set units=10, reason='recounted inventory'
24                              where item='B002';
NOTE: 1 row was updated in MYLIB.INVENTORY.

25   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           2.57 seconds
      cpu time            0.03 seconds


26
27   proc datasets lib=mylib;
28   audit inventory;
29   log admin_image=no;
30   suspend;
31   quit;

NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

```
32
33   proc sql;
34   select * from mylib.inventory(type=audit);
35   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.54 seconds
      cpu time           0.01 seconds


36
37   proc datasets lib=mylib;
37 !                        /* resume audit file */
38   audit inventory;
39   resume;
40   quit;

NOTE: PROCEDURE DATASETS used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds


41
42   /* additional step(s) which update the inventory dataset could go here*/
43
44   proc datasets lib=mylib;
44 !                        /* terminate audit file */
45   audit inventory;
46   terminate;
NOTE: Deleting MYLIB.INVENTORY (memtype=AUDIT).
47   quit;

NOTE: PROCEDURE DATASETS used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
```

出力例

アウトプット 15.24　MyLib ライブラリのインベントリコンテンツ

Directory	
Libref	MYLIB
Engine	V9
Physical Name	c:\mylib
Filename	c:\mylib
Owner Name	BUILTIN\Administrators
File Size	4KB
File Size (bytes)	4096

#	Name	Member Type	File Size	Last Modified
1	CLASS	DATA	128KB	08/16/2016 14:48:34
2	INVENTORY	DATA	128KB	08/16/2016 15:38:44
3	SALARY	DATA	128KB	04/30/2014 13:42:52

アウトプット 15.25　MyLib ライブラリの監査ファイルリスト

Directory	
Libref	MYLIB
Engine	V9
Physical Name	c:\mylib
Filename	c:\mylib
Owner Name	BUILTIN\Administrators
File Size	4KB
File Size (bytes)	4096

#	Name	Member Type	File Size	Last Modified
1	CLASS	DATA	128KB	08/16/2016 14:48:34
2	INVENTORY	DATA	128KB	08/16/2016 15:38:44
	INVENTORY	AUDIT	65KB	08/16/2016 15:38:44
3	SALARY	DATA	128KB	04/30/2014 13:42:52

アウトプット 15.26　MyLib ライブラリのインベントリデータファイルコンテンツ

vendor	item	description	units	reason	_ATDATETIME_	_ATOBSNO_	_ATRETURNCODE_	_ATUSERID_	_ATOPCODE_	_ATMESSAGE_
Bordens	B132	Milk	100	increase on hand	16AUG2016:15:38:44	4	.	suholm	DA	
Tropicana	B002	OrangeJuice	45		16AUG2016:15:38:44	2	.	suholm	DR	
Tropicana	B002	OrangeJuice	10	recounted inventory	16AUG2016:15:38:44	2	.	suholm	DW	

例 9: 拡張属性

要素:

PROC DATASETS ステートメントオプション
- LIB=
- NOLIST

CONTENTS ステートメント

MODIFY ステートメント

XATTR ADD DS ステートメント

XATTR ADD VAR ステートメント

プログラム

```
libname mylib '/home/userid/mylib';

data mylib.sales;
  purchase="car";
  age=10;
  income=200000;
  kids=3;
  cars=4;
run;

proc datasets lib=mylib nolist;
  modify sales;
    xattr add ds role="train" attrib="table";
    xattr add var purchase (role="target" level="nominal")
              age (role="reject")
              income (role="input" level="interval");

  contents data=sales;
  title  'The Contents of the Sales Data Set That Contains Extended Attributes';

run;
quit;
```

プログラムの説明

MyLib.Sales データセットを作成します。

```
libname mylib '/home/userid/mylib';
```

```
data mylib.sales;
  purchase="car";
  age=10;
  income=200000;
  kids=3;
  cars=4;
run;
```

データセットおよび変数に拡張属性を追加します。

```
proc datasets lib=mylib nolist;
  modify sales;
    xattr add ds role="train" attrib="table";
    xattr add var purchase (role="target" level="nominal")
                age (role="reject")
                income (role="input" level="interval");

  contents data=sales;
  title  'The Contents of the Sales Data Set That Contains Extended Attributes';

run;
quit;
```

ログの例

例のコード 15.7 拡張属性

```
355   libname mylib '/home/userid/mylib';
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:       V9
      Physical Name: /home/userid/mylib
356
357  data mylib.sales;
358     purchase = "car";
359     age = 10;
360     income = 200000;
361     kids = 3;
362     cars = 4;
363  run;

NOTE: The data set MYLIB.SALES has 1 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


364
365  proc datasets lib=mylib nolist ;
366     modify sales;
367       xattr add ds role= "train" attrib="table";
368       xattr add var purchase ( role="target" level="nominal" )
369                    age ( role="reject" )
370                    income ( role="input" level="interval" );
NOTE: MODIFY was successful for MYLIB.SALES.DATA.
371
372     contents data=sales;
373     title  'The Contents of the Sales Data Set That Contains Extended Attributes';
374
375  run;

376  quit;

NOTE: PROCEDURE DATASETS used (Total process time):
      real time           1.02 seconds
      cpu time            0.10 seconds
```

出力例

アウトプット 15.27　拡張属性を持つ Sales データセットのコンテンツ

The Contents of the Sales Data Set That Contains Extended Attributes

The DATASETS Procedure

Data Set Name	MYLIB.SALES	**Observations**	1
Member Type	DATA	**Variables**	5
Engine	V9	**Indexes**	0
Created	08/16/2016 15:46:14	**Observation Length**	40
Last Modified	08/16/2016 15:46:19	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label		**EXTATTR Segment Length**	256
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
2	age	Num	8
5	cars	Num	8
3	income	Num	8
4	kids	Num	8
1	purchase	Char	3

Alphabetic List of Data Set Extended Attributes		
Extended Attribute	Numeric Value	Character Value
attrib	.	table
role	.	train

Alphabetic List of Extended Attributes on Variables			
Extended Attribute	Attribute Variable	Numeric Value	Character Value
level	income	.	interval
level	purchase	.	nominal
role	age	.	reject
role	income	.	input
role	purchase	.	target

16

DATEKEYS プロシジャ

概要: DATEKEYS プロシジャ

概要: DATEKEYS プロシジャ

DATEKEYS プロシジャを使用すると、日付を参照する名前を使用して、単一の日付または時系列の一連の日付を処理できます。たとえば、日付キーに関連付けられた期間の特定や、日付キーの読み取り/書き込み、日付キー定義に関する詳細の提供に使用できます。

DATEKEYS プロシジャを使用して、日付キーに関連付けられた期間を特定できます。

DATEKEYS プロシジャの一般的な用途は、祝日期間、セールイベント、または営業時間の変更などを識別する日付キーを定義することです。

概念: DATEKEYS プロシジャ

概念: DATEKEYS プロシジャ

SAS 日付キーは、祝日やセール期間などの特別イベントや時間計算に関連付けられた日付または時間間隔を記述します。

日付キーは、名前、その日付キーに関連付けられた単一または一連の日付、および一連の修飾子で構成されます。

DATEKEYS プロシジャは、他の SAS プロシジャで解釈できる出力データセットに結果を提供します。SAS システムオプションの EVENTDS=と組み合わせて使用すると、事前定義された SAS 日付キーとして指定できる日付キーを定義できます。事前定義された SAS 日付キーを使用する場合と同様に、定義した日付キーを日付キーワードとして使用できます。

次の例では、日付キーを作成し、MyHolidays という名前の出力データセットに日付キー定義を書き込みます。SAS システムオプションの EVENTDS=を設定することで、日付キー定義を SAS High-Performance Forecasting のプロシジャで自動的に使用できるようになります。

```
proc datekeys;
```

```
  datekeydef SuperBowl=
          '15JAN1967'd '14JAN1968'd '12JAN1969'd '11JAN1970'd
  '17JAN1971'd '16JAN1972'd '14JAN1973'd '13JAN1974'd '12JAN1975'd
  '18JAN1976'd '09JAN1977'd '15JAN1978'd '21JAN1979'd '20JAN1980'd
  '25JAN1981'd '24JAN1982'd '30JAN1983'd '22JAN1984'd '20JAN1985'd
  '26JAN1986'd '25JAN1987'd '31JAN1988'd '22JAN1989'd '28JAN1990'd
  '27JAN1991'd '26JAN1992'd '31JAN1993'd '30JAN1994'd '29JAN1995'd
  '28JAN1996'd '26JAN1997'd '25JAN1998'd '31JAN1999'd '30JAN2000'd
  '28JAN2001'd '03FEB2002'd '26JAN2003'd '01FEB2004'd '06FEB2005'd
  '05FEB2006'd '04FEB2007'd '03FEB2008'd '01FEB2009'd '07FEB2010'd
  '06FEB2011'd '05FEB2012'd '03FEB2013'd '02FEB2014'd '01FEB2015'd
...'07FEB2016'd '05FEB2017'd '04FEB2018'd '03FEB2019'd '02FEB2020'd
  / PULSE=DAY ;

  datekeydef GoodFriday=Easter / shift=-2 pulse=day;
  datekeykey EasterMonday=Easter / shift=1 pulse=day;
  datekeydata out=MyHolidays condense;
run;

options eventds=(MyHolidays);

proc hpfevents data=sashelp.citiday;
  id date interval=day start='27JAN1991'd end='01APR1991'd;
  eventkey SuperBowl;
  eventkey GoodFriday;
  eventkey EasterMonday;
  eventdata out=MyHolidayEvents condense;
  eventdummy out=MyHolidayDates;
run;
```

次の出力は結果を示しています。

図 16.1　ユーザー定義の日付キーに基づくイベント定義データセット

The SAS System

Obs	_NAME_	_KEYNAME_
1	SuperBowl	SUPERBOWL
2	GoodFriday	GOODFRIDAY
3	EasterMonday	EASTERMONDAY

次のステートメントは、スーパーボウル(Super Bowl)、聖金曜日(Good Friday)、復活の月曜日(Easter Monday)の各イベントの変数を示す出力データセットを表示します。最初の出力には、Month=1 の場合の結果が示されます。2 番目の出力には、Month GE 3 の場合の結果が示されます。

```
proc print data=MyHolidayDates(where=(month(date)=1));
  var date SuperBowl GoodFriday EasterMonday;
run;

proc print data=MyHolidayDates(where=(month(date) GE 3));
  var date SuperBowl GoodFriday EasterMonday;
```

```
run;
```

図 16.2　ユーザー定義の日付キーに基づく出力データセット

The SAS System

Obs	DATE	SuperBowl	GoodFriday	EasterMonday
1	27JAN1991	1	0	0
2	28JAN1991	0	0	0
3	29JAN1991	0	0	0
4	30JAN1991	0	0	0
5	31JAN1991	0	0	0

図 16.3　ユーザー定義の日付キーに基づく出力データセット: Month GE 3

The SAS System

Obs	DATE	SuperBowl	GoodFriday	EasterMonday
34	01MAR1991	0	0	0
35	02MAR1991	0	0	0
36	03MAR1991	0	0	0
37	04MAR1991	0	0	0
38	05MAR1991	0	0	0
39	06MAR1991	0	0	0
40	07MAR1991	0	0	0
41	08MAR1991	0	0	0
42	09MAR1991	0	0	0
43	10MAR1991	0	0	0
44	11MAR1991	0	0	0
45	12MAR1991	0	0	0
46	13MAR1991	0	0	0
47	14MAR1991	0	0	0
48	15MAR1991	0	0	0
49	16MAR1991	0	0	0
50	17MAR1991	0	0	0
51	18MAR1991	0	0	0
52	19MAR1991	0	0	0
53	20MAR1991	0	0	0
54	21MAR1991	0	0	0
55	22MAR1991	0	0	0
56	23MAR1991	0	0	0
57	24MAR1991	0	0	0
58	25MAR1991	0	0	0
59	26MAR1991	0	0	0
60	27MAR1991	0	0	0
61	28MAR1991	0	0	0
62	29MAR1991	0	1	0
63	30MAR1991	0	0	0
64	31MAR1991	0	0	0
65	01APR1991	0	0	1

構文: DATEKEYS プロシジャ

PROC DATEKEYS<*options*>;
BY *variable(s)*;
DATEKEYCALENDAR OUT=*SAS-data-set* <SUMMARY=*SAS-variable-name*>;
DATEKEYDATA IN=*SAS-data-set* | OUT=*SAS-data-set* <*options*>;
DATEKEYDEF *SAS-variable-name=timing-value* </*qualifier-options*>;
DATEKEYDSOPT LOCALE=<(ONLY)>'*POSIX locale*';
DATEKEYKEY <*SAS-variable-name=*> *datekey-keyword* </*qualifier-options*>;
DATEKEYPERIODS OUT=*SAS-data-set*;
ID *SAS-variable-name* INTERVAL=*interval* <*options*>;
VAR *variable(s)*;

ステートメント	タスク	例
“PROC DATEKEYS ステートメント”	時間計算に関連付けられた日付キーを作成および管理する	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“BY ステートメント”	BY 変数によって定義されたオブザベーショングループごとに個別のカレンダー変数を取得する	
“DATEKEYCALENDAR ステートメント”	日付キーのアクティブ期間を示す変数を書き込む	Ex. 3, Ex. 4
“DATEKEYDATA ステートメント”	日付キーを入力および出力する	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DATEKEYDEF ステートメント”	他の SAS プロシジャで使用できる日付キーを定義する	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DATEKEYDSOPT ステートメント”	指定したロケールにデータセットの入出力処理を限定する	Ex. 4
“DATEKEYKEY ステートメント”	ユーザー定義または事前定義された SAS 日付キーを変更するか、別の日付キーから新しい日付キーを作成する	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DATEKEYPERIODS ステートメント”	入力時間 ID での日付キーのアクティブ期間をリストする変数を書き込む	Ex. 1
“ID ステートメント”	入力データセットと出力データセットで期間によってオブザベーションを識別する SAS 日付、日時または時間値を含む数値変数名を指定する	Ex. 1, Ex. 2, Ex. 4
“VAR ステートメント”	入力変数を出力カレンダー変数データセットにコピーする	

PROC DATEKEYS ステートメント

時間計算に関連付けられた日付キーを作成および管理します。

構文

PROC DATEKEYS <*options*>;

オプション引数

option
次のオプションを使用できます。

DATA=*SAS-data-set*
VAR、ID、および BY ステートメントで使用する変数を含む SAS データセットの名前を指定します。

ヒント DATA=オプションを指定しない場合、最後に作成された SAS データセットが使用されます。

LEAD=*number-of-periods*
カレンダー変数を入力時間 ID の範囲外に拡張するための期間数を指定します。LEAD=の値は、入力データセットの最後のオブザベーションを基準とします。BY 変数を指定した場合、LEAD=の値は各 BY グループの最後のオブザベーションを基準とします。

デフォルト 0

MAXERROR=*number*
プロシジャの実行中に生成される警告およびエラーメッセージの最大数を指定します。

デフォルト 25

ヒント このオプションは、BY グループ処理で特に役立ちます。このオプションを使用することで、繰り返されるメッセージを抑制できます。

SORTNAMES
出力データセットの日付キーと変数をアルファベット順に書き込むことを指定します。変数はグループ内で並べ替えられます。VAR ステートメントでリストされている変数は、VAR ステートメントでリストされている他の変数に対して並べ替えられます。カレンダー変数は他のカレンダー変数に対して並べ替えられます。

BY ステートメント

BY 変数によって定義されたオブザベーショングループごとに個別のカレンダー変数を取得します。

構文

BY *variable(s)*;

必須引数

variable
BY 変数によって定義されたオブザベーショングループごとに個別のカレンダー変数を取得する際に使用する変数を指定します。

ヒント BY ステートメントが指定されていると、プロシジャでは入力データセットが BY 変数順で並べ替えられていることが前提となります。入力データセットが昇順で並べ替えられていない場合は、次の代替方法のいずれかを使用します。同様の BY ステートメントで SORT プロシジャを使用してデータを並べ替えるか、DATASETS プロシジャを使用して BY 変数のインデックスを作成します。詳細については、*Base SAS プロシジャガイド*の DATASETS プロシジャを参照してください。

DATEKEYCALENDAR ステートメント

日付キーのアクティブ期間を示す変数を書き込みます。アクティブ期間は値 1 で示され、非アクティブ期間は値 0 で示されます。

構文

DATEKEYCALENDAR OUT=*SAS-data-set* <SUMMARY=*SAS-variable-name*>;

オプション引数の要約

SUMMARY SUM=*SAS-variable-name*

必須引数

OUT=*SAS-data-set*
ID ステートメントで指定した ID 情報に基づいて、指定した日付キーのカレンダー変数を含む出力データセットの名前を指定します。

ヒント *SAS-data-set* には、VAR、BY、および ID ステートメントで指定した変数も含まれます。

オプション引数

SUMMARY SUM=*SAS-variable-name*
日付キーのカレンダー変数を合計し、DATEKEYCALENDAR OUT=データセットの指定した変数に結果を配置することを指定します。

SUM=変数は、時間間隔ごとのアクティブなキー日付の数として解釈できます。SUM=オプションを指定しない場合、このような変数は OUT=データセットに含まれません。

DATEKEYDATA ステートメント

日付キーデータセットからの日付キーの入力、または日付キーデータセットへの日付キーの書き込みを行います。複数の DATEKEYDATA ステートメントを指定できます。

構文

DATEKEYDATA IN=*SAS-data-set* | OUT=*SAS-data-set* <*options*>;

オプション引数の要約

option

必須引数

IN=*SAS-data-set*
日付キー定義を含む入力データセットの名前を指定します。

OUT=*SAS-data-set*
DATEKEYDATA IN=データセットと、DATEKEYDEF および DATEKEYKEY ステートメントで指定した日付キー定義を含む出力データセットの名前を指定します。

ヒント LIST オプションを指定しない場合、他の SAS プロシジャとシステムオプションで OUT=データセットを使用して日付キーを定義できます。

オプション引数

option
次のオプションを使用できます。

CONDENSE
DATEKEYDATA OUT=データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

DATEKEYDATA IN=オプションは、簡略化されたデータセットと簡略化されていないデータセットの両方を読み取ります。 詳細については、“DATEKEYDATA OUT=データセットでの変数の識別” (555 ページ)を参照してください。

LIST
DATEKEYDATA OUT=データセットに、使用可能な日付キーのリストだけを含めることを指定します。LIST オプションを指定すると、日付キー定義に必要なパラメーターは出力データセットに含まれません。

NODEFAULTS
DATEKEYDATA OUT=データセットに、事前定義された SAS 日付キーを含めないことを指定します。

DATEKEYDEF ステートメント

他の SAS プロシジャで解釈できる日付キーを定義します。複数の DATEKEYDEF ステートメントを指定できます。

注: これらの日付キーを使用して、予測モデルに含めることができるイベントを作成できます。

構文

DATEKEYDEF *SAS-variable-name*=*timing-value* </*qualifier-options*>;

オプション引数の要約

qualifier-options

必須引数

SAS-variable-name
DATEKEYDEF ステートメントで名前を指定します。

timing-value list
1 つ以上の日付キー、日付、日時値、またはオブザベーション番号を指定します。また、*value-list* を指定することもできます。

value-list の指定方法は、変数の種類によって決まります。

integer value-list
整数変数の場合、*integer value-list* は、1 つ以上の整数の明示的なリストかまたは間隔増分を指定した開始値と終了値のどちらか、あるいはその両方の形式の組み合わせです。

- *n* <...*n*>

  次に例を示します。

```
10,11,12
```

- *n* TO *n* <BY *increment*>

  次に例を示します。

```
10 to 12 by 1
```

- *n* <...*n*> TO *n* TO *n* <BY increment> <*n*<...*n*> >

  次に、例を示します。

```
11 to 5, 5 to 10 by 1;
```

SAS keyword value-list
文字変数の場合、*SAS keyword value-list* は、空白で区切られた 1 つ以上の一意の文字値のリストです。

"value-1" <"value-2"..."value-n">

次に、例を示します。

```
"INDEPENDENCE"
or
"INDEPENDENCE" "EASTER" "NEWYEARS"
```

SAS date value-list* and *SAS datetime value-list
日時値の場合、*value-list* は次の形式を取ります。

- *"SAS-value"i<..."SAS-value"i>*

  次に、一部の例を示します。

```
'01Jan2000'd,'01Feb2000'd, '01Mar2000'd
or
'01Mar1990:15:03:00'dt
or
'01Jan:15:03:00'dt,'1Feb:15:03:00'dt, '01Mar:15:03:00'dt
```

- *"SAS-value"i* TO *"SAS-value"i*< BY *interval*>

  注: *"SAS-value"i* TO *"SAS-value"i* は、整数値、日付値、日時値、時間値のいずれかにする必要があります。種類を混在させないでください。

  次に、一部の例を示します。

```
'01Jan2000'd TO '01Mar2000'd BY month
or
'01Jan1990:15:03:00'dt TO '01Mar1990:15:03:00'd
```

オプション引数

qualifier-options
次の修飾子オプションを使用できます。

AFTER=(<DURATION=*value*>)
タイミング値の後の日付キー定義を制御するオプションを指定します。DURATION=サブオプションは、AFTER= ( )オプションのかっこ内で使用します。DURATION は、タイミング値の後の日付キーの期間を指定します。

BEFORE=(<DURATION=*value*>)
タイミング値の前の日付キー定義を制御するオプションを指定します。DURATION=サブオプションは、BEFORE= ( )オプションのかっこ内で使用します。DURATION は、タイミング値の前の日付キーの期間を指定します。

LABEL='*SAS-label*'
日付キーに関連付けるラベルを指定します。'*SAS-label*'は、引用符で囲まれたテキスト文字列です。このテキスト文字列には最大 256 文字使用できます。

デフォルトのラベルは、'*SAS-variable-name*'です。*SAS-variable-name* は、DATEKEYDEF ステートメントで指定した名前です。ラベルは、DATEKEYDATA OUT=データセットに保存されます。

LOCALE='*POSIX locale*'
日付キーに関連付けるロケールを指定します。ロケールは、POSIX ロケール値にする必要があります。ロケール値のデフォルトはありません。

PERIOD=*interval*
日付キーの度数の間隔を指定します。たとえば、PERIOD=YEAR の場合、年単位パターンで周期的な日付キーが生成されます。

PERIOD=オプションを省略すると、日付キーは周期的にはなりません。PERIOD=オプションは、周期的ではないオブザベーション番号や、独自の周期性を持つ日付キーワードには適用されません。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

PULSE=*interval*
日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

日付キーが時間 ID 変数に対して評価される場合、デフォルトのパルスは 1 オブザベーションになります。DURATION=の値を指定せずに、PULSE=オプションを指定した場合、DURATION=の値はゼロに設定されます。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

RULE=*value*
定義されている日付キーに、少なくとも 1 つの日付キーを含む複数のタイミング値があるときに実行するアクションを指定します。

日付キーのタイミング値が、SAS 日付値、SAS 日時値、またはオブザベーション番号だけで構成されている場合、RULE=オプションは適用されません。また、タイミング値リストが単一の日付キーで構成されている場合も、RULE=オプションは適用されません。RULE=オプションは、AND 値と OR 値を受け入れます。デフォルトは RULE=OR です。

RULE=オプションの例を次に示します。

```
datekeykey JANUARY            / pulse=month;
datekeydef RainyDays='11JUL2013'd '13JUL2013'd '21JUL2013'd;
datekeydef HotDays=  '11JUL2013'd '16JUL2013'd '17JUL2013'd
                '18JUL2013'd '19JUL2013'd;
datekeydef FridaysInJanuary=JANUARY FRIDAY / rule=and;
datekeydef JanuaryPlusFridays=JANUARY FRIDAY / rule=or;
datekeydef HotandRainyDays=RainyDays HotDays / rule=and;
```

最初のステートメントでは、タイミング値リストが単一の日付キーで構成されているため、RULE=オプションは適用されません。2 番目と 3 番目のステートメントでは、タイミング値リストが SAS 日付値だけで構成されているため、RULE=オプションは適用されません。2 つの SAS 日付値、日時値、またはオブザベーション値の間での演算は常に OR になります。4 番目のステートメントでは、RULE=AND オプションにより、月が 1 月かつ曜日が金曜日の日付だけが特定されます。5 番目のステートメントでは、RULE=OR オプションにより、1 月のすべての日付と曜日が金曜日のすべての日付が特定されます。6 番目のステートメントでは、RULE=AND オプションにより、雨天かつ暑い日が特定されます。RULE=AND は、RainyDays と HotDays の 2 つの日付キーに適用されます。

通常、2 つの別個の期間の間での AND 演算の結果は空の値になります。そのため、別個の期間の間では、常に OR 演算を使用します(例: '13JUL2013'd, '01Mar1990:15:03:00'dt, 3)。

次のテーブルに、各オブザベーションで RULE=オプションがどのように解釈されるのかを示します。

表 16.1　RULE=オプション値の定義

RULE=オプション	名前	定義
AND	および	すべての日付キーおよび個々のタイミング値のグループで識別される期間を特定します。
OR	または	任意の日付キーまたは個々のタイミング値のグループで識別される期間を特定します。

SHIFT=*number*

タイミング値 δ をシフトするパルス数を指定します。デフォルトでは、タイミング値をシフトしません(δ= 0)。SHIFT=オプションを使用すると、リストのすべてのタイミング値(日付キーワードによって生成されるタイミング値を含む)がシフトされます。これによって、EASTER で SHIFT=を使用して、復活祭に基づく教会の祝日を指定できます。たとえば、次のステートメントは、復活祭の 2 日前と定められている聖金曜日を指定しています(Montes 2001)。

```
datekeydef GoodFriday=EASTER / shift=-2 pulse=day;
```

DATEKEYDSOPT ステートメント

データセットの入出力処理を指定したロケールに限定します。

構文

DATEKEYDSOPT LOCALE= '*POSIX locale*'<(ONLY)>;

オプション引数の要約

(ONLY)

必須引数

LOCALE=
入力/出力データセットのフィルターに使用するロケールを指定します。

'POSIX locale'
POSIX ロケール値を指定します。ロケール値のデフォルトはありません。

オプション引数

(ONLY)
入力データセットと出力データセットの両方で、指定したロケールだけを処理することを指定します。(ONLY)を指定しない場合、入力データセットと出力データセットの両方で、指定したロケールとデフォルト値(ロケールの指定なし)が処理されます。

DATEKEYKEY ステートメント

ユーザー定義または事前定義された SAS 日付キーを変更するか、別の日付キーから新しい日付キーを作成します。複数の DATEKEYKEY ステートメントを指定できます。

参照項目: “DATEKEYKEY ステートメントの使用” (551 ページ)

構文

DATEKEYKEY <*SAS-variable-name*=> *datekey-keyword* </*qualifier-options*>;

オプション引数の要約

SAS-variable-name

必須引数

datekey-keyword
ユーザー定義または事前定義の日付キーに対してデフォルトの SAS 変数名を指定します。

オプション引数

SAS-variable-name
新しい日付キーキーワードの名前を指定します。

> ヒント　DATEKEYCALENDAR オプションを指定すると、この名前で変数が作成され、キーワードのアクティブ期間と非アクティブ期間が示されます。このオプションが指定されていない場合、*datekey-keyword* は、変更された日付キーの名前になります。

その他のオプション引数

qualifier-option
次のオプションを使用できます。

AFTER=(<DURATION=*value*>)
タイミング値の後の日付キー定義を制御するオプションを指定します。DURATION=サブオプションは、AFTER= ( )オプションのかっこ内で使用します。

AFTER=オプションで使用するときには、DURATION はタイミング値の後の日付キーの期間を指定します。

BEFORE=(<DURATION=*value*>)
タイミング値の前の日付キー定義を制御するオプションを指定します。DURATION=サブオプションは、BEFORE= ( )オプションのかっこ内で使用します。

BEFORE=オプションで使用するときには、DURATION はタイミング値の前の日付キーの期間を指定します。

LABEL='*SAS-label*'
日付キーに関連付けるラベルを指定します。'SAS-label'は、引用符で囲まれたテキスト文字列です。このテキスト文字列には最大 256 文字使用できます。デフォルトのラベルは'*SAS-variable-name*'です。*SAS-variable-name* は、DATEKEYKEY ステートメントで指定した名前です。DATEKEYKEY ステートメントで *SAS-variable-name* が指定されていない場合、事前定義された SAS 日付キーのデフォルトのラベルが使用されます。ラベルは、DATEKEYDATA OUT=データセットに保存されます。

LOCALE='*POSIX locale*'
日付キーに関連付けるロケールを指定します。ロケールは、POSIX ロケール値にする必要があります。ロケール値のデフォルトはありません。

PERIOD=*interval*
日付キーの度数の間隔を指定します。たとえば、PERIOD=YEAR の場合、年単位パターンで周期的な日付キーが生成されます。PERIOD=オプションを省略すると、日付キーは周期的にはなりません。PERIOD=オプションは、周期的ではないオブザベーション番号や、独自の周期性を持つ日付キーワードには適用されません。 指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

PULSE=*interval*
日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。日付キーが時間 ID 変数に対して評価される場合、デフォルトのパルスは 1 オブザベーションになります。DURATION=の値を指定せずに、PULSE=オプションを指定した場合、DURATION=の値はゼロに設定されます。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

RULE=*value*
定義されている日付キーに、少なくとも 1 つの日付キーを含む複数のタイミング値があるときに実行するアクションを指定します。日付キーのタイミング値が、SAS 日付、SAS 日時、またはオブザベーション番号だけで構成されている場合、RULE=オプションは適用されません。また、タイミング値リストが単一の日付キーで構成されている場合も、RULE=オプションは適用されません。RULE=オプションは、AND 値と OR 値を受け入れます。デフォルトは RULE=OR です。RULE=オプションの例を次に示します。

```
datekeykey JANUARY / pulse=month;
datekeydef RainyDays='11JUL2013'd '13JUL2013'd '21JUL2013'd;
datekeydef HotDays=  '11JUL2013'd '16JUL2013'd '17JUL2013'd
                     '18JUL2013'd '19JUL2013'd;
datekeydef FridaysInJanuary=JANUARY FRIDAY / rule=and;
datekeydef JanuaryPlusFridays=JANUARY FRIDAY / rule=or;
datekeydef HotandRainyDays=RainyDays HotDays / rule=and;
```

最初のステートメントでは、タイミング値リストが単一の日付キーで構成されているため、RULE=オプションは適用されません。2 番目と 3 番目のステートメントでは、タイミング値リストが SAS 日付値だけで構成されているため、RULE=オプションは適用されません。2 つの SAS 日付値、日時値、またはオブザベーション値の間での演算は常に OR になります。4 番目のステートメントでは、RULE=AND オプションにより、両方とも 1 月の日付でかつ曜日が金曜日の日付だけが特定されます。5 番目のステートメントでは、RULE=OR オプションにより、1 月のすべての日付と曜日が金曜日のすべての日付が特定されます。6 番目のステートメントでは、RULE=AND オプションにより、雨天かつ暑い日が特定されます。RULE=AND は、RainyDays と HotDays の 2 つの日付キーに適用されます。

通常、2 つの別個の期間の間での AND 演算の結果は空の値になります。そのため、別個の期間の間では、常に OR 演算を使用します(例: '13JUL2013'D, '01Mar1990:15:03:00'DT, 3)。

表 16.50 (537 ページ)は、各オブザベーションで RULE=オプションがどのように解釈されるのかを説明します。

SHIFT=*number*
タイミング値 δ をシフトするパルス数を指定します。デフォルトでは、タイミング値をシフトしません(δ= 0)。SHIFT=オプションを使用すると、リストのすべてのタイミング値(日付キーワードによって生成されるタイミング値を含む)がシフトされます。

DATEKEYPERIODS ステートメント

入力時間 ID での日付キーのアクティブ期間をリストする変数を書き込みます。アクティブ期間の日付、日時値、またはオブザベーション番号が、関連付けられている日付キーとともにリストされます。

構文

DATEKEYPERIODS OUT=*SAS-data-set*;

必須引数

OUT=*SAS-data-set*
ID ステートメントで指定した ID 情報に基づいて、指定した日付キーのアクティブな日付、日時値、またはオブザベーションを含む出力データセットの名前を指定します。OUT=データセットには、BY ステートメントと ID ステートメントで指定した変数も含まれます。

ID ステートメント

入力/出力データセットのオブザベーションを識別する数値変数を指定します。

参照項目: “ID ステートメントの詳細” (554 ページ)

構文

ID *SAS-variable-name* INTERVAL=*interval* <*options*>;

オプション引数の要約

option

必須引数

SAS-variable-name
入力/出力データセットのオブザベーションを識別する数値変数を指定します。*SAS-variable-name* には、SAS 日付値、SAS 時間値、SAS 日時値、またはオブザベーション番号を指定できます。

INTERVAL=*interval*
入力時間 ID の度数を指定します。たとえば、入力データセットの時間 ID が四半期ごとのオブザベーションで構成されている場合は、INTERVAL=QTR を使用します。指定できる間隔については、*SAS/ETS User's Guide* の第 4 章 Date Intervals, Formats, and Functions を参照してください。

オプション引数

option
次のオプションを使用できます。

ALIGN=*option*
出力オブザベーションの識別に使用する SAS 日付の調整を制御します。ALIGN=オプションは、BEGINNING | BEG | B、MIDDLE | MID | M、ENDING | END | E の各値を受け入れます。

デフォルト　BEGINNING

END=*value*
データの終わりを表す SAS 日付値、SAS 時間値、または SAS 日時値を指定します。時間 ID 変数の最後のオブザベーションの値が *value* よりも小さい場合、系列は欠損値で拡張されます。時間 ID 変数の最後のオブザベーションの値が *value* より大きい場合、系列は切り捨てられます。たとえば、END="&sysdate"D では、自動マクロ変数 SYSDATE を使用して、一連の末尾を現在の日付に拡張するか切り捨てます。このオプション、START=オプション、および時間 ID 変数の最後のオブザベーションを使用できます。次のいずれかが当てはまる場合、デフォルトは時間 ID 変数の最後のオブザベーションの値です。

- START=オプションが指定されていません
- 時間 ID 変数の最後のオブザベーションの値が START=の値以上である。

それ以外の場合、デフォルトは START=値と同じです。

FORMAT=*format*
時間 ID 値の SAS 出力形式を指定します。FORMAT=オプションを指定しない場合、デフォルトのフォーマットは INTERVAL=オプションから暗黙的に取得されます。

START=*value*
データの始まりを表す SAS 日付値、SAS 時間値、または SAS 日時値を指定します。時間 ID 変数の最初のオブザベーションの値が *value* より大きい場合、系列には欠損値の接頭辞が付けられます。時間 ID 変数の最初のオブザベーションの値が *value* よりも小さい場合、系列は切り捨てられます。このオプションと END=オプションおよび時間 ID 変数の最初のオブザベーションを指定できます。次のいずれかに該当する場合、デフォルトは時間 ID 変数の最初のオブザベーションの値です。

- END=オプションが指定されていません
- 時間 ID 変数の最初のオブザベーションの値が END=の値以下である。

それ以外の場合、デフォルトは END=値と同じです。

VAR ステートメント

入力変数を出力カレンダー変数データセットにコピーします。VAR ステートメントを省略すると、BY ステートメントまたは ID ステートメントに出現する変数を除く、すべての数値変数が選択されます。

構文

VAR *variable(s)*;

必須引数

variable
出力カレンダー変数データセットにコピーする数値入力変数を指定します。

使用法: DATEKEYS プロシジャ

DATEKEYS プロシジャの使用

DATEKEYS プロシジャの機能概要

次のテーブルに、DATEKEYS プロシジャを制御するステートメントとオプションの概要を示します。

説明	ステートメント	オプション
ステートメント		
BY グループ処理を指定します。	BY	該当なし

説明	ステートメント	オプション
カレンダー変数のデータセットを指定します。	DATEKEYCALENDAR	該当なし
日付キーデータセットを指定します。	DATEKEYDATA	該当なし
日付キー定義を指定します。	DATEKEYDEF	該当なし
既存の日付キーに基づいて日付キー定義を指定します。	DATEKEYKEY	該当なし
日付キー別のアクティブ期間のリストを含むデータセットを指定します。	DATEKEYPERIODS	該当なし
時間 ID 変数を指定します。	ID	該当なし
カレンダー変数データセットにコピーする変数を指定します。	VAR	該当なし
データセットオプション		
入力データセットを指定します。	PROC DATEKEYS	DATA=
日付キーのカレンダー変数を含む出力データセットを指定します。	DATEKEYCALENDAR	OUT=
日付キー入力データセットを指定します。	DATEKEYDATA	IN=
日付キー出力データセットを指定します。	DATEKEYDATA	OUT=
日付キー出力データセットを簡略化することを指定します。	DATEKEYDATA	CONDENSE
日付キー出力データセットに日付キーのリストだけを含めることを指定します。	DATEKEYDATA	LIST
日付キー出力データセットに事前定義された SAS 日付キーを含めないことを指定します。	DATEKEYDATA	NODEFAULTS
入力/出力データセットで、指定したロケールを処理することを指定します。	DATEKEYDSOPT	LOCALE=
日付キーのアクティブな日付を含む出力データセットを指定します。	DATEKEYPERIODS	OUT=
開始時間 ID 値を指定します。	ID	START=
終了時間 ID 値を指定します。	ID	END=

説明	ステートメント	オプション
ID 変数の出力形式を指定します。	ID	FORMAT=
カレンダー変数の出力形式オプション		
入力時間 ID の終わりを超えてカレンダー変数を拡張します。	PROC DATEKEYS	LEAD=
時間 ID 変数の度数を指定します。	ID	INTERVAL=
間隔調整を指定します。	ID	ALIGN=
その他のオプション		
出力データセットの変数を並べ替え後の順序にすることを指定します。	PROC DATEKEYS	SORTNAMES
エラーメッセージと警告メッセージを制限します。	PROC DATEKEYS	MAXERROR=

日付キー定義

日付キー定義は、参照日付キーに関連付ける期間を定義することを目的としています。これらの期間は、イベントなどの時間に依存する機能を定義する他の SAS プロシジャで解釈されます。日付キー定義の期間は、対象となる期間と比較されます。対象となる期間が日付キー定義の範囲内であれば、適切なアクションが実行されます。日付キー定義は、DATEKEYDATA ステートメントの OUT=オプションを使用することで、出力ファイルに書き込むことができます。

日付キーを定義したら、SAS 変数名を使用して参照します。DATEKEYDATA ステートメントを使用して、日付キー定義を出力ファイルに書き込むと、日付キーは SAS 変数名で識別されます。事前定義された SAS 日付キーと同様に、ユーザー定義の日付キー名を使用してイベントを指定すると、日付キー定義を使用してダミー変数が作成されます。ダミー変数名は日付キーの SAS 参照名と同じです。

各日付キーには、一意の SAS 変数名が必要です。2 つの日付キー定義が同じ名前の場合、次の規則が適用されます。

- 同じ名前を使用する 2 つの DATEKEYDEF ステートメントが存在する場合、2 番目のステートメントが使用されます。
- DATEKEYDEF ステートメントと、DATEKEYDATA ステートメントを使用して指定したデータセットの両方で日付キーが定義されている場合、DATEKEYDEF ステートメントでの定義が使用されます。

- 事前定義された SAS 日付キーではなく、DATEKEYDEF、DATEKKEYKEY、または DATEKEYDATA ステートメントを使用して定義された日付キーが使用されます。

タイミング値リスト指定の詳細

各 DATEKEYDEF ステートメントは、1 つ以上のタイミング値を使用して定義する必要があります。タイミング値は、リストを使用して指定できます。リストの各項目として、事前定義された SAS 日付キーワード、整数、SAS 日付、SAS 日時値、または *value-list* を指定できます。たとえば、次の DATEKEYDEF ステートメントは、これらの各方法を上記の順序で使用したタイミング値を指定しています。

```
datekeydef datekey1=USINDEPENDENCE 10 '25Dec2000'd
           '01Mar1990:15:03:00'dt
           '01Jan2000'd to '01Mar2000'd by month;
```

このタイミング値は、"該当する年の 7 月 4 日、時系列またはデータセット内で 10 番目のオブザベーション、2000 年 12 月 25 日、1990 年 3 月 1 日 3:03PM、2000 年 1 月 1 日、2000 年 2 月 1 日、2000 年 3 月 1 日"と解釈されます。

次の 2 つの DATEKEYDEF ステートメントは、同一のタイミング値を指定しています。

```
datekeydef MyFirstDATEKEY='01Jan2000'd to '01Mar2000'd by month;
datekeydef MyNextDATEKEY=('01Jan2000'd, '01Feb2000'd, '01Mar2000'd );
```

timing-value list はかっこで囲むことができ、リストの項目はカンマで区切ることができます。数字は整数にする必要があり、常にオブザベーション番号として解釈されます。*value-list* は、オブザベーション番号、SAS 日付、または SAS 日時値に基づくことができます。ただし、リストの最初と 2 番目の値は、同じ型である必要があります。SAS では、常に、2 番目の値の型が最初の値の型と同じであると想定し、その前提でステートメントの解釈を試みます。次のステートメントは、不安定な結果をもたらします。

```
datekeydef baddatekey='01Jan2000'd to '01Mar2000:00:00:00'dt by month;
```

DATEKEYS プロシジャは、予想よりもかなり長いリストを生成するか、メモリ不足で実行できなくなります。

注: *value-list* では日付、日時、整数、値の型を混在させないでください。

次のテーブルに、*timing-value list* で使用できる祝日の日付キーワードとその定義を示します。

表 16.2　祝日の日付キーワードと定義

日付キーワード	定義
BOXING	12 月 26 日
CANADA	7 月 1 日
CANADAOBSERVED	7 月 1 日(7 月 1 日が日曜日の場合は 7 月 2 日)

日付キーワード	定義
CHRISTMAS	12 月 25 日
COLUMBUS	10 月の第 2 月曜日
EASTER[1]	復活の主日
FATHERS	6 月の第 3 日曜日
HALLOWEEN	10 月 31 日
LABOR	9 月の第 1 月曜日
MLK	1 月の第 3 月曜日
MEMORIAL	5 月の最後の月曜日
MOTHERS	5 月の第 2 日曜日
N<*n*>W<*w*><MON>YR	NWKDOM(*n*、*w*、*m*、*year*)で指定した日付。*m* は MON で指定される月に対応し、*year* はそのデータに関連する任意の年を表します。 例: N4W5NOVYR は THANKSGIVING と同じです。
NEWYEAR	1 月 1 日
THANKSGIVING	11 月の第 4 木曜日
THANKSGIVINGCANADA	10 月の第 2 月曜日
USINDEPENDENCE	7 月 4 日
USPRESIDENTS	2 月の第 3 月曜日(1971 年以降)
VALENTINES	2 月 14 日
VETERANS	11 月 11 日
VETERANSUSG	月曜日～金曜日のスケジュールに対応する米国政府の振替休日
VETERANSUSPS	月曜日～土曜日のスケジュール(米国郵便局)に対応する米国政府の振替休日
VICTORIA	5 月 24 日以前の直近の月曜日

1. 復活祭の日付は、Montes (2001)で説明されている方法を使用して計算されます。

次のテーブルに、*timing value list* で使用できる季節的な日付キーワードとその定義を示します。

表 16.3　季節的な日付キーワードと定義

日付キーワード	定義
SECOND_1, ...SECOND_60	指定した秒。
MINUTE_1, ...MINUTE_60	指定した分の開始。
HOUR_1, ...HOUR_24	指定した時間の開始。
SUNDAY, ...SATURDAY	時系列でのすべての日曜日など。
WEEK_1, ...WEEK_53	その年の第 *n* 週目の最初の日。PULSE=WEEK.*n* は、*n* NE 1 に対してこの日付をシフトします。
TENDAY_1, ...TENDAY_36	該当する月の 1 日、11 日、または 21 日。
SEMIMONTH_1, ...SEMIMONTH_24	該当する月の 1 日または 16 日。
JANUARY, ...DECEMBER	指定した月の 1 日。
QTR_1、QTR_2、QTR_3、QTR_4	四半期の初日。PULSE=QTR.*n* は、*n* NE 1 に対してこの日付をシフトします。
SEMIYEAR_1, SEMIYEAR_2	半期の初日。PULSE=SEMIYEAR.*n* は、*n* NE 1 に対してこの日付をシフトします。

タイミング値は、ユーザーが指定した用途および関連する時間に対して評価されます。用途と一致するタイミング値を選択します。特に、日付または時間の情報が指定されていない場合、日付と日時のタイミング値は無視され、分析に使用できるのはオブザベーション番号だけになります。

オプションを使用したタイミング値の変更の詳細

修飾子オプションは、各タイミング値で適用される機能の変更を定義します。オプションは、次の順序で適用されます。

1. SHIFT=の値が指定されている場合、SHIFT=の値と PULSE=の値(指定されている場合)を使用してリスト内のタイミング値がシフトされます。
2. BEFORE=オプションまたは AFTER=オプションで DURATION=の値が指定されている場合、DURATION=と PULSE=の値に基づいてシフトされたタイミング値を中心にして、連続する間隔が定義されます。

3 PERIOD=の値が指定されている場合、ステップ 1 と 2 の結果であるタイミング値に基づいて周期的な値が生成されます。

タイミング値とオプションに基づくアクティブな間隔の識別の詳細

日付キーが時間 ID に対して評価されると、その時間 ID の間隔に対してアクティブな間隔が識別されます。そのため、アクティブ期間は、時間 ID と日付キー定義の両方に依存します。

シフトされたタイミング値によって指定されたオブザベーション(t_i)は、INTNX(interval,timing—value,s,'same')によって生成された日付を含むオブザベーションです。ここで、SHIFT=*s*、PULSE=*interval* です。PULSE=の値が指定されていない場合、デフォルトは PULSE=OBS です。これは、PULSE=*interval* と同じであり、*interval* は時間 ID の間隔を表します。

表 16.4 時系列に適用したときの日付キーの開始オブザベーションと終了オブザベーションの計算

BEFORE=(DURATION=*value*)	PULSE=*value*	t_b の定義
ALL	利用できません	t_b=1(データセットの最初のオブザベーション)、または t_b= START=で指定されたオブザベーション
m=0	指定なし	t_b=t_i(シフトされたタイミング値によって指定されたオブザベーション)
m>0	指定なし	t_b=t_i-*m*
m>=0	*interval*	t_b=日付で指定されたオブザベーション INTNX(*interval*, *timing-value*, —*m*, 'begin')
AFTER=(DURATION=*value*	PULSE=*value*	t_e の定義
ALL	利用できません	t_e=データセットの最後のオブザベーション、または t_e= END=で指定されたオブザベーション

BEFORE=(DURATION=*value*)	PULSE=*value*	t_b の定義
n=0	指定なし	t_e=t_i(シフトされたタイミング値で指定されたオブザベーション)
n>0	指定なし	t_e=t_i+*n*
n>=0	*interval*	t_e=日付で指定されたオブザベーション INTNX(*interval*, *timing-value*,*n*,'end')

次のテーブルに、日付キー定義のアクティブ期間を示します。

表 16.5　日付キー定義のアクティブ期間

BEFORE=(DURATION=*m*)	AFTER=(DURATION=*n*)	アクティブオブザベーション
m=ALL	*n*=ALL	ξ_{it}, for all *t*
m=finite	*n*=ALL	ξ_{it}, for all $t>=t_b$
m=ALL	*n*=finite	ξ_{it}, for all $t<=t_e$
m=finite	*n*=finite	ξ_{it}, if $t_b<=t<=t_e$

オブザベーションが日付キーのアクティブ期間内にない場合、そのオブザベーションは日付キーのアクションによって変更されません。

アクティブ期間は、常にシフトされたタイミング値で発生します。タイミング値の前に 3 つのオブザベーション、タイミング値に 1 つのオブザベーション、タイミング値の後に 4 つのオブザベーションをそれぞれ指定します。次のように、オブザベーションは合計 3+1+4=8 個になります。

```
datekeydef E1='01JAN1950'd / before=(duration=3)
                             after=(duration=4);
```

この例では、次のように、BEFORE=、AFTER=、PULSE=の各オプションの組み合わせを使用して、タイミング値の前に 3 週、タイミング値に 1 週、タイミング値の後に 4 週指定しています。

```
datekeydef E1='01JAN1950'd / before=(duration=3)
                             after=(duration=4)
                             pulse=week;
```

DURATION=ALL は、アクティブ期間を期間の始まり(BEFORE=)または終わり(AFTER=)まで拡大する必要があることを意味します。一方の DURATION=の値だけを指定した場合、もう一方の値はゼロと見なされます。どちらの DURATION=の値

も指定しない場合、両方の DURATION=の値がゼロに設定されます。
DURATION=ALL は、日付キー定義データセットで、"A"と表示される特殊な欠損値として表されます。詳細については、欠損変数値を参照してください。

DATEKEYKEY ステートメントの使用

既存の日付キー定義からの新しい日付キーの作成

DATEKEYKEY ステートメントを PROC DATEKEYS とともに使用して、既存の日付キー定義から新しい日付キーを作成し、処理に使用できます。ユーザー定義の日付キーに基づく SAS イベントは、EVENTDS=システムオプションを指定することで、PROC HPFDIAGNOSE と PROC HPFENGINE から直接使用することもできます。

ユーザー定義の日付キー変数には、ユーザーが定義したタイミング値と修飾子が含まれます。事前定義された SAS 日付キーを使用して定義された DATEKEYKEY 変数には、事前定義された日付キーキーワードに関連付けられたタイミング値と修飾子の事前定義されたセットが含まれます。ステートメントオプションを使用して、修飾子を再定義できます。オプションは DATEKEYDEF ステートメントと同じです。
“概念: DATEKEYS プロシジャ” (526 ページ)において、日付キーに基づくイベントのデフォルトの SAS 変数名は、日付キーキーワードです。ただし、日付キーに異なる SAS 名を指定することはできます。たとえば、CHRISTMAS の事前定義された日付キーの名前を XMAS に変更するには、次のステートメントを使用します。

```
datekeykey xmas=christmas;
```

事前定義された SAS 日付キーに関連付けられている修飾子を再定義し、日付キーの名前を変更しない場合、事前定義された SAS 日付キーの再定義という影響を及ぼします。この再定義が発生するのは、事前定義された SAS 定義よりもユーザー定義が優先されるためです。次の例では、ハロウィンのパルスが 1 日で、感謝祭のパルスが 1 か月である FALLHOLIDAYS という名前のイベントを生成します。

```
datekeykey thanksgiving / pulse=month;
eventcomb fallholidays=halloween thanksgiving;
```

次のテーブルに、事前定義された SAS 日付キーキーワードを作成する方法を示します。また、それらの事前定義された日付キーのデフォルトの修飾子オプションも示します。

表 16.6　DATEKEYKEY の事前定義されたイベントキーワードの定義

変数名または変数名の形式	説明	修飾子オプション
AO<obs>OBS AO<date>D AO<datetime>DT	外れ値	BEFORE=(DURATION=0) AFTER=(DURATION=0)
LS<obs>OBS	レベルシフト	BEFORE=(DURATION=0)

変数名または変数名の形式	説明	修飾子オプション
LS<date>D LS<datetime>DT		AFTER=(DURATION=ALL)
TLS<obs>OBS<n> TLS<date>D<n> TLS<datetime>DT<n>	一時レベルシフト	BEFORE=(DURATION=0) AFTER=(DURATION=<n>)
NLS<obs>OBS NLS<date>D NLS<datetime>DT	負のレベルシフト	BEFORE=(DURATION=0) AFTER=(DURATION=ALL)
CBLS<obs>OBS CBLS<date>D CBLS<datetime>DT	米国国勢調査局のレベルシフト	SHIFT=-1 BEFORE=(DURATION=ALL) AFTER=(DURATION=0)
TC<obs>OBS TC<date>D TC<datetime>DT	一時変更	BEFORE=(DURATION=0) AFTER=(DURATION=ALL)
<date keyword>	日付パルス	BEFORE=(DURATION=0) AFTER=(DURATION=0)
LINEAR QUAD CUBIC	多項式傾向	BEFORE=(DURATION=ALL) AFTER=(DURATION=ALL) デフォルトのタイミング値は 0 オブザベーション
INVERSE LOG	傾向	BEFORE=(DURATION=0) AFTER=(DURATION=ALL) デフォルトのタイミング値は 0 オブザベーション
<seasonal keywords>	季節的	PULSE=はキーワードに依存 BEFORE=(DURATION=0)

変数名または変数名の形式	説明	修飾子オプション
		AFTER=(DURATION=0) キーワードに基づくタイミング値

ユーザー定義の日付キーの変更または複製

DATEKEYKEY ステートメントを同様の方法で使用して、ユーザー定義の日付キーを変更または複製できます。次の例では、DATEKEYDEF ステートメントを使用して、SPRING という名前の単純なイベントを定義します。DATEKEYKEY ステートメントを使用して、SPRING イベントの定義を変更します。次に、DATEKEYKEY ステートメントを使用して、前に定義した SPRING というユーザーイベントに基づく SPRINGBREAK という名前の新しいイベントを作成します。したがって、この例では、合計 2 つの日付キー(SPRING と SPRINGBREAK)を定義します。DATEKEYKEY ステートメントを使用して、修飾子を変更できます。このステートメントを使用してタイミング値を変更することはできません。

```
datekeydef spring='20mar2005'd;
datekeykey spring / pulse=day;
datekeykey SPRINGBREAK=spring / pulse=week;
```

前出の日付キーを SPRINGHOLIDAYS という名前のデータセットに保存したら、次の例の最初の DATEKEYKEY ステートメントは、SPRING を FirstDayOfSpring という名前の日付キーとして複製しています。2 番目の DATEKEYKEY ステートメントは、SPRINGBREAK の日付キー名の大文字/小文字を変更しています。

```
datekeydata in=springholidays;
datekeykey FirstDayOfSpring=spring;
datekeykey Springbreak=springbreak;
```

前に定義した日付キーを参照する日付キー名では、大文字と小文字は区別されません。ただし、新しい日付キーを作成する際に使用する日付キー名は、DATEKEYDATA OUT=データセットの_NAME_変数で大文字と小文字の区別が保持されます。

SAS 日付キーキーワード

表 16.51 (546 ページ)に記載されている日付キーワードを、事前定義された SAS 日付キーキーワードとして DATEKEYDEF ステートメントで使用できます。タイミング値は、表 16.51 (546 ページ)での定義に従います。デフォルトの修飾子は、表 16.55 (551 ページ)に示されている通りです。 表 16.52 (548 ページ)は、事前定義された SAS 日付キーキーワードとして使用できる季節的なキーワードを示します。季節的なキーワードのデフォルトの修飾子は、表 16.55 (551 ページ)に示されています。表 16.56 (554 ページ)には、日付とオブザベーション番号を、AO、LS、TLS、NLS、CBLS、および TC タイプの事前定義された日付キーにどのようにエンコードするかが記述されています。

事前定義された SAS 日付キーワードには、アクティブ期間だけが含まれます。

表 16.7 EAO、LS、TLS、NLS、CBLS、TC タイプの DATEKEYKEY 変数名へのデータ情報のエンコーディング

変数名の形式	例	参照先
AO<int>OBS	AO15OBS	15 番目のオブザベーション
AO<date>D	AO01JAN2000D	'01JAN2000'D
AO<date>h<hr>m<min>s<sec>DT	AO01Jan2000h12m34s56DT	'01Jan2000:12:34:56'DT
TLS<int>OBS<n>	TLS15OBS10	15 番目のオブザベーション
TLS<date>D<n>	TLS01JAN2000D10	'01JAN2000'D
TLS<date>h<hr>m<min>s<sec>DT<n>	TLS01Jan2000h12m34s56DT10	'01Jan2000:12:34:56'DT

ID ステートメントの詳細

ID ステートメントは、SAS 変数名と間隔を受け入れます。ID 変数の値は、SAS 日付値、時間値、日時値、またはオブザベーション番号であることが前提となります。また、ID ステートメントでは、アクティブ期間と非アクティブ期間を調べるために必要な度数も指定します。指定された情報は、DATEKEYCALENDAR および DATEKEYPERIODS ステートメントを使用して生成されるすべての変数に影響します。DATEKEYCALENDAR ステートメントと DATEKEYPERIODS ステートメントのどちらも指定されていない場合、ID ステートメントは処理に影響を及ぼしません。これは、DATEKEYDEF の定義は時間 ID 値および度数とは無関係であるためです。ID ステートメントを指定する場合、INTERVAL=オプションも指定する必要があります。ID ステートメントを指定しない場合は、(BY グループに対する)オブザベーション番号が時間 ID として使用されます。この場合、オブザベーション番号に基づく日付キーのタイミング値だけが入力時間 ID に適用されて、カレンダー変数が作成されます。SAS 日付値または日時値に基づくタイミング値は無視されます。

データセットの出力

データセットの出力の作成

DATEKEYS プロシジャでは、DATEKEYCALENDAR OUT=、DATEKEYDATA OUT=、DATEKEYPERIODS OUT=の各データセットを作成できます。DATEKEYDATA OUT=データセットには、別の SAS プロシジャへの入力に使用できる日付キー定義が含まれます。DATEKEYDATA OUT=ステートメントで LIST オプションを指定すると、出力データセットに使用可能な日付キーのリストが含まれます。DATEKEYCALENDAR OUT=データセットには、入力時間 ID に関連する日付キーのアクティブ期間を示すカレンダーインジケーター変数が含まれます。DATEKEYPERIODS OUT=データセットには、入力時間 ID および関連付けられた日付に対応するアクティブな 日付キーに関する情報が含まれます。

DATEKEYCALENDAR OUT=データセットでの変数の識別

DATEKEYCALENDAR OUT=データセットには、BY ステートメントでリストされている変数、ID 変数、VAR ステートメントで定義されている変数、プロシジャによって生成されたカレンダー変数が含まれます。カレンダーインジケーター変数は、入力時間 ID に関連する日付キーのアクティブ期間を示します。DATEKEYCALENDAR ステートメントでは、SUM=オプションを指定できます。この場合、SUM=オプションで指定した変数が DATEKEYCALENDAR OUT=データセットに含まれます。この変数には、カレンダーインジケーター変数の合計が含まれます。

DATEKEYDATA OUT=データセットでの変数の識別

DATEKEYDATA OUT=データセットには、次の変数が含まれます。CONDENSE オプションのデフォルト値も提供されます。変数内のすべてのオブザベーションがデフォルト値と同じである場合、日付キー定義データセットからその変数を除外できます。

CLASS
: すべての日付キーのクラスが DATEKEY であることを指定します。_CLASS_のデフォルトは DATEKEY です。

DATEINTRVL
: 日付の *value-list* の間隔を指定します。_DATEINTRVL_のデフォルトは間隔なしです。これは、"."で表されます。

DTINTRVL
日時の *value-list* の間隔を指定します。_DTINTRVL_のデフォルトは間隔なしです。これは、"."で表されます。

_DUR_AFTER_
タイミング値の後の期間数を指定します。_DUR_AFTER_のデフォルトは 0 です。

_DUR_BEFORE_
タイミング値の前の期間数を指定します。_DUR_BEFORE_のデフォルトは 0 です。

ENDDATE
value-list で使用する最後の日付タイミング値を指定します。_ENDDATE_のデフォルトは日付なしです。これは欠損値で表されます。

ENDDT
value-list で使用する最後の日時タイミング値を指定します。_ENDDT_のデフォルトは日時なしです。これは欠損値で表されます。

ENDOBS
value-list で使用する最後のオブザベーション番号タイミング値を指定します。_ENDOBS_のデフォルトはオブザベーション番号なしです。これは欠損値で表されます。

KEYNAME
事前定義された日付キーキーワードまたはユーザー定義の日付キーキーワードを指定します。All_KEYNAME_の値は大文字で表示されます。ただし、_KEYNAME_の値がユーザー定義のキーワードを参照している場合、実際の名前では大文字と小文字を混在させることができます。_KEYNAME_のデフォルトはキー名なしです。これは、"."で表されます。

LABEL
日付キーのラベルまたは説明を指定します。ラベルを指定しない場合、デフォルトのラベル値は"."と表示されます。詳細については、*SAS システムオプション: リファレンス*の LABEL システムオプションを参照してください。

LOCALE
日付キーのロケールを指定します。_LOCALE_値は有効な POSIX ロケール値です。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*の LOCALE システムオプションを参照してください。デフォルトはありません。

NAME
日付キーの参照名を指定します。_NAME_は、保持されている文字種(大文字/小文字)で表示されます。_NAME_は SAS 変数名であるため、任意の文字種を使用して日付キーを参照できます。_NAME_変数は必須です。デフォルトはありません。

OBSINTRVL
オブザベーション番号の *value-list* の間隔の長さを指定します。_OBSINTRVL_のデフォルトは間隔なしです。これは、"."で表されます。

PERIOD
日付キーを繰り返す必要がある度数の間隔を指定します。この値がない場合、日付キーは周期的にはなりません。_PERIOD_のデフォルトは間隔なしです。これは、"."で表されます。

PULSE
DURATION の値の単位を定義する間隔を指定します。_PULSE_のデフォルトは間隔なし(1 オブザベーション)です。これは、"."で表されます。

RULE
日付キーのタイミング値を組み合わせるときに使用する規則を指定します。日付キーの_RULE_のデフォルトは OR です。

SHIFT
タイミング値をシフトする PULSE=interval の数を指定します。シフトは、正(時間を進める)にすることも負(時間を戻す)にすることもできます。PULSE=を指定しない場合、オブザベーションでシフトが発生します。_SHIFT_のデフォルトは 0 です。

STARTDATE
日付タイミング値または *value-list* で使用する最初の日付タイミング値を指定します。_STARTDATE_のデフォルトは日付なしです。これは欠損値で表されます。

STARTDT
日時タイミング値または *value-list* で使用する最初の日時タイミング値を指定します。_STARTDT_のデフォルトは日時なしです。これは欠損値で表されます。

STARTOBS
オブザベーション番号タイミング値または *value-list* で使用する最初のオブザベーション番号タイミング値を指定します。_STARTOBS_のデフォルトはオブザベーション番号なしです。これは欠損値で表されます。

DATEKEYPERIODS OUT=データセットでの変数の識別

DATEKEYPERIODS OUT=データセットには、次の変数が含まれます。

LOCALE
日付キーのロケールを指定します。_LOCALE_値は有効な POSIX ロケール値です。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*の LOCALE システムオプションを参照してください。デフォルトはありません。

NAME
日付キーの参照名を指定します。_NAME_は、保持されている文字種(大文字/小文字)で表示されます。_NAME_は SAS 変数名であるため、任意の文字種を使用して日付キーを参照できます。_NAME_変数は必須です。デフォルトはありません。

STARTDATE
日付タイミング値または *value-list* で使用する最初の日付タイミング値を指定します。_STARTDATE_のデフォルトは日付なしです。これは欠損値で表されます。

TIME ID
日付を指定します。

LIST オプションを使用した日付キーのリスト作成

DATEKEYDATA ステートメントで LIST オプションを使用して、日付キーのリストを作成できます。DATEKEYDATA OUT=ステートメントで LIST オプションを指定すると、出力データセットが変更されます。LIST オプションは、データセットで定義されている使用可能な日付キーをリストするデータセットを作成することを目的としています。LIST オプションを指定すると、_LOCALE_変数、_NAME_変数、および _LABEL_変数だけがデータセットに含まれます。データセットには、日付キーごとに 1 つのオブザベーションが含まれます。

BY ステートメント変数のデータセットの作成

DATEKEYPERIODS ステートメントを使用して、BY ステートメントのデータセットを作成できます。DATEKEYPERIODS OUT=データセットには、次の項目が含まれます。

- BY ステートメントでリストされている変数
- 入力時間 ID に関連付けられているアクティブな 日付キーのリスト
- アクティブな 日付キーに関連付けられている ID 変数の日付
- 時間 ID のアクティブ期間に関連付けられている開始日、日時、またはオブザベーション値

DATEKEYPERIODS OUT=データセットには、BY 変数と ID 変数に加え、次の変数も含まれます。

NAME
: 日付キーの参照名を指定します。_NAME_は、保持されている文字種(大文字/小文字)で表示されます。表示される日付キーは、ID 変数とオブザベーションの _STARTDATE_、_STARTDT_、または_STARTOBS_の値で指定された日付に対してアクティブです。

時間 ID 変数に基づいて、次のいずれかの変数が含まれます。

STARTDATE
: 時間 ID 変数の値に関連する INTERVAL=オプションで指定された期間の始まりに関連付ける日付を指定します。

STARTDT
: 時間 ID 変数の値に関連する INTERVAL=オプションで指定された期間の始まりに関連付ける日時を指定します。

STARTOBS
: 時間 ID 変数の値に関連する INTERVAL=オプションで指定された期間の始まりに関連付けるオブザベーション番号を指定します。

書き込まれる出力

DATEKEYS プロシジャには、ログに記録された警告メッセージとエラーメッセージ以外に書き込まれる出力はありません。

例: DATEKEYS プロシジャ

例 1: 日付キー定義データセットの作成方法

要素:

PROC DATEKEYS ステートメントオプション
ID
DATEKEYDEF
DATEKEYKEY
DATEKEYDATA
DATEKEYPERIODS

詳細

この例では、2 つの方法で日付キー定義データセットを作成します。最初の方法では、DATA ステップを使用して、商品が販売された金曜日の日付キーデータセットを作成します。2 番目の方法では、DATEKEYS プロシジャを使用して、日付キー定義データセットを作成します。TimeSeriesDates データセットには、時系列用の時間 ID 変数が含まれます。

DATA ステップの WhiteSaleDates で定義される日付は、DATEKEYS プロシジャを使用して定義できます。アクティブな日付は、アウトプット 16.89 (561 ページ)に示されているデータセットと同じです。ただし、DATEKEYS プロシジャの定義は連続しています。

プログラム: DATA ステップの使用

```
options eventds=(nodefaults);

data TimeSeriesDates(keep=date);
  set sashelp.citiday;
```

```
  format date date.;
run;

data WhiteSaleDates(keep=_name_ _startdate_);
  set TimeSeriesDates;
  _name_='WhiteSale';
  if (month(date)=1) then do;
    if (year(date)=1991 or year(date)=1992) then do;
      if (weekday(date)=6) then do;
        _startdate_=date;
      end;
      else delete;
      end;
    else delete;
    end;
  else delete;
  format _startdate_ date.;
run;

proc print data=WhiteSaleDates;
run;
```

プログラムの説明

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。NODEFAULTS オプションは、デフォルトのイベント定義を使用しないことを指定します。event-data-set リストで指定されたイベントだけが使用されます。

```
options eventds=(nodefaults);
```

TimeSeriesDates データセットを作成します。 この例のデータセットを作成するには、DATA ステップで sashelp.citiday を使用します。

```
data TimeSeriesDates(keep=date);
  set sashelp.citiday;
  format date date.;
run;
```

EVENTDS=システムオプションで使用するために、WhiteSalesDates という日付キーデータセットを作成します。 DATA ステップを使用して日付キーデータセットを作成するときは、特定の日付キーを定義するすべてのオブザベーションが連続している必要があります。必要に応じて、SORT プロシジャを使用して、_NAME_変数でデータセットを並べ替えます。DATA ステップでは、1991 年 1 月から 1992 年 1 月までの間に販売された商品を記述しています。ユーザー定義の日付キーは、商品が販売された金曜日を識別します。

```
data WhiteSaleDates(keep=_name_ _startdate_);
  set TimeSeriesDates;
  _name_='WhiteSale';
  if (month(date)=1) then do;
    if (year(date)=1991 or year(date)=1992) then do;
      if (weekday(date)=6) then do;
        _startdate_=date;
      end;
      else delete;
```

```
      end;
    else delete;
    end;
  else delete;
  format _startdate_ date.;
run;

proc print data=WhiteSaleDates;
run;
```

HTML 出力

アウトプット 16.1　ユーザー定義の日付キーデータセット

The SAS System

Obs	_NAME_	_STARTDATE_
1	WhiteSale	04JAN91
2	WhiteSale	11JAN91
3	WhiteSale	18JAN91
4	WhiteSale	25JAN91
5	WhiteSale	03JAN92
6	WhiteSale	10JAN92
7	WhiteSale	17JAN92
8	WhiteSale	24JAN92
9	WhiteSale	31JAN92

プログラム: DATEKEYS プロシジャの使用

```
proc datekeys data=sashelp.citiday;
  id date interval=day;
  datekeydef Years1991_1992='01JAN1991'd / pulse=year after=(duration=1);
  datekeykey January / pulse=month;
  datekeydef WhiteSale=JANUARY FRIDAY Years1991_1992 / rule=and;
  datekeydata out=WhiteSaleDefinitions condense;
  datekeyperiods out=WhiteSaleActiveDates;
run;
proc print data=WhiteSaleActiveDates(where=(_name_='WhiteSale'));
run;
```

```
proc print data=WhiteSaleDefinitions;
run;
```

プログラムの説明

DATEKEYS プロシジャを開始します。 DATA=オプションには、ID ステートメントで使用される変数が含まれます。

```
proc datekeys data=sashelp.citiday;
```

時間 ID 変数を指定します。 ID ステートメントは、データセットのオブザベーションを識別する数値変数の名前を指定します。ID ステートメントを使用する場合は、INTERVAL=オプションを使用する必要があります。この例では、INTERVAL=day です。

```
id date interval=day;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。PULSE=は、日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

```
datekeydef Years1991_1992='01JAN1991'd / pulse=year after=(duration=1);
```

日付キーを変更するか、新しい日付キーを作成します。 タイミング値リストで January を使用しているので、1 月 1 日を表します。DATEKEYKEY を PULSE=MONTH とともに使用すると、1 月全体を対象とする日付キーが作成されます。

```
datekeykey January / pulse=month;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。RULE=AND オプションにより、1 月の金曜日を特定します。

```
datekeydef WhiteSale=JANUARY FRIDAY Years1991_1992 / rule=and;
```

日付キーを出力データセットに書き込みます。 DATEKEYDATA OUT=データセットには、WhiteSale 日付キーの定義が含まれます。CONDENSE オプションは、出力データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

```
datekeydata out=WhiteSaleDefinitions condense;
```

入力時間 ID での日付キーのアクティブ期間をリストする変数を書き込みます。 アウトプット 16.90 (563 ページ)に示すように、DATEKEYPERIODS OUT=データセットには、WhiteSale 日付キーのアクティブ期間が含まれます。

```
datekeyperiods out=WhiteSaleActiveDates;
```

この例を実行します。 RUN ステートメントにより、プログラムが実行されます。

```
run;
```

アクティブな販売日データセットを書き込みます。 アウトプット 16.90 (563 ページ)に結果が示されます。

```
proc print data=WhiteSaleActiveDates(where=(_name_='WhiteSale'));
run;
```

WhiteSale 日付キーの定義を書き込みます。 アウトプット 16.91 (563 ページ)に結果が示されます。

```
proc print data=WhiteSaleDefinitions;
run;
```

HTML 出力

アウトプット 16.2　日付キーのアクティブ期間データセットの出力

The SAS System

Obs	_LOCALE_	DATE	_NAME_	_STARTDATE_
557	.	04JAN1991	WhiteSale	04JAN1991
558	.	11JAN1991	WhiteSale	11JAN1991
559	.	18JAN1991	WhiteSale	18JAN1991
560	.	25JAN1991	WhiteSale	25JAN1991
561	.	03JAN1992	WhiteSale	03JAN1992
562	.	10JAN1992	WhiteSale	10JAN1992
563	.	17JAN1992	WhiteSale	17JAN1992
564	.	24JAN1992	WhiteSale	24JAN1992
565	.	31JAN1992	WhiteSale	31JAN1992

次の出力は、WhiteSale 日付キーの DATEKEYDATA OUT=データセットの定義を示しています。

アウトプット 16.3　日付キー定義データセットの出力

The SAS System

Obs	_NAME_	_CLASS_	_KEYNAME_	_STARTDATE_	_PULSE_	_DUR_AFTER_	_RULE_
1	JANUARY	DATEKEY	JANUARY	.	MONTH	0	OR
2	Years1991_1992	DATEKEY	.	01JAN1991	YEAR	1	OR
3	WhiteSale	DATEKEY	JANUARY	.	.	0	AND
4	WhiteSale	DATEKEY	FRIDAY	.	.	0	AND
5	WhiteSale	DATEKEY	YEARS1991_1992	.	.	0	AND

例 2: 他の SAS プロシジャでのユーザー定義の SAS 日付キーキーワードの直接使用

要素:

PROC DATEKEYS ステートメント
- DATEKEYDEF
- DATEKEYKEY
- DATEKEYDATA

システムオプション
- EVENTDS=

その他のプロシジャ
- HPFDIAGNOSE

詳細

プロシジャが EVENT ステートメントをサポートしている場合は、PROC HPFEVENTS を使用せずに、“例 1: 日付キー定義データセットの作成方法” (559 ページ)に示すいずれかのデータセットを使用できます。PROC HPFEVENTS を使用すると、ユーザー定義の日付キーを使用できます。ユーザー定義の日付キーをイベントとして指定すると、イベントが自動的に作成されます。ただし、EVENTDS=システムオプションを使用して、ユーザー定義の日付キー定義データセットが指定されていることが前提となります。この例では、“例 1: 日付キー定義データセットの作成方法” (559 ページ)のデータセットを使用し、PROC HPFDIAGNOSE で EVENT ステートメントを使用します。

PROC HPFDIAGNOSE の出力は、WhiteSale イベントを含むモデルが選択されたことを示しています。このイベントは適していませんでしたが、REQUIRED=YES が指定されていました。REQUIRED=YES は、モデルの診断が失敗しない限り、イベントをモデルに含めることを指定します。

プログラム

```
proc datekeys;
  datekeydef Years1991_1992='01JAN1991'd / pulse=year after=(duration=1);
  datekeykey JANUARY / pulse=month;
  datekeydef WhiteSale=JANUARY FRIDAY Years1991_1992 / rule=and;
  datekeydata out=WhiteSaleDefinitions condense;
run;
options eventds=(WhiteSaleDefinitions);
proc hpfdiagnose data=sashelp.citiday
```

```
  print=all;
 id date interval=day;
 forecast snysecm;
 event WhiteSale / required=yes;
 arimax;
run;
```

プログラムの説明

DATEKEYS プロシジャを開始します。

```
proc datekeys;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。PULSE=は、日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

```
datekeydef Years1991_1992='01JAN1991'd / pulse=year after=(duration=1);
```

日付キーを変更するか、新しい日付キーを作成します。 DATEKEYKEY ステートメントは、ユーザー定義または事前定義された SAS 日付キーを変更するか、別の日付キーから新しい日付キーを作成します。PULSE=は、日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

```
datekeykey JANUARY / pulse=month;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。RULE=AND オプションにより、1991 年と 1992 年の 1 月の金曜日を特定します。

```
datekeydef WhiteSale=JANUARY FRIDAY Years1991_1992 / rule=and;
```

日付キーを出力データセットに書き込みます。 CONDENSE オプションは、出力データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

```
datekeydata out=WhiteSaleDefinitions condense;
```

DATEKEYS プロシジャを実行します。

```
run;
```

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。

```
options eventds=(WhiteSaleDefinitions);
```

HPFDIAGNOSE プロシジャを開始します。 HPFDIAGNOSE プロシジャは、時系列の統計特性を自動的に診断し、適切なモデルを特定します。

```
proc hpfdiagnose data=sashelp.citiday
   print=all;
```

ID ステートメントを指定します。 ID ステートメントは、入力/出力データセットのオブザベーションを識別する数値変数の名前を指定します。ID 変数の値は SAS 日付値です。また、ID ステートメントでは、時系列に関連付ける必要な度数も指定します。INTERVAL=オプションは、入力時間 ID の度数を指定します。

```
id date interval=day;
```

データセットの診断する変数をリストします。 FORECAST ステートメントは、DATA=オプションで指定されたデータセットの診断対象となる変数をリストします。これらの変数は、HPFENGINE プロシジャで予測する従属変数または応答変数です。

```
forecast snysecm;
```

イベントの名前を指定します。 EVENT ステートメント名によってイベントが識別されます。REQUIRED オプションは、モデルの診断が失敗しない限り、イベントをモデルに含めることを指定します。

```
event WhiteSale / required=yes;
```

適切な ARIMAX 指定を見つけます。 ARIMAX モデルは、イベントを含め、傾向変動要因、季節変動要因、および回帰変数をモデル化します。HPFDIAGNOSE プロシジャは、最初に断続性テストを実行します。系列が断続的でない場合、ARIMAX モデルはそのデータに適しています。

```
arimax;
```

HPFDIAGNOSE プロシジャを実行します。

```
run;
```

HTML 出力

次の出力は結果を示しています。

アウトプット 16.4　PROC HPFDIAGNOSE での EVENT ステートメントの使用

The SAS System

The HPFDIAGNOSE Procedure

Variable Information	
Name	SNYSECM
Label	STOCK MKT INDEX:NYSE COMPOSITE, (WSJ)
First	01JAN1988
Last	05FEB1992
Number of Observations Read	1497

Seasonal Dickey-Fuller Unit Root Test(Seasonality=7)				
Type	Rho	Pr < Rho	Tau	Pr < Tau
Zero Mean				
Single Mean				

Dickey-Fuller Unit Root Test Summary				
Variable	Seasonality	Zero Mean	Mean	Trend
SNYSECM	1	NO	NO	NO

Seasonal Dickey-Fuller Unit Root Test Summary				
Variable	Seasonality	Zero Mean	Mean	Trend
SNYSECM	7	NO	NO	

ARIMA Model Specification												
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Model Criterion	Statistic	Status
SNYSECM	NONE	YES	0	0	4	0	0	2	7	RMSE	2.7772	OK

ARIMA Event Selection				
Event Name	Selected	d	D	Status
WHITESALE	REQUIRED	0	0	Not Improved

ARIMA Outlier Selection					
Variable	Type	Obs	Time	Chi-Square	Approx Pr > ChiSq
SNYSECM	LS	677	07NOV1989	334.86	<.0001
	LS	1138	11FEB1991	183.43	<.0001

ARIMA Model Specification After Adjusting for Events and Outliers														
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Outlier	Event	Model Criterion	Statistic	Status
SNYSECM	NONE	YES	0	0	4	0	0	2	7	2	1	RMSE	2.2440	OK

例 3: DATEKEYS プロシジャを使用したカレンダー変数の取得

要素:

PROC DATEKEYS ステートメント
- DATEKEYDEF
- DATEKEYKEY
- DATEKEYDATA
- ID
- DATEKEYCALENDAR

システムオプション
- EVENTDS=

詳細

“概念: DATEKEYS プロシジャ” (526 ページ)に示された日付キー定義について考えてみます。DATEKEYS プロシジャを使用して、データセット MyHolidays で定義された日付キー定義のアクティブ期間を示すカレンダー変数を作成できます。

プログラム

```
options eventds=(nodefaults);

data Year2010;
  do date='01JAN2010'd to '31DEC2010'd;
    output;
  end;
  format date date.;
run;

proc datekeys;

  datekeydef SuperBowl=
            '15JAN1967'd '14JAN1968'd '12JAN1969'd '11JAN1970'd
   '17JAN1971'd '16JAN1972'd '14JAN1973'd '13JAN1974'd '12JAN1975'd
   '18JAN1976'd '09JAN1977'd '15JAN1978'd '21JAN1979'd '20JAN1980'd
   '25JAN1981'd '24JAN1982'd '30JAN1983'd '22JAN1984'd '20JAN1985'd
   '26JAN1986'd '25JAN1987'd '31JAN1988'd '22JAN1989'd '28JAN1990'd
   '27JAN1991'd '26JAN1992'd '31JAN1993'd '30JAN1994'd '29JAN1995'd
   '28JAN1996'd '26JAN1997'd '25JAN1998'd '31JAN1999'd '30JAN2000'd
   '28JAN2001'd '03FEB2002'd '26JAN2003'd '01FEB2004'd '06FEB2005'd
   '05FEB2006'd '04FEB2007'd '03FEB2008'd '01FEB2009'd '07FEB2010'd
   '06FEB2011'd '05FEB2012'd '03FEB2013'd '02FEB2014'd
                / pulse=day;

  datekeydef GoodFriday=Easter / shift=-2 pulse=day;
```

```
  datekeykey EasterMonday=Easter / shift=1 pulse=day;
  datekeydata out=MyHolidays condense;
run;
options eventds=(MyHolidays);
proc datekeys data=Year2010;
  id date interval=day;
  datekeycalendar out=MyHolidaysIn2010;
run;
proc print data=MyHolidaysIn2010(where=(month(date)=2));
run;
proc print data=MyHolidaysIn2010(where=(month(date)=4));
run;
```

プログラムの説明

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。NODEFAULTS オプションは、デフォルトのイベント定義を使用しないことを指定します。event-data-set リストで指定されたイベントだけが使用されます。

```
options eventds=(nodefaults);
```

SAS データセットを作成します。 一連の日付を含む Year2010 データセットを作成します。

```
data Year2010;
  do date='01JAN2010'd to '31DEC2010'd;
    output;
  end;
  format date date.;
run;
```

DATEKEYS プロシジャを開始します。

```
proc datekeys;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを定義します。PULSE=は、日付キーの幅を決定するために DURATION=オプションで使用する間隔を指定します。

```
  datekeydef SuperBowl=
          '15JAN1967'd '14JAN1968'd '12JAN1969'd '11JAN1970'd
 '17JAN1971'd '16JAN1972'd '14JAN1973'd '13JAN1974'd '12JAN1975'd
 '18JAN1976'd '09JAN1977'd '15JAN1978'd '21JAN1979'd '20JAN1980'd
 '25JAN1981'd '24JAN1982'd '30JAN1983'd '22JAN1984'd '20JAN1985'd
 '26JAN1986'd '25JAN1987'd '31JAN1988'd '22JAN1989'd '28JAN1990'd
 '27JAN1991'd '26JAN1992'd '31JAN1993'd '30JAN1994'd '29JAN1995'd
 '28JAN1996'd '26JAN1997'd '25JAN1998'd '31JAN1999'd '30JAN2000'd
 '28JAN2001'd '03FEB2002'd '26JAN2003'd '01FEB2004'd '06FEB2005'd
 '05FEB2006'd '04FEB2007'd '03FEB2008'd '01FEB2009'd '07FEB2010'd
 '06FEB2011'd '05FEB2012'd '03FEB2013'd '02FEB2014'd
```

```
/ pulse=day;
```

日付キーを定義します。 DATEKEYDEF ステートメントは、GoodFriday という名前の日付キーを定義し、Easter という名前の日付キーのタイミング値と等しいことを指定します。PULSE=オプションを指定すると、DURATION=の値がゼロに設定されます。SHIFT=は、タイミング値をシフトするパルス数を指定します。SHIFT=オプションを使用すると、すべてのタイミング値がシフトされます。

```
datekeydef GoodFriday=Easter / shift=-2 pulse=day;
```

日付キーを変更するか、新しい日付キーを作成します。 EasterMonday という日付キー名を指定した DATEKEYKEY ステートメントの値は、Easter という名前の日付キーです。Easter を日付キーのタイミング値として扱うときは、タイミング値の定義だけが使用され、修飾子(SHIFT や PULSE など)は定義には含まれません。Easter を日付キーとして扱うときは、選択した修飾子(SHIFT や PULSE など)が新しい定義に含まれます。PULSE=オプションを指定すると、DURATION=の値がゼロに設定されます。SHIFT=は、タイミング値をシフトするパルス数を指定します。SHIFT=オプションを使用すると、すべてのタイミング値がシフトされます。

```
datekeykey EasterMonday=Easter / shift=1 pulse=day;
```

日付キーを出力データセットに書き込みます。 DATEKEYDATA ステートメントは、MyHolidays という名前の出力データセットに日付キーを書き込みます。CONDENSE オプションは、出力データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

```
datekeydata out=MyHolidays condense;
```

DATEKEYS プロシジャを実行します。

```
run;
```

EVENTSDS=システムオプションを設定します。 EVENTSDS=システムオプションは、イベントを定義するデータセットを指定します。

```
options eventds=(MyHolidays);
```

DATEKEYS プロシジャを開始します。 DATA=オプションは、ID ステートメントで使用される変数を含むデータセットの名前を指定します。

```
proc datekeys data=Year2010;
```

時間 ID 変数を指定します。 ID ステートメントは、データセットのオブザベーションを識別する数値変数の名前を指定します。ID ステートメントを使用する場合は、INTERVAL=オプションを使用する必要があります。この例では、INTERVAL=day です。

```
id date interval=day;
```

日付キーのアクティブ期間を示す変数を含むデータセットを作成します。 DATEKEYCALENDAR OUT=ステートメントは、作成されるデータセットを指定します。このデータセットには、ID ステートメントで指定した情報に基づくカレンダー変数が含まれます。DATEKEYCALENDAR OUT=データセットには、MyHolidays データセットで定義された祝日を識別するカレンダー変数が含まれます。アウトプット 16.93 (572 ページ)には、2010 年 2 月の値が示されます。アウトプット 16.94 (573 ページ)には、2010 年 4 月の値が示されます。DATEKEYCALENDAR OUT=データセット MyHolidaysIn2010 には、入力時刻 ID に対応するアクティブ期間と非アクティブ期間のみが含まれます。入力時間 ID には 1 日の度数があり、2010 年に及ぶため、結果は 2010 年のカレンダー変数になります。

```
datekeycalendar out=MyHolidaysIn2010;
```

```
run;
```

2 月の MyHolidaysIn2010 データセットの出力を書き込みます。 アウトプット 16.93 (572 ページ)に結果が示されます。

```
proc print data=MyHolidaysIn2010(where=(month(date)=2));
```

DATEKEYS プロシジャを実行します。

```
run;
```

4 月の MyHolidaysIn2010 データセットの出力を書き込みます。 アウトプット 16.94 (573 ページ)に結果が示されます。

```
proc print data=MyHolidaysIn2010(where=(month(date)=4));
run;
```

HTML 出力

アウトプット 16.5　2010 年の日単位の時間 ID に基づく 2 月の祝日カレンダー

The SAS System

Obs	date	SuperBowl	GoodFriday	EasterMonday
32	01FEB2010	0	0	0
33	02FEB2010	0	0	0
34	03FEB2010	0	0	0
35	04FEB2010	0	0	0
36	05FEB2010	0	0	0
37	06FEB2010	0	0	0
38	07FEB2010	1	0	0
39	08FEB2010	0	0	0
40	09FEB2010	0	0	0
41	10FEB2010	0	0	0
42	11FEB2010	0	0	0
43	12FEB2010	0	0	0
44	13FEB2010	0	0	0
45	14FEB2010	0	0	0
46	15FEB2010	0	0	0
47	16FEB2010	0	0	0
48	17FEB2010	0	0	0
49	18FEB2010	0	0	0
50	19FEB2010	0	0	0
51	20FEB2010	0	0	0
52	21FEB2010	0	0	0
53	22FEB2010	0	0	0
54	23FEB2010	0	0	0
55	24FEB2010	0	0	0
56	25FEB2010	0	0	0
57	26FEB2010	0	0	0
58	27FEB2010	0	0	0
59	28FEB2010	0	0	0

アウトプット 16.6　2010 年の日単位の時間 ID に基づく 4 月の祝日カレンダー

The SAS System

Obs	date	SuperBowl	GoodFriday	EasterMonday
91	01APR2010	0	0	0
92	02APR2010	0	1	0
93	03APR2010	0	0	0
94	04APR2010	0	0	0
95	05APR2010	0	0	1
96	06APR2010	0	0	0
97	07APR2010	0	0	0
98	08APR2010	0	0	0
99	09APR2010	0	0	0
100	10APR2010	0	0	0
101	11APR2010	0	0	0
102	12APR2010	0	0	0
103	13APR2010	0	0	0
104	14APR2010	0	0	0
105	15APR2010	0	0	0
106	16APR2010	0	0	0
107	17APR2010	0	0	0
108	18APR2010	0	0	0
109	19APR2010	0	0	0
110	20APR2010	0	0	0
111	21APR2010	0	0	0
112	22APR2010	0	0	0
113	23APR2010	0	0	0
114	24APR2010	0	0	0
115	25APR2010	0	0	0
116	26APR2010	0	0	0
117	27APR2010	0	0	0
118	28APR2010	0	0	0
119	29APR2010	0	0	0
120	30APR2010	0	0	0

例 4: DATEKEYDSOPT ステートメントを使用したデータセットのフィルタリング

要素:

PROC DATEKEYS ステートメント
- DATEKEYKEY
- DATEKEYDEF
- DATEKEYDATA
- ID
- DATEKEYDSOPT
- DATEKEYCALENDAR

システムオプション
- EVENTDS=

詳細

DATEKEYS プロシジャを使用して、入力データセットをフィルタリングし、指定したロケールに関連付けられた日付キーデータだけを含むデータセットを作成できます。DATEKEYDSOPT ステートメントは、DATEKEYDATA、DATEKEYPERIODS、および DATEKEYCALENDAR ステートメントで指定したデータセットに適用されます。

この例では、Canada、CanadaObserved、Boxing の各祝日の英語/カナダの日付キー定義を作成します。また、USINDEPENDENCE の英語/米国の日付キー定義も作成します。NewYearsEve の定義では、ロケールは指定しません。

プログラム

```
options eventds=(nodefaults);

data December2010;
  do date='01DEC2010'd to '31DEC2010'd;
    output;
  end;
  format date date.;
run;

proc datekeys;

  datekeykey Canada / locale=en_CA;
  datekeykey CanadaObserved / locale=en_CA;
  datekeykey Boxing / locale='en_CA';
  datekeykey USINDEPENDENCE / locale=en_US;

  datekeydef NewYearsEve=NEWYEAR / shift=-1 pulse=day;

  datekeydata out=MyHolidays condense;
```

```
run;
options eventds=(MyHolidays);
proc datekeys data=December2010;
  id date interval=day;
 datekeydsopt locale=en_CA;
  datekeycalendar out=AllCAHolidaysInDecember2010;
run;
proc datekeys data=December2010;
  id date interval=day;
  datekeydsopt locale=(ONLY)en_CA;
  datekeycalendar out=OnlyCAHolidaysInDecember2010;
run;
proc print data=MyHolidays;
run;
proc print data=AllCAHolidaysInDecember2010;
run;
proc print data=OnlyCAHolidaysInDecember2010;
run;
```

プログラムの説明

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。NODEFAULTS オプションは、デフォルトのイベント定義を使用しないことを指定します。event-data-set リストで指定されたイベントだけが使用されます。

```
options eventds=(nodefaults);
```

SAS データセットを作成します。 一連の日付を含む December2010 データセットを作成します。

```
data December2010;
  do date='01DEC2010'd to '31DEC2010'd;
    output;
  end;
  format date date.;
run;
```

DATEKEYS プロシジャを開始します。

```
proc datekeys;
```

新しい日付キーを作成します。 DATEKEYKEY ステートメントは、新しい日付キーを作成します。LOCALE=オプションは、日付キーに関連付けるロケールを指定します。ロケールは、POSIX ロケール値にする必要があります。

```
  datekeykey Canada / locale=en_CA;
  datekeykey CanadaObserved / locale=en_CA;
  datekeykey Boxing / locale='en_CA';
```

```
datekeykey USINDEPENDENCE / locale=en_US;
```

日付キーを定義します。 DATEKEYDEF ステートメントは日付キーを識別します。SHIFT=オプションは、タイミング値 6 をシフトするパルス数を指定します。このオプションを使用すると、リストのすべてのタイミング値(日付キーワードによって生成されるタイミング値を含む)がシフトされます。PULSE=オプションは使用する間隔を指定します。

```
datekeydef NewYearsEve=NEWYEAR / shift=-1 pulse=day;
```

日付キーを出力データセットに書き込みます。 DATEKEYDATA ステートメントは、出力データセットを作成します。CONDENSE オプションは、出力データセットを簡略化することを指定します。デフォルト値だけを含む変数はデータセットから除外されます。

```
datekeydata out=MyHolidays condense;
```

DATEKEYS プロシジャを実行します。

```
run;
```

EVENTDS=システムオプションを設定します。 EVENTDS=システムオプションは、イベントを定義するデータセットを指定します。

```
options eventds=(MyHolidays);
```

DATEKEYS プロシジャを開始します。 DATEKEYS プロシジャは、時間計算に関連付けられた日付キーを作成します。DATA=オプションには、ID ステートメントで使用される変数が含まれます。

```
proc datekeys data=December2010;
```

ID ステートメントを指定します。 ID ステートメントは、入力/出力データセットのオブザベーションを識別する数値変数の名前を指定します。ID 変数の値は SAS 日付値です。また、ID ステートメントでは、時系列に関連付ける必要な度数も指定します。

```
id date interval=day;
```

特定のロケールのデータセットを作成します。 DATEKEYDSOPT ステートメントは、データセットの処理を指定したロケールに限定します。このステートメントを使用すると、ロケール値 en_CA が、指定された日付キー定義とロケールが指定されていない日付キー定義を使用して、AllCAHolidaysInDecember2010 データセットが作成されます。

```
datekeydsopt locale=en_CA;
```

日付キーのアクティブ期間を示す変数を含むデータセットを作成します。 DATEKEYCALENDAR ステートメントは、日付キーのアクティブ期間を示す変数を書き込みます。OUT=オプションは、作成されるデータセットを表します。このデータセットには、ID ステートメントで指定した情報に基づくカレンダー変数が含まれます。

```
datekeycalendar out=AllCAHolidaysInDecember2010;
```

DATEKEYS プロシジャを実行します。

```
run;
```

DATEKEYS プロシジャを開始します。 DATA=オプションには、ID ステートメントで使用される変数が含まれます。

```
proc datekeys data=December2010;
```

ID ステートメントを指定します。 ID ステートメントは、入力/出力データセットのオブザベーションを識別する数値変数の名前を指定します。ID 変数の値は SAS 日付値です。また、ID ステートメントでは、時系列に関連付ける必要な度数も指定します。

```
id date interval=day;
```

特定のロケールのデータセットを作成します。 DATEKEYDSOPT ステートメントは、指定したロケールに関連付けられた日付キーデータだけを含むデータセットを作成します。LOCALE=(ONLY)en_CA を指定すると、入力データセットと出力データセットの両方で、指定したロケールだけが処理されます。ロケールは、POSIX ロケール値にする必要があります。

```
datekeydsopt locale=(ONLY)en_CA;
```

日付キーのアクティブ期間を示す変数を含むデータセットを作成します。 DATEKEYCALENDAR ステートメントは、日付キーのアクティブ期間を示す変数を書き込みます。OUT=オプションは、作成されるデータセットを指定します。このデータセットには、ID ステートメントで指定した情報に基づくカレンダー変数が含まれます。

```
datekeycalendar out=OnlyCAHolidaysInDecember2010;
```

DATEKEYS プロシジャを実行します。

```
run;
```

MyHolidays データセットを書き込みます。 結果については、アウトプット 16.95 (578 ページ)を参照してください。

```
proc print data=MyHolidays;
run;
```

AllCAHolidaysInDecember2010 データセットを書き込みます。 結果については、アウトプット 16.96 (579 ページ)を参照してください。

```
proc print data=AllCAHolidaysInDecember2010;
run;
```

OnlyCAHolidaysInDecember2010 データセットを書き込みます。 結果については、アウトプット 16.97 (580 ページ)を参照してください。

```
proc print data=OnlyCAHolidaysInDecember2010;
run;
```

HTML 出力

アウトプット 16.7　MyHolidays で指定されたすべての祝日定義

The SAS System

Obs	_LOCALE_	_NAME_	_CLASS_	_KEYNAME_	_PULSE_	_SHIFT_
1	EN_CA	Canada	DATEKEY	CANADA	DAY	0
2	EN_CA	CanadaObserved	DATEKEY	CANADAOBSERVED	DAY	0
3	en_CA	Boxing	DATEKEY	BOXING	DAY	0
4	EN_US	USINDEPENDENCE	DATEKEY	USINDEPENDENCE	DAY	0
5	.	NewYearsEve	DATEKEY	NEWYEAR	DAY	-1

アウトプット 16.8　2010 年 12 月のカナダのすべての祝日を示す祝日カレンダー

The SAS System

Obs	date	Canada	CanadaObserved	Boxing	NewYearsEve
1	01DEC2010	0	0	0	0
2	02DEC2010	0	0	0	0
3	03DEC2010	0	0	0	0
4	04DEC2010	0	0	0	0
5	05DEC2010	0	0	0	0
6	06DEC2010	0	0	0	0
7	07DEC2010	0	0	0	0
8	08DEC2010	0	0	0	0
9	09DEC2010	0	0	0	0
10	10DEC2010	0	0	0	0
11	11DEC2010	0	0	0	0
12	12DEC2010	0	0	0	0
13	13DEC2010	0	0	0	0
14	14DEC2010	0	0	0	0
15	15DEC2010	0	0	0	0
16	16DEC2010	0	0	0	0
17	17DEC2010	0	0	0	0
18	18DEC2010	0	0	0	0
19	19DEC2010	0	0	0	0
20	20DEC2010	0	0	0	0
21	21DEC2010	0	0	0	0
22	22DEC2010	0	0	0	0
23	23DEC2010	0	0	0	0
24	24DEC2010	0	0	0	0
25	25DEC2010	0	0	0	0
26	26DEC2010	0	0	1	0
27	27DEC2010	0	0	0	0
28	28DEC2010	0	0	0	0
29	29DEC2010	0	0	0	0
30	30DEC2010	0	0	0	0
31	31DEC2010	0	0	0	1

アウトプット 16.9　2010 年 12 月のカナダの祝日のみを示す祝日カレンダー

The SAS System

Obs	date	Canada	CanadaObserved	Boxing
1	01DEC2010	0	0	0
2	02DEC2010	0	0	0
3	03DEC2010	0	0	0
4	04DEC2010	0	0	0
5	05DEC2010	0	0	0
6	06DEC2010	0	0	0
7	07DEC2010	0	0	0
8	08DEC2010	0	0	0
9	09DEC2010	0	0	0
10	10DEC2010	0	0	0
11	11DEC2010	0	0	0
12	12DEC2010	0	0	0
13	13DEC2010	0	0	0
14	14DEC2010	0	0	0
15	15DEC2010	0	0	0
16	16DEC2010	0	0	0
17	17DEC2010	0	0	0
18	18DEC2010	0	0	0
19	19DEC2010	0	0	0
20	20DEC2010	0	0	0
21	21DEC2010	0	0	0
22	22DEC2010	0	0	0
23	23DEC2010	0	0	0
24	24DEC2010	0	0	0
25	25DEC2010	0	0	0
26	26DEC2010	0	0	1
27	27DEC2010	0	0	0
28	28DEC2010	0	0	0
29	29DEC2010	0	0	0
30	30DEC2010	0	0	0
31	31DEC2010	0	0	0

参考文献

Montes, M. J. “Calculation of the Ecclesiastical Calendar.” 2001. URI: at http://www.smart.net/~mmontes/ec-cal.html.

Montes, M. J. “Algorithm for Calculating the Date of Easter in the Gregorian Calendar.” 2001. URI: at http://www.smart.net/~mmontes/nature1876.html.

17

DELETE プロシジャ

概要: DELETE プロシジャ

DELETE プロシジャの機能

DELETE プロシジャは、SAS ライブラリのメンバーを削除します。PROC DELETE を使用して次の作業を実行します。

- 永続 SAS ファイルまたは一時 SAS ファイルのいずれかを削除する

- ロードされた CAS テーブルを caslib から削除します。caslib で指定されたデータソースから元のファイルは削除されません。
- 次のように名前が同じで接尾辞が数値のデータセットのリストを削除する

  ```
  proc delete data=x1-x3;
  run;
  ```

- メンバーの種類を削除する。
- GENNUM=オプションを使用して、世代データセットを削除する。
- GENNUM=ALL オプションと ENCRYPTKEY=オプションを使用する場合に、AES の暗号化されたデータセットを削除する。

概念: DELETE プロシジャ

概念: DELETE プロシジャ

DATASETS プロシジャにおいて DELETE ステートメントではなく PROC DELETE を使用する利点の 1 つは、SAS データセットの削除にインメモリディレクトリを使用しないことです。その結果、DELETE プロシジャの方が処理速度が早くなります。

構文: DELETE プロシジャ

該当要素: 計算サーバー, CAS

ヒント: DATASETS プロシジャの DELETE ステートメントと同様の機能を実行できます。

PROC DELETE DATA=*SAS-file(s)* <*options*>;

ステートメント	タスク	例
“PROC DELETE ステートメント”	SAS ライブラリから SAS ファイルを削除する	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5

PROC DELETE ステートメント

SAS ライブラリから SAS ファイルを削除します。

構文

PROC DELETE <LIBRARY=*libref*> DATA=*SAS-file(s)*
(<GENNUM=ALL | HIST | REVERT | *integer*>
<MEMTYPE=*member-type*>
<ENCRYPTKEY=*key-value*>
<ALTER=*alter-password*>);

オプション引数の要約

ALTER=*alter-password*
変更保護されている SAS ファイルに Alter パスワードを付与します。

ENCRYPTKEY=*key-value*
ENCRYPTKEY=オプションは、AES 暗号化キー値により保護されているデータセットのロックを解除するものです。

GENNUM=ALL | HIST | REVERT | *integer*
世代データセットの処理を制限します。

LIBRARY=*libref*
削除するメンバーを含む SAS ライブラリを指定します。

MEMTYPE=(*member-type(s)*)
削除を特定の種類の SAS ファイルに制限します。

必須引数

DATA= *SAS-file(s)*
削除する SAS ファイルを 1 つ以上指定します。

注: 番号範囲リストを使用することもできます。詳細については、“データセット名リスト” (*SAS プログラマガイド: エッセンシャル*)を参照してください。コロンリストは使用できません。

ヒント　ライブラリ内のすべてのファイルを削除する場合は、PROC DATASETS KILL オプションを使用します。PROC DATASETS LIB=*library name* KILL を使用して、カタログを含むすべてのファイルを削除します。詳細については、“KILL” (379 ページ).を参照してください。

オプション引数

ALTER=*alter-password*
変更保護されている SAS ファイルに Alter パスワードを付与します。

参照項目: “DATASETS プロシジャでパスワードを使用する” (462 ページ)

ENCRYPTKEY=*key-value*
ENCRYPTKEY=オプションは、AES 暗号化キー値により保護されているデータセットのロックを解除するものです。ENCRYPTKEY=オプションは、データセットを開く必要がある場合にのみ必要です。

参照項目: “例 7: ENCRYPTKEY=オプションの使用” (591 ページ)

GENNUM=ALL | HIST | REVERT | *integer*
世代データセットの処理を制限します。各 SAS ファイルの名前の後にある丸かっこ内のオプションを使用します。有効な値は次のとおりです。

ALL	世代グループの基本バージョンとすべての 履歴バージョンを参照します。
HIST	世代グループの基本バージョンを除く、すべての履歴バージョンを参照します。
REVERT | 0	基本バージョンを削除し、存在する場合は最新履歴バージョンを基本バージョンに変更します。
integer	世代グループ内の特定のバージョンを参照する番号です。正の数を指定すると、データセット名に追加される特定の世代番号の絶対参照になります(つまり、**gennum=2** は MYDATA#002 を指定することになります)。

参照項目: “世代データセットの処理制限” (463 ページ)

例: “例 1: 複数の SAS データセットの削除” (588 ページ)

“例 2: 基本バージョンとすべての 履歴バージョンの削除” (589 ページ)

“例 3: 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更する” (589 ページ)

“例 4: 絶対数でバージョンを削除する” (590 ページ)

“例 5: すべての履歴バージョンの削除および基本バージョンの保留” (590 ページ)

LIBRARY=*libref*
削除するメンバーを含む SAS ライブラリを指定します。

別名: LIB=

MEMTYPE=(*member-type(s)*)
削除を 1 つ以上のメンバーの種類に制限します。たとえば、MyFile という名前のカタログとデータセットが MyLib ライブラリにあり、カタログのみを削除する場合は、MEMTYPE=オプションを使用します。

```
proc delete lib=MyLib data=MyFile (memtype=catalog);
run;
```

ACCESS
アクセスディスクリプターファイル (SAS/ACCESS ソフトウェアによって作成)

CATALOG
SAS カタログ

DATA
SAS データファイル

FDB
財務データベース

MDDB
多次元データベース

PROGRAM
保存されたコンパイル済み SAS プログラム

VIEW
SAS ビュー

別名 MTYPE=

MT=

デフォルト DATA

例 “例 6: MEMTYPE=オプションの使用” (591 ページ)

詳細

世代グループの操作

世代グループで作業しているとき、PROC DELETE を使用すれば次のバージョンを削除できます。

- 基本バージョンとすべての 履歴バージョンを削除する。“例 2: 基本バージョンとすべての 履歴バージョンの削除” (589 ページ)を参照してください。
- 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更する。“例 3: 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更する” (589 ページ)を参照してください。
- 絶対バージョンを削除する。“例 4: 絶対数でバージョンを削除する” (590 ページ)を参照してください。
- すべての履歴バージョンを削除し、基本バージョンは残す。“例 5: すべての 履歴バージョンの削除および基本バージョンの保留” (590 ページ)を参照してください。

特定のメンバーの種類の削除

MEMTYPE=オプションを使用して、削除するメンバーの種類を指定できます。MEMTYPE=オプションのデフォルトは DATA です。詳細については、“MEMTYPE=(*member-type(s)*)” (586 ページ).を参照してください。

一貫性制約の操作

DELETE プロシジャを使用しても、外部キー一貫性制約を持つ、または外部キー参照を含むプライマリキーを持つデータファイルは削除できません。外部キーを持つデータファイルについては、データファイルを削除する前に外部キーを削除する必要があります。外部キー参照を含むプライマリキーを持つデータファイルについては、データファイルを削除する前にそのプライマリキーを参照する外部キーを削除する必要があります。

例: DELETE プロシジャ

例 1: 複数の SAS データセットの削除

要素:

PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例は、次のタスクを示しています。

- ライブラリからデータセットを削除する
- 各データセットのすべての履歴バージョンを削除する

プログラム

A、B および C という名前の SAS データセットを MyLib という名前の SAS ライブラリから削除します。 GENNUM=オプションは、各データセットの履歴バージョンをすべて削除します。

```
proc delete data=MyLib.A MyLib.B MyLib.C (gennum=all);
run;
```

例 2: 基本バージョンとすべての履歴バージョンの削除

要素:

PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例は、次のタスクを示しています。

- ライブラリから 1 つのデータセットを削除する
- データセットのすべての履歴バージョンを削除する

プログラム

MyLib.A という名前のデータセットとすべての履歴バージョンを削除します。

```
proc delete data=MyLib.A (gennum=all);
run;
```

例 3: 基本バージョンを削除し、最新履歴バージョンを基本バージョンに名前変更する

要素:

PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例は、次のタスクを示しています。

- ライブラリから 1 つのデータセットを削除する
- 最新履歴バージョンの名前を変更する

次のステートメントは、MyLib.A という名前のデータセットを削除し、その最新履歴バージョンの名前を変更します。

プログラム

MyLib.A という名前のデータセットを削除し、最新履歴バージョンの名前を変更します。

```
proc delete data=MyLib.A(gennum=revert1);
run;
```

例 4: 絶対数でバージョンを削除する

要素: PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例では、データセットの最初の履歴バージョンを削除します。

プログラム

GENNUM=オプションで、MyLib.A とうい名前のデータセットの最初の履歴バージョンを削除します。 GENNUM=*integer* を使用して削除する履歴バージョンを選択します。

```
proc delete data=MyLib.A(gennum=1);
run;
```

例 5: すべての履歴バージョンの削除および基本バージョンの保留

要素: PROC DELETE ステートメントオプション
DATA=
GENNUM=

詳細

この例では、データセットの基本バージョンを除くすべての履歴バージョンを削除します。

プログラム

GENNUM=HIST オプションを使用してすべての履歴バージョンを削除し、MyLib.A データセットの基本バージョンは保留します。

```
proc delete data=MyLib.A(gennum=hist);
run;
```

例 6: MEMTYPE=オプションの使用

要素:

PROC DELETE ステートメントオプション
DATA=
MEMTYPE=

詳細

この例では、特定の SAS ライブラリの CATALOG ファイルを削除します。

プログラム

MEMTYPE=オプションでは、SAS ライブラリで削除するファイルのタイプを指定します。MyLib ライブラリ内に MyFile という名前のその他のメンバーの種類がある場合、それらは削除されません。

```
proc delete lib=MyLib data=MyFile (memtype=catalog);
run;
```

例 7: ENCRYPTKEY=オプションの使用

要素:

PROC DELETE ステートメントオプション
DATA=

ENCRYPTKEY=
GENNUM=

詳細

この例は、次のタスクを示しています。

- AES 暗号化データセットのロックを解除する
- すべての履歴バージョンと MyLib.A という名前の基本 AES データセットを削除する

プログラム

MyLib.A という名前の基本 AES データセットとすべての履歴バージョンを削除します。 ENCRYPTKEY=オプションは、データセットが AES 暗号化されている場合に使用する必要があります。

```
proc delete data=MyLib.A (gennum=ALL encryptkey=key-value);
run;
```

例 8: ALTER=オプションの使用

要素:

PROC DELETE ステートメントオプション
ALTER=
DATA=

詳細

この例では、パスワードで保護されているデータセットを削除します。

プログラム

MyLib.A とうい名前のパスワードで保護されているデータセットを削除します。 データセットがパスワードで保護されている場合は、パスワードを指定する必要があります。

```
proc delete data=MyLib.A (alter=alter-password);
run;
```

例 9: DATA=リスト機能の使用

要素: PROC DELETE ステートメントオプション
DATA=

プログラム

X1、X2、X3、X4、X5 という名前の複数のデータセットを削除します。 このリスト機能を使用するには、データセットの名前が同じで、その末尾が数値の接尾辞であることが必要です。LIBRARY=オプションが指定されていない場合、データセットは Work ライブラリから削除されます。

```
proc delete data=X1-X5;
run;
```

例 10: LIBRARY=オプションの使用

要素: PROC DELETE ステートメントオプション
DATA=
LIB=

詳細

次のステートメントでは、特定の SAS ライブラリからデータセットを削除します。

プログラム

指定された MyLib という名前の SAS ライブラリにある A データセットを削除します。 LIBRARY=オプションの別名は LIB=です。

```
proc delete lib=MyLib data=A;
run;
```

例 11: LIBRARY=オプションおよびリスト機能の使用

要素: PROC DELETE ステートメントオプション
DATA=
LIB=

詳細

この例では、指定された MyLib という名前の SAS ライブラリのデータセット X1、X2、X3、X4、X5 を削除します。LIBRARY=オプションの別名は LIB=です。

注: これらデータセットの末尾は数値の接尾辞とします。

プログラム

指定された MyLib という名前の SAS ライブラリからデータセット X1、X2、X3、X4、X5 を削除します。 このリスト機能を使用するには、すべてのデータセットの名前が同じで、その末尾が数値の接尾辞であることが必要です。LIBRARY=オプションの別名は LIB=です。

```
proc delete lib=MyLib data=X1-X5;
run;
```

18

DS2 プロシジャ

概要: DS2 プロシジャ

DS2 プロシジャの動作について

DS2 プロシジャを使用すると、SAS DS2 言語プログラムをサブミットできます。DS2 プロシジャを使用すると、DS2 プログラムを SAS およびサードパーティのデータソースにサブミットできます。または、DS2 プログラムを CAS サーバーにサブミットできます。

DS2 は、上級データ操作に適した SAS プログラミング言語です。DS2 は、コア機能を SAS DATA ステップと共有します。DS2 は、変数スコープ、ANSI SQL データ型、定義済みパッケージ、およびユーザー定義のメソッドとパッケージを追加することで、DATA ステップを強化します。DS2 SET ステートメントは、埋め込み型 FedSQL 構文を受け入れ、実行時生成型クエリは DS2 とサポートされるデータベース間で対話式にデータを交換できます。これによって、入力テーブルの SQL 前処理を行えるようになるため、2 つの言語の機能を効果的に組み合わせることができます。DS2 言語の詳細については、*SAS DS2 言語リファレンス*と *SAS DS2 プログラマガイド*を参照してください。

注: DS2 プロシジャでは、CAS への接続に CAS エンジンを使用しません。DS2 プログラムを CAS サーバーにサブミットするには、CASLIB ステートメントを使用して割り当てられた caslib を指定する必要があります。詳細については、“CAS サーバーへの接続” (609 ページ)を参照してください。

DS2 言語は、いくつかの例外を除いて、SAS およびサードパーティのデータソースで提供されるのと同じ機能を CAS サーバーでサポートします。

概念: DS2 プロシジャ

DS2 言語の利点

DS2 プログラムは次のようなアプリケーション用に作成されています。

- 言語で使用可能な追加のデータ型を使用した結果の精度が必要になる

- 新しい表現を使用することで利益を得たり、DS2 構文で使用可能なメソッドやパッケージを作成したりする
- DS2 プログラムから SAS FedSQL 言語を実行する必要がある
- SAS Viya の内部と外部(たとえば、SAS Federation サーバー上)で実行できる
- スレッド処理を活用する。

データソースサポート

DS2 プロシジャでは、次のデータソースがサポートされています。

- Amazon Redshift
- CAS サーバー
- Linux 動作環境の DB2
- Google BigQuery
- Greenplum
- Hadoop (Hive)
- Impala
- JDBC 準拠のデータベース
- Microsoft SQL Server
- MongoDB
- MySQL
- Netezza
- ODBC 準拠のデータベース
- Oracle
- PostgreSQL
- Salesforce
- SAP
- SAP HANA
- SAP IQ
- SAS データセット
- SAS Scalable Performance Data Engine データセット
- SAS Scalable Performance Data Server テーブル
- SingleStore
- Snowflake
- Spark
- Teradata

- Vertica
- Yellowbrick

PROC DS2 を使用して DBMS データソースにアクセスし、操作するには、適切な SAS/ACCESS ソフトウェアを構成する必要があります。PROC DS2 で処理するために、SAS およびサードパーティのデータを CAS サーバーにロードするには、適切な SAS データコネクタソフトウェアを構成する必要があります。

構文: DS2 プロシジャ

PROC DS2<*connection-options*><*processing-options*>;
...DS2 language statements
RUN;
RUN CANCEL;
QUIT;

ステートメント	タスク	例
“PROC DS2 ステートメント”	後続の入力が DS2 言語ステートメントであることを指定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
“RUN CANCEL ステートメント”	前の DS2 言語ステートメントをキャンセルします。	Ex. 3

PROC DS2 ステートメント

後続の入力が DS2 言語ステートメントであることを指定します。

制限事項: HTTP アクセス方式がロックダウン状態の場合、DS2 組み込みパッケージである HTTP は使用できません。サーバー管理者は、このアクセス方式がロックダウン状態でも使用できるように、同方式を再有効化できます。詳細については、“LOCKDOWN” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

要件 PROC DS2 ステートメントの後に DS2 言語ステートメントを続けます。DS2 言語ステートメントについての情報は、*SAS DS2 言語リファレンス*を参照してください。また、*SAS DS2 プログラマガイド*も参照してください。

操作: DS2 プロシジャでは、RUN ステートメントから DS2 ステートメントをサブミットする必要があります。つまり、SAS は RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。複数の RUN グループを指定することができます。プログラムの終了を知らせるには、QUIT の後にセミコロンを指定します。“RUN CANCEL ステートメント” (607 ページ)を参照してください。

デフォルトでは、DS2 プロシジャは CHAR(*n*)列と DOUBLE 列に存在しない値を SAS 欠損値として処理します。存在しない値が ANSI SQL null 値として処理されるように要求するには、MODE=ANSI を指定します。

PROC DS2 を使用してテーブルを作成する場合、デフォルトでは既存のテーブルを上書きできないことに注意してください。置換出力テーブルが作成される前に出力テーブルが削除されるように指定するには、OVERWRITE=YES テーブルオプションを使用します。詳細については、"OVERWRITE= テーブルオプション" (*SAS DS2 言語リファレンス*)を参照してください。

サポート: スレッド処理

注: 特に明記されていない限り、すべてのプロシジャオプションは、SAS Compute Server で実行される DS2 プログラムと、CAS で実行される DS2 プログラムでサポートされています。

例:
"例 1: DS2 コードの導入" (619 ページ)
"例 2: SAS データセットを作成する" (621 ページ)
"例 3: ラインプロンプトモードでの現在のステップの終了" (623 ページ)
"例 4: 値に基づいてテーブルにデータを送る" (624 ページ)
"例 5: CAS での DS2 プログラムの実行" (626 ページ)

構文

PROC DS2<*options*>;

オプション引数の要約

Connection

LIBS=*libref* | (*libref1libref2...librefn*)
データソース接続を、指定された 1 つまたは複数のライブラリ参照とデフォルトのライブラリに制限します。

SESSREF=*session-reference*
CAS セッションで DS2 ステートメントを実行することを指定します。CAS セッションは、そのセッション名によって識別されます。

SESSUUID="*session-uuid*"
CAS セッションで DS2 ステートメントを実行することを指定します。CAS セッションは、その汎用一意識別子(UUID)によって識別されます。

General Processing

ANSIMODE
固定長文字列と DOUBLE 列に存在しない値が ANSI SQL の null 値として処理されるように指定します。

ERRORSTOP

NOERRORSTOP
プロシジャにエラーが発生した場合、その実行を停止するかどうかを指定します。

LABEL

NOLABEL
列見出しとして列ラベルまたは列名を使用するかどうかを指定します。

LOGICALEXPR=STANDARD | OPTIMIZED
論理 AND および OR 式の評価方法を指定します。

MEMSIZE=n | nM | nG
割り当てられたメモリが他の PROC DS2 処理をサポートするために使用できるように、基本クエリ(SELECT ステートメントなど)に使用されるメモリ量を制限するように指定します。

MODE=ANSI | SAS
固定長文字列と DOUBLE 列に存在しない値が SAS 欠損値として処理されるのか、あるいは SAS ANSI NULL 値として処理されるのかを指定します。

NUMBER
Row という名前の列(行が取得される際のデータの行(オブザベーション)番号)を含めるように指定します。

STIMER
経過時間統計などのシステムパフォーマンス統計のサブセットが SAS ログに書き込まれるように指定します。

TREEALGORITHM=ITERATIVE | RECURSIVE
DS2 コンパイラが構文ツリーの構築に使用するデフォルトのアルゴリズムを変更します。

XCODE=ERROR | WARNING | IGNORE
NLS トランスコーディングが失敗した場合の SAS セッションの動作を制御します。

Message Control

MSGLIMIT=n | MIN | MAX
DS2 プログラムの実行中に SAS ログに書き込むことができるエラー、警告、および注意メッセージの最大数を指定します。

MSGORDER=STANDARD | TEMPORAL
DS2 が、エラー、警告、および注意メッセージを生成時に SAS ログに書き込むか、または DS2 プログラムの完了後に書き込むかを指定します。

REPORTLINE

NOREPORTLINE
DS2 プログラム実行メッセージでログ行番号を報告するかどうかを指定します。

SCOND=WARNING | NONE | NOTE | ERROR
DS2 変数宣言厳密モードの SAS ログで PROC DS2 が表示するメッセージのレベルを指定します。このモードでは、すべての変数を DS2 プログラム内で宣言する必要があります。

オプション引数

ANSIMODE
固定長文字列と DOUBLE 列に存在しない値が ANSI SQL の null 値として処理されるように指定します。 他のすべてのデータ型は、常に ANSI NULL セマンティクスを使用します。

デフォルト　存在しない値は SAS 欠損値として処理されます。これは、SAS プログラミングランタイム環境と一致しています。

重要 MODE=プロシジャオプションは ANSIMODE オプションよりも優先されます。

ERRORSTOP | NOERRORSTOP
プロシジャにエラーが発生した場合、その実行を停止するかどうかを指定します。 ERRORSTOP は、エラーが発生した後に、ステートメントの実行は停止しつつ、構文のチェックを続行するようにプロシジャに指示します。

NOERRORSTOP は、プロシジャにステートメントの実行を命令し、エラー発生後も構文のチェックを継続するように命令します。

デフォルト 対話型 SAS セッションでの NOERRORSTOP、バッチまたは非対話型セッションでの ERRORSTOP

制限事項 ERRORSTOP は、SASa バッチまたは非対話型実行モードで実行している場合にのみ有効です。

ヒント NOERRORSTOP は、エラー発生後もバッチジョブにプロシジャステートメントの実行を継続させる場合に役立ちます。

LABEL | NOLABEL
列見出しとして列ラベルまたは列名を使用するかどうかを指定します。 列にラベルがない場合、プロシジャでは列名が列見出しとして使用されます。

デフォルト LABEL

操作 列の別名を使用すると、列の別名が列見出しとしてラベルまたは列名に優先されます。

LIBS=*libref* | (*libref1libref2...librefn*)
データソース接続を、指定された 1 つまたは複数のライブラリ参照とデフォルトのライブラリに制限します。 LIBS=オプションは、多数の LIBNAME ステートメントが SAS セッションで定義されている場合に、DS2 プログラムのスコープをプログラムの適用先の LIBNAME ステートメントのみに限定する場合に便利です。このオプションを使用すると、Netezza などのネイティブカタログをサポートするデータソースに接続するときに、重複カタログエラーを回避することもできます。

複数のライブラリ参照名を指定する場合は、ライブラリ参照名をかっこで囲みます。各ライブラリ参照名はスペースで区切ります。

別名 LIBNAMES=

制限事項 LIBS=は、CAS サーバーではサポートされません。LIBS=と SESSREF= (または SESSUID=)の両方がプロシジャステートメントで指定されている場合、LIBS=は無視されます。

操作 LIBS=を指定しても、LIBS=に 1 つのライブラリ参照名のみが指定されている場合でも、DS2 プログラムでライブラリ参照名を指定する必要性がなくなるわけではありません。テーブル名にライブラリ参照名を指定しない場合は、デフォルトのライブラリが使用されます。詳細については、“SAS および DBMS データソースへの接続” (608 ページ)を参照してください。

プロシジャステートメントで LIBS=を複数回指定した場合、プロシジャオプションの最後のインスタンスで指定されたライブラリ参照名のみが適用されます。

ヒント LIBS=がライブラリ割り当てに及ぼす影響を知りたい場合は、LIBS=を指定して PROC DS2 要求を実行する前に MSGLEVEL=i システムオプションを設定します。このオプションで、各 LIBNAME ステートメントに対して Include および Ignore メッセージが生成されます。

例 次の PROC DS2 プロシジャステートメントは、SAS セッションで割り当てられた 4 つのライブラリ参照のうち 2 つとデフォルトのライブラリを使用するデータソース接続を作成することを指定します。

```
proc ds2 libs=(mylib3 mylib4);
```

LOGICALEXPR=STANDARD | OPTIMIZED
論理 AND および OR 式の評価方法を指定します。

STANDARD 結果を返す前に、DS2 が論理 AND または OR 式の両方のオペランドを評価できることを指定します。

OPTIMIZED DS2 が、可能であれば最初のオペランドに基づく結果を返すことによって、論理 AND および OR 式のオペランドの評価を最適化することを指定します。

デフォルト STANDARD

操作 論理 AND および OR 式の評価も、DS2_OPTIONS ステートメントで制御できます。DS2_OPTIONS ステートメントで LOGICALEXPR 値が設定されている場合、その値は、それが指定されている DATA、PACKAGE、または THREAD ブロックのプロシジャオプションよりも優先されます。詳細については、DS2_OPTIONS ステートメントを参照してください。

ヒント 2 番目のオペランドの評価をバイパスすると、無効になる計算を回避できます。

参照項目: “DS2 での短絡評価” (*SAS DS2 プログラマガイド*)

MEMSIZE=n | nM | nG
割り当てられたメモリが他の PROC DS2 処理をサポートするために使用できるように、基本クエリ(SELECT ステートメントなど)に使用されるメモリ量を制限するように指定します。 メモリ制限は 1 (バイト)、1,048,576 (MB)、または 1,073,741,824 (GB)の倍数で指定します。たとえば、23M という値を指定すると、24,117,248 バイトのメモリが指定されます。16G という値を指定すると、17,179,869,184 バイトのメモリが指定されます。

デフォルト この手順では、ホスト上のメモリ量に基づいて設定が最適化されます。

注 CAS サーバーでは、MEMSIZE=で単一のワーカーのメモリを指定します。

ヒント　通常、DS2 でメモリ問題エラーが報告されないかぎり、メモリ制限を指定する必要はありません。

MODE=ANSI | SAS
固定長文字列と DOUBLE 列に存在しない値が SAS 欠損値として処理されるのか、あるいは SAS ANSI NULL 値として処理されるのかを指定します。 他のすべてのデータ型は、常に ANSI NULL セマンティクスを使用します。

デフォルト　SAS

操作　DS2_OPTIONS ステートメントでは、存在しないデータを処理するための設定も指定できます。MODE=値が DS2_OPTIONS ステートメントに設定されている場合、ステートメントの値は、それが指定されているプログラムブロックのプロシジャオプションの値をオーバーライドします。詳細については、DS2_OPTIONS ステートメントを参照してください。

注　null と欠損値の違いを理解することが重要です。理解していないと、データが失われる可能性があります。処理の違いの詳細については、"存在しないデータの DS2 モード" (*SAS DS2 プログラマガイド*)を参照してください。

MSGLIMIT=n | MIN | MAX
DS2 プログラムの実行中に SAS ログに書き込むことができるエラー、警告、および注意メッセージの最大数を指定します。

n　DS2 プログラムの実行中に生成される最初の *n* 個のエラー、警告、および注意メッセージをログに書き込むことを指定します。メッセージ制限に達すると、DS2 は制限に達したことを示すエラーメッセージを SAS ログに書き込みます。プログラムは完了するまで実行を続けます。ただし、後続のエラー、警告、および注意メッセージは SAS ログに書き込まれません。

MIN　DS2 プログラムの実行中に生成されるエラー、警告、または注意メッセージがログに書き込まれないことを指定します。MIN の指定は、MSGLIMIT=0 を指定することと同じです。

MAX　DS2 プログラムの実行中に生成されるすべてのエラー、警告、および注意メッセージがログに書き込まれるように指定します。

デフォルト　プログラムの実行ごとに 1,024 メッセージ

制限事項　MSGLIMIT=オプションは、MSGORDER=TEMPORAL が設定されている場合にのみ有効です。

操作　メッセージの数は、DS2_OPTIONS ステートメントでも制御できます。DS2_OPTIONS ステートメントで MSGLIMIT=値が設定されている場合、その値はプロシジャオプションの設定をオーバーライドします。

注 *n* は、1 (*n*)、1,024(*n*K)、1,048,576 (*n* M)、または 1,073,741,824 (*n* G) で指定できる整数です。たとえば、値 8 では 8 個のメッセージ、値 8K では 8,192 個のメッセージが指定されます。

MSGLIMIT=オプションは、DS2 プログラムのコンパイル中に生成されるエラーメッセージ、PUT ステートメントによって生成されるメッセージ、および SAS ログ機能に書き込まれるメッセージには影響しません。

ヒント MSGLIMIT=によって抑制されるメッセージは、App.TableServices.DS2.Runtime.Log ロガーを有効にすることでキャプチャできます。プログラムを実行する前に、レベル INFO でロガーを有効にしてください。

MSGORDER=STANDARD | TEMPORAL
DS2 が、エラー、警告、および注意メッセージを生成時に SAS ログに書き込むか、または DS2 プログラムの完了後に書き込むかを指定します。

STANDARD DS2 プログラムの完了後に DS2 がメッセージを書き込むことを指定します。ログでは、PUT ステートメントメッセージは診断メッセージの前に置かれます。

TEMPORAL DS2 プログラムによって生成されるメッセージを DS2 が書き込むことを指定します。ログでは、PUT ステートメントメッセージには診断メッセージが散在しています。

デフォルト STANDARD

操作 メッセージの順序は、DS2_OPTIONS ステートメントでも制御できます。DS2_OPTIONS ステートメントで MSGORDER=値が設定されている場合、その値はプロシジャオプションの設定をオーバーライドします。

ヒント 一時的なメッセージ順序付けは、PUT ステートメントメッセージと、エラー、警告、および注意メッセージを生成された順序で報告することにより、DS2 プログラムのデバッグに役立ちます。DS2 プログラムのロジックフローの重要なポイントに PUT ステートメントを挿入すると、エラー、警告、または注意メッセージを生成するエラー状態の原因を診断するのに役立ちます。

NUMBER
Row という名前の列(行が取得される際のデータの行(オブザベーション)番号)を含めるように指定します。

デフォルト 行番号はありません。

REPORTLINE | NOREPORTLINE
DS2 プログラム実行メッセージでログ行番号を報告するかどうかを指定します。 ログ行番号は、エラー、警告、および注意メッセージに含まれており、メッセージがログに生成されたときに実行されていた DS2 ステートメントの場所を識別します。保存されたパッケージとスレッドについて報告される行番号は、SAS ログの行番号ではなく、パッケージまたはスレッドブロックの開始からオフセットされた行番号です。

NOREPORTLINE は、行番号の生成を抑制します。

デフォルト REPORTLINE

操作 プログラム実行メッセージの行番号も、DS2_OPTIONS ステートメントで制御できます。DS2_OPTIONS ステートメントで REPORTLINE 値が設定されている場合、その値は DS2 プロシジャオプションよりも優先されます。詳細については、DS2_OPTIONS ステートメントを参照してください。

注 NOREPORTLINE は、DS2 コンパイルメッセージの行番号には影響しません。

SCOND=WARNING | NONE | NOTE | ERROR
DS2 変数宣言厳密モードの SAS ログで PROC DS2 が表示するメッセージのレベルを指定します。このモードでは、すべての変数を DS2 プログラム内で宣言する必要があります。 DS2 変数宣言厳密モードの詳細については、*SAS DS2 言語リファレンス*を参照してください。

WARNING
出力を SAS ログに書き出します。

NONE
SAS ログにメッセージは書き込まれません。

NOTE
ノートを SAS ログに書き出します。

ERROR
エラーメッセージを SAS ログに書き出します。

デフォルト デフォルトは、DS2SCOND=システムオプションによって決定されます。DS2SCOND=のデフォルトは WARNING です。DS2SCOND=システムオプションの詳細については、*SAS DS2 言語リファレンス*を参照してください。

操作 PROC DS2 ステートメントで SCOND=オプションを指定すると、DS2SCOND=システムオプションよりもそれが優先されます。

DS2 変数宣言厳密モード設定は、DS2_OPTIONS ステートメントでも制御できます。DS2_OPTIONS ステートメントで SCOND 値が設定されている場合、その値は、それが指定されている DATA、PACKAGE、または THREAD ブロックの DS2 プロシジャオプションよりも優先されます。優先規則の詳細については、DS2_OPTIONS ステートメントを参照してください。

SESSREF=*session-reference*
CAS セッションで DS2 ステートメントを実行することを指定します。CAS セッションは、そのセッション名によって識別されます。

要件 ご使用のシステム用に CAS サーバーを構成する必要があります。

CAS セッションは、前もって CAS ステートメントを使用して確立しておく必要があります。指定された CAS セッションが存在しない場合、プロシジャは終了します。

操作 CAS セッションに接続するには、SESSREF=(または SESSUUID=)を使用します。両方のオプションが指定されている場合、プロシジャステートメントの最後のオプションが適用されます。

参照項目: “CAS サーバーへの接続” (609 ページ)

“SAS Cloud Analytic Services での DS2 機能” (611 ページ)

“例 5: CAS での DS2 プログラムの実行” (626 ページ)

SESSUUID="*session-uuid*"
CAS セッションで DS2 ステートメントを実行することを指定します。CAS セッションは、その汎用一意識別子(UUID)によって識別されます。

要件 ご使用のシステム用に CAS サーバーを構成する必要があります。

CAS セッションは、前もって CAS ステートメントを使用して確立しておく必要があります。CAS ステートメントでは、UUID 値が生成されます。指定された CAS セッションが存在しない場合、プロシジャは終了します。

操作 CAS セッションに接続するには、SESSUUID=(または SESSREF=)を使用します。両方のオプションが指定されている場合、プロシジャステートメントの最後のオプションが適用されます。

参照項目: “CAS サーバーへの接続” (609 ページ)

“SAS Cloud Analytic Services での DS2 機能” (611 ページ)

例 SESSUUID=を指定している PROC DS2 プロシジャステートメントを次に示します。

```
proc ds2 sessuuid="76904741-fb09-554d-a8de-6cbce2a0e0e5";
```

UUID 値は、単一引用符または二重引用符で囲むことができます。

STIMER
経過時間統計などのシステムパフォーマンス統計のサブセットが SAS ログに書き込まれるように指定します。 STIMER が有効な場合、このプロシジャは各ステップと SAS セッション全体に使用されるコンピューターリソースのリストを SAS ログに書き込むように指定します。

デフォルト パフォーマンス統計は SAS ログに書き込まれません。

操作 SAS システムオプション FULLSTIMER が有効な場合、コンピューターリソースの詳細リストが SAS ログに書き込まれます。

TREEALGORITHM=ITERATIVE | RECURSIVE
DS2 コンパイラが構文ツリーの構築に使用するデフォルトのアルゴリズムを変更します。 RECURSIVE オプションを使用しているときにスタックオーバーフローエラーが発生した場合、アルゴリズムを変更するとスタックオーバーフローエラーが解決される場合があります。

デフォルト SAS プログラミングランタイム環境: RECURSIVE

CAS: ITERATIVE

操作 デフォルトのツリーウォーカーアルゴリズムは、DS2_OPTIONS ステートメントでも変更できます。DS2_OPTIONS ステートメントで TREEWALKER=値が設定されている場合、ステートメントの値は、それが指定されているプログラムブロックのプロシジャオプションの 値をオーバーライドします。詳細については、DS2_OPTIONS ステートメントを参照してください。

XCODE=ERROR | WARNING | IGNORE
NLS トランスコーディングが失敗した場合の SAS セッションの動作を制御します。 トランスコーディングの失敗は、行の入力または出力操作中や、ストリングの割り当て中に発生する可能性があります。トランスコーディングは、あるエンコーディングから別のエンコーディングに文字データを変換するプロセスです。

ERROR
実行時エラーが発生し、行処理が停止するように指定します。エラーメッセージが SAS ログに書き込まれます。これはデフォルトの動作です。

WARNING
互換性のない文字が代替文字に設定されるように指定します。この結果、警告メッセージが SAS ログに書き込まれます。

IGNORE
互換性のない文字が代替文字に設定されるように指定します。SAS ログにメッセージは書き込まれません。

デフォルト ERROR

RUN CANCEL ステートメント

前の DS2 言語ステートメントをキャンセルします。

ヒント: RUN CANCEL ステートメントは、エラーが発生した場合にプログラムの一部の処理をキャンセルするのに役立ちます。

例: “例 3: ラインプロンプトモードでの現在のステップの終了” (623 ページ)

構文

RUN CANCEL;

使用法: DS2 プロシジャ

SAS および DBMS データソースへの接続

ライブラリ参照名を指定して、PROC DS2 を SAS または DBMS データソースに接続します。

1 まず、接続するデータソースの SAS および SAS/ACCESS エンジンを指定する LIBNAME ステートメントをサブミットします。特定のデータソースの LIBNAME ステートメントを定義する方法については、次のリソースを参照してください。

SAS データセット
SAS グローバルステートメント: リファレンス

リレーショナル DBMS データソース
SAS/ACCESS for Relational Databases: リファレンス

MongoDB と Salesforce
SAS/ACCESS for Relational Databases: リファレンス

SPD Engine データセット
SAS Scalable Performance Data Engine: リファレンス

SPD Server テーブル
SAS Scalable Performance Data Server: ユーザーガイド

注: LIBNAME ステートメントで指定するライブラリパスは、SAS Compute Server に認識されている必要があります。

注: PROC DS2 で CAS エンジンライブラリ参照名を使用しないでください。CAS サーバーに対して DS2 プログラムを実行する手順については、"CAS サーバーへの接続" (609 ページ)を参照してください。

2 DS2 プログラムでは、テーブルを参照するために *libref.table-name* の形式で 2 部構成の名前を使用します。libref は、テーブルを作成または検索する場所を DS2 言語に指示します。

3 次に、PROC DS2 をサブミットします。

この例では、以前割り当てた libref の属性を使用して PROC DS2 がデータソースにどのようにアクセスするかを示します。

```
libname myfiles v9 'library-path'; 1

proc ds2;
  data myfiles.table1; 2
    dcl double j j2;
    method run();
      do j = 1 to 1000;
        j2 = 2*j;
        output;
      end;
    end;
  enddata;
run;
quit;
```

1 LIBNAME ステートメントは、libref Myfiles を割り当て、SAS データセットの物理的場所を指定します。

2 次に、DS2 プログラムは SAS データセット Myfiles.Table1 を作成します。プログラムは、*libref.table-name* 形式の名前を使用して、出力テーブルの場所を設定します。

DS2 プロシジャを使用してプログラムをサブミットすると、プロシジャはすべてのアクティブな libref を含むデータソース接続文字列を内部的に構築します。次に、プロシジャはこの内部接続文字列を DS2 プログラムに送信します。DS2 言語は、プログラム内の libref を使用して、この内部接続文字列内から特定のデータソース接続のプロパティを検索します。

この接続文字列には、主にデータソース接続の libref 属性(エンジン、物理的な場所、および認証など)が含まれます。文字列には、動作を定義する libref 属性が含まれていない可能性があります。たとえば、SAS V9 エンジンに以前にサブミットした LIBNAME ステートメントによって、SAS データセットが圧縮されるように指定されている場合、圧縮属性は DS2 プロシジャによって使用されません。圧縮を要求するには、DS2 プログラムで COMPRESS=テーブルオプションを指定する必要があります。いくつかの動作属性は、プロシジャによって使用されます。

LIBNAME ステートメントをサブミットする前に MSGLEVEL=i システムオプションを設定することにより、プロシジャが使用する LIBNAME オプションを判別できます。すべての LIBNAME ステートメントオプションに対応するテーブルオプションがあるわけではありません。

DS2 プログラムで libref なしでテーブルが参照される場合、または LIBNAME ステートメントからエンジン名が省略されている場合は、デフォルトの SAS V9 エンジンが使用されます。libref なしで参照されるテーブルは、SAS Work ライブラリに作成されます。User ライブラリが定義されている場合は、User ライブラリに作成されます。

CAS サーバーへの接続

DS2 プログラムを CAS サーバーにサブミットするには、次のようにします。

1 CAS ステートメントをサブミットして、CAS セッションを確立します。

2 caslib を割り当てるか、セッションで対話するデータソースに事前定義された caslib を使用します。CASLIB ステートメントを使用して、caslib を割り当て、CAS セッションで使用可能な caslib をリストすることができます。caslib は、SAS Data Connector (または SAS Data Connector Accelerator)を使用してデータにアクセスします。SAS Data Connector については、 *SAS Cloud Analytic Services: ユーザーガイド*を参照してください。

3 PROC DS2 ステートメントで、CAS セッション名またはセッション識別子を使用して SESSREF=(または SESSUUID=)プロシジャオプションを指定します。SESSREF= (または SESSUUID=)を指定すると、デフォルトの接続メカニズムと LIBS=プロシジャオプションの両方が無視されます。そのかわりに、プロシジャは、指定された CAS セッションに接続します。

4 caslib でテーブルを参照できるようになりました。アクティブな caslib にあるテーブルは、名前だけで参照できます。アクティブな caslib にないテーブルにアクセスするには、 *caslib.table-name* 形式の 2 レベルの名前が必要です。

次の例は、CAS サーバーへの PROC DS2 要求を示しています。

```
cas; 1

caslib castera desc='Teradata Caslib' 2
  datasource=(srctype='teradata'
    username='myname'
    password='mypw'
    server='testserver',
    db='testdb');

proc ds2 sessref=casauto; 3
  data newtable;
    method run();
      set {select * from employees}; 4
    end;
enddata;
run;
quit;
```

1 この例では、CAS セッションが確立されます。この CAS セッションのデフォルト名は Casauto です。

2 CASLIB ステートメントは、caslib Castera を割り当てます。Castera がアクティブな caslib になります。

3 PROC DS2 ステートメントで、SESSREF=プロシジャオプションと CAS セッション名 Casauto が指定されます。

4 DS2 SET ステートメントの FedSQL SELECT ステートメントは、Teradata テーブル Employees を名前で識別します。アクティブな caslib に存在するテーブルの名前を caslib で修飾する必要はありません。アクティブな caslib では、テーブル名での caslib の使用はオプションです。

CAS ステートメント、CASLIB ステートメント、および CASUTIL プロシジャの詳細については、*SAS Cloud Analytic Services: ユーザーガイド*を参照してください。

前の例では、データコネクタを使用してデータを CAS に自動的にロードしています。caslib を使用してテーブルを参照すると、そのテーブルは、以前にロードされ

ていなければ、指定された CAS セッションに自動的にロードされます。パスベースの場所からデータを CAS にロードすることもできます。データを CAS セッションに明示的にロードする例については、“例 5: CAS での DS2 プログラムの実行” (626 ページ)を参照してください。

PROC DS2 を使用して作成および操作する CAS テーブルは、インメモリセッションテーブルです。つまり、このテーブルは CAS セッションの間は使用可能であり、現在のセッションにのみアクセス可能です。PROC DS2 では、データソースとテーブルを保持したり、他の CAS セッションとテーブルを共有したりする方法は提供されていません。CAS テーブルを保持または共有するには、CASUTIL プロシジャを使用します。

CAS テーブルはインメモリテーブルですが、初期出力テーブルを置換出力テーブルで上書きするには、OVERWRITE=テーブルオプションを指定する必要があります。

CAS での DS2 の使用に関する詳細は、“SAS Cloud Analytic Services での DS2 機能” (611 ページ)を参照してください。

SAS Cloud Analytic Services での DS2 機能

SAS Cloud Analytic Services は、DS2 要求を処理するための代替環境です。SESSREF= (または SESSUUID=)オプションを指定すると、PROC DS2 は、CAS サーバーで DS2 プログラムを準備し、実行します。PROC DS2 を SESSREF=または SESSUUID=と一緒に実行した場合、実際は runDS2 アクションを使用することになります。詳細については、*SAS Viya プラットフォーム: システムプログラミングガイド*の runDS2 アクションを参照してください。

CAS サーバーは、対称型マルチプロセッシング(SMP)サーバーです。CAS サーバー上で DS2 プログラムを実行すると、複数のデータオブザベーションに対する操作を同時に実行し、大規模なデータセットの処理に必要な時間を減少させられます。DS2 プログラムの構造に基づいて、DS2 コンパイラーは、複数のオブザベーションで同時に実行できる操作を判別します。また、各観測に順番に適用する必要がある操作も決定します。

DS2 プログラムは、順次プログラム、並列プログラム、または並列順次プログラムのいずれかに分類されます。CAS サーバーの恩恵を受けるためには、DS2 プログラムは、DS2 並列プログラムか DS2 並列順次プログラムのいずれかのように構成されている必要があります。

- DS2 並列プログラムには、観測全体のデータ依存関係を持つ操作は含まれていません。したがって、複数のデータ観測を並行して処理できます。各 CAS ワーカーは、入力データのサブセットを処理し、結果セットのサブセットを生成することができます。
- DS2 並列順次プログラムには、オブザベーション間にデータ依存関係がある動作もあれば、データ依存関係のない動作もあります。動作の処理は、並列ステージと順次ステージの 2 ステージに分かれています。並列ステージでは、各 CAS ワーカーが、入力データセットのサブセットを処理し、中間データセットのサブセットを生成します。順次ステージでは、1 つの CAS ワーカーが、完全な中間データセットを処理し、完全な結果セットを生成します。

DS2 言語のほとんどの機能は、CAS サーバーでの使用がサポートされています。ただし、例外もいくつかあります。CAS でサポートされている DS2 機能の詳細については、*SAS DS2 プログラマガイド*を参照してください。

DS2 並列プログラムの CAS サーバーへのサブミットの例については、“例 5: CAS での DS2 プログラムの実行” (626 ページ)を参照してください。

CAS の DS2 出力テーブルはインメモリテーブルです。テーブルは、ユーザーの CAS セッションで作成されます。他の CAS アクションを使用して、CAS でグローバルに使用するために出力テーブルをプロモートするか、データを caslib データソースに保存する必要があります。

RUN グループ処理

PROC DS2 は RUN グループ処理をサポートしています。RUN グループ処理では、プロシジャを終了しなくても RUN グループをサブミットできます。

RUN グループ処理を使用するには、プロシジャを起動して複数の RUN グループをサブミットします。RUN グループは、1 つ以上のアクションステートメントを含み、RUN ステートメントで終わるステートメントのグループです。プロシジャを終了しないかぎり、アクティブなまま持続し、PROC ステートメントを再サブミットする必要はありません。

注: PROC DS2 を使用している場合、DS2 プログラムは RUN ステートメントによって区切られます。RUN ステートメントの後に別の DS2 コードが検出されると、このコードによって、前の RUN ステートメントよりも前に新しい別個の DS2 プログラムが DS2 プログラムから構成されます。

RUN グループ処理を終了するには、RUN CANCEL ステートメントをサブミットします。サブミットされなかったステートメントは終了されます。プロシジャを停止するには、QUIT ステートメントをサブミットします。サブミットされなかったステートメントも終了されます。

DS2 テーブルオプションの適用

PROC DS2 を使用してデータソースにアクセスする場合、後続の DS ステートメントで DS2 テーブルオプションを適用できます。テーブルオプションは、バッファーページサイズの割り当てやパスワードの指定などの処理をテーブル上で実行できるようにするアクションを指定します。

DS2 テーブルオプションは、SAS データセットオプションが DATA ステップ処理のオプションを適用するのと同様に、DS2 言語処理のオプションを適用するために使用されます。SAS データセットオプションは DS2 言語では使用できません。たとえば、次のコードは新しいテーブルの恒久バッファーページのサイズを指定するために、テーブルオプションを SAS データセットに適用します。

```
libname myfiles v9 'library-path';
```

```
proc ds2;
  data myfiles.table1 (bufsize=16k);
    dcl double j j2;
    method run();
      do j = 1 to 1000;
        j2 = 2*j;
        output;
      end;
    end;
enddata;
run;
quit;
```

テーブルオプションはデータソース固有です。使用可能なテーブルオプションのリストについては、“データソース別の DS2 ステートメントテーブルオプション” (*SAS DS2 言語リファレンス*)を参照してください。.

リテラル文字列でのマクロ変数の使用

注: この情報は、CAS サーバー上での DS2 の使用には適用されません。

マクロ変数を指定すると、シンボルの置換によってプログラム内でテキストを動的に変更できます。プログラム内でマクロ変数を参照すると、マクロプロセッサによって参照値は指定したマクロ変数の値に置き換えられます。

PROC DS2 では、後続の DS ステートメントでマクロ変数を使用できます。ただし、マクロ変数がリテラル文字列内にある場合は、文字列を二重引用符で囲むことはできません。マクロプロセッサでは、マクロ変数参照を解決するために二重引用符が必要です。DS2 ステートメントは、二重引用符で囲まれた文字列をテーブル名または列名などの区切り識別記号(大文字/小文字を区別する)とみなします。

リテラル文字列でマクロ変数を参照するには、SAS マクロ関数%TSLIT を使用します。%TSLIT は、リテラル文字列を二重引用符で囲む必要性を無効にし、入力値を単一引用符で囲みます。たとえば、次のステートメントには&SYSHOSTNAME マクロ変数を指定するための%TSLIT 関数が含まれており、これによって、それが実行されるコンピューターのホスト名が返されます。

```
if hostname = %tslit(&syshostname) then ...
```

%TSLIT マクロ関数は、デフォルトのオートコールマクロライブラリに保存されます。詳細については、*SAS DS2 プログラマガイド*にある“Referencing a Macro Variable in a Delimited Identifier”を参照してください。

DS2 自動変数

DS2 言語では、各ステートメントの実行後、特定の値を使用して自動変数が設定されます。これらの自動変数は、DS2 スレッド間で問題の原因を診断するときに有用

です。また、PUT ステートメントでのデバッグ時にコンテキストを提供するためにも有用です。

表 18.1　DS2 自動変数

変数	説明
HOSTNAME	DS2 プログラムが実行されているワーカーノードまたはホストの名前を返します。
NTHREADS	プログラムで実行中の DS2 スレッドの合計数を返します。並列環境では、_NTHREADS_は DS2 プログラムが実行されているすべてのノードにわたる DS2 スレッドの総数です。
THREADID	_THREADID_を返します。シリアルプログラム(スレッドコンポーネントを含まない)には、_THREADID_=0 が割り当てられます。並列プログラムでは、実行中のデータプログラムに_THREADID_=0 が割り当てられ、実行中の各スレッドプログラムには 1 からスレッド数までの一意の_THREADID_が割り当てられます。

自動変数の使用例を次に示します。

```
proc ds2;
thread thd / overwrite = yes;
  dcl double x;
  method init();
    put 'THREAD: thread' _threadid_ 'on' _hostname_;
  end;
endthread;

data _null_;
  dcl thread thd t;
  method init();
    put 'DATA: thread' _threadid_ 'on' _hostname_;
  end;
  method run();
    set from t threads=8;
  end;
enddata;
run;
quit;
```

詳細については、“DS2 スレッドで役立つ自動変数” (*SAS DS2 プログラマガイド*)を参照してください。

パスワード

この情報は、CAS サーバー上での DS2 の使用には適用されません。

SAS ソフトウェアを使用して、ファイルに SAS パスワードを割り当てることで、SAS データセットや SPD Engine データセットへのアクセスを制限できます。読み取り、書き込み、変更という、3 つのレベルの保護を 指定できます。

PROC DS2 では、DS2 テーブルオプション(ALTER=、PW=、READ=、および WRITE=)を使用してデータソースのパスワードを割り当てるか、指定します。たとえば、READ、WRITE、および ALTER の各パスワードを SAS データセットに割り当てる場合、次のコードは DS2 テーブルオプション PW=を適用します。

```
libname myfiles v9 'library-path';

proc ds2;
  data myfiles.table1 (pw=luke);
    dcl double j j2;
    method run();
      do j = 1 to 1000;
        j2 = 2*j;
        output;
      end;
    end;
enddata;
run;
quit;
```

SAS パスワードは、SAS 以外での SAS ファイルへのアクセスを制御しません。SAS の外部にある SAS ファイルへのアクセスを制御するには、オペレーティングシステムで提供されているユーティリティやファイルシステムセキュリティを 使用する必要があります。SAS パスワードの詳細については、"ファイルの保護" (*SAS プログラマガイド: エッセンシャル*)を参照してください。

CAS テーブルは SAS パスワードをサポートしていません。CAS テーブルにパスワードを割り当てることはできません。CAS からパスワードで保護されたデータにアクセスする場合、パスワードはデータにアクセスするために使用される CAS 言語要素で指定されます。たとえば、データを CAS にロードする CASUTIL プロシジャ、または CASLIB ステートメントでパスワードを指定できます。詳細については、*SAS Cloud Analytic Services: ユーザーガイ ド*の SAS Data Connector ドキュメントを参照してください。

暗号化

SAS を使用すると、SAS データセット、SPD Engine データセット、および SPD Server テーブルのコンテンツを暗号化できます。SAS では、SAS 独自の暗号化と AES 暗号化がサポートされています。

SAS 独自の暗号化は、ENCRYPT=テーブルオプションを PW=または READ=テーブルオプションで指定することによって実行されます。SAS 独自の暗号化で暗号化されたデータセットまたはテーブルは、適切なパスワードを PW=または READ=テーブルオプショで指定して解読する必要があります。

AES 暗号化は、ENCRYPT=table オプションを ENCRYPTKEY=table オプションと一緒に指定することによって実行されます。AES 暗号化で暗号化されたデータセットまたはテーブルは、後で ENCRYPTKEY=テーブルオプションを適切なキー値で指定

することによって解読されます。SAS は、次の 2 つのレベルの AES 暗号化をサポートしています: AES および AES2。AES2 は、AES よりも新しく安全な暗号化規格を満たしています。詳細については、“ENCRYPT= テーブルオプション” (*SAS DS2 言語リファレンス*)を参照してください。

DS2 では現在、CAS テーブルの暗号化属性はサポートされていません。CAS から SAS で暗号化されたデータセットや AES で暗号化されたデータセットにアクセスする際、パスワードおよび暗号化キーは、データにアクセスするために使用される CAS 言語要素で指定されます。たとえば、データを CAS にロードする CASUTIL プロシジャ、または CASLIB ステートメントで、パスワードと暗号化キーを指定できます。詳細については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector ドキュメントを参照してください。

SAS データセットの DS2 データ型のサポート

PROC DS2 では、DS2 ステートメントをサブミットするときに、すべての DS2 言語データ型がサポートされます。ただし、データの読み取りと作成の際、DS2 データ型はターゲットデータソースのデータ型との間で変換されます。次のテーブルには、DS2 データ型と、SAS データ型への変換方法またはその逆に SAS データ型からの変換方法が一覧表示されています。

表 18.2　SAS データセットの DS2 データ型の変換

DS2 データ型	レガシー SAS データ型	説明
BIGINT	SAS 数値	SAS 数値は、正確な数値データ型ではなく近似数値データ型である DOUBLE であるため、精度が失われる可能性があります。
BINARY(*n*)	SAS 文字	SAS フォーマット$n を適用します。
CHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。
DATE	SAS 数値 [1]	SAS 出力形式 DATE9 を適用します。
DECIMAL|NUMERIC(*p*,*s*)	SAS 数値	
DOUBLE	SAS 数値	
FLOAT	SAS 数値	
INTEGER	SAS 数値	
NCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。
NVARCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。

DS2 データ型	レガシー SAS データ型	説明
REAL	SAS 数値	
SMALLINT	SAS 数値	
TIME(*p*)	SAS 数値	SAS 出力形式 TIME8 を適用します。
TIMESTAMP(*p*)	SAS 数値	SAS 出力形式 DATETIME19.2 を適用します。
TINYINT	SAS 数値	
VARBINARY(*n*)	SAS 文字	SAS フォーマット$n を適用します。
VARCHAR(*n*)	SAS 文字	SAS フォーマット$n を適用します。

1 有効な SAS 日付値は、1582-01-01 から 9999-12-31 までです。SAS 日付範囲外の日付はサポートされていないため、無効な日付として処理されます。

DS2 データ型の詳細については、*SAS DS2 言語リファレンス*を参照してください。DBMS データソースのデータ型のサポートの詳細については、"データ型リファレンス" (*SAS DS2 言語リファレンス*)を参照してください。

CAS テーブルの DS2 データ型のサポート

DS2 プログラムが CAS で実行されると、DS2 データ型は CAS データ型に変換されます。CAS サーバーは、SAS データエンジンよりも多くのデータ型をサポートしています。DS2 は、CAS で BIGINT、CHAR(*n*)、DOUBLE、INTEGER、VARBINARY、および VARCHAR 型の列を作成、読み取り、および書き込みできます。一部の DS2 データ型は、CAS ではサポートされていません。次のテーブルには、DS2 データ型と、CAS データ型への変換方法が一覧表示されています。

表 18.3 CAS テーブルの DS2 データ型の変換

DS2 データ型	CAS のデータ型	説明
BIGINT	INT64	大きな符号付き、正確な整数。
BINARY(*n*)	サポート対象外 [1]	
CHAR(*n*)	CHAR	SAS 出力形式$*n* を適用します。
DATE	DOUBLE[2]	SAS 出力形式 DATE9 を適用します。

DS2 データ型	CAS のデータ型	説明
DECIMAL|NUMERIC(*p*,*s*)	サポート対象外	
DOUBLE	DOUBLE	符号付きの近似する倍精度浮動小数点数を保存します。SAS Cloud Analytic Services の場合、これは 64 ビット倍精度浮動小数点数です。
FLOAT	DOUBLE	
INTEGER	INT32	正規の符号付き、正確な整数。
NCHAR(*n*)	CHAR	SAS 出力形式$*n* を適用します。
NVARCHAR(*n*)	VARCHAR	
REAL	DOUBLE	
SMALLINT	INT32	小さい符号付き、正確な整数。
TIME(*p*)	DOUBLE	SAS 出力形式 TIME8 を適用します。
TIMESTAMP(*p*)	DOUBLE	SAS 出力形式 DATETIME25.6 を適用します。
TINYINT	INT32	非常に小さい符号付き、正確な整数。
VARBINARY(*n*)	VARBINARY	可変長のバイナリ不透明(OPAQUE)データ。このデータ型は、画像データ、音声、ドキュメント、およびその他の非構造化データを格納するために使用されます。
VARCHAR(*n*)	VARCHAR	可変長文字列を保存します。

1 CAS で実行される DS2 プログラムでは、BINARY データ型の列を作成できます。ただし、CAS テーブルから BINARY データ列を読み取ったり、CAS テーブルの既存の BINARY 列に書き込んだりすることはできません。DS2 は、そうしようとするとエラーを返します。BINARY 列を含む CAS テーブルの読み取りまたは書き込み時には、DROP データセットオプションを使用して既存の BINARY 列を除外します。

2 有効な SAS 日付値は、1582-01-01 から 9999-12-31 までです。SAS 日付範囲外の日付はサポートされていないため、無効な日付として処理されます。

CAS テーブルは、デフォルトで UTF-8 文字セットを使用します。

CAS テーブルに DATE、TIME、および TIMESTAMP 列を作成するように指定すると、DS2 は代わりに DOUBLE 列を作成し、必要に応じて、テーブルに記載されている SAS の日付、時刻、および日時形式を列に適用します。日付、時刻、およびタイムスタンプの値を含む数値列を SAS データセットおよび SPD Engine データセットから CAS に読み込む場合、DS2 は定義されている既存の形式を保持します。

例: DS2 プロシジャ

例 1: DS2 コードの導入

要素:

PROC DS2 ステートメント
LIBS=プロシジャオプション
QUIT ステートメント
DS2 言語ステートメント

詳細

この例では、SAS ログ内で **Hello World!**を表示する簡単な DS2 プログラムが使用されます。この例は、DS2 と SAS DATA ステップの基本的な違いを示しています。このコードは SAS DATA ステップと似ていますが、デフォルトのシステムメソッドなど、異なる構文要素が含まれています(INIT、RUN、および TERM)。また、DS2 では DECIMAL、INTEGER、および VARCHAR などの非常に一般的な SQL データ型がサポートされるため、DBMS データに対してよりネイティブに処理が行われます。

プログラム

```
proc ds2 libs=work;

data _null_;
  method init();
    dcl varchar(16) str;
    str = 'Hello World!';
    put str;
  end;
enddata;

run;

quit;
```

プログラムの説明

PROC DS2 ステートメントを実行します。 LIBS=接続オプションは、SAS Work ライブラリで要求を実行するように指定します。PROC DS2 ステートメントは DS2 言語ステートメントをサブミットするための環境を設定します。

```
proc ds2 libs=work;
```

DS2 言語ステートメントを入力します。 DS2 DATA ステートメントの_NULL_によって、自動的に出力が生成されないように指定されます。DS2 PUT ステートメントによって、SAS ログへの書き込みが行われます。

```
data _null_;
  method init();
    dcl varchar(16) str;
    str = 'Hello World!';
    put str;
  end;
enddata;
```

DS2 ステートメントをサブミットします。 RUN ステートメントによって、DS2 ステートメントがサブミットされます。RUN ステートメントは必須です。SAS は、RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

```
run;
```

プロシジャを停止します。 QUIT ステートメントは、プロシジャを終了します。

```
quit;
```

例のコード 18.1　Hello World!を表示する SAS ログ

```
48   proc ds2 libs=work;
49   data _null_;
50     method init();
51       dcl varchar(16) str;
52       str = 'Hello World!';
53       put str;
54     end;
55   enddata;
56   run;
Hello World!
NOTE: Execution succeeded.No rows affected.
57   quit;

NOTE: PROCEDURE DS2 used (Total process time):
      real time           16.38 seconds
      cpu time            0.15 seconds
```

例 2: SAS データセットを作成する

要素:

PROC DS2 ステートメント
LIBS=プロシジャオプション
QUIT ステートメント
LIBNAME ステートメント
DS2 言語ステートメント
PROC PRINT

詳細

この例では、DS2 プロシジャをサブミットしてから DS2 言語ステートメントをサブミットすることで、SAS セッション内で SAS データセットが作成されます。出力には、データセットの最初の 10 行が示されます。

プログラム

```
libname myfiles base 'library-path';
proc ds2 libs=myfiles;


  data myfiles.basetable;
    declare double j j2;
    method run();
      do j = 1 to 1000;
        j2 = 2*j;
        output;
      end;
    end;
  enddata;
run;
quit;
proc print data=myfiles.basetable (obs=10);
run;
```

プログラムの説明

ライブラリ参照を作成する SAS データセットに割り当てます。 LIBNAME ステートメントは libref Myfiles を割り当て、SAS V9 エンジンを指定し、SAS データセットの物理的場所を指定します。

```
libname myfiles base 'library-path';
```

PROC DS2 ステートメントを実行します。 PROC DS2 ステートメントは、ibref Myfiles を使用してデータソースに接続し、DS2 言語ステートメントをサブミットするための環境を設定します。

```
proc ds2 libs=myfiles;
```

DS2 言語ステートメントを入力します。 DS2 DATA ステートメントによって、Myfiles.BaseTable という名前の出力テーブルが作成されます。DATA ステートメント内の 2 レベルの名前では、カタログ識別子 Myfiles が指定されます。DECLARE ステートメントによって、データ型 DOUBLE が変数 J および J2 に割り当てられます。METHOD ステートメントによって、出力の作成に使用される RUN システムメソッドが識別されます。OUTPUT ステートメントによって、DO ループを実行するたびに 1 行がテーブル Myfiles.BaseTable に書き込まれます。

```
data myfiles.basetable;
  declare double j j2;
  method run();
    do j = 1 to 1000;
      j2 = 2*j;
      output;
    end;
  end;
enddata;
```

DS2 言語ステートメントをサブミットします。 RUN ステートメントによって、DS2 ステートメントがサブミットされます。RUN ステートメントは必須です。SAS は、RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

```
run;
```

プロシジャを停止します。 QUIT ステートメントは、プロシジャを終了します。

```
quit;
```

SAS データセットを出力します。 PRINT プロシジャによって、オブザベーションが SAS データセットに出力されます。OBS=データセットオプションによって、出力オブザベーション数は 10 に制限されます。

```
proc print data=myfiles.basetable (obs=10);
run;
```

出力: SAS データセットを作成する

アウトプット 18.1　Myfiles.V9Table の PROC PRINT 出力

The SAS System

Obs	j	j2
1	1	2
2	2	4
3	3	6
4	4	8
5	5	10
6	6	12
7	7	14
8	8	16
9	9	18
10	10	20

例 3: ラインプロンプトモードでの現在のステップの終了

要素:

PROC DS2 ステートメント
RUN CANCEL ステートメント
QUIT ステートメント
DS2 言語ステートメント

詳細

次の例は、ラインプロンプトモードセッションでの RUN CANCEL ステートメントの有効性を示しています。コード内の 6 番目のステートメントには、列(Y ではなく Z)の無効な値が含まれています。RUN CANCEL は PROC DS2 ステップを終了し、実行されないようにします。

プログラム

```
proc ds2;
data xy_data;
  declare double x y;
  method init();
    do x = 1 to 5;
      z = 2*x;
    end;
  end;
enddata;
run cancel;
quit;
```

例のコード 18.2　キャンセルされた PROC DS2 ステップを示す SAS ログ

```
17   proc ds2 libs=work;
NOTE: Writing HTML Body file: sashtml1.htm
18   data xy_data;
19     dcl double x y;
20     method init();
21       do x = 1 to 5;
22         z = 2*x;
23       end;
24     end;
25   enddata;
26   run cancel;
NOTE: DS2 query cancelled per user request.
27   quit;

NOTE: PROCEDURE DS2 used (Total process time):
      real time          14.33 seconds
      cpu time           0.71 seconds
```

例 4: 値に基づいてテーブルにデータを送る

要素:

PROC DS2 ステートメント
QUIT ステートメント
DS2 言語ステートメント

詳細

この例では、条件に基づいてテーブルを作成する方法を示します。プログラム 1 および 2 によって、Dept1_Items および Dept2_Items という 2 つのテーブルが作成され、2 つの部門によって使用されるアイテムのコストが格納されます。3 番目のプログラムによって、2 つのアイテムテーブル内のアイテムのコストに基づいて、

Highcosts および Lowcosts という 2 つのテーブルが作成されます。プログラム 4 および 5 によって、コストテーブルのコンテンツが出力されます。

プログラム

```
proc ds2;
/* Program 1 */
data dept1_items (overwrite=yes);
  dcl varchar(20) item;
  dcl double cost;
  method init();
    item = 'staples';   cost =  1.59; output;
    item = 'pens';      cost =  3.26; output;
    item = 'envelopes'; cost = 11.42; output;
  end;
enddata;
run;
/* Program 2 */
data dept2_items (overwrite=yes);
  dcl varchar(20) item;
  dcl double cost;
  method init();
    item = 'erasers'; cost =  5.43; output;
    item = 'paper';   cost = 26.92; output;
    item = 'toner';   cost = 62.29; output;
  end;
enddata;
run;
/* Program 3 */
data lowCosts (overwrite=yes) highCosts (overwrite=yes);
  method run();
    set dept1_items dept2_items;
    if cost <= 10.00 then
      output lowCosts;
    else
      output highCosts;
    end;
enddata;
run;
/* Program 4 */
data;
  method run();
    set lowCosts;
  end;
enddata;
run;
/* Program 5 */
data;
  method run();
     set highCosts;
  end;
enddata;
run;
quit;
```

出力: 条件に基づいたテーブルの作成

アウトプット 18.2　コストテーブルの出力

The SAS System

item	cost
staples	1.59
pens	3.26
erasers	5.43

The SAS System

item	cost
envelopes	11.42
paper	26.92
toner	62.29

例 5: CAS での DS2 プログラムの実行

要素:

CAS システムオプション
CAS ステートメント
CASLIB ステートメント
CASUTIL プロシジャ
PROC DS2 ステートメント
SESSREF=プロシジャオプション
DS2 言語ステートメント

詳細

次に、CAS で実行される DS2 並列プログラムの例を示します。DS2 並列プログラムは、スレッドプログラムとデータプログラムを含み、データ操作ステートメントを含まない DS2 プログラムです。つまり、このデータプログラムには、SET FROM と OUTPUT 以外のステートメントは含まれていません。スレッドプログラムの動作は、複数のデータオブザベーションに並列適用されます。各 CAS ワーカーは、データセットのサブセットを処理し、結果セットのサブセットを生成します。

プログラム

```
cas;

caslib casdata datasource=(srctype=path)
 path="testdata/cas";

proc casutil;
 load casdata="cars_single.sashdat" incaslib="casdata" casout="cars_single";
run;

proc ds2 sessref=casauto;

thread cars_thd / overwrite=yes;
  method run();
    set cars_single;
    if (msrp > 100000) then do;
      put make= model= msrp=;
      output;
    end;
  end;
endthread;

data cars_luxury /overwrite=yes;
  dcl thread cars_thd t;
  method run();
    set from t threads=4;
  end;
enddata;


run;

quit;
```

プログラムの説明

CAS セッションを確立します。 CAS ステートメントをサブミットすることによって、CAS セッションを確立します。この CAS セッションは、デフォルト名 Casauto で作成されます。

```
cas;
```

入力データにアクセスするための caslib を定義します。 Cars_Single.sashdat という名前のファイルにアクセスする必要があります。このファイルは、CAS サーバーが実行されているコンピューターの Testdata/Cas サブディレクトリにあります。CASLIB ステートメントでは、ディレクトリの場所に caslib Casdata が割り当てられます。別のコンピューター上のディレクトリにアクセスするには、絶対パス名を指定します。

```
caslib casdata datasource=(srctype=path)
 path="testdata/cas";
```

処理のために CAS にテーブルをロードします。 CASUTIL プロシジャを使用して、CAS セッションにテーブル Cars_Single.sashdat をロードします。CASOUT=パラメーターでは、ロードされたテーブルに名前 Cars_Single が割り当てられます。

```
proc casutil;
 load casdata="cars_single.sashdat" incaslib="casdata" casout="cars_single";
run;
```

PROC DS2 ステートメントを発行し、SESSREF=プロシジャオプションを指定します。 SESSREF=プロシジャオプションで、CAS セッション名を指定します。SESSREF=オプションは Casauto を指定します。SESSREF=オプションは、DS2.runDS2 アクションに続く DS2 プログラムを渡し、CAS セッション Casauto でそれらを処理するようにプロシジャに指示します。

```
proc ds2 sessref=casauto;
```

PROC DS2 ステートメントを入力します。 DS2 THREAD ステートメントは、ロードされたテーブル Cars_Single からデータを選択するための基準を指定する Cars_Thd というスレッドプログラムを作成します。DS2 DATA ステートメントは、出力テーブル Cars_Luxury を作成し、新しいテーブルの内容としてスレッドプログラムの結果を設定するように指定します。データプログラムは、4 つのスレッドを使用してスレッドプログラムを実行するように指定します。

```
thread cars_thd / overwrite=yes;
 method run();
  set cars_single;
  if (msrp > 100000) then do;
   put make= model= msrp=;
   output;
  end;
 end;
endthread;

data cars_luxury /overwrite=yes;
 dcl thread cars_thd t;
 method run();
  set from t threads=4;
 end;
enddata;
```

DS2 言語ステートメントをサブミットします。 RUN ステートメントによって、DS2 ステートメントがサブミットされます。RUN ステートメントは必須です。SAS は、RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

```
run;
```

プロシジャを停止します。 QUIT ステートメントは、プロシジャを終了します。

```
quit;
```

19

DSTODS2 プロシジャ

概要: DSTODS2 プロシジャ

DSTODS2 プロシジャの機能

DSTODS2 プロシジャを使用すると、SAS DATA ステップコードのサブセットを DS2 コードに変換できます。次に、プログラムを改訂して DS2 の機能を利用し、PROC DS2 を使用してプログラムをサブミットすることができます。

詳細については、18 章, “DS2 プロシジャ,” (595 ページ)を参照してください。

概念: DSTODS2 プロシジャ

入力ファイルと出力ファイル

PROC DSTODS2 は、入出力用のテキストファイルで動作し、これらのファイルは直接名前を付けたファイルまたは SAS ファイル参照名として表示されます。

PROC DSTODS2 には、変換するソースを含む入力ファイルが必要です。入力ファイルは IN=引数で指定します。PROC DSTODS2 には、変換ソースが書き込まれる出力ファイル名も必要です。このファイルは、OUT=引数で指定します。

注: 結果の出力ファイルが構文的に完全でない可能性があります。コンパイルして実行できる DS2 プログラムを作成するには、出力ファイルをクリーンアップする必要があります。

注: PROC DSTODS2 は、このコード行を出力ファイルに追加します。これは、クリーンアップ中に.ds2 ファイルから削除しないでください。

```
_return: ;
```

サポートされている構文とサポートされていない構文

概要

PROC DSTODS2 は、すべての 可能な DATA ステップ構文を変換することはできません。その主な目的は、完全な DATA ステップ構文のサブセットである典型的な SAS Enterprise Miner スコアリング構文をサポートすることです。

サポートされている構文

PROC DSTODS2 がサポートする DATA ステップ構文には、次の項目が含まれます。

- 次の文がサポートされています。

  ARRAY ステートメントと配列イニシャライザー。暗黙の配列は警告を出します。

  配列とスカラーを使用する代入文。

  属性ステートメントは、PROC DSTODS2 でセマンティックな意味を持ちません。構文だけがサポートされています。つまり、構文エラーが発生しません。

  BY

  コメント

  CONTINUE

  DATA ステートメントおよび関連するデータセット

  DO (DO OVER を除くすべての形式)

  DROP

  END

  FORMAT (LENGTH 文もプログラムに含まれている場合のみ)

  GOTO

  IF および IF-THEN/ELSE

  KEEP

  LABEL

  LEAVE

  LENGTH

  MERGE

  Null

  OUTPUT

  基本的な PUT ステートメント機能: 変数、配列、_ALL_、およびフォーマットされた値の出力。他の PUT 引数はインラインでコメントアウトされます。

  RENAME

  RETAIN

  RETURN

  RUN (構文的にはサポートされていますが、セマンティックな意味はありません)

  SELECT

  SET

  STOP

  合計

- ハッシュオブジェクトの部分的なサポート。
- DROP、IN、KEEP、および RENAME データセットオプション
- すべての DATA ステップ式

- すべての関数とフォーマットはそのまま変換されます。ただし、DS2 でサポートされている関数とフォーマットだけが有効です。詳細については、“DS2 関数” (*SAS DS2 言語リファレンス*)を参照してください。
- 定数リスト(IN 句で使用される)
- リスト内の型修飾子を除く変数リスト。たとえば、 **x-numeric-a**

注: DATA ステップ言語の深さのために、このリストは網羅的ではありません。

サポートされていない構文

PROC DSTODS2 でサポートされていない DATA ステップ構文には、次のアイテムが含まれます。

- 次のステートメントはサポートされていません。

  ABORT
  ATTRIB
  CALL
  CARDS
  CARDS4
  DATALINES
  DATALINES4
  DELETE
  DESCRIBE
  DISPLAY
  DO OVER
  ERROR
  EXECUTE
  FILE
  FORMAT(宣言されていない変数)
  INFILE
  INFORMAT
  INPUT (すべてのフォーム)
  LINK
  LIST
  LOSTCARD
  MODIFY
  任意の書式の PUT ステートメント
  PUTLOG
  REMOVE
  REPLACE
  UPDATE
  WHERE
  WINDOW

- DROP、IN、KEEP、および RENAME 以外のデータセットオプション
- DEBUG や PASSTHRU のような DATA ステップの実行可能オプション
- 次の I/O オプションはサポートされていません。

  CONTROL
  CUROBS
  END
  INDSNAME
  KEY
  NOBS
  OPEN
  POINT
  KEYRESET

- Implicit arrays
- INPUT および PUT 修飾子
- Hash オブジェクト以外のすべてのオブジェクト

- すべてのメソッドにおける引数タグ
- プログラムシンタックスの表示と保存
- 定数リストの定数範囲(1:100 など)
- フォーマットジャスティファイアー(L、R、C)
- DMNORM および STREAMINIT を除く CALL ステートメント
- ビットストリング式
- その他のプロシジャ

注: DATA ステップ言語の深さのために、このリストは網羅的ではありません。

PROC DSTODS2 に関する考慮事項と制限事項

- 一度に変換できる DATA ステッププログラムは 1 つだけです。
- 結果の DS2 プログラムには、最終的な DS2 プログラムを保存する前に削除する必要がある 2 行のコードが含まれています。

```
ds2_options sas tkgmac;
_return: ;
```

- 既存のコメントは削除されます。
- 変換できないコード行はコメントに置かれます。

  これらのコメントは、元のコードを、変換されたコードと簡単に比較できるように、可能な限り元の位置にインラインで配置されます。しかし、コメントや他のコードが他の位置に移動される可能性のある状況があります。たとえば、コードをブロックの最上部(最も外側のスコープ)に移動することができます。
- 結果の DS2 プログラムは構文的に完全ではないかもしれません。結果のプログラムは、次の結果のいずれかになります。
  - コンパイルしてエラーなしで実行します。新しい DATA ステップ構文のいずれかが使用されている場合には、その可能性はほとんどありません。
  - コンパイルしますが、実行に失敗します。これは、コメントアウトされたセクションを含むプログラムの場合に特に当てはまります。
  - コンパイルしません。これは、警告があったか、構文的には変換されますが、DS2 では対応するサポートがないために発生する場合があります。これは、特定の関数やフォーマットの場合に当てはまる場合があります。これは、変数リスト機能の一部にも発生する場合があります。
  - 致命的な構文エラーが発生します。これは以前に発見されていない機能や前述の変数リスト構文の一定の側面で発生する場合があります。
- 結果の DS2 プログラムは、SAS In-Database Code Accelerator を起動しません。並行して実行できるコードの部分を取り、データプログラムに付随するスレッドプログラムを作成する必要があります。
- 32767 文字より長いコード行は、切り捨てられます。

- DATA ステップの固定長文字はバイト単位で測定されます。DS2 固定長文字は文字単位で測定されます。データにマルチバイトエンコーディングがある場合は、PROC DSTODS2 を実行する前に LENGTH ステートメントを調整して、切り捨てを避けるために変数の長さを文字数で考慮する必要があります。シングルバイトエンコーディングには問題ありません。
- CAS サーバーで実行する SESSREF=オプションを指定して DATA ステッププログラムで PROC DSTODS2 を実行する場合は、CAS サーバー上で実行する前に SESSREF=オプションを DATA ステートメントから PROC DS2 ステートメントに移動する必要があります。

構文: DSTODS2 プロシジャ

制限事項: このプロシジャは、CAS サーバーではサポートされません。

PROC DSTODS2 IN=*datastep-program-filename* OUT=*ds2-program-filename* <OUTDIR="*output-directory-name*">

RUN;

<QUIT;>

ステートメント	タスク	例
“PROC DSTODS2 ステートメント”	DATA ステップコードを DS2 コードに変換します。	Ex. 1, Ex. 2, Ex. 3

PROC DSTODS2 ステートメント

DATA ステップコードを DS2 コードに変換します。

構文

PROC DSTODS2 IN="*datastep-program-filename*" OUT="*ds2-program-filename*" <OUTDIR="*output-directory-name*">

必須引数

IN=*datastep-program-filename*
DATA ステップファイルの名前または SAS ファイル参照名を指定します。

操作 *datastep-program-filename* にファイル参照名を使用すると、PROC DSTODS2 はファイル参照名オプション(たとえば、エンコーディング)を調べません。

ヒント フルパス名を指定することができます。

OUT=*ds2-program-filename*
作成される DS2 ファイルの名前を指定します。

制限事項 出力ファイル名は 1 つだけ指定する必要があります。パス名を含めることはできません。

注 出力.ds2 プログラムから次の 2 行を削除する必要があります。

```
ds2_options sas tkgmac;
_return: ;
```

ヒント OUTDIR=ディレクトリを指定できない、または指定しない場合、出力ファイルは自動的に現在の作業ディレクトリに書き込まれます。このコードを使用して、現在の作業ディレクトリを見つけることができます。

```
data _null_;
file 'name.txt';
put x;
run;
```

オプション引数

OUTDIR="*output-directory-name*"
ファイルの出力ディレクトリ名を示します。

例: DSTODS2 プロシジャ

例 1: データ入力

要素: PROC DSTODS2 ステートメント

詳細

この例では、PROC DSTODS2 を使用して、SET ステートメントと BY ステートメントを持つ次の DATA ステッププログラム(dsEx1.sas)を変換します。

```
data _null_;
  length x y z w 8;
  set x;
  by x--z w;
  put x=;
run;
```

PROC DSTODS2 ステートメントを実行します。 現在割り当てられた libref がない場合、PROC DSTODS2 ステートメントは DS2 言語ステートメントをサブミットするための環境を設定します。

```
proc dstods2 in="dsEx1.sas" out="ds2Ex1.ds2";
```

DSTODS2 ステートメントをサブミットします。 RUN ステートメントによって、DSTODS2 ステートメントがサブミットされます。RUN ステートメントは必須です。SAS は、RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

```
run;
```

例のコード 19.1　結果を表示する SAS ログ

```
4   proc dstods2 in="dsEx1.sas" out="ds2Ex1.ds2";
5   run;

NOTE: PROCEDURE DSTODS2 used (Total process time):
      real time          0.20 seconds
      cpu time           0.07 seconds
```

Output: ds2Ex1.ds2 Program

PROC DSTODS2 は、dsEx1.sas を DS2 コードに変換して結果を dsEx1.ds2 に格納します。ご覧の通り、LENGTH ステートメントは DECLARE ステートメントに変換され、METHOD ステートメントが追加され、変数リスト **X--Z** がコメントアウトされました。

```
data _NULL_;
dcl double X;
dcl double Y;
dcl double Z;
dcl double W;
method run();
set X;
by  /* X--Z */ W;
put X=;
;
_return: ;
end;
enddata;
```

例 2: 行指定子を使用した PUT ステートメント

要素: PROC DSTODS2 ステートメント

詳細

この例では、PROC DSTODS2 を使用して、DO ステートメントといくつかの PUT ステートメントを持つ次の DATA ステッププログラム(dsEx2.sas)を変換します。

```
data _null_;
  file temp linesize=32600;
  do i = 1 to 5330;
    put i 6. @;
  end;
  put @31990 'N1=72' @;
  put @32580 'N2=32413' @;
  put @32001 'N4=32 N3=783424123' @;
  put @32477 'N5=1977' @;
  put @32222 'N6=1981' ;
run;
```

PROC DSTODS2 ステートメントを実行します。 現在割り当てられた libref がない場合、PROC DSTODS2 ステートメントは DS2 言語ステートメントをサブミットするための環境を設定します。

```
proc dstods2 in="dsEx2.sas" out="ds2Ex2.ds2";
```

DSTODS2 ステートメントをサブミットします。 RUN ステートメントによって、DSTODS2 ステートメントがサブミットされます。RUN ステートメントは必須です。SAS は、RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

```
run;
```

例のコード 19.2 結果を表示する SAS ログ

```
4   proc dstods2 in="dsEx2.sas" out="ds2Ex2.ds2";
5   run;

NOTE: PROCEDURE DSTODS2 used (Total process time):
      real time          0.18 seconds
      cpu time           0.06 seconds
```

Output: ds2Ex2.ds2 Program

PROC DSTODS2 は、dsEx2.sas を DS2 コードに変換して結果を dsEx2.ds2 に格納します。ご覧の通り、METHOD ステートメントが追加され、FILE ステートメントがコメントアウトされました。PUT ステートメントは変換されますが、@ラインホールド指定子と定数はコメントアウトされました。

```
data _NULL_;
method run();
/* FILE TEMP LINESIZE = 32600 */;
do I = 1.0 to 5330.0;
put I 6. /* Put statement contains unsupported feature(s) @  */ ;
end;
put  /* Put statement contains unsupported feature(s) @  31990  'N1=72'
 @  */ ;
put  /* Put statement contains unsupported feature(s) @  32580  'N2=32413'
 @  */ ;
put  /* Put statement contains unsupported feature(s) @  32001  'N4=32
N3=783424123'
 @  */ ;
put  /* Put statement contains unsupported feature(s) @  32477  'N5=1977'
 @  */ ;
put  /* Put statement contains unsupported feature(s) @  32222  'N6=1981'
 */ ;
;
_return: ;
end;
 enddata;
```

例 3: 配列

要素: PROC DSTODS2 ステートメント

詳細

この例では、PROC DSTODS2 を使用して、ARRAY ステートメントを持つ次の DATA ステッププログラム(dsEx2.sas)を変換します。

```
data _null_;
  array a(*) a1-a5;
  retain a1-a5 (1*(10 10 10 10 10));
  put _all_;
run;
```

PROC DSTODS2 ステートメントを実行します。 現在割り当てられた libref がない場合、PROC DSTODS2 ステートメントは DS2 言語ステートメントをサブミットするための環境を設定します。

```
proc dstods2 in="dsEx3.sas" out="ds2Ex3.ds2";
```

DSTODS2 ステートメントをサブミットします。 RUN ステートメントによって、DSTODS2 ステートメントがサブミットされます。RUN ステートメントは必須です。SAS は、RUN ステートメントに到達するまで 1 つのタスクに関連付けられたプログラムステートメントを読み取ります。

```
run;
```

例のコード 19.3　結果を表示する SAS ログ

```
6   proc dstods2 in="dsex3.sas" out="ds2ex3.ds2";
7   run;

NOTE: PROCEDURE DSTODS2 used (Total process time):
      real time          1.83 seconds
      cpu time           1.14 seconds
```

Output: ds2Ex3.ds2 Program

PROC DSTODS2 は、dsEx3.sas を DS2 コードに変換して結果を dsEx3.ds2 に格納します。ご覧の通り、METHOD ステートメントが追加され、ARRAY ステートメントが VARARRAY ステートメントに変換されました。

```
data _NULL_;
retain A1-A5 (1 * (10, 10, 10, 10, 10)) ;
vararray double A[*] A1-A5;
 method run();
put _ALL_;
;
_return: ;
end;
 enddata;
```

20

EXPORT プロシジャ

概要: EXPORT プロシジャ

EXPORT プロシジャの機能

EXPORT プロシジャは SAS データセットまたはテーブルからデータを読み込み、外部データソースに書き込みます。外部データソースには次のものがあります。

- dBase ファイル
- 区切りファイル
- JMP ファイル

- Paradox ファイル
- SPSS ファイル
- Stata ファイル

CAS エンジンは、CAS サーバー上のメモリ内の CAS テーブルを PC ファイル形式にエクスポートするための JMP、SPSS、および Stata ファイルもサポートしています。CAS エンジンを使用して CAS データをインポートまたはエクスポートするには、CAS サーバーへの接続を確立する必要があります。CAS テーブルをエクスポートすると、すべての VARCHAR データ型が PC ファイル形式の CHAR データ型に変換されます。CAS エンジンと CAS LIBNAME ステートメントの詳細については、*SAS Cloud Analytic Services: ユーザーガイド*を参照してください。

区切りファイルでは、区切り文字(ブランク、カンマ、タブなど)でデータ値の列が区切られます。SAS/ACCESS Interface to PC Files のライセンスを持っている場合、Microsoft Access データベース、Microsoft Excel ワークブックおよび DBF ファイルなど、別のファイル形式にエクスポートすることもできます。

ユーザーは SAS データセットを JMP 7 以降のファイルにエクスポートできます。また、JMP 変数には最大 255 文字を使用できます。拡張属性が自動的に使用されるようになり、META=ステートメントは JMP ファイルでサポートされなくなりました。詳細については、*"JMP Files" (SAS/ACCESS Interface to PC Files: Reference for the SAS Viya Platform)*を参照してください。

EXPORT プロシジャは、次のいずれかの方法を使用してデータをエクスポートします。

- 生成された DATA ステップコード
- 生成された SAS/ACCESS コード

出力データソースに特有のオプションやステートメントを指定して、結果を制御します。EXPORT プロシジャは指定された出力ファイルを生成し、エクスポートに関する情報を SAS ログに書き込みます。ログには EXPORT プロシジャによって生成される DATA ステップまたは SAS/ACCESS コードが表示されます。変換エンジンを使用する場合は、コードはサブミットされません。

SAS Viya プラットフォームでのフォーマットカタログのエンコーディング

SAS Viya プラットフォームでは、UTF-8 エンコーディングのみがサポートされています。

フォーマットカタログのエンコーディングの詳細については、*SAS Viya プラットフォームでの UTF-8 へのデータの移行*および *SAS/ACCESS Interface to PC Files: SAS Viya プラットフォームリファレンス*を参照してください。

VARCHAR データ型のサポート

CAS では、PROC EXPORT で VARCHAR データ型がサポートされています。VARCHAR では、可変長の文字変数が保存されます。変数に指定する長さは、保存する文字の最大数を表します。

VARCHAR データ型は、CHAR データ型に似ています。CHAR 変数の長さは、バイト単位で測定されます。VARCHAR 変数の長さは、バイト単位ではなく文字単位で測定されます。VARCHAR の使用法については、 *SAS Cloud Analytic Services: DATA ステッププログラミング*を参照してください。

次の例では、CAS Engine を LENGTH ステートメントと一緒に使用して、VARCHAR 変数と CHAR 変数を作成します。VARCHAR 変数 X の長さは 30 で、CHAR 変数 Y の長さも 30 です。

```
libname mycas cas;
data mycas.string;
  length x varchar(30);
  length y $30;
  x = 'abc'; y = 'def';
run;
proc contents data=mycas.string; run;
```

このコードで生成される出力を次に示します。

The CONTENTS Procedure

Data Set Name	MYCAS.STRING	Observations	1
Member Type	DATA	Variables	2
Engine	CAS	Indexes	0
Created	09/11/2016 13:48:30	Observation Length	48
Last Modified	09/11/2016 13:48:30	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Limit	100MB

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	x	Varchar	30
2	y	Char	30

構文: EXPORT プロシジャ

制限事項: EXPORT プロシジャは次の動作環境で使用できます。

- Linux

ファイルのパス名には、最大で 201 文字を使用できます。

ヒント: CAS テーブルでは、PROC EXPORT で VARCHAR データ型がサポートされています。詳細については、“VARCHAR データ型のサポート” (643 ページ)を参照してください。

PROC EXPORT DATA=<*libref* | *caslib.tablename*.>*SAS data set* <(*SAS data set options*)>

OUTFILE="*filename*" | OUTTABLE="*tablename*"

<DBMS=*identifier*> <REPLACE> <LABEL>;

statements for exporting to delimited files

DELIMITER=*char* | '*nn*'x;

PUTNAMES=YES | NO;

JMP ファイルをエクスポートするステートメント

DBENCODING=*12-char SAS encoding-value*;

FMTLIB=<*libref.*>*format-catalog*;

META=*libref.member-data-set*;

ステートメント	タスク	例
“PROC EXPORT ステートメント”	SAS データセットを外部データファイルにエクスポートします。	Ex. 1, Ex. 2
“DBENCODING ステートメント”	JMP ファイルにデータを保存するために使用するエンコーディングを示します。	
“DELIMITER ステートメント”	区切り文字で区切られた出力ファイルでデータ列を区切る文字を指定します。	Ex. 1
“FMTLIB ステートメント”	フォーマットカタログで定義された SAS 出力形式の値を JMP ファイルの値ラベルに書き込みます。	
“META ステートメント”	SAS メタデータ情報を JMP ファイルに書き込みます。	
“PUTNAMES ステートメント”	エクスポートするデータファイルの最初の行に列見出しとして SAS 変数名を書き込みます。	Ex. 3

PROC EXPORT ステートメント

SAS データセットを外部データファイルにエクスポートします。

ヒント: PROC EXPORT は、FILENAME ステートメントで使用可能なアクセスの種類をすべてサポートしています。

CAS テーブルでは、PROC EXPORT で VARCHAR データ型がサポートされています。詳細については、“VARCHAR データ型のサポート” (643 ページ)を参照してください。

参照項目: “FILENAME ステートメント” (*SAS グローバルステートメント: リファレンス*)

構文

PROC EXPORT DATA=<*libref SAS data set | caslib.tablename.*> <(*SAS data set options*)>

OUTFILE="*filename*" | OUTTABLE="*tablename*"

<DBMS=*identifier*> <REPLACE> <LABEL>;

オプション引数の要約

(*SAS data set options*)
SAS データセットオプションを指定します。

DBMS=*identifier*
EXPORT プロシジャが作成する外部データソースのタイプを指定します。DBMS テーブルにエクスポートする場合は、有効なデータベース識別子を使用して DBMS オプションを指定する必要があります。

LABEL
変数ラベル名を指定します。

REPLACE
既存のファイルを上書きします。

必須引数

DATA=<*libref.*>*SAS data set | caslib.tablename*
入力力 SAS データセットを 1 レベルまたは 2 レベルの SAS 名(ライブラリとメンバー名)で識別します。1 レベルの名前を指定した場合、EXPORT プロシジャはデフォルトで USER ライブラリ(割り当てられている場合)または WORK ライブラリのどちらかを使用します。

caslib.tablename は、入力 CAS テーブルを指定します。CAS テーブルには 1 レベルの名前だけを指定することはできないため、DATA=2 レベルの名前(caslib およびテーブル)を使用する必要があります。

EXPORT プロシジャは、データターゲットが SAS データセットの出力形式をサポートしている場合のみ、SAS データセットをエクスポートできます。データの容量もデータターゲットの制限以内でなければなりません。たとえば、データファイルの中には行または列数に最大値が設定されているものもあります。SAS のユーザー定義の出力形式と入力形式をサポートできないデータファイルもあります。エクスポートする SAS データセットがターゲットファイルの制限を超える場合は、EXPORT プロシジャによって正しくエクスポートできない場合もあります。多くの場合、プロシジャは可能な限り最善の方法でデータを変換します。ただし、種類によっては変換できないものもあります。

VALIDMEMNAME=EXTEND システムオプションも指定されている場合、SAS データセット名には単一引用符を含めることができます。VALIDMEMNAME=を使用すると、SAS データセット名など、特定の SAS メンバーの名前に関するルールが拡張されます。詳細については、“SAS データセットと変数の命名規則” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

デフォルト
エクスポートする SAS データセットを指定しない場合、EXPORT プロシジャは直近に作成された SAS データセットを使用します。SAS は、システム変数_LAST_を使用してデータセットを追跡します。EXPORT プロシジャが確実に正しいデータセットを使用するようにするには、SAS データセットを指定する必要があります。

制限事項
PROC EXPORT は、DBMS=CSV | TAB | DLM の名前範囲リストで DROP | KEEP データセットオプションをサポートしていません。

例
“例 1: 区切り文字で区切られた外部データソースにエクスポートする” (652 ページ)

“例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする” (657 ページ)

OUTFILE="*filename*" | "*fileref*"
出力 PC ファイル、スプレッドシート、または区切り文字で区切られた外部ファイルの完全パスとファイル名またはファイル参照名を指定します。ファイル参照名は物理的なファイルの場所に関連付けられた SAS 名です。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

ファイル参照名を指定した場合、または完全パスとファイル名に特殊文字(パスのバックスラッシュなど)、小文字、またはスペースが含まれない場合は、引用符を省略できます。

別名
FILE=

制限事項
EXPORT プロシジャでは、DISK を除き、FILENAME ステートメントのデバイスの種類またはアクセスメソッドはサポートされません。たとえば、EXPORT プロシジャでは一時外部ファイルを作成する TEMP デバイスの種類はサポートされていません。

PROC EXPORT は、DBMS=CSV | TAB | DLM の名前範囲リストで DROP| KEEP データセットオプションをサポートしていません。

参照項目:
PC ファイル形式の詳細については、*SAS/ACCESS Interface to PC Files: SAS Viya プラットフォームリファレンス*を参照してください。

例 “例 1: 区切り文字で区切られた外部データソースにエクスポートする” (652 ページ)

“例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする” (657 ページ)

OUTTABLE="*tablename*"
出力 DBMS テーブルのテーブル名を指定します。名前に特殊文字(疑問符など)、小文字、またはスペースが含まれない場合は、引用符を省略できます。DBMS テーブル名では大文字と小文字が区別される場合もあります。

別名 TABLE=

要件 DBMS テーブルをエクスポートするには SAS/ACCESS Interface to PC Files のライセンスが必要です。

DBMS テーブルをエクスポートするときは、DBMS オプションを指定する必要があります。

オプション引数

DBMS=*identifier*
EXPORT プロシジャが作成する外部データソースのタイプを指定します。DBMS テーブルにエクスポートする場合は、有効なデータベース識別子を使用して DBMS オプションを指定する必要があります。

CSV
出力データソースは、カンマ区切りのファイルです。ファイル拡張子は.CSV です。

DBF
dBASE 5.0、IV、III+、および III ファイル。ファイル拡張子は.dbf です。

DBFMEMO
メモ付きの dBASE 5.0、IV、III+、および III ファイル。

メモ付きの FoxPro および Visual FoxPro ファイル。

ファイル拡張子は、.dbf、.dbt、.fbt です。

DLM
出力データソースは、区切り文字で区切られたファイルです。
DELIMITER='*char*'を使用して、エクスポートするデータのタイプを指定できます。

ファイル拡張子は.*です。

デフォルト デフォルトの区切り文字は空白(スペース>です。

DTA
Stata ファイル。ファイル拡張子は.dta です。

JMP
出力データソースは、バージョン 7 以降の形式の JMP ファイルです。ファイル拡張子は.JMP です。

PARADOX

Paradox DB ファイル。ファイル拡張子は.db です。

SAV

SPSS ファイル、圧縮および非圧縮のバイナリファイル。ファイル拡張子は.sav です。

TAB

出力データソースはタブ区切りファイルです。ファイル拡張子は.txt です。

XLS

ファイル形式を使用する Microsoft Excel 97、2000、2002、または 2003 スプレッドシート。ファイル拡張子は.xls です。

XLSX

Excel 2007、2010、およびそれ以降のスプレッドシート(.xlsx ファイル形式を使用)。ファイル拡張子は.xlsx です。

トランスコーディングは、DBMS=XLS ではサポートされていません。.xls ファイルを.xlsx ファイルに保存し、DBMS=XLSX を使用してトランスコーディングをサポートします。

Linux で Excel ワークブックを読み書きするために、DBMS=XLSX と DBMS=XLS を指定できます。SAS PC Files Server を使用する必要はありません。

注: SAS は、サポートと機能強化のために.xlsx を使用することをお勧めします。

既存の Excel ワークブック.xls または.xlsx ファイルにエクスポートする場合、DBMS=XLS (または DBMS=XLSX)を使用できます。ただし、.bak ファイルは、DBMS=XLS または DBMS=XLSX が使用されている場合にのみ作成されます。

この表は、開いて読み取ることができるさまざまなバージョンの Microsoft Excel を使用して SAS が作成するファイルを示しています。

表 20.1 エクスポートされるデータ: Microsoft Excel ワークブックの可読性

ファイル拡張子	Excel 2007 以降	Excel97、2000、2002、2003
.xlsx	はい	いいえ
.xls	はい	はい

DBMS=XLSX で EXPORT プロシジャを使用し、ページに他のページを参照する数式がある場合、エラーメッセージがログに書き込まれます。このエラーは、他のページを参照する式が含まれているため、ページを置き換えることができないことを示しています。

この表は、Excel ファイルに使用する DBMS データソース識別子を指定しています。

表 20.2 DBMS 識別子

DBMS=	Linux
EXCEL	いいえ
XLS	はい
XLSX	はい

制限事項 .xlsx 形式を使用できるのは、Excel 2007 以降のみです。

出力外部データソースを使用できるかどうかは、動作環境、および場合によっては前の表で指定されているプラットフォームによって決まります。この可用性は、サイトに SAS/ACCESS Interface to PC Files のライセンスがあるかどうかにも依存します。ライセンスがない場合、区切り文字で区切られたファイルと JMP ファイルのみを使用できます。

例 "例 1: 区切り文字で区切られた外部データソースにエクスポートする" (652 ページ)

LABEL
変数ラベル名を指定します。SAS はこれらを、エクスポートされたテーブルの列名として書き込みます。ラベル名が存在しない場合、SAS はエクスポートされたテーブルに書き込みます。

REPLACE
既存のファイルを上書きします。REPLACE を指定しなければ、EXPORT プロシジャは既存のファイルを上書きしません。

例 "例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする" (657 ページ)

(*SAS data set options*)
SAS データセットオプションを指定します。たとえば、エクスポートするデータセットにパスワードが割り当てられている場合、ALTER=、PW=、READ=、または WRITE=データセットオプションを使用できます。指定された条件を満たすデータのサブセットをエクスポートするには、WHERE オプションを使用します。SAS データセットオプションの詳細については、*SAS データセットオプション: リファレンス*を参照してください。

例 "例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする" (657 ページ)

DBENCODING ステートメント

JMP ファイルにデータを保存するために使用するエンコーディングを示します。

操作: DBENCODING ステートメントは、DBMS=JMP の場合にのみ有効です。

構文

DBENCODING=*12-char SAS encoding-value*;

必須引数

12-char SAS encoding-value
JMP ファイルにデータを保存するために使用するエンコーディングを示します。エンコーディングによって、文字セットの各文字が一意の数値表現にマッピングされ、コードポイントのテーブルが構成されます。各文字は、エンコーディングごとに異なる数値表現がされます。この値には最大で 12 文字を使用できます。

DELIMITER ステートメント

出力ファイルでデータ列を区切る文字を指定します。

デフォルト: ブランクスペース

操作: DBMS=DLM を指定する場合は、DELIMITER ステートメントも指定する必要があります。

ヒント: *nn* を単一引用符または二重引用符で囲みます。

例: “例 1: 区切り文字で区切られた外部データソースにエクスポートする” (652 ページ)

構文

DELIMITER=*char*'' | '*nn*x;

必須引数

char | '*nn*'x
出力ファイルで値を区切るのために使用する区切り文字を指定します。区切り文字は、単一 BYTE 文字または 16 進値として指定できます。たとえば、列の値をアンパサンドで区切る場合は、DELIMITER='&'を指定します。DBCS 環境での表現に 2 バイトを必要とする単一文字は無効です。

FMTLIB ステートメント

フォーマットカタログで定義された SAS 出力形式の値を JMP ファイルの値ラベルに書き込みます。

操作: FMTLIB ステートメントは、DBMS=JMP の場合のみ有効です。

構文

FMTLIB=*<libref> format-catalog*;

必須引数

<libref.>format-catalog
JMP ファイルに書き込むフォーマットカタログを指定します。

META ステートメント

SAS メタデータ情報を JMP ファイルに書き込みます。(非推奨)

操作: META ステートメントは、DBMS=JMP の場合のみ有効です。

構文

META=*libref.member-data-set*;

必須引数

libref.member-data-set
JMP ファイルに書き込むメタデータ情報を含む SAS データセットを指定します。

META ステートメントはサポートされなくなったため、無視されます。代わりに、拡張属性が自動的に使用されます。SAS データセットを JMP にエクスポートすると、PROC EXPORT は SAS データセット上の拡張属性を探します。その拡張属性が存在すれば、プロシジャはその属性を使用して新しい JMP ファイルを作成します。

META ステートメントはプログラム内に残っている可能性がありますが、META が拡張属性に置換され、無視されることを示す NOTE をログに生成します。

PUTNAMES ステートメント

エクスポートするデータファイルの最初の行に列見出しとして SAS 変数名を書き込みます。

デフォルト: YES

制限事項: EXPORT プロシジャの場合にのみ有効です。

注: LABEL=オプションを指定すると、SAS 変数ラベル(変数名ではない)は列見出しとして書き込まれます。

例: “例 3: PUTNAMES=ステートメントを使用したタブで区切られたファイルへのエクスポート” (660 ページ)

構文

PUTNAMES=YES | NO;

必須引数

YES
EXPORT プロシジャで次のタスクが行われるように指定します。

- エクスポートするデータファイルの最初の行に列見出し(ヘディング)として SAS 変数名を書き込みます。
- エクスポート対象のデータファイルの 2 番目の行に、SAS データセットの最初の行を書き込みます。

NO
EXPORT プロシジャで、エクスポート対象のデータファイルの最初の行に、SAS データセット値の最初の行が書き込まれるように指定します。

例: EXPORT プロシジャ

例 1: 区切り文字で区切られた外部データソースにエクスポートする

要素: PROC EXPORT ステートメントオプション
DATA=

DBMS=
OUTFILE=
REPLACE

DELIMITER=ステートメント

詳細

この例では、Sashelp.Class データセットが、区切り文字で区切られた外部ファイルにエクスポートされます。次の例は、エクスポートされる前の Sashelp.Class データセットを示しています。

アウトプット 20.1　Sashelp.Class の PROC PRINT

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

プログラム

```
proc export data=sashelp.class
  outfile="c:\myfiles\class"
  dbms=dlm replace;

  delimiter='&';
run;
```

プログラムの説明

入力データセットを指定します。 ファイル名に拡張子は含まれません。
DBMS=DLM によって、出力ファイルは区切り文字で区切られたファイルであることが指定されています。

```
proc export data=sashelp.class
  outfile="c:\myfiles\class"
  dbms=dlm replace;
```

DELIMITER オプションで、出力ファイルでは&(アンパサンド)によってデータフィールドが区切られることが指定されています。

```
  delimiter='&';
run;
```

ログ

これは SAS ログの一部ですが、生成された SAS DATA ステップを含む、正常なエクスポート処理に関する情報が表示されています。

例のコード 20.1　区切り文字で区切られたファイルを作成する場合の SAS ログ

```
1   proc export data=sashelp.class outfile="c:\myfiles\class" dbms=dlm replace;
1 !                                          delimiter='&' ; run;

2    /*************************************************************************
3    *   PRODUCT:   SAS
4    *   VERSION:   9.3
5    *   CREATOR:   External File Interface
6    *   DATE:      31JAN11
7    *   DESC:      Generated SAS Datastep Code
8    *   TEMPLATE SOURCE:  (None Specified.)
9    *************************************************************************/
10      data _null_;
11      %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
12      %let _EFIREC_ = 0;     /* clear export record count macro variable */
13      file 'c:\myfiles\class' delimiter='&' DSD DROPOVER lrecl=32767;
14      if _n_ = 1 then        /* write column names or labels */
15       do;
16         put
17           "Name"
18         '&'
19           "Sex"
20         '&'
21           "Age"
22         '&'
23           "Height"
24         '&'
25           "Weight"
26         ;
27       end;
28     set  SASHELP.CLASS   end=EFIEOD;
29         format Name $8.;
30         format Sex $1.;
31         format Age best12.;
32         format Height best12.;
33         format Weight best12.;
34       do;
35         EFIOUT + 1;
36         put Name $ @;
37         put Sex $ @;
38         put Age @;
39         put Height @;
40         put Weight ;
41         ;
42       end;
43      if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro variable */
44      if EFIEOD then call symputx('_EFIREC_',EFIOUT);
45      run;

NOTE: The file 'c:\myfiles\class' is:
      Filename=c:\myfiles\class,
      RECFM=V,LRECL=32767,File Size (bytes)=0,
      Last Modified=31Jan2011:09:37:14,
      Create Time=31Jan2011:09:37:14
```

出力

EXPORT プロシジャによって、この外部ファイルが作成されます。

アウトプット 20.2　外部ファイル

```
Name&Sex&Age&Height&Weight
Alfred&M&14&69&112.5
Alice&F&13&56.5&84
Barbara&F&13&65.3&98
Carol&F&14&62.8&102.5
Henry&M&14&63.5&102.5
James&M&12&57.3&83
Jane&F&12&59.8&84.5
Janet&F&15&62.5&112.5
Jeffrey&M&13&62.5&84
John&M&12&59&99.5
Joyce&F&11&51.3&50.5
Judy&F&14&64.3&90
Louise&F&12&56.3&77
Mary&F&15&66.5&112
Philip&M&16&72&150
Robert&M&12&64.8&128
Ronald&M&15&67&133
Thomas&M&11&57.5&85
William&M&15&66.5&112
```

例 2: CSV ファイルにオブザベーションのサブセットをエクスポートする

要素:

PROC EXPORT ステートメントオプション
- DATA=
- DBMS=
- OUTFILE=
- REPLACE

詳細

この例では、SAS データセット Sashelp.Class が、区切り文字で区切られたファイルにエクスポートされます。

プログラム

エクスポートするデータセットを指定します。 WHERE オプションはオブザベーションのサブセットを要求します。OUTFILE=オプションによって出力ファイルが指定されます。DBMS=オプションによって、出力ファイルが CSV ファイルであり、存在する場合はそれによってターゲット CSV が上書きされることが指定されます。

```
proc export data=sashelp.class (where=(sex='F'))
    outfile="c:\myfiles\Femalelist.csv"
    dbms=csv
    replace;
run;
```

ログ

これは SAS ログの一部ですが、生成された SAS DATA ステップを含む、正常なエクスポート処理に関する情報が表示されています。

例のコード 20.2 CSV ファイルへのエクスポート

```
579  proc export data=sashelp.class (where=(sex='F'))
580      outfile="c:\myfiles\Femalelist.csv"
581      dbms=csv
582      replace;
583  run;

584  /************************************************************************
585  *   PRODUCT:   SAS
586  *   VERSION:   9.4
587  *   CREATOR:   External File Interface
588  *   DATE:      18APR14
589  *   DESC:      Generated SAS Datastep Code
590  *   TEMPLATE SOURCE:  (None Specified.)
591  ************************************************************************/
592     data _null_;
593     %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
594     %let _EFIREC_ = 0;     /* clear export record count macro variable */
595     file 'c:\myfiles\Femalelist.csv' delimiter=',' DSD DROPOVER lrecl=32767
595! ;
596     if _n_ = 1 then       /* write column names or labels */
597      do;
598        put
599          "Name"
600        ','
601          "Sex"
602        ','
603          "Age"
604        ','
605          "Height"
606        ','
607          "Weight"
608        ;
609      end;
610    set  SASHELP.CLASS(where=(sex='F'))   end=EFIEOD;
611        format Name $8.;
612        format Sex $1.;
613        format Age best12.;
614        format Height best12.;
615        format Weight best12.;
616      do;
617        EFIOUT + 1;
618        put Name $ @;
619        put Sex $ @;
620        put Age @;
621        put Height @;
622        put Weight ;
623        ;
624      end;
625     if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection
625! macro variable */
626     if EFIEOD then call symputx('_EFIREC_',EFIOUT);
627     run;

NOTE: The file 'c:\myfiles\Femalelist.csv' is:
      Filename=c:\myfiles\Femalelist.csv,
      RECFM=V,LRECL=32767,File Size (bytes)=0,
      Last Modified=18Apr2014:11:32:07,
      Create Time=18Apr2014:11:32:07
```

出力

EXPORT プロシジャによって、この外部 CSV ファイルが作成されます。

アウトプット 20.3　CSV ファイル

Name	Sex	Age	Height	Weight
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112

例 3: PUTNAMES=ステートメントを使用したタブで区切られたファイルへのエクスポート

要素:

PROC EXPORT ステートメントオプション
- DATA=
- DBMS=
- OUTFILE=
- PUTNAMES=
- REPLACE

詳細

この例では、SAS データセット WORK.INVOICE をタブで区切られたファイルにエクスポートする様子が示されています。最初のプログラムは PUTNAMES=ステートメントを指定して PROC EXPORT を使用していますが、2 番目のプログラムはそうではありません。このステートメントを使用すると、タブで区切られたファイルの列見出しにどのような影響が及ぼされるかを示しています。

次に、SAS データセット WORK.INVOICE がタブ区切りファイルにエクスポートされる前の状態を示します。

アウトプット 20.4　WORK.INVOICE の PROC PRINT

Obs	Invoice_ID	Billed_To	Amount_Billed_in_Local_Currency	Country	Amount_Billed_in_US_Dollars	Billed_By	Billed_On	Paid_On
1	11270	39045213	8738600640.00	Brazil	3675848666.21	239185	05OCT2004	18OCT2004
2	11271	18543489	11063836.00	USA	11063836.00	457232	05OCT2004	.
3	11273	19783482	252148.50	USA	252148.50	239185	06OCT2004	11NOV2004
4	11276	14324742	1934460.00	USA	1934460.00	135673	06OCT2004	20OCT2004
5	11278	14898029	1400825.00	USA	1400825.00	239185	08OCT2004	19OCT2004
6	11280	39045213	8738600640.00	Brazil	3675848666.21	423286	07OCT2004	20OCT2004
7	11282	19783482	252148.50	USA	252148.50	457232	07OCT2004	25OCT2004
8	11285	38763919	2234301.30	Argentina	772847.05	239185	10OCT2004	30NOV2004
9	11286	43459747	14936604.08	Australia	11366352.06	423286	10OCT2004	.
10	11287	15432147	252148.50	USA	252148.50	457232	11OCT2004	04NOV2004
11	12051	39045213	8738600640.00	Brazil	3675848666.21	457232	02NOV2004	.
12	12102	18543489	11063836.00	USA	11063836.00	239185	17NOV2004	.
13	12263	19783482	252148.50	USA	252148.50	423286	05DEC2004	.
14	12468	14898029	1400825.00	USA	1400825.00	135673	24DEC2004	02JAN2005
15	12471	39045213	8738600640.00	Brazil	3675848666.21	457232	27DEC2004	.
16	12476	38763919	2234301.30	Argentina	772847.05	135673	24DEC2004	.
17	12478	15432147	252148.50	USA	252148.50	423286	24DEC2004	02JAN2005

プログラム

```
PROC PRINT DATA=WORK.INVOICE;
RUN;

PROC EXPORT DATA=WORK.INVOICE
  OUTFILE="c:\temp\invoice_names.txt"
     DBMS=TAB REPLACE;
  PUTNAMES=YES;
RUN;
PROC PRINT;
RUN;

PROC EXPORT DATA=WORK.INVOICE
  OUTFILE="c:\temp\invoice_data_1st.txt"
     DBMS=TAB REPLACE;
  PUTNAMES=NO;
RUN;

PROC PRINT;
RUN;
```

プログラムの説明

EXPORT プロシジャで PUTNAMES=YES ステートメントを使用します。
WORK.INVOICE が出力された後、PUTNAMES=YES ステートメントを使用して、エクスポート対象の区切りファイル Invoice_names.txt の最初の行に、列名として

SAS 変数名を書き込みます。データの最初の行が、区切りファイルの 2 番目の行に書き込まれます。

```
PROC PRINT DATA=WORK.INVOICE;
RUN;

PROC EXPORT DATA=WORK.INVOICE
  OUTFILE="c:\temp\invoice_names.txt"
     DBMS=TAB REPLACE;
  PUTNAMES=YES;
RUN;
PROC PRINT;
RUN;
```

PUTNAMES=NO ステートメントの影響。 このステートメントを NO に設定すると、PROC EXPORT によって、エクスポート対象の区切りファイルの最初の行に、データの最初の行が書き込まれます。このため、SAS 変数名はスキップされ、列はラベルが設定されないままになります。

```
PROC EXPORT DATA=WORK.INVOICE
  OUTFILE="c:\temp\invoice_data_1st.txt"
    DBMS=TAB REPLACE;
  PUTNAMES=NO;
RUN;

PROC PRINT;
RUN;
```

SAS ログ

この SAS ログには、生成された SAS DATA ステップを含む、正常なエクスポート処理に関する情報が表示されています。ログは複数のセクションに分かれていますが、これはドキュメント内での見やすさを考慮しただけです。

```
490  PROC EXPORT DATA= WORK.INVOICE
491            OUTFILE= "c:\temp\invoice_names.txt"
492            DBMS=TAB REPLACE;
493       PUTNAMES=YES;
494  RUN;

495  /*************************************************************************
496  *   PRODUCT:   SAS
497  *   VERSION:   9.4
498  *   CREATOR:   External File Interface
499  *   DATE:      24MAY14
500  *   DESC:      Generated SAS Datastep Code
501  *   TEMPLATE SOURCE:  (None Specified.)
502  *************************************************************************/
503     data _null_;
504     %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
505     %let _EFIREC_ = 0;     /* clear export record count macro variable */
506     file 'c:\temp\invoice_names.txt' delimiter='09'x DSD DROPOVER lrecl=32767;
507     if _n_ = 1 then        /* write column names or labels */
508      do;
509        put
510          "INVNUM"
511        '09'x
512          "BILLEDTO"
513        '09'x
514          "AMTBILL"
515        '09'x
516          "COUNTRY"
517        '09'x
518          "AMTINUS"
519        '09'x
520          "BILLEDBY"
521        '09'x
522          "BILLEDON"
523        '09'x
524          "PAIDON"
525        ;
526      end;
527     set  WORK.INVOICE   end=EFIEOD;
528        format INVNUM best12.;
529        format BILLEDTO $8.;
530        format AMTBILL dollar18.2 ;
531        format COUNTRY $20.;
532        format AMTINUS dollar18.2 ;
533        format BILLEDBY best12.;
534        format BILLEDON date9.;
535        format PAIDON date9.;
536      do;
537        EFIOUT + 1;
538        put INVNUM @;
539        put BILLEDTO $ @;
540        put AMTBILL @;
541        put COUNTRY $ @;
542        put AMTINUS @;
543        put BILLEDBY @;
544        put BILLEDON @;
545        put PAIDON ;
546        ;
547      end;
548     if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro variable */
549     if EFIEOD then call symputx('_EFIREC_',EFIOUT);
550     run;
```

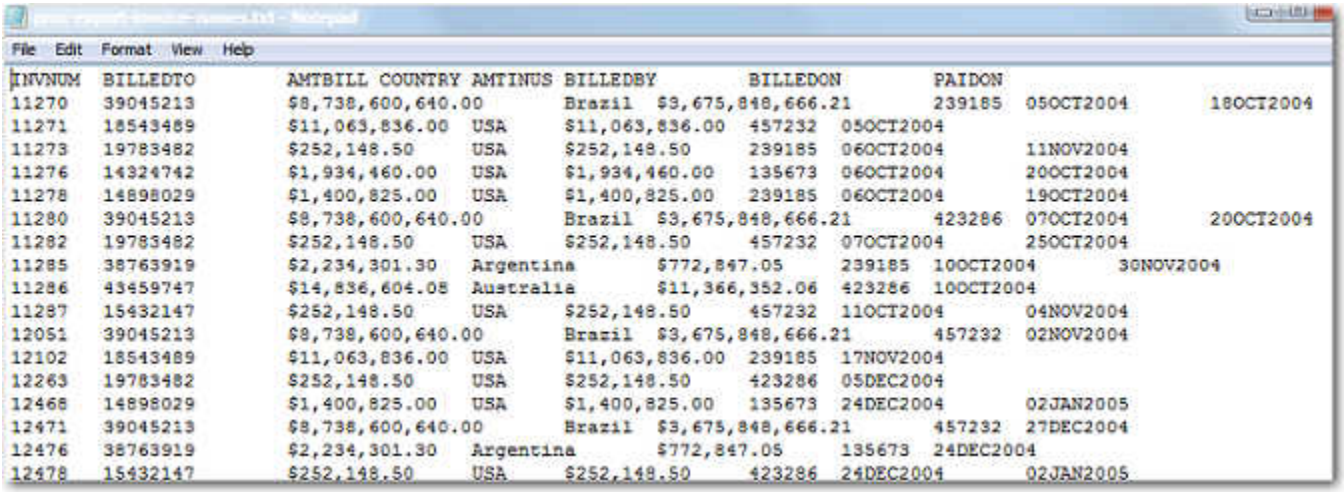
```
NOTE: The file 'c:\temp\invoice_names.txt' is:
      Filename=c:\temp\invoice_names.txt,
      RECFM=V,LRECL=32767,File Size (bytes)=0,
      Last Modified=24May2014:15:46:36,
      Create Time=24May2014:15:46:36

NOTE: 18 records were written to the file 'c:\temp\invoice_names.txt'.
      The minimum record length was 60.
      The maximum record length was 84.
NOTE: There were 17 observations read from the data set WORK.INVOICE.
NOTE: DATA statement used (Total process time):
      real time          0.04 seconds
      cpu time           0.03 seconds

17 records created in c:\temp\invoice_names.txt from WORK.INVOICE.

NOTE: "c:\temp\invoice_names.txt" file was successfully created.
NOTE: PROCEDURE EXPORT used (Total process time):
      real time          0.20 seconds
      cpu time           0.17 seconds

551  PROC PRINT; RUN;

NOTE: No observations in data set SASUSER.SASMBC.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds

552
553
554  PROC EXPORT DATA= WORK.INVOICE
555              OUTFILE= "c:\temp\invoice_data_1st.txt"
556              DBMS=TAB REPLACE;
557       PUTNAMES=NO;
558  RUN;
```

```
559  /*************************************************************************
560  *   PRODUCT:   SAS
561  *   VERSION:   9.4
562  *   CREATOR:   External File Interface
563  *   DATE:      24MAY14
564  *   DESC:      Generated SAS Datastep Code
565  *   TEMPLATE SOURCE:  (None Specified.)
566  *************************************************************************/
567     data _null_;
568     %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
569     %let _EFIREC_ = 0;     /* clear export record count macro variable */
570     file 'c:\temp\invoice_data_1st.txt' delimiter='09'x DSD DROPOVER lrecl=32767;
571    set  WORK.INVOICE   end=EFIEOD;
572        format INVNUM best12.;
573        format BILLEDTO $8.;
574        format AMTBILL dollar18.2 ;
575        format COUNTRY $20.;
576        format AMTINUS dollar18.2 ;
577        format BILLEDBY best12.;
578        format BILLEDON date9.;
579        format PAIDON date9.;
580      do;
581        EFIOUT + 1;
582        put INVNUM @;
583        put BILLEDTO $ @;
584        put AMTBILL @;
585        put COUNTRY $ @;
586        put AMTINUS @;
587        put BILLEDBY @;
588        put BILLEDON @;
589        put PAIDON ;
590        ;
591      end;
592     if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro variable */
593     if EFIEOD then call symputx('_EFIREC_',EFIOUT);
594     run;

NOTE: The file 'c:\temp\invoice_data_1st.txt' is:
      Filename=c:\temp\invoice_data_1st.txt,
      RECFM=V,LRECL=32767,File Size (bytes)=0,
      Last Modified=24May2014:15:46:36,
      Create Time=24May2014:15:46:36

NOTE: 17 records were written to the file 'c:\temp\invoice_data_1st.txt'.
      The minimum record length was 60.
      The maximum record length was 84.
NOTE: There were 17 observations read from the data set WORK.INVOICE.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time           0.03 seconds

17 records created in c:\temp\invoice_data_1st.txt from WORK.INVOICE.

NOTE: "c:\temp\invoice_data_1st.txt" file was successfully created.
NOTE: PROCEDURE EXPORT used (Total process time):
      real time          0.07 seconds
      cpu time           0.07 seconds

595
596  PROC PRINT; RUN;

NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

出力例

PROC EXPORT PUTNAMES=YES ステートメントを使用して、SAS 変数名はタブで区切られたファイルの列見出しにマッピングされます。これはデフォルトの動作です。

アウトプット 20.5　PUTNAMES=YES ステートメントを使用してタブで区切られたファイルにエクスポートされる SAS データ



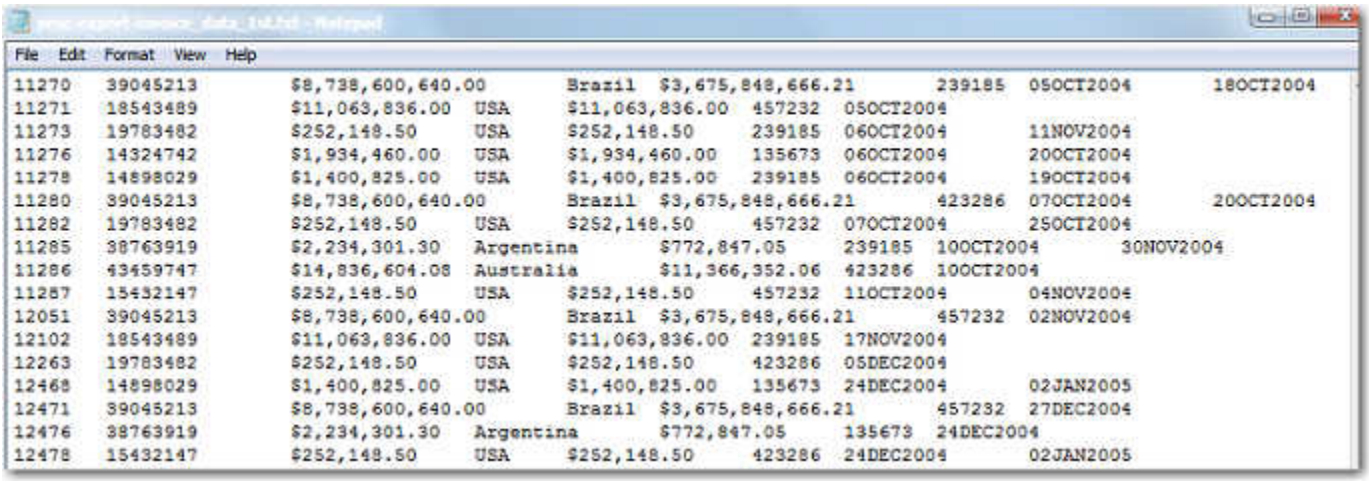
File Edit Format View Help

INVNUM	BILLEDTO	AMTBILL	COUNTRY	AMTINUS	BILLEDBY	BILLEDON	PAIDON		
11270	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	239185	05OCT2004	18OCT2004
11271	18543489	$11,063,836.00	USA		$11,063,836.00	457232	05OCT2004		
11273	19783482	$252,148.50	USA		$252,148.50	239185	06OCT2004	11NOV2004	
11276	14324742	$1,934,460.00	USA		$1,934,460.00	135673	06OCT2004	20OCT2004	
11278	14898029	$1,400,825.00	USA		$1,400,825.00	239185	06OCT2004	19OCT2004	
11280	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	423286	07OCT2004	20OCT2004
11282	19783482	$252,148.50	USA		$252,148.50	457232	07OCT2004	25OCT2004	
11285	38763919	$2,234,301.30	Argentina		$772,847.05	239185	10OCT2004	30NOV2004	
11286	43459747	$14,836,604.08	Australia		$11,366,352.06	423286	10OCT2004		
11287	15432147	$252,148.50	USA		$252,148.50	457232	11OCT2004	04NOV2004	
12051	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	457232	02NOV2004	
12102	18543489	$11,063,836.00	USA		$11,063,836.00	239185	17NOV2004		
12263	19783482	$252,148.50	USA		$252,148.50	423286	05DEC2004		
12468	14898029	$1,400,825.00	USA		$1,400,825.00	135673	24DEC2004	02JAN2005	
12471	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	457232	27DEC2004	
12476	38763919	$2,234,301.30	Argentina		$772,847.05	135673	24DEC2004		
12478	15432147	$252,148.50	USA		$252,148.50	423286	24DEC2004	02JAN2005	

出力例

PROC EXPORT PUTNAMES=NO ステートメントを使用すると、タブ区切りのファイルに名前のない列が含まれます。

アウトプット 20.6　PUTNAMES=NO ステートメントを使用してタブで区切られたファイルにエクスポートされる SAS データ



File Edit Format View Help

11270	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	239185	05OCT2004	18OCT2004
11271	18543489	$11,063,836.00	USA		$11,063,836.00	457232	05OCT2004		
11273	19783482	$252,148.50	USA		$252,148.50	239185	06OCT2004	11NOV2004	
11276	14324742	$1,934,460.00	USA		$1,934,460.00	135673	06OCT2004	20OCT2004	
11278	14898029	$1,400,825.00	USA		$1,400,825.00	239185	06OCT2004	19OCT2004	
11280	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	423286	07OCT2004	20OCT2004
11282	19783482	$252,148.50	USA		$252,148.50	457232	07OCT2004	25OCT2004	
11285	38763919	$2,234,301.30	Argentina		$772,847.05	239185	10OCT2004	30NOV2004	
11286	43459747	$14,836,604.08	Australia		$11,366,352.06	423286	10OCT2004		
11287	15432147	$252,148.50	USA		$252,148.50	457232	11OCT2004	04NOV2004	
12051	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	457232	02NOV2004	
12102	18543489	$11,063,836.00	USA		$11,063,836.00	239185	17NOV2004		
12263	19783482	$252,148.50	USA		$252,148.50	423286	05DEC2004		
12468	14898029	$1,400,825.00	USA		$1,400,825.00	135673	24DEC2004	02JAN2005	
12471	39045213	$8,738,600,640.00			Brazil	$3,675,848,666.21	457232	27DEC2004	
12476	38763919	$2,234,301.30	Argentina		$772,847.05	135673	24DEC2004		
12478	15432147	$252,148.50	USA		$252,148.50	423286	24DEC2004	02JAN2005	

21

FCMP プロシジャ

概要: FCMP プロシジャ

FCMP プロシジャの機能

SAS 関数コンパイラ(FCMP)プロシジャを使用すると、SAS 関数や CALL ルーチンやサブルーチンを、他の SAS プロシジャや DATA ステップで使う前に、作成、テスト、そして保存できます。PROC FCMP ではデータセットに保存される DATA ステップシンタックスを使って関数や CALL ルーチンやサブルーチンを作成できます。プロシジャは DATA ステップステートメントの少々のバリエーションは受け入れ、PROC FCMP で作成する関数や CALL ルーチンには SAS プログラム言語のほとんどの関数を使用できます。他の SAS 関数や CALL ルーチンやサブルーチンを呼ぶのと変わらず、DATA ステップから PROC FCMP 関数と CALL ルーチンを呼ぶことができます。これによりプログラマは独立していて再利用可能なサブルーチンで、複雑なコードも容易に読めて、書けて、メンテナンスすることができます。PROC FCMP ルーチンは、保存場所にアクセス可能であれば、どの DATA ステップまたは SAS プロシジャでも使用できます。

FCMP プロシジャでは、SAS 言語コンパイラを使用して SAS プログラムをコンパイルし実行します。このコンパイラーサブシステムは、SAS が実行されているコンピューターのマシン言語コードを生成します。CMPOPT オプションで値を指定することで、マシン言語コードが効率的に実行されるように最適化できます。SAS 言語コンパイラで使用するこの種のコード作成最適化の詳細については、“CMPOPT= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

PROC FCMP で作成した関数やサブルーチンは、DATA ステップや WHERE ステートメント、Output Delivery System (ODS)、および次のプロシジャで使用できます。

PROC CALIS	PROC OPTMODEL
PROC FORMAT	PROC PHREG

PROC GA	PROC QUANTREG
PROC GENMOD	PROC REPORT COMPUTE ブロック
PROC GLIMMIX	SAS リスクディメンションプロシジャ
PROC MCMC	PROC SEVERITY
PROC MODEL	PROC SIMILARITY
PROC NLIN	PROC SQL (配列引数を使う関数はサポートしていません)
PROC NLMIXED	PROC SURVEYPHREG
PROC NLP	PROC TMODEL
PROC OPTLSO	PROC VARMAX

概念: FCMP プロシジャ

関数とサブルーチンの作成

PROC FCMP では、DATA ステップシンタックスを使って関数や CALL ルーチンやサブルーチンを作成できます。PROC FCMP 関数と CALL ルーチンは、データセットに保存され、DATA ステップからだけでなく、NLIN、MODEL、および NLP プロシジャといった、複数の SAS/STAT、SAS/ETS、または SAS/OR プロシジャからも呼び出すことができます。単一の FCMP プロシジャステップで複数の関数と CALL ルーチンを作成できます。

関数は他のプログラミング言語で使用されるルーチンと同等です。それらは省略可能な引数(複数可)を持つ、個々に独立した計算ブロックです。サブルーチンは戻り値が任意の特別な種類の関数です。関数またはサブルーチンブロック内で作成されたすべての変数はそのサブルーチンのローカルとなります。

関数とサブルーチンの作成: 例

次の例では関数とサブルーチンを定義します。関数は FUNCTION ステートメントで始まり、サブルーチンは SUBROUTINE ステートメントで始まります。DAY_DATE

関数は、日付を数値の曜日に変換し、INVERSE サブルーチンは単一の逆関数を計算します。最後は ENDSUB ステートメントで終わります。

```
proc fcmp outlib=work.MySubs.Fncs;
 function day_date(indate);
  wkday=weekday(indate);
  return(wkday);
 endsub;

 subroutine inverse(in, inv);
  outargs inv;
  if in=0
    then inv=.;
    else inv=1/in;
  endsub;
quit;

option cmplib=work.MySubs;

data _null_;
daysDate=day_date(today());
put daysDate=;

CALL inverse(12, inverseValue);
put inverseValue=;
run;
```

これらのステートメントは、次の結果を生成します。

```
daysDate=2
inverseValue=0.0833333333
```

注: today()の値に基づいて出力が変化します。

関数とサブルーチンは、DATA ステップ構文の後に続きます。現在の FCMP プロシジャステップですでに定義されている関数とサブルーチン、および DATA ステップ関数のほとんどは、これらのルーチン内から呼び出すことができます。上記の例で、DATA ステップ関数 WEEKDAY は、DAY_DATE によって呼び出されます。

本例のルーチンは、MathFncs というパッケージ内の work.MySubs データセットに保存されます。パッケージは、ユーザーによって指定される関連ルーチンの集まりです。このように、データセット内の関連するサブルーチンと関数がグループ化されます。PROC FCMP は、PROC FCMP ステートメントの OUTLIB=オプションによりコンパイルするサブルーチンの保存場所を認識し、LIBRARY=オプションによりライブラリの読み込み場所(C または SAS)を認識します。

注: 関数名とサブルーチン名は、1 つのパッケージ内で一意である必要があります。ただし、異なるパッケージに、同じ名前のサブルーチンと関数を含めることはできます。あいまいさがある場合に特定のルーチンを選択するには、サブルーチン名の接頭辞としてパッケージ名と期間を使用します。たとえば、INVERSE の MthFncs バージョンにアクセスするには、MthFncs.inverse を使用します。

独自の関数の作成

独自の関数と CALL ルーチンを作成する利点

PROC FCMP では、DATA ステップシンタックスを使って関数や CALL ルーチンやサブルーチンを作成できます。独自の関数と CALL ルーチンを作成することの利点には、次のことがあります。

- 関数や CALL ルーチンによって、プログラムは読みやすく、書きやすく、変更しやすくなります。
- 関数や CALL ルーチンは独立しています。ルーチンを呼ぶプログラムは、ルーチンの実装には影響されません。
- 関数や CALL ルーチンは再利用できます。関数やルーチンが保存されているデータセットにアクセスできれば、どんなプログラムでもルーチンを呼び出せます。

注: 作成する PROC FCMP ルーチンは存在するビルトイン SAS 関数と同じ名前は持てません。もしも名前が同じであれば、SAS は同じ名前のビルトイン SAS 関数またはサブルーチンが存在していますというエラーメッセージを生成します。

ユーザー定義の関数の作成

次のプログラムでは、DATA ステップから PROC FCMP 関数を作成および呼ぶのに使うシンタックスを示します。この例では、治験期間の実験日数を計算します。

この例では、TRIAL という名前のパッケージの中に STUDY_DAY という名前の関数を作成します。パッケージとは、固有名を持ち、work.Funcs データセットに保存されているルーチンの集合です。STUDY_DAY は *intervention_date* と *event_date* の 2 個の数値引数を持てます。ルーチン内では、DATA ステップの構文を使って次の 2 つの日付の差を計算します。*intervention_date* より前の日数は-1 から始まって減少し、*intervention_date* 以降の日付は 1 から始まり増加します。この関数は実験日数に 0 を返すことはありません。

STUDY_DAY は DATA ステップコードから他の関数と何も変りなく呼ぶことができます。DATA ステップで STUDY_DAY への呼び出しが発生すると、この関数は関数の従来のライブラリで検出されません。SAS は STUDY_DAY を含むパッケージを探すのに、CMPLIB=システムオプションに指定されているライブラリやデータセットを検索します。この例では、STUDY_DAY は work.Funcs.Trial にあります。プログラムは関数を呼び、*start_date* と *end_date* の変数値を渡し、変数 **sd** で結果を返します。

```
proc fcmp outlib=work.funcs.trial;
  function study_day(intervention_date, event_date);
```

```
    n=event_date-intervention_date;
    if n <= 0 then
    n=n+1;
    return(n);
  endsub;

options cmplib=work.funcs;
data _null_;
  start_date='15Feb2006'd;
  end_date='27Mar2006'd;
  sd=study_day(start, today);
  put sd=;
run;
```

例のコード 21.1　ユーザー定義関数 STUDY_DAY の結果

```
sd=40
```

ライブラリオプションの使用

PROC FCMP は OUTLIB=または INLIB=オプションと一緒に使用できます。このプロシジャのシンタックスは次の形式になります。

```
proc fcmp
outlib=libname.dataset.package
       inlib=in-libraries;
routine-declarations;
```

OUTLIB=オプションは必須で、*routine-declarations* セクションで宣言されたルーチンが保存されているパッケージを指定します。

routine-declarations セクションで宣言されたルーチンは、他のパッケージにある FCMP ルーチンを呼ぶこともできます。ルーチンを探して、そして呼び出しの妥当性を確認するために、SAS は INLIB=オプションで指定されているデータセットを検索します。INLIB=オプションの形式は次のようになります。

```
inlib=library.dataset
inlib=(library1.dataset1 library2.dataset2 ... libraryN.datasetN)
inlib=library.datasetM - library.datasetN
```

宣言されているルーチンが他のパッケージの FCMP ルーチンを呼ばない場合は、INLIB=オプションを指定する必要はありません。

関数の宣言

プログラムの *routine-declarations* セクションに、関数(複数可)と CALL ルーチンを宣言します。ルーチンには 4 つのパーツがあります。

- 名前
- パラメーター(複数可)

- コードの本体
- RETURN ステートメント

これらのキーワードを FUNCTION または SUBROUTINE キーワードと ENDSUB キーワードの間に指定します。関数のシンタックスは次の形式になります。

```
function
name(argument-1 <, argument-2, ...>);
  program-statements;
  return(expression);
endsub;
```

FUNCTION キーワードの後に、関数名と引数を指定します。関数の宣言の引数は仮引数とよばれ、関数の本体で使用できます。文字列引数を指定するには、引数名の後にドルマーク($)を付加してください。関数では、すべての引数は値で渡されます。つまり、実際の引数の値、変数または呼び出し側の環境から関数に渡される値は、関数がそれを使用する前にコピーされます。コピーすることによって、関数による仮引数の変更がオリジナルの値から変更していないと確認できます。

RETURN ステートメントは関数に値を返すのに使用されます。RETURN ステートメントはかっこで囲まれた表現を受け入れ、そして呼び出し側の環境に返す値を保持しています。関数の宣言は ENDSUB ステートメントで終わります。

CALL ルーチンの宣言

CALL ルーチンは、FUNCTION キーワードのかわりに SUBROUTINE キーワードを使って *routine-declarations* 内で宣言されます。関数と CALL ルーチンは同じ型ですが、CALL ルーチンは値を返さず、また CALL ルーチンではパラメーターを変更できる点が違います。OUTARGS ステートメントで変更する引数を指定します。CALL ルーチンの宣言のシンタックスは次のようになります。

```
subroutine
name(<argument-1 , argument-2, ...>);
  outargs <out-argument-1 , out-argument-2, ...>;
  program-statements;
  return;
endsub;
```

OUTARGS ステートメントにある仮引数は、値ではなく参照として渡されます。つまり、CALL ルーチンによる仮引数の変更により、渡された元の変数が変更されるということです。また、CALL ルーチンの起動時に値がコピーされません。コピー数を減らすことで、CALL ルーチンと呼び出し環境の間で大量のデータを渡すときのパフォーマンスを向上させることができます。

RETURN ステートメントは、CALL ルーチンの定義内のオプションです。RETURN ステートメントの実行時、実行はすぐに呼び出し側に返されます。CALL ルーチン内の RETURN ステートメントは、値を返しません。

プログラムステートメントの作成

プログラムのプログラムステートメントセクションは、関数または CALL ルーチンによって実行される作業を説明する、一連の DATA ステップおよび/または FCMP ス

テートメントです。ほとんどの DATA ステップステートメントと関数は、PROC FCMP ルーチンからアクセス可能です。DATA ステップファイルとデータセット I/O ステートメント(INPUT、FILE、SET、MERGE など)は、PROC FCMP ルーチンから使用できません。ただし、PUT ステートメントの一部の機能はサポートされています。詳細については、“PROC FCMP と DATA ステップの相違点” (695 ページ)を参照してください。

出力形式としての関数の使用

PROC FCMP により、関数を使用して、最初に値の関数を実行することにより値をフォーマットできます。値をフォーマットする関数を使用して、カスタマイズされた出力形式を作成できます。

PROC FCMP との DATA ステップステートメントの使用

DATA ステップステートメントを PROC FCMP と使用できます。ただし、PROC FCMP に対する構文と機能にはいくつかの相違点があります。詳細については、“PROC FCMP と DATA ステップの相違点” (695 ページ)を参照してください。

DROP、KEEP、FORMAT、LENGTH ステートメントの動作は、PROC FCMP と DATA ステップで同じです。

次の DATA ステップステートメントは、PROC FCMP でサポートされていません。

- DATA
- SET
- MERGE
- UPDATE
- MODIFY
- INPUT
- INFILE

FILE ステートメントのサポートは、PROC FCMP の LOG 出力先および PRINT 出力先に制限されています。OUTPUT ステートメントは PROC FCMP でサポートされていますが、関数またはサブルーチン内ではサポートされていません。

次のステートメントは PROC FCMP でサポートされていますが、DATA ステップではサポートされていません。

- FUNCTION
- STRUCT
- SUBROUTINE
- OUTARGS

構文: FCMP プロシジャ

PROC FCMP *options*;

ABORT ;

ARRAY *array-name* [*dimensions*] < /NOSYMBOLS> |<*variable(s)*>| <*constant(s)*>|<*initial-values)*>;

ATTRIB *variable(s)*< FORMAT=*format-name* > < LABEL='*label*'> < LENGTH=*length* > ;

DELETEFUNC *function-name* ;

DELETESUBR *subroutine-name* ;

FUNCTION *function-name(argument(s))*< VARARGS> < $> <*length*> <KIND | GROUP='*string*' < LABEL='*string-2*'>> ;

LABEL *variable* ='*label*' ;

LISTFUNC *function-name* ;

LISTSUBR *subroutine-name* ;

STRUCT *structure-name variable* ;

SUBROUTINE *subroutine-name (argument(s))*< VARARGS> <LABEL='*label*'> <KIND | GROUP='*string*'>;

OUTARGS *out-argument(s)* ;

ステートメント	タスク	例
“PROC FCMP ステートメント”	その他の SAS プロシジャが使用するための SAS 関数を作成、テスト、保存します	Ex. 1, Ex. 2, Ex. 8, Ex. 7
“ABORT ステートメント”	現在の DATA ステップ、SAS ジョブまたは SAS セッションの実行を終了します	
“ARRAY ステートメント”	名前を変数および定数のリストと関連付けます	Ex. 8
“ATTRIB ステートメント”	変数の出力形式、ラベル、長さ情報を指定します	
“DECLARE ステートメント”	オブジェクトを宣言します。	
“DELETEFUNC ステートメント”	OUTLIB オプションで指定される関数ライブラリから関数を削除します	
“DELETESUBR ステートメント”	OUTLIB オプションで指定される関数ライブラリからサブルーチンを削除します	
“FUNCTION ステートメント”	変更された変数値を返します	Ex. 1, Ex. 2, Ex. 7

ステートメント	タスク	例
“LABEL ステートメント”	変数のラベルを指定します	
	文字変数や数値変数を格納する場合はバイト数を、VARCHAR 変数を格納する場合は文字数を指定します。	
“LISTFUNC ステートメント”	SAS リスティングの関数のソースコードを書き込みます	
“LISTSUBR ステートメント”	SAS リスティングのサブルーチンに対するソースコードを書き込みます	
“STATIC ステートメント”	変数が再度割り当てられるまで、前の呼び出しからの変数の値が保持されます。	
“STRUCT ステートメント”	構造の種類を宣言(作成)します	
“SUBROUTINE ステートメント”	コードの独立計算ブロックを宣言(作成)します	Ex. 8
“OUTARGS ステートメント”	(SUBROUTINE ステートメントとのみ使用します。)サブルーチンが更新する必要がある引数リストから引数を指定します	Ex. 2, Ex. 8

PROC FCMP ステートメント

SAS 関数、CALL ルーチン、サブルーチンを作成、テスト、保存します。

例:
- “例 1: 関数の作成と、DATA ステップからの関数の呼び出し” (728 ページ)
- “例 2: PROC FCMP による関数の作成および保存” (730 ページ)
- “例 7: ユーザー定義関数での Graph Template Language (GTL)の使用” (735 ページ)
- “例 8: データセットの各行を標準化する” (738 ページ)

構文

PROC FCMP *options*;

オプション引数

DATA=*filename*
入力データセットを PROC FCMP ステップに読み込みます。

注: DATA オプションは、PROC FCMP ステップで定義された関数またはサブルーチンに入力を送信できます。PROC FCMP ステップは、各観測を反復処理し、各反復中に関数またはサブルーチンを呼び出します。

注: DATA=オプションおよび OUT=オプションの詳細については、“Data セットの入力および出力” (696 ページ)を参照してください。

注: この例では、PROC FCMP ステートメントで DATA=および OUT=オプションを使用する方法を示しています。

例
```
proc fcmp outlib=work.funcs.trial;
  function priceWithTax_func(price);
   static taxRate;
    if (taxRate EQ .) then
      taxRate = (7.25 / 100);
   taxedPrice = (price * (1 + taxRate));
    return (taxedPrice);
  endsub;
run;

option CMPLIB=work.funcs;

proc fcmp data=sashelp.cars out=work.carPriceWithTax;
  format msrpWithTax DOLLAR8.;
  msrpWithTax = priceWithTax_func(msrp);
run;

proc print data=work.carPriceWithTax(obs=5 keep=make model
invoice msrp msrpWithTax);
run;
```

ENCRYPT
データセットのソースコードをエンコードするように指定します。

別名 HIDE

FLOW
プログラムの実行時にプログラムのステートメントごとのメッセージの印刷を指定します。このオプションにより、拡張出力が生成されます。

GETCASCMPLIB

別名 GETCMPLIB

注 別名 GETCMPLIB は 2024.05 から使用可能です。

CAS および特定の環境の cmplib 設定を表示するように指定します。結果は、SAS ログに書き込まれます。このオプションは 2024.02 以降でサポートされています。

GETCASCMPOPT

別名 GETCMPOPT

注　別名 GETCMPOPT は 2024.05 から使用可能です。

CAS および特定の環境の cmpopt 設定を表示するように指定します。結果は、SAS ログに書き込まれます。このオプションは 2024.02 以降でサポートされています。

LIBRARY=*library.dataset*
LIBRARY=(*library-1.dataset library-2.dataset ... library-n.dataset)*
LIBRARY=*library.datasetM - library.datasetN*
前にコンパイルされたライブラリがプログラムにリンクされるように指定します。これらのライブラリは、前の PROC FCMP ステップによって、または PROC PROTO (外部 C ルーチン)を使用することによって作成されます。

別名　INLIB

ヒント　ライブラリは OUTLIB=オプションによって作成され、種類が CMPSUB の SAS ライブラリのメンバーとして保存されます。LIBRARY=オプションの使用時は、サブルーチンと関数のみプログラムに読み込まれます。

宣言されているルーチンがその他のパッケージの PROC FCMP ルーチンを呼び出さない場合、LIBRARY=オプションを指定する必要はありません。*libref.dataset* 出力形式を使用して、ライブラリの 2 レベルの名前を指定します。名前 *libref* と *dataset* は、8 文字未満の有効な SAS 名である必要があります。

LIBRARY=オプションを使用してファイルのリストを指定できます。また、数値接尾辞を使用して、名前の範囲を指定できます。複数のファイルを指定する場合、単一の名前範囲の場合を除き、リストをかっこで囲む必要があります。構文の例は次のとおりです。

```
proc fcmp library=work.exsubs

                        ;
proc fcmp library=(work.exsubs

                         work.examples

                        );
proc fcmp library=lib1-lib10

                        ;
```

LIST
LISTSOURCE と LISTPROG オプションが両方ともに有効になるように指定します。

ヒント　割り当てが正しくコンパイルされたことを確認する 1 つの方法として、ソースコードとコンパイル済コードを印刷してから、割り当てステートメントの 2 つのリストを比較します。

LISTALL
LISTCODE、LISTPROG、LISTSOURCE オプションが有効になるように指定します。

LISTCODE
コンパイル済プログラムコードが印刷されるように指定します。LISTCODE には、コンパイラによって生成される一連の操作が表示されます。

ヒント LISTCODE 出力は読みにくいため、LISTPROG オプションを使用してより読みやすいコンパイル済プログラムコードのリストを取得します。

LISTFUNCS
表示可能なすべての FCMP 関数に対するプロトタイプ、またはサブルーチンが SAS リストに書き込まれるように指定します。

LISTPROG
コンパイル済プログラムが印刷されるように指定します。割り当てステートメントのリストが、コンパイラによって生成される一連の操作から生成されます。ソースステートメントテキストは、その他のステートメントに対して印刷されます。

ヒント LISTPROG オプションによって印刷される式は、その式が実際に計算される方法を必ずしも示してはいません。共通の部分式の中間結果が再利用可能ためです。ただし、式は LISTPROG オプションによる拡張出力形式で印刷されます。式が実際にどのように計算されるかについては、LISTCODE オプションからのリストを参照してください。

LISTSOURCE
プログラムのソースコードステートメントが印刷されるように指定します。

OUT=*filename*
出力データセットを作成します。

OUTFILE=*filename*
参照された関数とメインプログラムをテキストファイルに書き込みます。マクロ変数を含む PROC FCMP によって解析されたプログラムをエクスポートできます。

OUTITEMSTORE=*path name*
記号、参照された関数、およびメインプログラムを、指定アイテムストアにエクスポートされます。OUTITEMSTORE はファイル参照名をサポートしていません。引用符で囲んだパス名を使用する必要があります。

OUTLIB=*libname.dataset.package*
PROC FCMP ステップの終了時にコンパイル済サブルーチンと関数が書き込まれる出力データセットの 3 レベルの名前を指定します。この引数は必須です。構文の例は次のとおりです。

```
proc fcmp outlib=work.fcmpsubs.pkt1

                ;

proc fcmp outlib=work.mysubs.math

                ;
```

ヒント このオプションは、サブルーチンと関数を出力ライブラリに保存する場合に使用します。

現在の PROC FCMP ステップ内で宣言されるサブルーチンのみが出力ファイルに保存されます。LIBRARY=オプションを使用してロードされるサブルーチンは、出力ファイルに保存されません。OUTLIB=オプションを指定しない場合、現在の PROC FCMP ステップで宣言されるサブルーチンは保存されません。

PRINT
プログラムの実行時にプログラムのステートメントごとの結果の印刷を指定します。このオプションにより、拡張出力が生成されます。

SETCASCMPLIB="*library-1.dataset library-2.dataset library-n.dataset*"

別名 SETCMPLIB

注 別名 SETCMPLIB は 2024.05 から使用可能です。

CAS および特定の環境のコンパイル時に含めるコンパイラーサブルーチンを含む 1 つ以上の SAS データセットを指定します。このオプションは 2024.02 以降でサポートされています。

SETCASCMPOPT="*option-1 option-2 option-n*"

別名 SETCMPOPT

注 別名 SETCMPOPT は 2024.05 から使用可能です。

CAS および特定の環境に対して 1 つ以上のコード生成の最適化を指定します。このオプションは 2024.02 以降でサポートされています。コード生成の最適化のリストについては、 CMPOPT を参照してください。

TRACE
プログラムの実行時にプログラムの各ステートメントの各操作の結果の印刷を指定します。これらの結果は、FLOW オプションによって印刷される情報に加えて生成されます。TRACE オプションは、拡張出力を生成します。

注 TRACE を指定することは、FLOW、PRINT、PRINTALL を指定することと同じです。

TRACE オプションは、PROC FCMP にデータが含まれている場合に機能し、DATA ステップから FCMP 関数を呼び出すときには機能しません。この例と出力は、TRACE オプションの機能を示しています。

例
```
proc fcmp outlib=work.funcs.trial TRACE;
  function study_day(intervention_date, event_date);
  n=event_date - intervention_date;
  if n >= 0 then n=n + 1;
    return(n);
  endsub;
  start='15Feb2010'd;
  today='27Mar2010'd;
  sd=study_day(start, today);
run;
```

```
                        The FCMP Procedure

--- Program Execution Starting.
   1    1 (20:4)    Executing Stmt          : ASSIGN start =
```

```
   1     (20:9)    start = 18308
   1    2 (21:4)    Executing Stmt          : ASSIGN today =
   1     (21:9)    today = 18348
   1    3 (22:4)    Executing Stmt          : ASSIGN sd =


--- Subroutine study_day Execution Starting.
   1    1 (15:4)    Executing Stmt          : FUNCTION
   1    2 (16:4)    Executing Stmt          : ASSIGN n =
   1     (16:17)   n = (event_date=18348) - (intervention_date=18308)
= 40
   1    3 (17:4)    Executing Stmt          : IF
   1     (17:10)   _temp1 = (n=40) >= 0
   1    4 (17:17)   Executing Stmt          : ASSIGN n =
   1     (17:21)   n = (n=41) + 1 = 41
   1    5 (18:7)    Executing Stmt          : RETURN _study_day_ =
   1     (18:14)   _study_day_ = (n=41) = 41
   1    6 (19:4)    Executing Stmt          : ENDSUB
--- Subroutine study_day Execution Finished.

   1     (22:16)   sd = study_day( start=18308, today=18348 ) = 41
--- Program Execution Finished.
```

ABORT ステートメント

現在の DATA ステップ、ジョブ、SAS セッションを終了します。

構文

ABORT;

引数なし

PROC FCMP の ABORT ステートメントに引数はありません。

ARRAY ステートメント

名前を変数と定数のリストと関連付けます。

例: “例 8: データセットの各行を標準化する” (738 ページ)

重要: VARCHAR 配列は 2024.03 以降でサポートされています。

構文

ARRAY *array-name*[*dimensions*] <*$length* | *VARCHAR (length)* | *VARCHAR(*)*> < / NOSYMBOLS> |<*variable(s)*> |<*constant(s)*> |<*initial-values*>;

必須引数

array-name
配列の名前を指定します。

dimensions
一次元配列の要素数、または多次元配列の各次元の要素数を数値で表したものです。

オプション引数

$
直前にある変数が、CHAR データタイプの文字変数であることを指定します。

length
変数値を格納するための数値定数を指定します。数値変数と文字変数の場合、この定数は変数に格納される最大バイト数です。

VARCHAR
直前にある変数が VARCHAR 型であることを指定します。

length
VARCHAR 変数に格納される最大文字数を指定します。

*****
許可される最大長:の 536,870,911 の UTF-8 文字をサポートするように指定します。

variable
配列の変数を指定します。

constant
固定値を示す数値、または文字列を指定します。文字定数は単一引用符で囲みます。

initial-values
配列内の対応する要素に初期値を指定します。内部値をかっこ内に指定できます。

スラッシュ区切り文字に続くオプション

/NOSYMBOLS
数値または文字値の配列が関連付けられている要素変数なしで作成されるように指定します。この場合、配列添字によってのみ配列の要素にアクセスできます。

ヒント /NOSYMBOLS は、_TEMPORARY_とまったく同じように使用されます。

名前別の個別の配列要素変数にアクセスする必要がなければ、メモリを節約できます。

詳細

ARRAY ステートメントの基本

PROC FCMP の ARRAY ステートメントは、DATA ステップで使用される ARRAY ステートメントに似ています。ARRAY ステートメントは、名前を変数と定数のリストと関連付けます。サブスクリプトを含む配列名を使用して、配列のアイテムを参照します。

PROC FCMP で使用される ARRAY ステートメントは、DATA ステップの ARRAY ステートメントのすべての機能をサポートしているわけではありません。PROC FCMP にのみ適用される相違点のリストは、次のとおりです。

- すべての配列参照には、明示的な添字演算子が含まれている必要があります。
- PROC FCMP は名前の後にかっこを使用して、関数呼び出しを表します。配列を参照する場合は、角かっこ[ ]または中かっこ{ }を使用します。
- PROC FCMP の ARRAY ステートメントは、下限指定をサポートしていません。
- 配列には最大 6 次元を使用できます。
- 2024.03 以降、PROC FCMP は VARCHAR 配列をサポートしています。

PROC FCMP で使用される ARRAY ステートメントの配列要素として、変数と定数の両方を使用できます。要素を定数配列に割り当てることはできません。次元指定と要素リストはオプションですが、これらの値のいずれかを入力する必要があります。配列に対し要素のリストを指定しない場合、または配列のサイズよりも少ない要素をリストする場合、PROC FCMP は数値接尾辞を配列の要素に追加し、要素リストを完了することにより、配列変数を作成します。

配列参照を PROC FCMP ルーチンに渡す

配列を CALL ルーチンに渡し、CALL ルーチンで配列の値を変更しようとする場合、CALL ルーチンの OUTARGS ステートメントで配列引数の名前を指定する必要があります。

例

ARRAY ステートメントの例は、次のとおりです。

```
array spot_rate[3] 1 2 3;
array spot_rate[3] (1 2 3);
array y[4] y1-y4;
array xx[2,3] x11 x12 x13 x21 x22 x23;
array pp p1-p12;
array q[1000] /nosymbols;
array foo[1] $ 30
array bar[5] varchar(10);
```

注: VARCHAR 配列は 2024.03 以降でサポートされています。

ATTRIB ステートメント

変数の出力形式、ラベル、長さの情報を指定します。

構文

ATTRIB *variable(s)*< FORMAT=*format-name* LABEL='*label*' LENGTH=*length*>;

必須引数

variable
属性と関連付ける変数を指定します。

オプション引数

FORMAT=*format-name*
出力形式を *variable* 引数の変数と関連付けます。

LABEL='*label*'
ラベルを *variable* 引数の変数と関連付けます。

LENGTH=*length*
variable 引数の変数の長さを指定します。

例

ATTRIB ステートメントの例は、次のとおりです。

```
attrib x1 format=date7. label='variable x1' length=5;
```

```
attrib x1 format=date7. label='variable x1' length=5
    x2 length=5
    x3 label='var x3' format=4.
    x4 length=$2 format=$4.;
```

DECLARE ステートメント

オブジェクトを宣言します。

別名: DCL

構文

形式 1: **DECLARE***object-type object-reference* ;

形式 2: **DECLAREOBJECT***object-reference* (*object-type*)

必須引数

object-reference
オブジェクトの参照名を指定します。

object-type
宣言するデータの種類を指定します。

詳細

詳細

DECLARE ステートメントの形式 1 は、Hash、Hiter、および Dictionary オブジェクトに適用されます。

DECLARE ステートメントの形式 2 は、Python および ASTORE オブジェクトに適用されます。

関連項目

- “DECLARE ステートメント: ハッシュオブジェクトとハッシュ反復子オブジェクト” (*SAS コンポーネントオブジェクト: リファレンス*)
- “DECLARE ステートメント: ディクショナリオブジェクト” (*SAS コンポーネントオブジェクト: リファレンス*)
- “DECLARE ステートメント: Python オブジェクト” (*SAS コンポーネントオブジェクト: リファレンス*)
- “PROCFCMP および ASTORE”

DELETEFUNC ステートメント

OUTLIB オプションで指定されている関数ライブラリから関数が削除されます。

構文

DELETEFUNC *function-name*;

必須引数

function-name
OUTLIB オプションで指定されている関数ライブラリから関数の名前が削除されるように指定します。

DELETESUBR ステートメント

OUTLIB オプションで指定されている関数ライブラリからサブルーチンが削除されます。

構文

DELETESUBR *subroutine-name*;

必須引数

subroutine-name
OUTLIB オプションで指定されている関数ライブラリからサブルーチンの名前が削除されるように指定します。

FUNCTION ステートメント

値を返すルーチンに対し、サブルーチン宣言を指定します。

例:
“例 1: 関数の作成と、DATA ステップからの関数の呼び出し” (728 ページ)
“例 2: PROC FCMP による関数の作成および保存” (730 ページ)
“例 3: FUNCTION ステートメントでの数値データの使用” (732 ページ)
“例 4: FUNCTION ステートメントでの文字データの使用” (733 ページ)
“例 5: 変数の引数を配列として使用する” (734 ページ)
“例 7: ユーザー定義関数での Graph Template Language (GTL)の使用” (735 ページ)

構文

FUNCTION *function-name*(*argument-1*< , *argument-2, ...*>) < VARARGS> < *$* | *VARCHAR* > <*length*>
<KIND | GROUP='*string*'>< LABEL=*'string-2'*>;

... more-program-statements ...

RETURN (*expression*);

ENDSUB;

必須引数

function-name
関数の名前を指定します。

argument
関数に対し引数を指定します。文字引数を指定するには、引数名の後にドル記号($)を付けます。VARCHAR 引数を指定するには、引数名の後に *VARCHAR* を付けます。次の例 function myfunct(arg1, arg2 $, arg3, arg4 $);では、arg1 および arg3 は数値引数で、arg2 および arg4 は文字引数です。

expression
関数から返される値を指定します。

オプション引数

VARARGS
関数が可変個引数をサポートするように指定します。VARARGS を指定すると、関数の最後の引数は配列である必要があります。

制限事項 VARARGS 引数を使用して数値変数を指定する必要があります。

参照項目: “例 5: 変数の引数を配列として使用する” (734 ページ)

$
関数が文字値を返すように指定します。$または VARCHAR のどちらも指定されていない場合、関数は数値を返します。

VARCHAR
関数が VARCHAR 値を返すように指定します。VARCHAR も$も指定されていない場合、関数は数値を返します。

length
文字または VARCHAR 値の長さを指定します。VARCHAR 値の長さが指定されていない場合は、許可される最大 VARCHAR 長が使用されます。

デフォルト Character 32

VARCHAR(*)

KIND='*string*'
GROUP='*string*'
特定の属性を含む項目の集合を指定します。文字数は 32 文字以下です。

LABEL='*string-2*'
ブランクを含む最大 256 文字のラベルを指定します。

詳細

FUNCTION ステートメントは、値を返すサブルーチン宣言の特殊なケースです。関数の呼び出しに CALL ステートメントを使用しません。関数の定義は、FUNCTION ステートメントで始まり、ENDSUB ステートメントで終わります。

LABEL ステートメント

最大 256 文字のラベルを指定します。

構文

LABEL *variable*='*label*';

必須引数

variable
ラベル付けする変数の名前を指定します。

'*label*'
ブランクを含む最大 256 文字のラベルを指定します。

例

LABEL ステートメントの例は、次のとおりです。

```
label date='Maturity Date';
label bignum='Very very large numeric value';
```

LENGTH ステートメント

文字変数や数値変数を格納する場合はバイト数を、VARCHAR 変数を格納する場合は文字数を指定します。

構文

LENGTH *variable-specification(s)*<DEFAULT= *n*>;

必須引数

variable-specification
この引数は必須です。次の形式を使用します。

variable(s) <$ *length* | *length* | *VARCHAR(length)* | *VARCHAR(*)*>

variable
長さを割り当てる 1 つ以上の変数を指定します

$
直前にある変数が、CHAR データタイプの文字変数であることを指定します。

length
変数値を格納するための数値定数を指定します。数値変数と文字変数の場合、この定数は変数に格納される最大バイト数です。

範囲 文字変数の場合、1〜32767 バイト。数値変数の場合、3〜8 バイト。数値変数に指定できる長さの最小値は、ユーザーのシステムが使用する浮動小数点形式によって異なります。ほとんどのシステムでは IEEE 浮動小数点形式が使用され、最小値は 3 バイトです。

制限事項 CAS では、8 バイトより短い数値変数は 8 バイトとして扱われます。

VARCHAR
直前にある変数が VARCHAR 型であることを指定します。

length
VARCHAR 変数に格納される最大文字数を指定します。

たとえば、length foo varchar(100);は、*foo* VARCHAR 変数に最大 100 文字が格納されることを意味します。

デフォルト *

範囲 VARCHAR 変数の場合、1 から 536,870,911 の UTF-8 文字までです。

制限事項 VARCHAR 変数に文字定数を割り当てる場合、その文字定数は 32767 バイトまでに制限されます。

*****
許可される最大長:の 536,870,911 の UTF-8 文字をサポートするように指定します。たとえば、length foo varchar(*);は、*foo* VARCHAR 変数の最大長を設定することを意味します。

オプション引数

DEFAULT = n
新しく作成する数値変数の値を保存するときに使用するデフォルトのバイト数を変更します。

デフォルト 8 バイト

範囲 3 から 8 バイト

LISTFUNC ステートメント

関数のソースコードをが SAS リストに書き込まれます。

構文

LISTFUNC *function-name*</NODEPENDENTS>;

必須引数

function-name
ソースコードが SAS リストに書き込まれている関数の名前を指定します。

注: デフォルトでは、指定された関数に加えて、すべての依存関数が返されます。

オプション引数

/NODEPENDENTS
function-name で指定された関数のみが返され、依存関数は返されないことを指定します。

別名 /NODEPS

LISTSUBR ステートメント

サブルーチンのソースコードが SAS リストに書き込まれます。

構文

LISTSUBR *subroutine-name*;

必須引数

subroutine-name
ソースコードが SAS リストに書き込まれているサブルーチンの名前を指定します。

OUTARGS ステートメント

サブルーチンが更新する引数リストの引数を指定します。

制限事項: 多くの SAS 分析プロシジャが FCMP 関数で解析微分法を実行します。この方法で関数を使用する場合は、OUTARGS ステートメントを使用しないでください。ほとんどの場合、OUTARGS ステートメントを SUBROUTINE ステートメントとともに使用してください。

例: “例 2: PROC FCMP による関数の作成および保存” (730 ページ)

“例 8: データセットの各行を標準化する” (738 ページ)
“例 6: SUBROUTINE ステートメントの CALL ステートメントとの併用” (734 ページ)

構文

OUTARGS *out-argument-1*< , *out-argument-2*, ...>;

必須引数

out-argument
サブルーチンにより更新する引数リストから引数を指定します。

ヒント 配列がルーチン内の OUTARGS ステートメントに掲載されている場合は、配列は“参照別”に渡されます。掲載されていない場合は、"値別"に渡されます。

例 サブルーチンでの OUTARGS ステートメントの使用例については、“SUBROUTINE ステートメント” (694 ページ)を参照してください。

STATIC ステートメント

変数が再度割り当てられるまで、前の呼び出しからの変数の値が保持されます。

構文

STATIC*variables* , <*initial-value(s)*>;

必須引数

variables
保持する値の変数名、変数リスト、または配列名を指定します。

オプション引数

initial-values
直前に指定した 1 つまたは複数の要素に対し、初期値(数値または文字)を指定します。

詳細

STATIC ステートメントを使用して変数を初期化できます。

通常、関数またはサブルーチン内のローカル変数は関数またはサブルーチンへの呼び出し間で保持されません。コストのかかる初期化が必要となる場合は、STATIC 変数を使用して初期化できます。STATIC 変数はスタックで割り当てられないため、大規模なローカル配列に使用して、スタックで再割り当てやオーバーフローが発生しないようにすることができます。

例

例 1

数値の static の例を示します。

```
proc fcmp;
  function fdef1(in);
    static x1  1;
    if x1 = 1 then do;
      x1 = 2;
    return(in);
  end;

    return (in*2);
  endsub;

  ans = fdef1( 1);
  put "Answer should be 1" ans=;
  ans = fdef1( 1);
  put "Answer should be 2" ans=;

run;
```

例 2

文字の static の例を示します。

```
proc fcmp;
  function char_func( in $) $;
  length c1 $ 32;
  static c1  "Elephant";
    if c1 = "Elephant" then
      do;
       c1 = in || c1;
   return (c1);
   end;
   return( in);
  endsub;
  length ans $ 32;
  ans = char_func( "Big ");
  put "Answer should be >>Big Elephant<<" ans=;
  ans = char_func( "Big ");
```

```
  put "Answer should be >>Big<<" ans=;
run;
quit;
```

例 3

配列の static の例を示します。

```
proc fcmp ;
 function array_func( in ) ;
 array a[5] ;
 array foo[5];
 static a  first 1;
 put a[1]= foo[1]=;
   If first  then do;
     do i=1 to dim(a);
       a[i]=i;
       foo[i]=i;
 end;
 first =0;
 end;
 else do;
  do i=1 to dim(a);
     a[i]=a[i]+1;
 end;
 end;
 put a[5];
 return( in);

 endsub;

 ans = array_func( 4);

/* should increase by 1 */
  ans = array_func( 4);

run;
```

STRUCT ステートメント

C 言語パッケージに定義されている構造タイプを宣言(作成)します。

構文

STRUCT *structure-name variable*;

必須引数

structure-name
C 言語パッケージで定義され PROC FCMP で宣言されている構造の名前を指定します。

variable
この構造タイプとして宣言する変数を指定します。

例

次に、STRUCT ステートメントの例を示します。

```
struct DATESTR matdate;
matdate.month=3;
matdate.day=22;
matdate.year=2009;
```

SUBROUTINE ステートメント

CALL ステートメントを用いて呼び出せるコードの独立計算ブロックを宣言(作成)します。

例: “例 8: データセットの各行を標準化する” (738 ページ)
“例 2: PROC FCMP による関数の作成および保存” (730 ページ)

構文

SUBROUTINE *subroutine-name* (*argument-1*< , *argument-2*, ...>) < VARARGS>
<KIND | GROUP='*string*'>;

OUTARGS *out-argument-1*< , *out-argument-2* , ...> ;

... more-program-statements ...

ENDSUB;

必須引数

subroutine-name
サブルーチンの名前を指定します。

argument
そのサブルーチンに対して 1 つ以上の引数を指定します。引数名の後にドル記号($)を配置して、文字引数を指定します。VARCHAR 引数は、引数名の後に *VARCHAR* を配置して指定します。次の例
subroutine mysub(arg1, arg2 $, arg3, arg4 $);では、arg1 および arg3 は数値引数で、arg2 および arg4 は文字引数です。

OUTARGS
　サブルーチンを通して更新する、引数リスト内の引数を指定します。

out-argument
　サブルーチンにより更新する引数リストから引数を指定します。

オプション引数

VARARGS
　サブルーチンが可変個引数をサポートするように指定します。VARARGS を指定する場合は、サブルーチンの最後の引数は配列である必要があります。

GROUP='*string*'
KIND='*string*'
　特定の属性を含む項目の集合を指定します。文字数は 32 文字以下です。

詳細

SUBROUTINE ステートメントにより、CALL ステートメントで呼び出せるコードの独立計算ブロックを宣言(作成)できます。サブルーチンの定義は、SUBROUTINE ステートメントで始まり、ENDSUB ステートメントで終わります。SUBROUTINE ステートメントの OUTARGS ステートメントを使用すれば、サブルーチンにより更新する必要がある引数リストから引数を指定できます。

使用法: FCMP プロシジャ

PROC FCMP と DATA ステップの相違点

PROC FCMP と DATA ステップの相違点の概要

PROC FCMP は本来、複数の SAS/STAT プロシジャ、SAS/ETS プロシジャ、SAS/OR プロシジャに対するプログラミング言語として開発されました。その実装は DATA ステップとまったく同じというわけではなく、これら 2 つの言語には相違点があります。次のセクションでは、PROC FCMP と DATA ステップの相違点をいくつか取り上げます。

PROC FCMP と DATA ステップの相違点

ABORT ステートメント

PROC FCMP の ABORT ステートメントは引数を受け入れません。

ABORT ステートメントは PROC FCMP の関数またはサブルーチン内では無効となります。プロシジャ本体でのみ有効です。

配列

PROC FCMP は名前の後にかっこを使用して、関数呼び出しを表します。配列を参照する場合は、角かっこ[ ]または中かっこ{ }の使用をお勧めします。ARR という名前の配列の場合、コードは ARR[i]または ARR{i}となります。PROC FCMP では、配列の次元数が 6 に制限されます。

PROC FCMP の ARRAY ステートメントの相違点の詳細については、“詳細” (683 ページ)を参照してください。

Data セットの入力および出力

PROC FCMP は、出力データセットの作成およびこれへの書き込みにおいて、DATA ステートメントと OUTPUT ステートメントをサポートしています。PROC FCMP ステートメントは、データ入力および出力データセットを指定において、DATA=および OUT=オプションをサポートしています。データセット入力は、SET ステートメント、MERGE ステートメント、UPDATE ステートメント、MODIFY ステートメントをサポートしていません。一般に、データはパラメーターを使用して PROC FCMP ルーチンに変換されます。大量のデータを変換する必要がある場合は、配列を PROC FCMP ルーチンに渡せます。

DATA ステップデバッガー

DATA ステップデバッガーを使用するとき、PROC FCMP は他のルーチンと同様に機能します。つまり、デバッガー時にはその関数にはステップインできません。かわりにそのルーチン内で PUT ステートメントを使用してください。

DO ステートメント

DO ステートメントの次のタイプは、PROC FCMP にサポートされています。

```
do i=1, 2, 3;
```

PROC FCMP の DO ステートメントは文字ループ制御変数をサポートしていません。次のコードは PROC FCMP ではなく DATA ステップで実行できます。

```
do i='a', 'b', 'c';
```

DO ステートメントは文字インデックス変数をサポートしていません。したがって、次のコードは PROC FCMP ではサポートされていません。

```
do i='one', 'two', 'three';
```

ファイルの入力および出力

PROC FCMP は PUT ステートメントと FILE ステートメントをサポートしていますが、FILE ステートメントは LOG 出力先と PRINT 出力先に限られています。PROC FCMP には INFILE ステートメントも INPUT ステートメントもありません。

IF 式

IF 式により IF-THEN/ELSE 条件を式内で評価できます。IF 式は、PROC FCMP にはサポートされていますが DATA ステップにはサポートされていません。IF 式は IF-THEN/ELSE ステートメントに分割する必要がないため、一部の式を単純化できます。たとえば、次の 2 つのコードは同等ですが、IF 式(最初の例)は複素数式ではありません。

```
x=if y < 100 then 1 else 0;
```

```
if y < 100 then
  x=1;
else
  x=0;
```

IF 式の代替は式です。つまり、操作のグループ化には DO/END ブロックではなく、かっこが使用されます。

PUT ステートメント

PUT ステートメントの構文は PROC FCMP および DATA ステップで似ていますが、操作は異なる可能性があります。PROC FCMP で、PUT ステートメントは通常プログラムデバッグに使用されます。DATA ステップで、PUT ステートメントはレポートとして、またはファイル作成ツール、デバッグツールとして使用されます。次のリストに、その他の相違点を示します。

- PROC FCMP の PUT ステートメントは、デフォルトで出力を SAS 出力ウィンドウに書き込みます。DATA ステップの PUT ステートメントは、デフォルトで出力を SAS ログに書き込みます。
- PROC FCMP の PUT ステートメントは、行ポインター、形式修飾子、列出力、因数分解型リスト、反復係数、重ね打ち、_INFILE_オプション、または特殊文字$をサポートしていません。DLM=、DSD などの FILE ステートメントオプションによって提供される機能をサポートしていません。
- PROC FCMP の PUT ステートメントは、式の評価および結果の書き込みを式をかっこ内に置くことによりサポートしています。ただし、DATA ステップは、PUT ステートメントでの式の評価はサポートしていません。PROC FCMP の次の例では、式 x/100 と sqrt(y)/2 が評価され、結果が SAS ログに書き込まれます。

  ```
  put (x/100) (sqrt(y)/2);
  ```

  かっこは PROC FCMP で式評価に使用されるため、DATA ステップのように変数またはフォーマットリストに使用できません。
- PROC FCMP の PUT ステートメントは、かっこで囲まれている場合を除いて、添字配列名をサポートしていません。たとえば、ステートメント put (A[i]);は配列 A の i 番目の要素を書き込みますが、ステートメント put A[i];ではエラーメッセージが表示されます。

- 配列名は、サブスクリプトなしの PUT ステートメントで使用できます。そのため、次のステートメントが有効です。
  - put A=; (A が配列であるとき)は、各値が要素変数の名前でラベル付けされている配列 A のすべての要素を書き込みます。
  - put (A)*=;は、put A=;と同じ出力を書き込みます。
  - put A;は、配列 A のすべての要素を書き込みます。
- PROC FCMP の PUT ステートメントがスペースを 含む各項目の出力に続きます。これは、DATA ステップのリストモード出力に似ています。列および行の位置に関する詳細な制御は、DATA ステップより少ない程度にサポートされています。
- PROC FCMP の PUT ステートメントは、印刷項目_PDV_をサポートし、ルーチンのプログラムデータベクトルのすべての 変数のフォーマットされたリストを印刷します。ステートメント put _PDV_;により印刷される変数リストは、ステートメント put _ALL_;により印刷されるものよりもはるかに読みやすくなります。

PUT 関数

PUT 関数の構文は PROC FCMP および DATA ステップで似ていますが、結果は異なる可能性があります。PROC FCMP では、文字列の長さとフォーマット幅の最小値を使用して、出力のフォーマット済み文字列幅が決定されます。DATA ステップでは、指定されたフォーマット幅のみを使用して、フォーマットされた文字列の出力幅が決定されます。文字列の長さがフォーマット長よりも短い場合、PROC FCMP は有効な情報をできるだけ多く返そうとします。文字列の長さがフォーマット長よりも短いという同じ状況では、DATA ステップはフォーマットスタイルで情報を返します。次の例は、PROC FCMP の PUT 関数と DATA ステップの違いを示しています。

例のコード 21.1　可変長がフォーマット長より短い

```
proc fcmp;
/*Create variables with different lengths*/
length s2 $2;
length s3 $3;
length s4 $4;
length s5 $5;
length s6 $6;
length s7 $7;
length s8 $8;
length s9 $9;
length s10 $10;

/*Apply the same length=10 format to the different length variables*/
s2 = put('01Mar2022'd, mmddyy10.);
s3 = put('01Mar2022'd, mmddyy10.);
s4 = put('01Mar2022'd, mmddyy10.);
s5 = put('01Mar2022'd, mmddyy10.);
s6 = put('01Mar2022'd, mmddyy10.);
s7 = put('01Mar2022'd, mmddyy10.);
s8 = put('01Mar2022'd, mmddyy10.);
s9 = put('01Mar2022'd, mmddyy10.);
s10 = put('01Mar2022'd, mmddyy10.);

/*Display the results*/
put 's2=' s2;
```

```
put 's3=' s3;
put 's4=' s4;
put 's5=' s5;
put 's6=' s6;
put 's7=' s7;
put 's8=' s8;
put 's9=' s9;
put 's10=' s10;
run; quit;


/*Repeat the same steps for the DATA step*/
data _null_;
length s2 $2;
length s3 $3;
length s4 $4;
length s5 $5;
length s6 $6;
length s7 $7;
length s8 $8;
length s9 $9;
length s10 $10;

s2 = put('01Mar2022'd, mmddyy10.);
s3 = put('01Mar2022'd, mmddyy10.);
s4 = put('01Mar2022'd, mmddyy10.);
s5 = put('01Mar2022'd, mmddyy10.);
s6 = put('01Mar2022'd, mmddyy10.);
s7 = put('01Mar2022'd, mmddyy10.);
s8 = put('01Mar2022'd, mmddyy10.);
s9 = put('01Mar2022'd, mmddyy10.);
s10 = put('01Mar2022'd, mmddyy10.);


put 's2=' s2;
put 's3=' s3;
put 's4=' s4;
put 's5=' s5;
put 's6=' s6;
put 's7=' s7;
put 's8=' s8;
put 's9=' s9;
put 's10=' s10;
run;
```

アウトプット 21.1　PROC FCMP は次の結果を返します。

```
s2= 03
s3= 03
s4= 0301
s5= 03/01
s6= 030122
s7= 030122
s8= 03/01/22
s9= 03/01/22
s10= 03/01/2022
```

例のコード 21.2　DATA ステップは次の結果を返します。

```
s2= 03
s3= 03/
s4= 03/0
s5= 03/01
s6= 03/01/
s7= 03/01/2
s8= 03/01/20
s9= 03/01/202
s10= 03/01/2022
```

結果は、文字列の長さがフォーマット長よりも短い場合、PROC FCMP は結果文字列にできるだけ多くの有効な情報を入れることを優先することを示しています。DATA ステップでは、フォーマットスタイルが優先されます。たとえば、変数 s4 の文字列の長さが 4 に設定され、フォーマット長が 10 の場合、PROC FCMP は月と日 0301 を返しますが、DATA ステップ 03/0 を返します。文字列の長さがフォーマット長以上の場合、PROC FCMP と DATA ステップは同じ結果を返します。

WHEN ステートメントと OTHERWISE ステートメント

WHEN と OTHERWISE ステートメントでは、複数のターゲットステートメントを利用できます。つまり、DO/END グループは複数の WHEN ステートメントに不要です。次に例を示します。

```
SELECT;
  WHEN(expression-1

              )
  statement-1

              ;
  statement-2

              ;
  WHEN (expression-2

              )
  statement-3

              ;
  statement-4

              ;
END;
```

演算子

PROC FCMP 演算子は、式が有効な場所であればどこでも有効です。たとえば、if-then 式、do-loop 式、割り当てステートメントなどはすべて、演算子の有効な場所です。

IS <NOT> MISSING と IS <NOT> NULL は欠損値または null 値をテストするために使用されます。IS 演算子は、PROC FCMP では式が有効な場所であればどこでも有効です。次に、いくつかの例を示します。

```
if my-var IS MISSING then do;
  expression-1;
end;

do WHILE my-var IS NOT NULL;

my-test = my-var IS MISSING;
```

PROC FCMP は、CONTAINS 演算子をサポートしています。*target-string* <NOT> CONTAINS *test-string* は、指定されたテスト文字列がターゲット文字列の一部であるかどうかをテストします。CONTAINS 演算子は、PROC FCMP で式が有効な場所であればどこでも有効です。次に例を示します。

```
proc fcmp;
length s $10;
s = "abc";
x = s contains "b";
y = NOT(s contains "b");
if s contains "b" then
  PUT "Contains Match";
else
  PUT "Contains does not match";
put x= y=;
run; quit;
```

SAS ログは次の結果を返します。

```
Contains Match
x=1 y=0
```

PROC FCMP は、BETWEEN 演算子をサポートしています。使用すると、*value* <NOT> BETWEEN *range-1* AND *range-2* は、値が指定された値の範囲内にあるかどうかをテストします。BETWEEN 演算子は、PROC FCMP で式が有効な場所であればどこでも有効です。次に例を示します。

```
proc fcmp;
x = 10;
s= 'cat';
r1 = x between 1 and 20;
put r1=;
r2 = x between 90 and 100;
put r2=;
r3= upcase(s) between 'CAMEL' and 'DOG';
put r3=;
if (x not between 80 and 70) then put 'out of range';
run; quit;
```

SAS ログは次の結果を返します。

```
r1=1
r2=0
r3=1
out of range
```

PROC FCMP は、LIKE 演算子をサポートしています。LIKE 演算子は、パターンマッチング指定を使用して、テスト文字列をターゲット文字列と比較します。LIKE 演算子は、PROC FCMP では式が有効な場所であればどこでも有効です。パターンマッ

チングの詳細について-は、“LIKE 演算子” (*SAS SQL プロシジャユーザーガイド*)を参照してください。次に例を示します。

```
proc fcmp;
  length s $ 100;
  s = 'hello there';
  r = s not like '%dog%';
  put r=;

  if s like 'hello%' then
    put "Match";
  else put "No Match";
run;
quit;
```

SAS ログは次の結果を返します。

```
r=1
Match
```

2023.03 以降、PROC FCMP は SOUNDS-LIKE 演算子をサポートしています。SOUNDS-LIKE 演算子は、似ている単語を識別するための SOUNDEX アルゴリズムに基づいています。SOUNDEX アルゴリズムは英語中心に機能するため、英語以外の言語ではあまり有効ではありません。SOUNDEX アルゴリズムの詳細については、SOUNDS-LIKE 演算子を使用した値の取得.を参照してください。 SOUNDS-LIKE 演算子は、PROC FCMP では、式が有効な場所であればどこでも有効です。SOUNDS-LIKE 演算子の構文を次のテーブルと例に示します。

表 21.1　SOUNDS-LIKE 演算子の構文

演算子	構文
SOUNDS-LIKE	=*
NOT SOUNDS-LIKE	^=*
NOT SOUNDS-LIKE	~=*
NOT SOUNDS-LIKE	NE=*

```
proc fcmp;
length x $ 100 y $ 100;
x= Johnson;
y= Johnsen;
test = x =* y;
put test=;
run;
quit;
```

SAS ログは次の結果を返します。

```
test=1
```

PROC FCMP の追加機能

PROC REPORT と計算ブロック

PROC REPORT は DATA ステップを使用して、計算ブロックを評価します。DATA ステップは PROC FCMP ルーチンを呼び出すことができるため、これらのルーチンを PROC REPORT 計算ブロックから呼び出すこともできます。

関数の暗黙値の計算

PROC FCMP は、SOLVE 関数を関数の暗黙値の計算に使用します。

PROC FCMP と Microsoft Excel

通常は SAS で使用できない多くの Microsoft Excel 関数が、PROC FCMP で実装されています。これらの関数は、sashelp.slkwxl データセットにあります。これらの関数は、SAS の Excel 関数で表示できます。

これらの関数は、次の SAS コードを使用することでも表示できます。

```
proc fcmp inlib=sashelp.slkwxl listall;
run;
```

次の例では、ODD_SLK 関数が使用されています。

```
options cmplib=sashelp.slkwxl;
data _null_;
  num    =4.2;
  odd_num=odd_slk(num);
  put 'Odd number nearest to' num ' is ' odd_num;
run;
```

```
Odd number nearest to 4.2  is 5
```

配列の操作

配列を渡す

デフォルトでは、PROC FCMP はルーチン間で“値別”の配列を渡します。ただし、配列がルーチン内の OUTARGS ステートメントでリストされる場合、配列は“参照別”に渡されます。

つまり、関数別の正式なパラメーターへの変更により、渡される配列が変更されます。参照別の配列を渡すと、関数と呼び出し環境の間で大量データを効率的に渡すことができます。データをコピーする必要がないためです。正式な配列を指定するための構文には、次の形式があります。

```
function
name(numeric-array-parameter[*],
character-array-parameter[*] $,
varchar-array-parameter[*] varchar);
```

DATA ステップ一時配列を PROC FCMP ルーチンに渡すことができます。

配列のサイズ変更

組み込み CALL ルーチン DYNAMIC_ARRAY を呼び出して、PROC FCMP ルーチンの配列をサイズ変更できます。この CALL ルーチンの構文には、次の形式があります。

```
call dynamic_array(array, new-dim1-size <, new-dim2-size, ...>);
```

SAS は DYNAMIC_ARRAY CALL ルーチンに、サイズ変更される配列と、配列の各次元の新しいサイズを渡します。動的配列で、ルーチンは必要なメモリ量を割り当てることができます。すべての可能なケースの処理に十分な大きさの配列を作成する必要はありません。

動的配列のサポートは、PROC FCMP ルーチンに制限されています。配列のサイズが変更されると、配列はそのサイズ変更を行ったルーチンでのみ使用できます。DATA ステップ配列をサイズ変更することも、PROC FCMP 動的配列を DATA ステップに返すこともできません。

PROC FCMP ルーチンを含むマクロの使用

%SYSFUNC と%SYSCALL マクロを使用して、PROC FCMP で作成するルーチンを呼び出すことができます。すべての SAS CALL ルーチンは、%SYSCALL ステートメントを使って呼び出すことができます。ただし、LABEL、VNAME、SYMPUT、EXECUTE ルーチンは例外です。%SYSFUNC と%SYSCALL マクロは、最大 32 文字の SAS 関数名をサポートしています。

PROC FCMP ルーチンの変数のスコープ

変数のスコープの概念

ルーチンとプログラムを互いに独立させる重要な部分は、変数のスコープです。変数のスコープは、変数の値が使用可能なコードのセクションです。PROC FCMP ルーチンの場合、ルーチン外で宣言される変数は、ルーチン内ではアクセスできませ

ん。ルーチン内で宣言される変数は、ルーチン外ではアクセスできません。ルーチン内で作成される変数は、ローカル変数といいます。そのスコープがルーチンに対して“ローカル”であるためです。

関数はローカル変数を計算中時にスクラッチ変数として使用し、その変数は関数が返すときに使用できません。関数が呼び出されると、ローカル変数のスペースは、コールスタックでプッシュされます。関数が返す場合、ローカル変数によって使用されるスペースがコールスタックから削除されます。

別のルーチンと同じ名前のローカル変数がある場合

異なるルーチンでローカル変数の名前が同じ場合、変数のスコープの概念がわかりにくい可能性があります。この場合、各ローカル変数は重複しません。次の例では、DATA ステップ、CALL ルーチン subA および subB には、x という名前のローカル変数が含まれています。x はそれぞれその他の x 変数と異なります。プログラムの実行時に、DATA ステップは subA を呼び出し、subA は subB を呼び出します。それぞれの環境は x の値をログに書き込みます。ログ出力には、それぞれの x がどのようにその他と異なっているかが表示されます。

```
proc fcmp outlib=work.funcs.math;
  subroutine subA();
    x=5;
    call subB();
    put 'In subA: ' x=;
  endsub;

  subroutine subB();
    x='subB';
    put 'In subB: ' x=;
  endsub;
run;

options cmplib=work.funcs;
data _null_;
  x=99;
  call subA();
  put 'In DATA step: ' x=;
run;
```

例のコード 21.3　名前が同じローカル変数が別のルーチンにある

```
In subB:  x=subB
In subA:  x=5
In DATA step: x=99
```

再帰

PROC FCMP ルーチンは、再帰である可能性があります。再帰は、問題をより解決しやすい小さなものに単純化してから、より単純な解決の結果を統合して完全な解

決を作り上げていく、問題解決方法です。再帰関数は、関数自体を直接的または間接的に呼び出す関数です。

ルーチンが呼び出されるたびに、ローカル変数のスペースは、コールスタックでプッシュされます。コールスタックのスペースにより、呼び出しごとのローカル変数の独立性が保証されます。ルーチンが返すとき、コールスタックに割り当てられているスペースが削除され、ローカル変数によって使用されるスペースが解放されます。再帰は、完全な解決に向けた進捗を保存するコールスタックに依存します。

ルーチンがそれ自体を呼び出すとき、呼び出しルーチンと、呼び出し中のルーチンには、中間結果のための独自のローカル変数が含まれている必要があります。呼び出しルーチンで呼び出し中のルーチンのローカル変数が変更可能な場合、再帰ソリューションのプログラムが難しくなります。コールスタックにより、呼び出しごとのローカル変数の独立性が保証されます。

次の例では、PROC FCMP の ALLPERMK ルーチンには *n* および *k* の 2 つの引数があり、このルーチンにより、*n* 要素のうち *k* を含むすべての C(*n* , *k*) = *n* ! /(*n* - *k*)!の組み合わせが書き込まれます。要素は、バイナリ値(0, 1)と表されます。関数 ALLPERMK は再帰関数 PERMK を呼び出し、全体のソリューションスペースを 表示して、特定のフィルターに一致するアイテムのみ出力します。

```
proc fcmp outlib=work.funcs.math;
  subroutine allpermk(n, k);
    array scratch[1] / nosymbols;
    call dynamic_array(scratch, n);
    call permk(n, k, scratch, 1,0);
  endsub;

subroutine permk(n, k, scratch[*], m, i);
  outargs scratch;
  if m–1=n then do;
    if i=k then
      put scratch[*];
    end;
  else do;
    scratch[m]=1;
    call permk(n, k, scratch, m+1, i+1);
    scratch[m]=0;
    call permk(n, k, scratch, m+1, i);
  end;
endsub;
run;
quit;

options cmplib=work.funcs;
data _null_;
  call allpermk(5,3);
run;
```

例のコード 21.4 再帰処理例の結果

```
1 1 1 0 0
1 1 0 1 0
1 1 0 0 1
1 0 1 1 0
1 0 1 0 1
1 0 0 1 1
0 1 1 1 0
0 1 1 0 1
0 1 0 1 1
0 0 1 1 1
```

このプログラムでは ARRAY ステートメントの/NOSYMBOLS オプションを使用し、配列要素ごとに変数のない配列を作成します。/NOSYMBOLS 配列は、配列参照、scratch[*m*]によってのみアクセス可能で、DATA step _temporary_ array と同じです。/NOSYMBOLS 配列では通常の配列よりも少ないメモリが使用されます。スペースが変数に対して割り当てられていないためです。ALLPERMK では、PROC FCMP 動的配列も使用されます。

ディレクトリトラバーサル

ディレクトリトラバーサルの概要

DATA ステップまたはマクロを使用する場合、関数によるディレクトリ階層の表示を可能にする機能の実装は難しくなります。DATA ステップとマクロコードを使用した再帰または類似再帰は、コード化が容易ではありません。このセクションでは、ディレクトリ階層のすべてのファイルの完全パス名を配列に入力する、DIR_ENTRIES という名前のルーチンの開発方法を説明します。この例では、PROC FCMP と DATA ステップ構文の間の類似性を表し、PROC FCMP ルーチンがプログラムを簡素化し、独立した再利用可能なコードを作成する方法を強調します。DIR_ENTRIES では、入力として次のパラメーターが使用されます。

- 相対ディレクトリ
- パス名を入力する結果配列
- 結果配列に配置されるパス名の数である、出力パラメーター
- 結果配列の大きさが十分でなかったためにすべての結果セットが切り捨てられたかどうかを表す出力パラメーター

DIR_ENTRIES の制御のフローは、次のとおりです。

1 相対ディレクトリを開きます。

2 ディレクトリのエントリごとに、次のタスクのうちいずれかを行います。
   - エントリがディレクトリの場合、DIR_ENTRIES を呼び出し、結果配列にサブディレクトリのパス名を入力します。

- その他の場合、エントリはファイルで、ファイルのパスを結果配列に追加する必要があります。

3 相対ディレクトリを閉じます。

ディレクトリトラバーサルの例

ディレクトリの開閉

ディレクトリの開閉は、CALL ルーチン DIROPEN と DIRCLOSE によって処理されます。DIROPEN はディレクトリパスを受け入れ、次の制御フローを含みます。

1 FILENAME 関数を使用して、パスに対しファイル参照名を作成します。

2 FILENAME 関数が失敗すると、エラーメッセージがログに書き込まれ、返されます。

3 その他の場合、DOPEN 関数を使用して、ディレクトリを開き、ディレクトリ ID を取得します。

4 ディレクトリファイル参照名をクリアします。

5 ディレクトリ ID を返します。

DIRCLOSE CALL ルーチンにディレクトリ ID が渡され、DCLOSE に渡されます。ディレクトリが閉じられた後にプログラムでそのディレクトリ ID が使用される場合にエラーが発生するように、DIRCLOSE は渡されたディレクトリ ID を欠損に設定します。次のコードが、DIROPEN ルーチンと DIRCLOSE CALL ルーチンを実装します。

```
proc fcmp outlib=work.funcs.dir;
  function diropen(dir $);
  length dir $ 256 fref $ 8;
  rc=filename(fref, dir);
  if rc=0 then do;
    did=dopen(fref);
    rc=filename(fref);
  end;
  else do;
    msg=sysmsg();
    put msg '(DIROPEN(' dir= ')';
    did=.;
  end;
  return(did);
endsub;

subroutine dirclose(did);
  outargs did;
  rc=dclose(did);
  did=.;
endsub;
```

ファイル名の収集

ファイルパスは、DIR_ENTRIES CALL ルーチンによって収集されます。
DIR_ENTRIES では、次の引数が使用されます。

- 相対ディレクトリ
- 入力する結果配列
- 結果配列のエントリ数を入力する出力パラメーター
- 0 に設定する出力パラメーター(すべてのパス名が結果配列に収まる場合)、または 1 に設定する出力パラメーター(一部のパス名が配列に収まらない場合)

DIR_ENTRIES の本文は、DATA ステップのこの機能性の実行に使用されるコードとほぼ同じです。また DIR_ENTRIES は、いくつかのプログラムで簡単に再利用される CALL ルーチンでもあります。

DIR_ENTRIES は DIROPEN を呼び出し、ディレクトリを開いて、ディレクトリ ID を取得します。次にルーチンは DNUM を呼び出し、ディレクトリのエントリ数を取得します。ディレクトリのエントリごとに、DREAD が呼び出され、エントリ名が取得されます。エントリ名が利用可能で、ルーチンは MOPEN を呼び出し、エントリがファイルかディレクトリかどうかを決定します。

エントリがファイルの場合、MOPEN は正の値を返します。この場合、ファイルへの完全パスが結果配列に追加されます。結果配列がいっぱいになると、切り捨て出力引数が 1 に設定されます。

エントリがディレクトリの場合、MOPEN は 0 以下の値を返します。この場合、DIR_ENTRIES はサブディレクトリのエントリのパス名を収集します。パス名の収集は、DIR_ENTRIES を再帰的に呼び出し、サブディレクトリのパスを開始パスとして渡して行います。DIR_ENTRIES が返す場合、結果配列にはサブディレクトリのエントリのパスが含まれています。

```
subroutine dir_entries(dir $, files[*] $, n, trunc);
  outargs files, n, trunc;
  length dir entry $ 256;

  if trunc then return;

  did=diropen(dir);
  if did <= 0 then return;

  dnum=dnum(did);
  do i=1 to dnum;
    entry=dread(did, i);
      /* If this entry is a file, then add to array, */
      /* else entry is a directory, recurse.        */
    fid=mopen(did, entry);
    entry=trim(dir) || '\' || entry;
    if fid > 0 then do;
      rc=fclose(fid);
      if n < dim(files) then do;
        trunc=0;
        n=n + 1;
        files[n]=entry;
      end;
```

```
      else do;
        trunc=1;
        call dirclose(did);
        return;
      end;
    end;
    else
      call dir_entries(entry, files, n, trunc);
  end;

  call dirclose(did);
  return;
endsub;
```

DATA ステップからの DIR_ENTRIES の呼び出し

その他の DATA ステップ CALL ルーチンのように DIR_ENTRIES を起動します。検出されるすべてのファイルの保持に十分なエントリを含む配列を宣言します。次に、DIR_ENTRIES ルーチンを呼び出します。ルーチンが返すとき、結果配列はループされ、配列の各エントリは SAS ログに書き込まれます。

```
options cmplib=work.funcs;
data _null_;
  array files[1000] $ 256 _temporary_;
  dnum=0;
  trunc=0;
  call dir_entries("c:\logs", files, dnum, trunc);
  if trunc then put 'ERROR: Not enough result array entries. Increase array
size.';
  do i=1 to dnum;
    put files[i];
  end;
run;
```

例のコード 21.5　DATA ステップからの DIR_ENTRIES の呼び出しの結果

```
c:\logs\2004\qtr1.log
c:\logs\2004\qtr2.log
c:\logs\2004\qtr3.log
c:\logs\2004\qtr4.log
c:\logs\2005\qtr1.log
c:\logs\2005\qtr2.log
c:\logs\2005\qtr3.log
c:\logs\2005\qtr4.log
c:\logs\2006\qtr1.log
c:\logs\2006\qtr2.log
```

この例に、PROC FCMP 構文と DATA ステップ間の類似点を示します。たとえば、数値式と制御フローステートメントは同じです。PROC FCMP 関数への DIROPEN の抽出により、DIR_ENTRIES が簡素化されます。作成されるすべての PROC FCMP ルーチンは、新しいコンテキストで機能するルーチンを変更することなく、その他の DATA ステップによって再利用可能です。

コンパイルされた関数とサブルーチンの場所の特定: CMPLIB=システムオプション

CMPLIB=システムオプションの概要

SAS システムオプション CMPLIB=は、前にコンパイルされた関数とサブルーチンの検索場所を指定します。FCMP 関数とサブルーチンの使用をサポートするすべてのプロシジャ(FCMP を含む)では、このシステムオプションを使用します。

関数とサブルーチンをサポートするすべてのプロシジャステートメントで LIBRARY=オプションを指定するかわりに、CMPLIB=システムオプションを使用して、すべてのプロシジャによって 使用可能なライブラリを設定できます。

_DISPLAYLOC_オプションは、SAS での関数のロード元データセットの名前を該当するログに書き込みます。_NO_DISPLAYLOC_オプションは、データセット名がログに書き込まれないようにします。

CMPLIB=システムオプションの構文

CMPLIB=オプションの構文には、次の形式があります。

OPTIONS CMPLIB=*library*

OPTIONS CMPLIB=*(library-1 <, library-2, ...>)*

OPTIONS CMPLIB=*list-1 <, list-2, ...*

OPTIONS CMPLIB=_DISPLAYLOC_

OPTIONS CMPLIB=_NO_DISPLAYLOC_

次の説明では、前述の構文について触れます。

OPTIONS
ステートメントを OPTIONS ステートメントとして識別します。

library
前にコンパイルされたライブラリがプログラムにリンクされるように指定します。

list
ライブラリのリストを指定します。

DISPLAYLOC
PROC FCMP の使用時に指定すると、関数ロード元のデータセットが SAS ログに表示されます。

デフォルト _NO_DISPLAYLOC_

_NO_DISPLAYLOC_
PROC FCMP の使用時に指定すると、関数ロード元のデータセットは SAS ログに表示されず、CMPLIB=オプション値としてのライブラリ指定はすべて削除されます。

デフォルト _NO_DISPLAYLOC_

ヒント _DISPLAYLOC_オプションなしで CMPLIB=library-specification を指定する場合は、SAS では SAS ログのデータセット名が表示されません。

例 1: CMPLIB=システムオプションの設定

次の例に、CMPLIB=システムオプションの設定方法を示します。

```
options cmplib=work.funcs;
options cmplib=(work.funcs work.functions mycat.funcs);
options cmplib=(work.func1 - work.func10);
```

例 2: 関数のコンパイルと使用

次の例では、PROC FCMP は SIMPLE 関数をコンパイルし、それを work.Models データセットに保存します。次に、CMPLIB=システムオプションが設定され、関数が PROC MODEL によって呼び出されます。

この例からの出力は、複数のページにわたります。この出力は 5 ページに分割されます。

```
proc fcmp outlib=work.models.yval;
  function simple(a, b, x);
    y=a+b*x;
    return(y);
  endsub;
run;

options cmplib=work.models nodate ls=80;

data a;
  input y @@;
  x=_n_;
  datalines;
08 06 08 10 08 10
;

proc model data=a;
  y=simple(a, b, x);
  fit y / outest=est1 out=out1;
quit;
```

アウトプット 21.2　関数のコンパイルと使用: パート 1

The SAS System

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	2
Equations	1
Number of Statements	1

Model Variables	y
Parameters	a b
Equations	y

The Equation to Estimate is	
y =	F(a, b)

NOTE: At OLS Iteration 1 CONVERGE=0.001 Criteria Met.

アウトプット 21.3　関数のコンパイルと使用: パート 2

The SAS System

The MODEL Procedure
OLS Estimation Summary

Data Set Options	
DATA=	A
OUT=	OUT1
OUTEST=	EST1

Minimization Summary	
Parameters Estimated	2
Method	Gauss
Iterations	1

Final Convergence Criteria	
R	0
PPC	0
RPC(a)	64685.48
Object	0.984333
Trace(S)	1.67619
Objective Value	1.11746

Observations Processed	
Read	6
Solved	6

アウトプット 21.4　関数のコンパイルと使用: パート 3

The SAS System

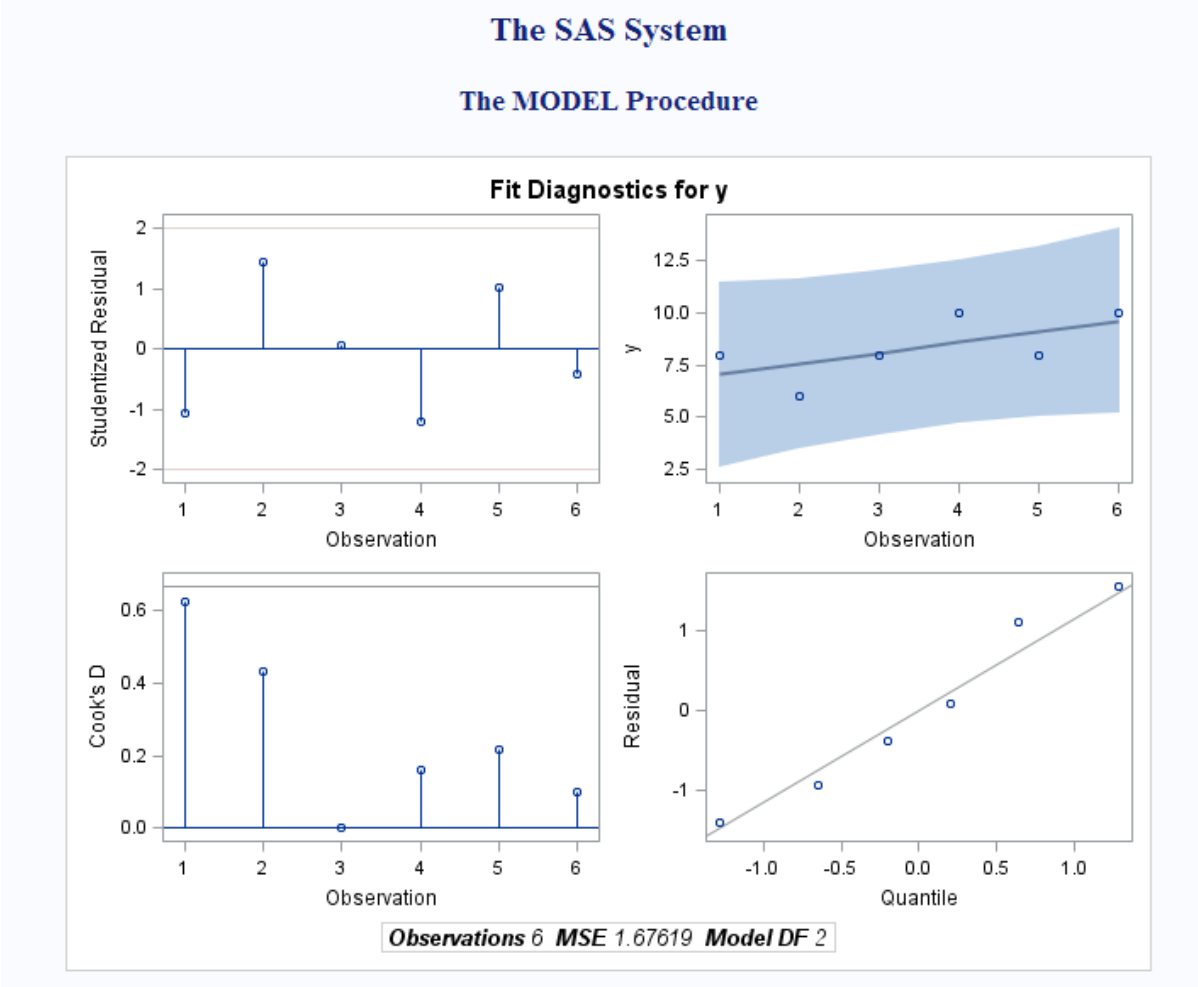
The MODEL Procedure

Nonlinear OLS Summary of Residual Errors							
Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
y	2	4	6.7048	1.6762	1.2947	0.4084	0.2605

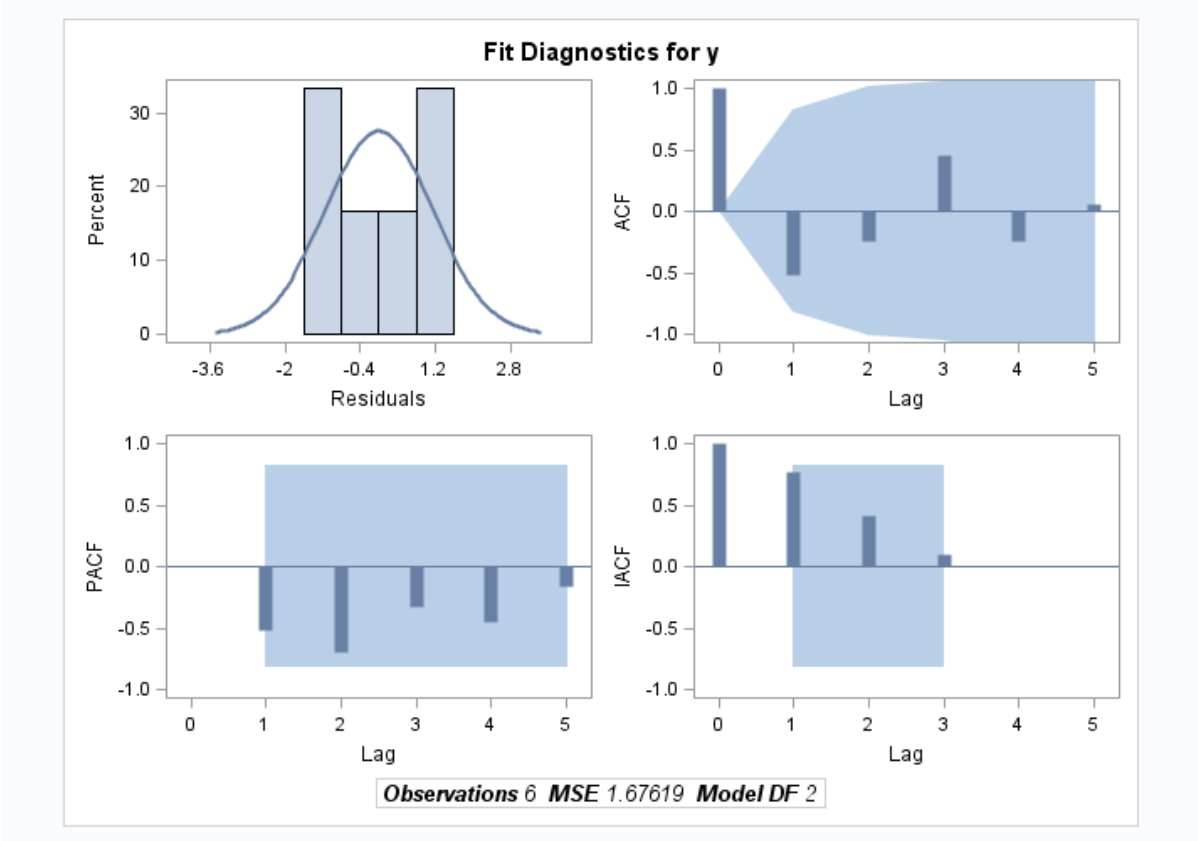
Nonlinear OLS Parameter Estimates				
Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a	6.533333	1.2053	5.42	0.0056
b	0.514286	0.3095	1.66	0.1719

Number of Observations		Statistics for System	
Used	6	Objective	1.1175
Missing	0	Objective*N	6.7048

アウトプット 21.5　関数のコンパイルと使用: パート 4



アウトプット 21.6　関数のコンパイルと使用: パート 5



PROC MODEL の詳細については、*SAS/ETS User's Guide* を参照してください。

例 3: SAS での関数のロード元データセット名の特定

次の例では、_DISPLAYLOC_オプションと_NO_DISPLAYLOC_オプションを使用します。_DISPLAYLOC_オプションを使用する場合、SAS では、機能をロードする場所にあったデータセットの名前が該当するログに書き込まれます。_NO_DISPLAYLOC_オプションでは、データセット名はログに書き込まれません。

```
proc fcmp outlib=work.myfuncs1.pkg;
  function myfunc();
    return(1);
  endsub;
run;

proc fcmp outlib=work.myfuncs2.pkg;
  function myfunc();
    return(2);
  endsub;
run;

proc fcmp outlib=work.myfuncs3.pkg;
  function myfunc();
```

```
    return(3);
  endsub;
run;

option CMPLIB=(myfuncs1-myfuncs3 _DISPLAYLOC_);

proc fcmp;

 a = myfunc();
 put a=;
run;

/*- turning _DISPLAYLOC_ off -*/
option CMPLIB=(myfuncs1-myfuncs3);


proc fcmp;

 a = myfunc();
 put a=;
run;

option CMPLIB=(myfuncs1 myfuncs2 _DISPLAYLOC_);

proc fcmp;

 a = myfunc();
 put a=;
run;

option CMPLIB=_DISPLAYLOC_;

proc fcmp inlib=work.myfuncs1;
 a = myfunc();
 put a=;
run;

option CMPLIB=_NO_DISPLAYLOC_;

proc fcmp inlib=work.myfuncs1;
 a = myfunc();
 put a=;
run;
```

次の結果は SAS ログの一部を示しています。_DISPLAYLOC_オプションと _NO_DISPLAYLOC オプションでは、結果が異なります。

```
116  option CMPLIB=(myfuncs1-myfuncs3 _DISPLAYLOC_);
117
118  proc fcmp;
119
120    a = myfunc();
121    put a=;
122  run;

NOTE: Function 'myfunc' loaded from WORK.myfuncs3.PKG.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.06 seconds
      cpu time           0.04 seconds


123
124  /*- turning _DISPLAYLOC_ off -*/
125  option CMPLIB=(myfuncs1-myfuncs3);
126
127
128  proc fcmp;
129
130    a = myfunc();
131    put a=;
132  run;

NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.06 seconds
      cpu time           0.06 seconds


133
134  option CMPLIB=(myfuncs1 myfuncs2 _DISPLAYLOC_);
135
136  proc fcmp;
137
138    a = myfunc();
139    put a=;
140  run;

NOTE: Function 'myfunc' loaded from WORK.myfuncs2.PKG.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.05 seconds
      cpu time           0.04 seconds


141
142  option CMPLIB=_DISPLAYLOC_;
143
144  proc fcmp inlib=work.myfuncs1;
145    a = myfunc();
146    put a=;
147  run;

NOTE: Function 'myfunc' loaded from work.myfuncs1.PKG.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.04 seconds
      cpu time           0.03 seconds
```

```
149  option CMPLIB=_NO_DISPLAYLOC_;
150
151  proc fcmp inlib=work.myfuncs1;
152    a = myfunc();
153    put a=;
154  run;

NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.04 seconds
      cpu time           0.03 seconds
```

PROC FCMP コンポーネントオブジェクトの使用

ハッシュオブジェクトとハッシュ反復子オブジェクト

ハッシュ法は FCMP プロシジャを介したユーザー定義サブルーチンで利用できます。ハッシュ法を使用すると、プログラムのスコープを拡張できるため、簡明さを損なうことなくより大きな問題を解決できます。PROC FCMP 関数にハッシュオブジェクトを埋め込むことで、既存のプログラムが簡素化され、パフォーマンスが改善されます。

ハッシュオブジェクトとハッシュ反復子コンポーネントオブジェクトの詳細については、"PROC FCMP ハッシュオブジェクトと PROC FCMP ハッシュ反復子オブジェクトの使用" (*SAS コンポーネントオブジェクト: リファレンス*)を参照してください。

辞書

辞書は数値データと文字データへの参照を作成し、配列、他の辞書、PROC FCMP ハッシュオブジェクトへの高速なインメモリハッシングも提供します。

PROC FCMP で辞書を使用する方法、および例を確認する方法については、"FCMP ディクショナリオブジェクトの使用" (*SAS コンポーネントオブジェクト: リファレンス*)および辞書: 新しい PROC FCMP データタイプのリファレンス

PROCFCMP および ASTORE

CMP は、ASTORE (アナリティックストア)スコアリングモデルをサポートしています。データの移動は、同じ TKCMP プログラム内でスコアと計算を一緒に実行することで軽減されます。PROC FCMP は、SAS クライアント上で ASTORE モデルをサポートします。

アナリティックストアとは何ですか?

アナリティックストアは、予測分析プロシジャの状態を含むバイナリファイルです。ランダムフォレストなどの予測分析手順からのこの状態は、モデル開発のトレーニングフェーズの結果を使用して作成されます。アナリティクスストアの重要な機能は、ホスト間を簡単に移送できることです。アナリティックストアは、コンパクトでユニバーサルなファイル形式です。ストアは、トレーニング後メモリを復元できる計算(メモリ)の状態を復元できる唯一の SAS コンポーネントの名前です。スコアリングのためのウォームリスタートとも呼ばれます。

状態とは何ですか?

状態とは、スコアリングなど、次のタスクに関連するすべてのメモリ項目のコピーです。SAVESTATE はこの情報のスナップショットを取ります。

- 公開情報(つまり、すべての分析エンジンに共通の情報)。パブリック情報の例には、入力変数のリスト、出力変数のリスト、形式、およびその他の要素が含まれます。
- 個人情報(つまり、ランダムなフォレストなどの特定の分析に固有の情報)。個人情報の例には、木の数、木自体、得点、および他の種類が含まれます。

スコアリングとは何ですか?

履歴データは、結果が分かっている場所です。この履歴データを使用してモデルをトレーニングし、入力変数を使用して新しいデータにスコアを付けます。新しいデータのスコアリングとは、履歴データを使用して構築されたモデルを使用して、新しいデータで結果を予測することです。

PROC ASTORE は何をしますか?

PROC ASTORE は入力データセットをスコアリングし、指定したアナリティックストアを使用して出力データセットを生成します。これは、各ステートメントがすぐに実行される対話型プロシジャです。PROC ASTORE は、これらの出力を生成します。

- DS2 手順を使用してローカルに実行できる DS2 スコアリングコードの種類
- SAS Viya で実行できる DS2 スコアリングコード。

PROC ASTORE は、アナリティックストアをクライアントとサーバーの間で移動することもでき、アナリティックストアに関する説明的な情報を提供できます。構文は次のとおりです。

SCORE
モデルのスコアリングを行います。

DESCRIBE
アナリティクスストアの名前を指定し、DS2 の基本スコアリングコードを生成します。

DOWNLOAD
指定されたアナリティックストアを CAS セッションから取得し、ローカルファイルシステムに格納します。

UPLOAD
指定したアナリティックストアをローカルファイルシステムから CAS のデータテーブルに移動します。

注: PROC ASTORE の詳細については、The ASTORE Procedure を参照してください。

例: PROCFCMP での ASTORE の使用

この例では、PROC FCMP で ASTORE を使用する方法を示します。また、ASTORE モデルの TKCMP スコアリングも示しています。

注: CMP オブジェクトを宣言するためのコードは、すべてのプラットフォームで同じです。

```
declare object myscore(astore);
```

注: SAS クライアントでは、score()メソッドは単一のパラメーターファイルパスを取ります。

- Windows:

```
call myscore.score("C:\models\_va_model208");
```

- UNIX:

```
call myscore.score("/userid/models/_va_model208");
```

注: CAS では、score()メソッドには、caslib と CAS テーブル名の 2 つの引数が必要です。

```
call myscore.score('CASUSER','_va_model208');
```

注: score オブジェクトは、ASTORE から入出力変数をログに出力する describe()メソッドもサポートしています。このメソッドは引数を取りません。

```
call myscore.describe();
```

このコードは、以前に作成された SVMCAS アクションを FCMP で実行します。

例のコード 21.2 ASTORE.SAS

```
title 'CMP ASTORE Test 1';

libname mydata "C:\project\my_data";
libname mymodel "C:\project\my_models";

/* Test a simple object model using the describe method */
/* Note: No input dataset is specified.   */
proc fcmp;
  declare object myscore(astore);
  call myscore.score("C:\project\my_models\_va_model208");

  /* Can use Proc FCMP to describe the local model */
  call myscore.describe();
run;quit;


/* Run the ASTORE model over an input dataset.  */
/* Write the results to an output dataset.      */
proc fcmp data=mydata.hmeq out=astore_fcmp_out;
  declare object myscore(astore);
  call myscore.score("C:\project\my_models\_va_model208");
run;quit;

proc print data=astore_fcmp_out(obs=10);
run;

title 'CMP ASTORE Test 2';

/* Using ASTORE scoring from a FCMP function */
proc fcmp outlib=work.score.funcs;

/* Return the probability value */
function astore(clage, clno, debtinc, delinq, ninq, value);
  declare object myscore(astore);
  call myscore.score("C:\project\my_models\_va_model208");
  return(_P_);
endsub;

/* Pass input values as arguments into the function */
/* Return all probability values from the score     */
subroutine astore2(clage, clno, debtinc, delinq, ninq, value,
                   _P_, P__EVENT_0, P__EVENT_1, I__EVENT_ $, _WARN_ $);
  outargs _P_, P__EVENT_0, P__EVENT_1, I__EVENT_, _WARN_;

  declare object myscore(astore);
  call myscore.score("C:\project\my_models\_va_model208");
endsub;
run;
quit;

/* Call these functions from Proc FCMP */

title 'CMP ASTORE Test 3';
```

```
proc fcmp data=mydata.hmeq inlib=work.score out=astore_fcmp_out2;
ds_p2 = astore(clage, clno, debtinc, delinq, ninq, value);
run;
quit;
proc print data=astore_fcmp_out2(obs=10);
run;

title 'CMP ASTORE Test 4';

proc fcmp data=mydata.hmeq(obs=10) inlib=work.score out=astore_fcmp_out3;
_P_=.;
P__EVENT_0=.;
P__EVENT_1=.;
I__EVENT_="";
_WARN_="";

call astore2(clage, clno, debtinc, delinq, ninq, value,
          _P_, P__EVENT_0, P__EVENT_1, I__EVENT_, _WARN_);
run;
quit;
proc print data=astore_fcmp_out3(obs=10);
run;


title 'CMP ASTORE Test 5';

/* You can call the function from the data step   */
/* We can perform ASTORE scoring using the data step */
options cmplib=work.score;
data astore_ds_out;
set mydata.hmeq;
ds_p = astore(clage, clno, debtinc, delinq, ninq, value);
run;
proc print data=astore_ds_out(obs=10);
run;
```

PROC ASTORE は、CAS からローカル分析ストア(ローカルファイル)にデータを移送します。そのフォームに入ると、データは CMP と新しい ASTORE オブジェクトによって使用されます。

例のコード 21.3　ASTORE_CAS.SAS

```
title 'CMP ASTORE Test 1';

/* CASPORT=0 means generate a port number for session connection */
/*options casuser=<userid> cashost="<machine>" casport=<port>;*/
options casuser=<userid> cashost="<machine>" casport=<port>;

/* This is how to start a CAS session */
/* CAS sessions live for the life of the SAS session, unless redefined or closed. */
/* Unix: cas mysession AUTHINFO='/userid/.authinfo';*/
/* Windows */
cas mysession authinfo='/userid/.authinfo';
libname sascas1 cas sessref="mysession" caslib="CASUSER";

libname mydata "C:\project\my_data";
```

```
libname mymodel "C:\project\my_models";

/* Our input data for SVM; Load into CAS */
data sascas1.hmeq;
set mydata.hmeq;
run;

/* Run action from VDMML */
proc cas;
   action builtins.loadactionset / actionSet='tkaasvm';
   action tkaasvm.svmtrain result=r /
c=1.0,
code={comment=false,fmtWdth=15,lineSize=200},
includeMissing=false,
maxiter=25,
noscale=false,
savestate={caslib="CASUSER",name="_va_model208"},
table={caslib="CASUSER",compOnDemand="false",compPgm="_va_calculated_208_1=round('BAD'n,1.0);
if (('_va_calculated_208_1'n = 0.0))then do;
_va_calculated_208_11= 0.0;
end;
else do;
_va_calculated_208_11= 1.0;
end;
;
_EVENT_=_va_calculated_208_11;
_va_FILTER_=(NOT(MISSING('_va_calculated_208_1'n))
AND NOT(MISSING('CLAGE'n)) AND NOT(MISSING('CLNO'n)) AND NOT(MISSING('DEBTINC'n))
AND NOT(MISSING('DELINQ'n)) AND NOT(MISSING('NINQ'n)) AND NOT(MISSING('VALUE'n)));
_va_calculated_208_14=NOT(('_va_FILTER_'n = 0.0));;",compVars={"_va_calculated_208_1","
_va_calculated_208_11","_EVENT_","_va_FILTER_","_va_calculated_208_14"},name="HMEQ",
onDemand="false",where="NOT('_va_calculated_208_14'n = 0)"},
emtarget={name="_EVENT_",options={levelType="BINARY"}},
tolerance=1.0E-6,
var={"CLAGE","CLNO","DEBTINC","DELINQ","NINQ","VALUE"}
outputTables={names={nobs="NObs",modelinfo="ModelInfo"}, replace="TRUE"};
run;

quit;

/* Use Proc ASTORE to describe the model in CAS */
/* Download the model locally                   */
proc astore;
  describe rstore=sascas1._va_model208;
  download rstore=sascas1._va_model208;
 store="C:\project\my_models\_va_model208";
  describe store="C:\project\my_models\_va_model208";
run;
quit;

options pagesize=max;
proc cas;
   loadactionset "table";
  table.fetch
  format=false
  maxRows=1
```

```
  sasTypes=TRUE
  table = {
    compOnDemand=TRUE
    caslib="CASUSER"
     name="hmeq"
    compPgm=
"declare object myscore(astore);
call myscore.score('CASUSER','_va_model208');"
     singlePass=TRUE
     compVars={"_P_", "P__EVENT_0" , "P__EVENT_1" , "I__EVENT_" ,"_WARN_"}
  };
run;
quit;
```

SETOPTION メソッド

CMP ASTORE メソッド SETOPTION は、異なるオプション値をとる ASTORE オブジェクトにオプションを渡します。PROC FCMP を使用したコードは次のとおりです。

```
routineCode = "
  declare object model_glmstore(astore);
  call model_glmstore.setoption('alpha', 0.05);
  call model_glmstore.setoption('COMPUTE_CONFIDENCE_LIMIT', 1);
  call model_glmstore.score('CASUSER','glmstore');
"
;
run;
quit;
```

FCMPACT アクションを使用するコードは次のとおりです。

```
routineCode = "
declare object model_glmstore(astore);
call model_glmstore.setoption('alpha', 0.05);
call model_glmstore.setoption('COMPUTE_CONFIDENCE_LIMIT', 1);
call model_glmstore.score('CASUSER','glmstore');
"
;
run;
quit;
```

PROC FCMP での Python 機能の使用

PROC FCMP は、Python オブジェクトを使用して Python で記述された関数のサブミットと実行をサポートします。Python オブジェクトを使用するための完全なガイドは、Python オブジェクトの使用の章(SAS コンポーネントオブジェクト: リファレンス)にあります。

VNAME 引数を使用して名前で変数を返す

2023.04 以降、PROC FCMP は、PROC FCMP 関数およびサブルーチンでの VNAME 引数の使用をサポートします。デフォルトでは、PROC FCMP 関数またはサブルーチンが呼び出されるとき、変数は値で渡されます。PROC FCMP VNAME 引数を使用すると、VNAME 関数のような関数やサブルーチンに名前を付けて変数を渡すことができます。PROC FCMP VNAME 引数を作成する場合は、引数名の後に vname を指定します。

> 重要　PROC FCMP VNAME 引数は OUTARGS リストでは使用できません。

> 重要　PROC FCMP VNAME 引数は変数名でなければなりません。式または配列添字の指定は、PROC FCMP VNAME 引数ではサポートされていません。

次の例では、PROC FCMP 関数 FOO とサブルーチン FOO_SUB が、PROC FCMP VNAME 引数を使用して定義されています。追加関数 FOO_VNAME は、VNAME 引数の代わりに VNAME 関数を使用して比較するために定義されています。入力変数は **callarg="hello"**です。関数やサブルーチンを呼び出すと、値**"hello"**の代わりに callarg が返されます。

例のコード 21.4　PROC FCMP VNAME 引数の使用

```
proc fcmp;

function foo(defarg vname) $;
length s $40;
s = """ || trim(defarg) || """;
return(s);
endfunc;

function foo_vname(defarg $) $;
length s $40;
s = """ || trim(defarg) || """;
return(s);
endfunc;

subroutine foo_sub(retarg $, defarg vname);
outargs retarg;
length s $40;
s = """ || trim(defarg) || """;
returnarg = s;
endsub;

length x $20;
length y $20;
length z $20;
length callarg $20;
callarg = "hello";
```

```
x = foo(callarg);
y = foo_vname(vname(callarg));
call foo_sub(z, callarg);

put x= y= z=;

if (x = y) then
  put 'Success on function!';
else
  put 'Failure on function';

if (z = y) then
  put 'Success on subroutine!';
else
  put 'Failure on subroutine';

run;
quit;
```

SAS ログは次の結果を返します。

```
x='callarg' y='callarg' z='callarg'
Success on function!
Success on subroutine!
```

例: FCMP プロシジャ

例 1: 関数の作成と、DATA ステップからの関数の呼び出し

要素:

PROC FCMP ステートメントオプション
    OUTLIB=
DATA ステップ

詳細

この例では、PROC FCMP で関数を作成してこれを DATA ステップで使用することにより、治験における研究日を計算する方法を示します。

プログラム

```
proc fcmp outlib=work.funcs.trial;

  function study_day(intervention_date, event_date);

    n=event_date - intervention_date;
      if n >= 0 then
        n=n + 1;
      return(n);
  endsub;

options cmplib=work.funcs;

data _null_;
  start='15Feb2010'd;
  today='27Mar2010'd;
  sd=study_day(start, today);

  put sd=;

run;
```

プログラムの説明

コンパイルされた関数と CALL ルーチンが書き込まれる出力パッケージの名前を指定します。 パッケージは、データセット work.Funcs に保存されます。

```
proc fcmp outlib=work.funcs.trial;
```

STUDY_DAY という関数を作成します。 STUDY_DAY が Trial というパッケージに作成され、2 つの数値入力引数が含まれています。

```
function study_day(intervention_date, event_date);
```

DATA ステップの IF ステートメントを使用して EVENT_DATE を計算します。 DATA ステップ構文を使用して、EVENT_DATE と INTERVENTION_DATE の間の差異を計算します。INTERVENTION_DATE より前の日は-1 で始まり、それ以下になります。INTERVENTION_DATE 以降の日は 1 で始まり、それ以上になります(この関数は、研究日に対し 0 を返しません)。

```
  n=event_date - intervention_date;
    if n >= 0 then
      n=n + 1;
    return(n);
endsub;
```

CMPLIB=システムオプションを使用して、プログラムコンパイラ時に含めるコンパイラサブルーチンを含む SAS データセットを指定します。

```
options cmplib=work.funcs;
```

DATA ステップを作成して、関数 STUDY_DAY に対して値を生成します。 関数は、開始日と今日の日付を使用して、値を計算します。STUDY_DAY は、DATA ステップから呼び出されます。DATA ステップで STUDY_DAY への呼び出しが発生すると、この関数は関数の従来のライブラリで検出されません。STUDY_DAY を含むパッケージに対し CMPLIB システムオプションで指定される各データセットが検索されます。この場合、work.Funcs.Trial の STUDY_DAY が検出されます。

```
data _null_;
  start='15Feb2010'd;
  today='27Mar2010'd;
  sd=study_day(start, today);
```

出力を SAS ログに書き出します。

```
  put sd=;
```

SAS プログラムを実行します。

```
run;
```

ログ

例のコード 21.6　関数の作成および DATA ステップからの呼び出しの結果

```
sd=41
```

例 2: PROC FCMP による関数の作成および保存

要素:

PROC FCMP ステートメントオプション
  OUTLIB=
OUTARGS ステートメント

詳細

この例では、PROC FCMP を使用して本例で使用される関数を作成し保存する方法を示します。

プログラム

```
proc fcmp outlib=work.exsubs.pkt1;

  subroutine calc_years(maturity, current_date, years);
    outargs years;
    years=(maturity - current_date) / 365.25;
  endsub;

function garkhprc(type$, buysell$, amount,E, t, S, rd, rf, sig);
    if buysell="Buy" then sign=1.;
    else do;
      if buysell="Sell" then sign=-1.;
      else sign=.;
```

```
    end;

    if type="Call" then
      garkhprc=sign * amount * garkhptprc(E, t, S, rd, rf, sig);
    else do;
      if type="Put" then
        garkhprc=sign * amount * garkhptprc(E, t, S, rd, rf, sig);
      else garkhprc=.;
    end;

  return(garkhprc);

  run;

  endfunc;
```

プログラムの説明

関数パッケージ情報が保存されるエントリを指定します。 パッケージは、3 レベルの名前です。

```
  proc fcmp outlib=work.exsubs.pkt1;
```

満期までの年数を計算するための関数を作成します。 CALC_YEARS という汎用関数が宣言され、日数として保存される日付変数から満期までの年数が計算されます。OUTARGS ステートメントは、CALC_YEARS によって更新される変数を指定します。

```
    subroutine calc_years(maturity, current_date, years);
      outargs years;
      years=(maturity - current_date) / 365.25;
    endsub;
```

FX オプションの Garman-Kohlhagen 価格設定のための関数を作成します。 GARKHPRC という関数が宣言されます。これは FX オプションに対する Garman-Kohlhagen 価格設定を計算します。関数は、SAS 関数 GARKHCLPRC と GARKHPTPRC を使用します。

```
  function garkhprc(type$, buysell$, amount,E, t, S, rd, rf, sig);
    if buysell="Buy" then sign=1.;
    else do;
      if buysell="Sell" then sign=-1.;
      else sign=.;
    end;

    if type="Call" then
      garkhprc=sign * amount * garkhptprc(E, t, S, rd, rf, sig);
    else do;
      if type="Put" then
        garkhprc=sign * amount * garkhptprc(E, t, S, rd, rf, sig);
      else garkhprc=.;
    end;
```

RETURN ステートメントは、GARKHPRC 関数の値を返します。

```
  return(garkhprc);
```

FCMP プロシジャを実行します。 RUN ステートメントは、FCMP プロシジャを実行します。

```
run;
```

関数を閉じる。 endfunc ステートメントは関数を閉じます。

```
endfunc;
```

ログ

例のコード 21.7　関数の保存場所

```
注: garkhprc 関数は work.exsubs.pkt1 に保存されました。
注: calc_years 関数は work.exsubs.pkt1 に保存されました。
```

例 3: FUNCTION ステートメントでの数値データの使用

詳細

次の例では、PROC FCMP の FUNCTION ステートメントへの入力として、数値データを使用します。

プログラム

```
proc fcmp;
  function inverse(in);
    if in=0 then inv=.;
    else inv=1/in;
    return(inv);
  endfunc;
run;
```

例 4: FUNCTION ステートメントでの文字データの使用

詳細

次の例では、PROC FCMP の FUNCTION ステートメントへの入力として、文字データを使用します。FUNCTION TEST からの出力には 12 バイト長が割り当てられています。

プログラム

```
options cmplib=work.funcs;

proc fcmp outlib=work.funcs.math;
  function test(x $) $ 12;
  if x='yes' then
    return('si si si');
    else
    return('no');
  endfunc;
run;

data _null_;
  spanish=test('yes');
  put spanish=;
run;
```

ログ

例のコード 21.8　PROC FCMP の FUNCTION ステートメントで文字データを使用した結果

```
spanish=si si si
```

例 5: 変数の引数を配列として使用する

詳細

次の例に、変数の引数を受け入れる配列を示します。例では、summation 関数が次のように呼び出し可能であることを示します: sum=summation(1,2,3,4,5);

注: DATA ステップからこの関数を呼び出すときは、配列として *VARARGS* を指定する必要があります。

プログラム

```
options cmplib=work.funcs;

proc fcmp outlib=work.funcs.temp;
function summation (b[*]) varargs;
   total=0;
   do i=1 to dim(b);
      total=total + b[i];
   end;
return(total);
endfunc;
sum=summation(1, 2, 3, 4, 5);
  put sum=;
run;
```

例 6: SUBROUTINE ステートメントの CALL ステートメントとの併用

詳細

次に、SUBROUTINE ステートメントの例を示します。SUBROUTINE ステートメントは、CALL ステートメントと使用できるコードの独立した計算ブロックを作成します。

プログラム

```
proc fcmp outlib=work.funcs.temp;
subroutine inverse(in,inv) group="generic";
  outargs inv;
  if in=0 then inv=.;
  else inv=1/in;
endsub;

options cmplib=work.funcs;
data _null_;
  x=5;
  call inverse(x, y);
  put x= y=;
run;
```

ログ

例のコード 21.9　PROC FCMP で SUBROUTINE ステートメントを使用した結果

```
x=5 y=0.2
```

例 7: ユーザー定義関数での Graph Template Language (GTL)の使用

要素:

PROC FCMP 関数
- OSCILLATE
- OSCILLATEBOUND

その他のプロシジャ
- PROC TEMPLATE
- PROC SGRENDER

詳細

次の例に、GTL EVAL 関数での関数の使用方法を示します。ここでは新しい曲線タイプ(oscillate と oscillateBound)を定義する関数の定義方法を示します。これらの関数は、GTL EVAL 関数で seriesplot と bandplot で表される新しい列の計算に使用できます。

プログラム

```
proc fcmp outlib=work.funcs.curves;
  function oscillate(x,amplitude,frequency);
       if amplitude le 0 then amp=1; else amp=amplitude;
         if frequency le 0 then freq=1; else freq=frequency;
       y=sin(freq*x)*constant("e")**(-amp*x);
       return (y);
 endfunc;

 function oscillateBound(x,amplitude);
     if amplitude le 0 then amp=1; else amp=amplitude;
     y=constant("e")**(-amp*x);
     return(y);
 endfunc;
run;


 options cmplib=work.funcs;

data range;
  do time=0 to 2 by .01;
  output;
  end;
run;

proc template ;
  define statgraph damping;
  dynamic X AMP FREQ;
  begingraph;
     entrytitle "Damped Harmonic Oscillation";
     layout overlay / yaxisopts=(label="Displacement");
       if (exists(X) and exists(AMP) and exists(FREQ))
            bandplot x=X  limitlower=eval(-oscillateBound(X,AMP))
            limitupper=eval(oscillateBound(X,AMP));
            seriesplot x=X y=eval(oscillate(X,AMP,FREQ));
        endif;
      endlayout;
  endgraph;
  end;
  run;

 proc sgrender data=range template=damping;
    dynamic x="Time" amp=10 freq=50 ;
 run;
```

プログラムの説明

OSCILLATE 関数を作成します。

```
proc fcmp outlib=work.funcs.curves;
  function oscillate(x,amplitude,frequency);
       if amplitude le 0 then amp=1; else amp=amplitude;
         if frequency le 0 then freq=1; else freq=frequency;
       y=sin(freq*x)*constant("e")**(-amp*x);
```

```
      return (y);
 endfunc;
```

OSCILLATEBOUND 関数を作成します。

```
function oscillateBound(x,amplitude);
    if amplitude le 0 then amp=1; else amp=amplitude;
    y=constant("e")**(-amp*x);
    return(y);
 endfunc;
run;
```

PROC SGRENDER によって使用される Range と呼ばれるデータセットを作成します。

```
options cmplib=work.funcs;

data range;
  do time=0 to 2 by .01;
  output;
  end;
run;
```

TEMPLATE プロシジャを使用して、SAS 出力の表示をカスタマイズできます。

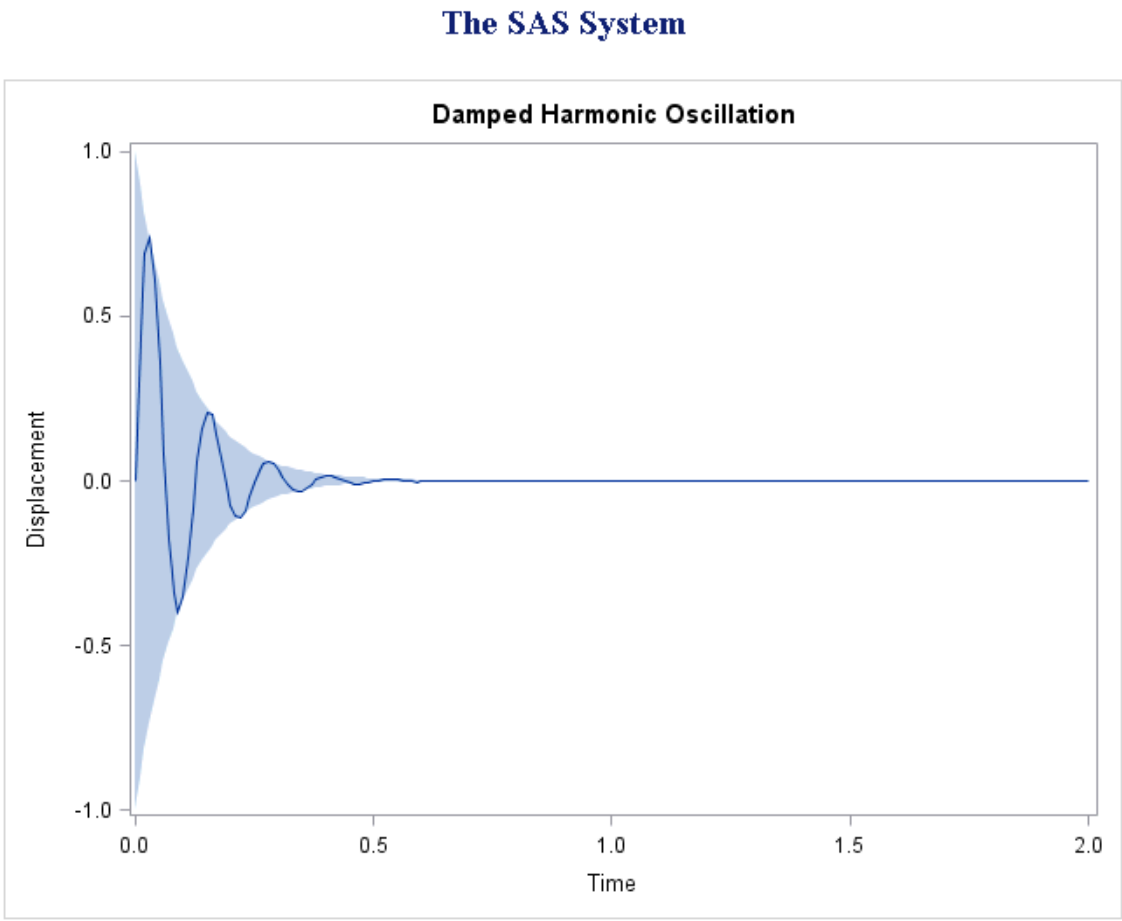
```
proc template ;
  define statgraph damping;
  dynamic X AMP FREQ;
  begingraph;
    entrytitle "Damped Harmonic Oscillation";
    layout overlay / yaxisopts=(label="Displacement");
      if (exists(X) and exists(AMP) and exists(FREQ))
          bandplot x=X  limitlower=eval(-oscillateBound(X,AMP))
          limitupper=eval(oscillateBound(X,AMP));
          seriesplot x=X y=eval(oscillate(X,AMP,FREQ));
      endif;
     endlayout;
  endgraph;
  end;
  run;
```

SGRENDER プロシジャを使用して、入力変数を含むデータセットを識別し、出力用の statgraph テンプレートを割り当てます。

```
proc sgrender data=range template=damping;
    dynamic x="Time" amp=10 freq=50 ;
run;
```

出力: 関数での GTL の使用

アウトプット 21.7　ユーザー定義関数で GTL を使用した結果



例 8: データセットの各行を標準化する

要素:　PROC FCMP 関数
- RUN_MACRO
- RUN_SASFILE
- READ_ARRAY
- WRITE_ARRAY

詳細

この例では、データセットの各行の標準化の方法を示します。

プログラム

```
data numbers;
  drop i j;
  array a[5];
  do j=1 to 5;
  do i=1 to 5;
      a[i] = rand('Uniform') * (i+123.234);
    end;
    output;
    end;
  run;

%macro standardize;
%let dsname=%sysfunc(dequote(&dsname));
%let colname=%sysfunc(dequote(&colname));
proc standard data=&dsname mean=&MEAN std=&STD out=_out;
  var &colname;
run;
data &dsname;
  set_out;
run;
%mend standardize;

proc fcmp outlib=work.ds.functions;
  subroutine standardize(x[*], mean, std);
    outargs x;

    rc=write_array('work._TMP_', x, 'x1');
    dsname='work._TMP_';
      colname='x1';
      rc=run_macro('standardize', dsname, colname, mean, std);
    array x2[1]_temporary_;
      rc=read_array('work._TMP_', x2);
    if dim(x2)=dim(x) then do;
      do i=1 to dim(x);
        x[i]=x2[i];
       end;
      end;
  endsub;
run;

options cmplib=(work.ds);
data numbers2;
  set numbers;
  array a[5];
  array t[5]_temporary_;
  do i=1 to 5;
    t[i]=a[i];
  end;
  call standardize(t, 0, 1);
  do i=1 to 5;
    a[i]=t[i];
  end;
  output;
run;

proc print data=work.numbers2;
```

```
run;
```

プログラムの説明

5 行の乱数を含むデータセットを作成します。

```
data numbers;
  drop i j;
  array a[5];
  do j=1 to 5;
  do i=1 to 5;
      a[i] = rand('Uniform') * (i+123.234);
    end;
    output;
    end;
  run;
```

データセットを MEAN および STD の指定値で標準化するためのマクロを作成します。

```
%macro standardize;
%let dsname=%sysfunc(dequote(&dsname));
%let colname=%sysfunc(dequote(&colname));
proc standard data=&dsname mean=&MEAN std=&STD out=_out;
  var &colname;
run;
data &dsname;
  set_out;
run;
%mend standardize;
```

FCMP 関数を使用して、WRITE_ARRAY を呼び出します。これにより、データがデータセットに書き込まれます。RUN_MACRO を呼び出して、データセットのデータを標準化します。WRITE_ARRAY を呼び出して、データをデータセットに書き込みます。READ_ARRAY を呼び出して、標準化されたデータを配列に読み込みます。

```
proc fcmp outlib=work.ds.functions;
  subroutine standardize(x[*], mean, std);
    outargs x;

    rc=write_array('work._TMP_', x, 'x1');
    dsname='work._TMP_';
      colname='x1';
      rc=run_macro('standardize', dsname, colname, mean, std);
    array x2[1]_temporary_;
      rc=read_array('work._TMP_', x2);
    if dim(x2)=dim(x) then do;
      do i=1 to dim(x);
        x[i]=x2[i];
       end;
      end;
  endsub;
run;
```

DATA ステップの行ごとに関数を実行します。

```
options cmplib=(work.ds);
data numbers2;
  set numbers;
  array a[5];
  array t[5]_temporary_;
  do i=1 to 5;
    t[i]=a[i];
  end;
  call standardize(t, 0, 1);
  do i=1 to 5;
    a[i]=t[i];
  end;
  output;
run;
```

出力を書き込みます。

```
proc print data=work.numbers2;
run;
```

出力: データセットの行の標準化

アウトプット 21.8　データセットの各行を標準化した結果

The SAS System

Obs	a1	a2	a3	a4	a5
1	45.088	93.3237	104.908	35.152	23.5725
2	90.552	9.7548	92.696	89.987	97.9810
3	60.596	22.7409	19.284	50.079	58.9264
4	106.778	49.1589	22.885	20.641	30.1756
5	34.812	71.3746	44.248	101.808	79.3731

参考文献

SAS Institute Inc. 2013. *SAS コンポーネントオブジェクト: リファレンス*. Cary, NC: SAS Institute Inc.

Henrick, A., D. Erdman, and S. Christian. 2013. “Hashing in PROC FCMP to Enhance Your Productivity.” *Proceedings of the SAS Global Forum 2013 Conference*, Cary, NC.SAS Institute Inc., 1–15. Available at http://support.sas.com/resources/papers/proceedings13/129-2013.pdf.

22

FCMP 特殊関数と CALL ルーチン

特殊関数と CALL ルーチンの概要

FCMP プロシジャは、特殊用途関数の小規模なセットを提供します。これらの関数はユーザー定義 FCMP 関数から呼び出すことができますが、DATA ステップから直接呼び出すことはできません。これらの関数を DATA ステップで使用するには、別のユーザー定義 FCMP 関数内で特殊関数をラップする必要があります。

注: 特殊関数を直接プロシジャで呼び出すことができますが、DATA ステップで呼び出すことはできません。

カテゴリ別の関数と CALL ルーチン

表 22.1 FCMP 関数と CALL ルーチンのカテゴリ

カテゴリ	説明
関数内からの SAS コードの呼び出し	SAS コードを関数内から呼び出せる 2 つの関数が使用できます。RUN_MACRO 関数は、事前定義された SAS マクロを実行します。RUN_SASFILE 関数は、指定するファイル参照名から SAS コードを実行します。
C Helper	複数の Helper 関数には、PROC FCMP での C 言語構築を処理するパッケージが提供されています。ほとんどの C 言語構築は、構築が参照可能になる前、または PROC FCMP によって使用可能になる前に PROC PROTO によって作成されるパッケージで定義する必要があります。ISNULL 関数、SETNULL ルーチン、および STRUCTINDEX CALL ルーチンが SAS 言語を拡張し、SAS 言語に合わない C 言語構築を処理するために追加されました。
配列	PROC FCMP では、配列を読み込むための READ_ARRAY 関数、配列をデータセットに書き込むための WRITE_ARRAY 関数を提供しています。この機能により、PROC FCMP 配列データを SAS プログラム、マクロ、プロシジャによって 処理できるようになります。
行列操作	FCMP プロシジャでは、宣言された配列での単一の行列操作を実行するための多くの CALL ルーチンを提供しています。これらの CALL ルーチンは、自動的に FCMP プロシジャによって提供されます。ZEROMATRIX、FILLMATRIX、IDENTITY を除く、次にリストされている CALL ルーチンでは、欠損値を含む行列または配列をサポートしていません。

カテゴリ	説明
特殊な目的	FCMP プロシジャでは、2 つの特殊な目的のある関数 INVCDF と LIMMOMENT が提供されます。INVCDF 関数は、累積分布関数(CDF)を定義した分布から分位点を計算します。LIMMOMENT 関数は、累積分布関数(CDF)を定義した分布の制限されたモーメントを計算します。

カテゴリ	言語要素	説明
C Helper	CALL SETNULL ルーチン (p. 759)	構造のポインター要素を Null に設定します。
	CALL STRUCTINDEX ルーチン (p. 760)	構造の配列の各構造要素にアクセスできます。
	ISNULL 関数 (p. 767)	構造のポインター要素が Null かどうかを決定します。
暗黙値の計算	SOLVE 関数 (p. 781)	Gauss-Newton 法を使用して、関数の暗黙値を計算します。
関数内からの SAS コードの呼び出し	RUN_MACRO 関数 (p. 775)	事前定義された SAS マクロを実行します。
	RUN_SASFILE 関数 (p. 780)	指定するファイル参照名で SAS コードを実行します。
行列操作	CALL ADDMATRIX ルーチン (p. 746)	2 つの行列、または 1 つの行列およびスカラーの elementwise 加算を実行します。
	CALL CHOL ルーチン (p. 747)	指定対称行列に対する Cholesky 分解を計算します。
	CALL DET ルーチン (p. 749)	正方である必要がある指定行列の行列式を計算します。
	CALL ELEMMULT ルーチン (p. 751)	2 つの行列の elementwise 乗算を実行します。
	CALL EXPMATRIX ルーチン (p. 752)	行列 etA を返します(入力行列 A、乗数 t の場合)。
	CALL FILLMATRIX ルーチン (p. 754)	入力行列のすべての要素値を指定値と置き換えます。
	CALL IDENTITY ルーチン (p. 755)	入力行列を単位行列に変換します。
	CALL INV ルーチン (p. 756)	正則な正方行列でなければならない、提供されている入力行列の逆行列である行列を計算します。
	CALL MULT ルーチン (p. 757)	2 つの入力行列の乗法積を計算します。
	CALL POWER ルーチン (p. 758)	正方行列を指定されているスカラー値に累乗します。
	CALL SUBTRACTMATRIX ルーチン (p. 761)	2 つの行列、または 1 つの行列およびスカラーの element-wide 減算を実行します。

カテゴリ	言語要素	説明
	CALL TRANSPOSE ルーチン (p. 762)	行列の転置を返します。
	CALL ZEROMATRIX ルーチン (p. 763)	数値入力行列のすべての要素値を 0 と置き換えます。
特殊な目的のある関数	INVCDF 関数 (p. 764)	累積分布関数(CDF)を定義した分布の分位点を計算します。
	LIMMOMENT 関数 (p. 769)	累積分布関数(CDF)を定義した分布の制限したモーメントを計算します。
配列	CALL DYNAMIC_ARRAY ルーチン (p. 750)	関数内で宣言される配列のサイズを効率的に変更できます。
	NDIMS 関数 (p. 772)	配列にある次元数を返します。
	READ_ARRAY 関数 (p. 773)	データを SAS データセットから PROC FCMP 配列変数に読み込みます。
	WRITE_ARRAY 関数 (p. 785)	データを PROC FCMP 配列変数から SAS プログラム、マクロ、プロシジャによって 使用可能なデータセットに書き込みます。

ディクショナリ

CALL ADDMATRIX ルーチン

2 つの行列、または 1 つの行列およびスカラーの elementwise 加算を実行します。

カテゴリ: 行列操作

要件 すべての 入力行列と出力行列には、同じ次元が含まれている必要があります。

構文

CALL ADDMATRIX(*X*, *Y*, *Z*);

必須引数

X
次元 *m* x *n* (つまり X[*m*, *n*])を含む入力行列、またはスカラーを指定します。

Y
次元 *m* x *n* (つまり Y[*m*, *n*])を含む入力行列、またはスカラーを指定します。

Z
次のような次元 *m* x *n* (つまり Z[*m*, *n*])を含む出力行列を指定します。次のようになります。

$$Z = X + Y$$

例

次の例では ADDMATRIX CALL ルーチンを使用しています。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
  array mat2[3,2] (0.2, 0.38, -0.12, 0.98, 2, 5.2);
  array result[3,2];
  call addmatrix(mat1, mat2, result);
  call addmatrix(2, mat1, result);
  put result=;
quit;
```

アウトプット 22.1　ADDMATRIX CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

result[1, 1]=2.3 result[1, 2]=1.22 result[2, 1]=1.18 result[2, 2]=2.54
result[3, 1]=3.74 result[3, 2]=3.2
```

CALL CHOL ルーチン

指定対称行列に対する Cholesky 分解を計算します。

カテゴリ:　行列操作

別名:　CHOLESKY_DECOMP

要件　入力行列と出力行列は正方で、同じ次元が含まれている必要があります。X は対称正定値で、Y は下三角行列である必要があります。

構文

CALL CHOL(*X*, *Y* <, *validate*>);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む対称正定値入力行列を指定します。

Y
次元 *m* x *m* (つまり、*m*, *m*])を含む出力行列を指定します。次のように、この変数には Cholesky 分解が含まれています。

$$Z = YY^*$$

Y は正の対角エントリを含む下三角行列で、Y*は Y の共役転置を示します。

注: X が対称正定値でない場合、Y には欠損値が入力されます。

オプション引数

validate
エラーチェックを行わないことによって処理速度を上げることができる任意の引数を指定します。引数は、次の値を受け取ることができます。

0 行列 X の対称性がチェックされます。これは、*validate* 引数が省略された場合のデフォルトです。

1 行列は、対称と見なされます。

例

次の例では、CHOL CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array xx[3,3] 2 2 3 2 4 2 3 2 6;
  array yy[3,3];
  call chol(xx, yy, 0);
  do i=1 to 3;
    put yy[i, 1] yy[i, 2] yy[i, 3];
  end;
run;
```

アウトプット 22.2　PROC FCMP ルーチンおよび CHOL CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

1.4142135624 0 0
1.4142135624 1.4142135624 0
2.1213203436 -0.707106781 1
```

CALL DET ルーチン

正方である必要がある指定行列の行列式を計算します。

カテゴリ: 行列操作

要件 入力行列 X は、正方にしてください。

構文

CALL DET(*X*, *a*);

必須引数

X
次元 *m* x *n* (つまり、X[*m*, *n*])を含む入力行列を指定します。

a
次のように、返された特定値を指定します。

$a = |X|$

詳細

行列式、固有値の積は、単一の数値です。行列の行列式がゼロの場合、行列は特異です(つまり、逆行列は含まれていません)。メソッドは LU 分解を実行し、対角線の積を収集します(Forsythe, Malcolm, and Moler 1967)。詳細については、*SAS/IML User's Guide* を参照してください。

例

次の例では、DET CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,3] (.03, -0.78, -0.82, 0.54, 1.74,
                   1.2, -1.3, 0.25, 1.49);
  call det(mat1, result);
  put result=;
quit;
```

アウトプット 22.3　DET CALL ルーチンの結果

```
                                The SAS System                          1

                              The FCMP Procedure

result=-0.052374
```

CALL DYNAMIC_ARRAY ルーチン

関数内で宣言される配列のサイズを効率的に変更できます。

カテゴリ:　配列

構文

CALL DYNAMIC_ARRAY(*array–name*, *new-dimension1–size* <, *new-dimension2–size*, ...>);

必須引数

array-name
　一時配列の名前を指定します。

new-dimension-size
　一時配列に新しいサイズを指定します。

詳細

関数と CALL ルーチンで宣言された配列は、/NOSYMBOLS オプションで宣言された配列と同様にサイズ変更可能です。その他の配列はサイズ変更できません。

DYNAMIC_ARRAY CALL ルーチンは配列のサイズを、提供するターゲットの次元に合わせて動的に変更しようとします。つまり、配列は動的である必要があります。配列は関数またはサブルーチンのいずれか、または/NOSYMBOLS オプションで宣言する必要があります。

DYNAMIC_ARRAY CALL ルーチンにはサイズ変更する配列と、配列の次元ごとに新しいサイズが渡されます。ALLPERMK ルーチンでは、並べ替えている要素数のサイズであるスクラッチ配列が必要になります。関数の作成時はこの値はパラメーター *n* として渡されるため、不明です。動的配列で、ルーチンは必要なメモリ量を割り当てることができます。すべての可能なケースの処理に十分な大きさの配列を作成する必要はありません。

動的配列の使用時は、サポートは PROC FCMP ルーチンに制限されます。配列のサイズが変更されると、サイズ変更された配列はサイズ変更を行ったルーチン内での

み使用できます。DATA ステップ配列をサイズ変更することも、PROC FCMP 動的配列を DATA ステップに返すこともできません。

例

次の例では、TEMP という一時配列を作成します。配列領域のサイズは、関数に渡されるパラメーターによって異なります。

```
proc fcmp;
  function avedev_wacky(data[*]);

  length=dim(data);
  array temp[1] /nosymbols;
  call dynamic_array(temp, length);

  mean=0;
  do i=1 to length;
    mean += data[i];
    if i>1 then temp[i]=data[i-1];
    else temp[i]=0;
  end;
  mean=mean/length;

  avedev=0;
  do i=1 to length;
    avedev += abs((data[i])-temp[i] /2-mean);
  end;
  avedev=avedev/length;

  return(avedev);
endsub;

array data[10];

do i = 1 to 10;

  data[i] = i;
end;

avedev = avedev_wacky(data);
run;
```

CALL ELEMMULT ルーチン

2 つの行列の elementwise 乗算を実行します。

カテゴリ: 行列操作

要件 すべての 入力行列と出力行列には、同じ次元が含まれている必要があります。

構文

CALL ELEMMULT(*X*, *Y*, *Z*);

必須引数

X
次元 *m* x *n* (つまり、X[*m*, *n*])を含む入力行列を指定します。

Y
次元 *m* x *n* (つまり、Y[*m*, *n*])を含む入力行列を指定します。

Z
次元 *m* x *n* (つまり、Z[*m*, *n*])を含む出力行列を指定します。

例

次の例では、ELEMMULT CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
  array mat2[3,2] (0.2, 0.38, -0.12, 0.98, 2, 5.2);
  array result[3,2];
  call elemmult(mat1, mat2, result);
  call elemmult(2.5, mat1, result);
  put result=;
 quit;
```

アウトプット 22.4　ELEMMULT CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

result[1, 1]=0.75 result[1, 2]=-1.95 result[2, 1]=-2.05 result[2, 2]=1.35
result[3, 1]=4.35 result[3, 2]=3
```

CALL EXPMATRIX ルーチン

行列 e^{tA} を返します(入力行列 *A*、乗数 *t* の場合)。

カテゴリ:　行列操作

要件　入力行列と出力行列は正方で、同じ次元が含まれている必要があります。*t* は、任意のスカラー値です。

構文

CALL EXPMATRIX(*X*, *t*, *Y*);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む入力行列を指定します。

t
倍精度のスカラー値を指定します。

Y
次のように、次元 *m* x *m* (つまり、Y[*m*, *m*])を含む出力行列を指定します。

$$Y = \varepsilon^{tX}$$

詳細

EXPMATRIX CALL では、Padé 概算アルゴリズム(Golub and van Loan (1989), p. 558 に記載)を使用します。このモジュールは、行列の各エントリを累乗しないことに注意してください。詳細については、*SAS/IML User's Guide* の EXPMATRIX ドキュメントを参照してください。

例

次の例では、EXPMATRIX CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,3] (0.3, -0.78, -0.82, 0.54, 1.74,
                   1.2, -1.3, 0.25, 1.49);
  array result[3,3];
  call expmatrix(mat1, 3, result);
  put result=;
quit;
```

アウトプット 22.5　EXPMATRIX CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

result[1, 1]=365.58043585 result[1, 2]=-589.6358476 result[1, 3]=-897.1034008
result[2, 1]=-507.0874798 result[2, 2]=838.64570481 result[2, 3]=1267.3598426
result[3, 1]=-551.588816 result[3, 2]=858.97629382 result[3, 3]=1324.8187125
```

CALL FILLMATRIX ルーチン

入力行列のすべての要素値を指定値と置き換えます。

カテゴリ: 行列操作

注: FILLMATRIX CALL ルーチンを多次元数値配列と使用できます。

構文

CALL FILLMATRIX(*X*, *Y*);

必須引数

X
入力数値行列を指定します。

Y
行列に入力する数値を指定します。

例

次の例では、FILLMATRIX CALL ルーチンを使用します。

```
options pageno=1 nodate ls=80 ps=64;

proc fcmp;
  array mat1[3, 2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
  call fillmatrix(mat1, 99);
  put mat1=;
quit;
```

アウトプット 22.6　FILLMATRIX CALL ルーチンの結果

```
                                The SAS System                                1

                               The FCMP Procedure

mat1[1, 1]=99 mat1[1, 2]=99 mat1[2, 1]=99 mat1[2, 2]=99 mat1[3, 1]=99
mat1[3, 2]=99
```

CALL IDENTITY ルーチン

入力行列を単位行列に変換します。

カテゴリ: 行列操作

要件 入力行列は正方にしてください。

注: 行列の対角線要素値が 1 に設定され、その他の値が 0 に設定されます。

構文

CALL IDENTITY(*X*);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む入力行列を指定します。

例

次の例では、IDENTITY CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,3] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2,
                   -1.3, 0.25, 1.49);
  call identity(mat1);
  put mat1=;
quit;
```

アウトプット 22.7 IDENTITY CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

mat1[1, 1]=1 mat1[1, 2]=0 mat1[1, 3]=0 mat1[2, 1]=0 mat1[2, 2]=1 mat1[2, 3]=0
mat1[3, 1]=0 mat1[3, 2]=0 mat1[3, 3]=1
```

CALL INV ルーチン

正則な正方行列でなければならない、提供されている入力行列の逆行列である行列を計算します。

カテゴリ: 行列操作

要件 入力行列と出力行列は正方で、同じ次元が含まれている必要があります。

構文

CALL INV(*X*, *Y*);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む入力行列を指定します。

Y
次のように、次元 *m* x *m* (つまり、Y[*m*, *m*])を含む出力行列を指定します。

$$Y[m,m] = X'[m,m]$$

'は逆行列を示し、

$$X \times Y = Y \times X = I$$

I は単位行列です。

例

次の例では、INV CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,3] (0.3, -0.78, -0.82, 0.54, 1.74,
                   1.2, -1.3, 0.25, 1.49);
  array result[3,3];
  call inv(mat1, result);
  put result=;
quit;
```

アウトプット 22.8　INV CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

result[1, 1]=4.0460407887 result[1, 2]=1.6892917399 result[1, 3]=0.8661767509
result[2, 1]=-4.173108283 result[2, 2]=-1.092427483 result[2, 3]=-1.416802558
result[3, 1]=4.230288655 result[3, 2]=1.6571719011 result[3, 3]=1.6645841716
```

CALL MULT ルーチン

2 つの入力行列の乗法積を計算します。

カテゴリ:　行列操作

要件　最初の入力行列の列数が 2 番目の行列の行数と同じである必要があります。

構文

CALL MULT(*X*, *Y*, *Z*);

必須引数

X
次元 *m* x *n* (つまり、X[*m*, *n*])を含む入力行列を指定します。

Y
次元 *n* x *p* (つまり、Y[*n*, *p*])を含む入力行列を指定します。

Z
次のように、次元 *m* x *p* (つまり、Z[*m*, *p*])を含む出力行列を指定します。

$$Z[m,p] = X[m,n] \times Y[n,p]$$

例

次の例では、MULT CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[2,3] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
  array mat2[3,2] (1, 0, 0, 1, 1, 0);
  array result[2,2];
  call mult(mat1, mat2, result);
  put result=;
quit;
```

アウトプット 22.9　MULT CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

result[1, 1]=-0.52 result[1, 2]=-0.78 result[2, 1]=1.74 result[2, 2]=1.74
```

CALL POWER ルーチン

正方行列を指定されているスカラー値に累乗します。

カテゴリ:　行列操作

制限事項:　行列乗算ルーチンの POWER CALL ルーチンの内部使用により数値精度の問題が発生することがあるため、大きなスカラー値は避ける必要があります。

要件　入力行列と出力行列は正方で、同じ次元が含まれている必要があります。

構文

CALL POWER(*X*, *a*, *Y*);

必須引数

X
次元 *m* x *m* (つまり、X[*m*, *m*])を含む入力行列を指定します。

a
整数のスカラー値(累乗)を指定します。

Y
次のように、次元 *m* x *m* (つまり、Y[*m*, *m*])を含む出力行列を指定します。

$$Y = X^a$$

詳細

スカラーは整数でない場合、整数に切り捨てられます。スカラーは 0 未満の場合、0 に変更されます。詳細については、*SAS/IML User's Guide* を参照してください。

例

次の例では、POWER CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,3] (0.3, -0.78, -0.82, 0.54, 1.74,
                   1.2, -1.3, 0.25, 1.49);
  array result[3,3];
  call power(mat1, 3, result);
  put result=;
quit;
```

アウトプット 22.10　POWER CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

result[1, 1]=2.375432 result[1, 2]=-4.299482 result[1, 3]=-6.339638
result[2, 1]=-3.031224 result[2, 2]=6.272988 result[2, 3]=8.979036
result[3, 1]=-4.33592 result[3, 2]=5.775695 result[3, 3]=9.326529
```

CALL SETNULL ルーチン

構造のポインター要素を Null に設定します。

カテゴリ: C Helper

構文

CALL SETNULL(*pointer-element*);

必須引数

pointer-element
構造へのポインターです。

例

次の例では、PROC PROTO を使用して定義された LINKLIST 構造を前提としています。CALL SETNULL ルーチンを使用して、NEXT 要素を null に設定できます。

```
struct linklist list;
call setnull(list.next);
```

CALL STRUCTINDEX ルーチン

構造の配列の各構造要素にアクセスできます。

カテゴリ: C Helper

構文

CALL STRUCTINDEX(*structure-array*, *index*, *structure-element*);

必須引数

structure-array
配列を指定します。

index
1 から始まるインデックスです。ほとんどの SAS 配列で使用されます。

structure-element
配列の要素を示します。

例

この例の最初の部分では、次の構造と関数が PROC PROTO を使用して定義されます。

```
proc proto package=sasuser.mylib.str2;
struct point{
  short s;
  int i;
  long l;
  double d;
};

struct point_array {
  int          length;
  struct point p[2];
  char         name[32];
};
run;
```

この例の 2 番目の部分では、PROC FCMP コードセグメントに、STRUCTINDEX CALL ルーチンを使用して POINT_ARRAY 構造の P という配列の各ポイント構造要素を取得および設定するための方法が示されます。

```
options pageno=1 nodate ls=80 ps=64;

proc fcmp libname=sasuser.mylib;
```

```
  struct point_array pntarray;
  struct point pnt;

  pntarray.length=2;
  pntarray.name="My funny structure";

  /* Get each element using the STRUCTINDEX CALL routine and set values. */
do i=1 to 2;
  call structindex(pntarray.p, i, pnt);
  put "Before setting the" i "element: " pnt=;
  pnt.s=1;
  pnt.i=2;
  pnt.l=3;
  pnt.d=4.5;
  put "After setting the" i "element: " pnt=;
end;
run;
```

アウトプット 22.11　配列のポイント構造要素の設定の結果

```
                                The SAS System                                1

                              The FCMP Procedure

Before setting the 1 element:  pnt {s=0, i=0, l=0, d=0}
After setting the 1 element:  pnt {s=1, i=2, l=3, d=4.5}
Before setting the 2 element:  pnt {s=0, i=0, l=0, d=0}
After setting the 2 element:  pnt {s=1, i=2, l=3, d=4.5}
```

CALL SUBTRACTMATRIX ルーチン

2 つの行列、または 1 つの行列およびスカラーの element-wide 減算を実行します。

カテゴリ: 行列操作

要件 すべての 入力行列と出力行列には、同じ次元が含まれている必要があります。

構文

CALL SUBTRACTMATRIX(*X*, *Y*, *Z*);

必須引数

X
次元 *m* x *n* (つまり X[*m*, *n*])を含む入力行列、またはスカラーを指定します。

Y
次元 *m* x *n* (つまり Y[*m*, *n*])を含む入力行列、またはスカラーを指定します。

Z
次のような次元 *m* x *n* (つまり Z[*m*, *n*])を含む出力行列を指定します。次のようになります。

$$Z = X - Y$$

例

次の例では、SUBTRACTMATRIX CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
  array mat2[3,2] (0.2, 0.38, -0.12, 0.98, 2, 5.2);
  array result[3,2];
  call subtractmatrix(mat1, mat2, result);
  call subtractmatrix(2, mat1, result);
  put result=;
 quit;
```

アウトプット 22.12　SUBTRACTMATRIX CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

result[1, 1]=1.7 result[1, 2]=2.78 result[2, 1]=2.82 result[2, 2]=1.46
result[3, 1]=0.26 result[3, 2]=0.8
```

CALL TRANSPOSE ルーチン

行列の転置を返します。

カテゴリ:　行列操作

構文

CALL TRANSPOSE(*X*, *Y*);

必須引数

X
次元 *m* x *n* (つまり、X[*m*, *n*])を含む入力行列を指定します。

Y
次元 *n* x *m* (つまり、Y[*n*, *m*])を含む出力行列を指定します。

詳細

$$Y = X'$$

入力行列の行数が出力行列の列数と同じで、出力行列の行数が入力行列の列数と同じである必要があります。

例

次の例では、TRANSPOSE CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,2] (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
  array result[2,3];
  call transpose(mat1, result);
  put result=;
quit;
```

アウトプット 22.13　TRANSPOSE CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

result[1, 1]=0.3 result[1, 2]=-0.82 result[1, 3]=1.74 result[2, 1]=-0.78
result[2, 2]=0.54 result[2, 3]=1.2
```

CALL ZEROMATRIX ルーチン

数値入力行列のすべての要素値を 0 と置き換えます。

カテゴリ: 行列操作

注: ZEROMATRIX CALL ルーチンを多次元数値配列と使用できます。

構文

CALL ZEROMATRIX(*X*);

必須引数

X
数値入力行列を指定します。

例

次の例では、ZEROMATRIX CALL ルーチンを使用します。

```
options pageno=1 nodate;

proc fcmp;
  array mat1[3,2]  (0.3, -0.78, -0.82, 0.54, 1.74, 1.2);
  call zeromatrix(mat1);
  put mat1=;
quit;
```

アウトプット 22.14　ZEROMATRIX CALL ルーチンの結果

```
                                The SAS System                                1

                              The FCMP Procedure

mat1[1, 1]=0 mat1[1, 2]=0 mat1[2, 1]=0 mat1[2, 2]=0 mat1[3, 1]=0 mat1[3,
2]=0
```

INVCDF 関数

累積分布関数(CDF)を定義した分布の分位点を計算します。

カテゴリ: 特殊な目的のある関数

注: INVCDF は、FCMP プロシジャによって 参考のために自動的に提供される特殊な目的のある関数です。

確率 *p* を CDF を含む分布(*F(x; <parameters>)*で示される)に対して指定する場合、INVCDF 関数は分位点 *q* (*F(q;<parameters>) = p* を満たす)を返します。つまり、$q=F^{-1}(p)$となります。

構文

quantile=**INVCDF**('*CDF-function-name*', *options-array*, *cumulative-probability*, *parameter-1 <, parameter-2, ...>*);

必須引数

quantile
INVCDF 関数から返される分位点を指定します。

'CDF-function-name'
CDF 関数の名前を指定します。*CDF-function-name* を引用符で囲みます。

要件 *CDF-function-name* は、FCMP プロシジャを使用して定義される関数にしてください。次のように、シグネチャが含まれている必要があります。

```
function <CDF-function-name> (x, parameter-1 <, parameter-2, ...>);

endsub;
```

注 CDF が連続関数であることが推奨されます。CDF が不連続である場合、INVCDF 関数で分位点を計算できないことがあります。

options-array
INVCDF 関数と使用するオプションの配列を指定します。*Options-array* は、CDF の反転プロセスの制御とモニターに使用されます。*Options-array* には欠損値(.)が可能です。または次の順序の次の要素のうち 4 つまでを含めることができます。

initial-value
反転プロセスが開始する分位点に対し初期予測を指定します。これは、分位点に対する推定値(CDF の経験推定からなど)がある場合に有用です。

デフォルト 0.1

desired-accuracy
分位点の必要な相対的な正確性を指定します。範囲(0,0.1)の値を指定します。小さい値を指定すると、結果として分位点の推定はより正確になりますが、CDF の反転により時間がかかることがあります。

デフォルト 1.0e-8

domain-type
CDF 関数のドメインを指定します。欠損値または値 0 は、正のサポート、[0,∞)を示します。その他の値は実数直線全体にわたるサポート、(-∞,∞)を示します。

デフォルト 0

return-code
リターンステータスを指定します。*options-array* が次元 4 以上の場合、4 番目の要素にはリターンステータスが含まれています。*Return-code* の値には、次のうちいずれかが可能です。

<=0
成功を示します。負の場合、絶対値は分位点を計算するために評価された CDF 関数の回数です。絶対値が大きくなると、収束時間が長くなります。

1
分位点を計算できなかったことを示します。

cumulative-probability
分位点が必要となる累積確率値を指定します。

範囲 [0,1)

parameter
分位点が必要となる分布のパラメーターを指定します。指定された CDF 関数によって要求されるものと同じパラメーター数を指定する必要があります。これらは、指定された CDF 関数によって要求されるものと同じ順序で表示する必要があります。

詳細

INVCDF 関数は、累積分布関数が *CDF-function-name* 引数によって指定される分布から、指定された累積確率に対する分位点を検索します。つまり、次の式が True の CDF 関数を反転します。

cumulative-probability = CDF-function-name(*quantile*,<*parameters*>)

ε累積確率 *p* の分位数の望ましい精度を表す場合、INVCDF は、$|p - F(q)| < \varepsilon p$,*F(x)*が *x* で評価された CDF を表す、分位数 *q* を計算しようとします。

反転プロセスをさまざまなオプションで制御できます。次に、オプション配列の例を示します。

array opts[4] *initial* epsilon *support* (1.5 1.0e-6 0);

これらの値は次の前の命令行を参照しています。

initial(*initial-value*)=1.5

epsilon(*desired-accuracy*)=1.0e-6

support(*domain-type*)=0

opts[4]をチェックして、関数のリターンステータスを検証できます。

比較

この関数を QUANTILE 関数の一般的な拡張とすることができます。特定の分布からのみ分位点を計算します。分布の CDF 関数がプログラムで定義可能な限り、INVCDF 関数を使用して連続分布からの分位点を計算できます。QUANTILE 関数とは異なり、この関数は DATA ステップで直接使用できません。FCMP 関数またはサブルーチンの定義内でのみ使用できます。ただし、これは制限ではありません。この関数を使用する FCMP 関数を DATA ステップから起動できるためです。次の例を参照してください。

例: 指数分布からの無作為抽出の生成

次の例に、INVCDF 関数を使用して DATA ステップでパラメトリックな分布から無作為抽出を生成する方法を示します。例では指数分布を説明に使用しますが、プロ

グラムで CDF 関数が定義可能な分布まで拡張できます。次のステートメントでは、指数分布からの分布点の計算に INVCDF 関数と CDF 関数を使用する FCMP 関数 EXP_QUANTILE を定義します。

```
proc fcmp library=work.mycdf outlib=work.myquantile.functions;
  function exp_quantile(cdf, theta, rc);
    outargs rc;
    array opts[4] / nosym(0.1 1.0e-8 .);
    q=invcdf("exp_cdf", opts, cdf, theta);
    rc=opts[4];  /* return code */
    return(q);
  endsub;
quit;
```

前出のコードでは、次のように PROC FCMP ステップを使用して EXP_CDF 関数の定義を Work.Mycdf という FCMP ライブラリに保存したとみなします。

```
proc fcmp outlib=work.mycdf.functions;
  function exp_cdf(x, theta);
    return(1.0 - exp(-x/Theta));
  endsub;
quit;
```

EXP_QUANTILE 関数を DATA ステップから起動し、値が 50 のスケールパラメーター(theta)によって指数分布から無作為抽出を生成できるようになりました。DATA ステップを実行する前に、EXP_CDF 関数と EXP_QUANTILE 関数の場所を CMPLIB= オプションに対する適切な値で指定する必要があります。

```
options cmplib=(work.mycdf work.myquantile);

data exp_sample(keep=q);
  n=0;k=0;
  do k=1 to 500;
    if (n=100) then leave;
    rcode=.;
    q=exp_quantile(rand('UNIFORM'), 50, rcode);
    if (rcode <= 0) then do;
      n=n+1;
      output;
    end;
  end;
run;
```

ISNULL 関数

構造のポインター要素が Null かどうかを決定します。

カテゴリ: C Helper

構文

numeric-variable = **ISNULL** (*pointer-element*);

必須引数

numeric-variable
数値を指定します。

pointer-element
別の変数のアドレスを含む変数を指定します。

例

例 1: リンクされたリストの生成

次の例で、LINKLIST 構造と GET_LIST 関数が PROC PROTO を使用して定義されます。GET_LIST 関数は、必要とされる数の要素を含むリンクされたリストを生成する外部 C ルーチンです。

```
struct linklist{
  double value;
  struct linklist * next;
};

struct linklist * get_list(int);
```

例 2: ISNULL C Helper 関数のループでの使用

次のコードセグメントに、ISNULL C Helper 関数が GET_LIST によって作成されるリンクされたリストにわたりループし、要素を書き出すことを示します。

```
proc proto package=sasuser.mylib.str2;
struct linklist{
  double value;
  struct linklist * next;
};

struct linklist * get_list(int);

externc get_list;
  struct linklist * get_list(int len){
    int i;
    struct linklist * list=0;
    list=(struct linklist*)
      malloc(len*sizeof(struct linklist));
    for (i=0;i<len-1;i++){
      list[i].value=i;
      list[i].next=&list[i+1];
    }
    list[i].value=i;
    list[i].next=0;
    return list;
  }
externcend;
run;

options pageno=1 nodate ls=80 ps=64;
```

```
proc fcmp libname=sasuser.mylib;
  struct linklist list;
  list=get_list(3);
  put list.value=;

  do while (^isnull(list.next));
    list=list.next;
    put list.value=;
  end;
run;
```

アウトプット 22.15　ISNULL C Helper 関数使用の結果

```
                                The SAS System                          1

                              The FCMP Procedure

list.value=0
list.value=1
list.value=2
```

LIMMOMENT 関数

累積分布関数(CDF)を定義した分布の制限したモーメントを計算します。

カテゴリ: 特殊な目的のある関数

注: LIMMOMENT は、FCMP プロシジャによって 参考のために自動的に提供される特殊な目的のある関数です。

順序 *k* と上限 *u* を指定する場合、LIMMOMENT 関数は制限されたモーメントを $E[min(X,u)^k]$として計算します。この場合、*E* は、指定された CDF 関数によって定義される確率変数 X の分布を引き継いだ期待を示します。

構文

imom=**LIMMOMENT**('*CDF-function-name*', *options-array*, *order*, *limit*,
parameter-1 <, parameter-2, ...>);

必須引数

imom
LIMMOMENT 関数から返される制限されたモーメントを指定します。

'CDF-function-name'
CDF 関数の名前を指定します。*CDF-function-name* を引用符で囲みます。

要件 *CDF-function-name* は、FCMP プロシジャを使用して定義される関数にしてください。次のように、シグネチャが含まれている必要があります。

function <*CDF-function-name*> (*x*, *parameter-1* <, *parameter-2*, ...>);

endsub;

注 CDF が連続関数であることが推奨されます。CDF が不連続である場合、LIMMOMENT 関数は制限されたモーメントを計算できないことがあります。

options-array
LIMMOMENT 関数と使用するオプションの配列を指定します。*Options-array* は、限度額の計算に使用される数値積分のプロセスの制御とモニターに使用されます。*Options-array* には欠損値(.)が可能です。または次の順序の次の要素のうち 4 つまでを含めることができます。

desired-accuracy
数値積分の必要な正確性を指定します。範囲(0,0.1)の値を指定します。小さい値を指定すると、結果としてモーメントの推定はより正確になりますが、desired-accuracy の計算にはより多くの時間がかかります。

デフォルト 1.0e-8

initial-step-size
数値積分プロセスによって最初に使用されるステップサイズを指定します。値が増えると、被積分関数が評価される回数の線が少なくなります。通常、デフォルト値 1 を使用すると、よい結果が生成されます。

デフォルト 1

maximum-iterations
必要な正確性を実現するため統合結果の絞り込みに使用される反復の最大数を指定します。この値が増えると、被積分関数が評価される回数の指数が増えます。

デフォルト 8

return-code
リターンステータスを指定します。*options-array* が次元 4 以上の場合、4 番目の要素にはリターンステータスが含まれています。*Return-code* の値には、次のうちいずれかが可能です。

<=0
成功を示します。負の場合、絶対値は制限されたモーメントを計算するために被積分関数が評価された回数です。絶対値が大きくなると、収束時間が長くなります。

1
制限されたモーメントを計算できなかったことを示します。

order
必要な制限されたモーメントの順序を指定します。

範囲 [1,10]

limit
必要な制限されたモーメントの計算に使用される上限を指定します。

要件 *limit* の値は 0 より大きいものとします。

parameter
制限されたモーメントが必要となる分布のパラメーターを指定します。指定された CDF 関数によって要求されるものと同じパラメーター数を指定する必要があります。これらは、指定された CDF 関数によって要求されるものと同じ順序で表示する必要があります。

詳細

確率変数 *X* に確率密度関数 *f(x;θ)*と累積分布関数 *F(x;θ)*を含む確率分布が含まれるようにします。 θ は分布のパラメーターを示します。指定された上限 *u* に対し、この分布の k^{th} (番目)の順序の制限されたモーメントが次のように定義されます。

$$E[(X \wedge u)^k] = \int_0^u x^k f(x)dx + u^k \int_u^\infty f(x)dx = \int_0^u x^k f(x)dx + u^k(1 - F(u))$$

LIMMOMENT 関数は、次の代替式を使用します。

$$E[(X \wedge u)^k] = k \int_0^u x^{k-1}[1 - F(x)]dx$$

式には *F(x)*のみ必要なため、分布に対し CDF 関数のみ指定する必要があります。制限されたモーメントは、保険アプリケーションで保険範囲が特定の値で設定されている場合に支払われる最大金額の計算に使用されます。

数値積分プロセスをさまざまなオプションで制御できます。次に、オプション配列の例を示します。

array opts[4] epsilon *initial* maxiter (1.0e-5 1 6);

これらの値は前の命令行を参照します。

epsilon(*desired-accuracy*)=1.0e-5

initial(*initial-step*)=1

maxiter(*maximum-iterations*)=6

opts[4]をチェックして、関数のリターンステータスを検証できます。

例: 対数正規分布の制限されたモーメントの計算

この例では、LIMMOMENT 関数を使用して DATA ステップでパラメトリックな分布に対する制限されたモーメントを計算する方法を示します。例では対数正規分布を説明に使用しますが、プログラムで CDF 関数が定義可能な分布まで拡張できます。次のステートメントでは、対数正規分布からの制限されたモーメントの計算に LIMMOMENT 関数と CDF 関数を使用する FCMP 関数 LOGN_LIMMOMENT を定義します。

```
proc fcmp library=work.mycdf outlib=work.mylimmom.functions;
    function logn_limmoment(order, limit, mu, sigma, rc);
      outargs rc;
      array opts[4] / nosym (1.0e-8 . . .);
```

```
      m=limmoment("logn_cdf", opts, order, limit, mu, sigma);
      rc=opts[4]; /* return code */

      return(m);
    endsub;
  quit;
```

前出のコードでは、次のように PROC FCMP ステップを使用して LOGN_CDF 関数の定義を Work.Mycdf という FCMP ライブラリに保存したとみなします。

```
proc fcmp outlib=work.mycdf.functions;
    function logn_cdf(x, Mu, Sigma);
      if (x >= constant('MACEPS')) then do;
        z=(log(x) - Mu)/Sigma;
        return(CDF('NORMAL',z));
      end;
      return (0);
    endsub;
  quit;
```

次のように、LOGN_LIMMOMENT 関数を DATA ステップから起動できるようになりました。DATA ステップを実行する前に、LOGN_CDF 関数と LOGN_LIMMOMENT 関数の場所を CMPLIB=オプションに対し適切な値で指定する必要があります。

```
options cmplib=(work.mycdf work.mylimmom);

  data _null_;
    do order=1 to 3;
      rcode=.;
      m=logn_limmoment(order, 100, 5, 0.5, rcode);
      if (rcode > 0) then
        put "ERROR: Limited moment could not be computed.";
      else
        put 'Moment of order ' order ' with limit 100 = ' m;
    end;
  run;
```

NDIMS 関数

配列にある次元数を返します。

カテゴリ: 配列

構文

rc= **NDIMS**(*array-name*)

必須引数

array-name
配列の名前を指定します。

rc
次元数を返す文字変数を指定します。

詳細

NDIMS 関数は、多次元配列の次元数を返すか、1 次元配列の場合は 1 を返します。

例

```
proc fcmp;
  array my_onearray[3] 1 2 3;
  array my_multiarray[2,3] x11 x12 x13 x21 x22 x23;
  n_one=ndims(my_onearray);
  n_multi=ndims(my_multiarray);
  put n_one= n_multi=;
run;
quit;
```

READ_ARRAY 関数

データを SAS データセットから PROC FCMP 配列変数に読み込みます。

カテゴリ: 配列

構文

rc = **READ_ARRAY**(*data_set_name*, *array_variable* <, '*column_name_1*', '*column_name_2*' , ... >);

必須引数

rc
関数がデータセットを正常に読み込める場合は 0 です。

data_set_name
配列データの読み込み元のデータセットの名前を指定します。*data_set_name* は、読み込み元のデータセットのメンバー名(libname.memname)が含まれている文字リテラルまたは変数にしてください。

array_variable
データの読み込み先の PROC FCMP 配列変数を指定します。*array_variable* は必ず、ローカル一時配列変数にしてください。関数はデータセットのサイズにあわせてそのサイズを拡大、縮小する必要があるためです。

オプション引数

column_name
読み込まれるデータセットの特定の列に対し任意の名前を指定します。

column_name を指定した場合、引用符で囲まれたリテラル文字列にしてください。*column_name* は PROC FCMP 変数にはなりません。列名が指定されない場合、PROC FCMP はデータセットのすべての列を読み込みます。

詳細

配列とデータセット間の変換時に、配列は[row,column]のようにインデックス付けされます。

関数と CALL ルーチンで宣言された配列は、/NOSYMBOLS オプションで宣言された配列と同様にサイズ変更可能です。その他の配列はサイズ変更できません。

READ_ARRAY 関数は、入力データセットの次元に合わせて動的に配列をサイズ変更しようとします。つまり、配列は動的である必要があります。配列は関数または CALL ルーチンのいずれか、または/NOSYMBOLS オプションで宣言する必要があります。

例

この例では、SAS データセットを作成し、FCMP 配列変数に読み込みます。

```
options nodate pageno=1;

data account;
  input acct price cost;
  datalines;
1 2 3
4 5 6
;
run;

proc fcmp;
  array x[2,3] / nosymbols;
  rc=read_array('account',x);
  put x=;
run;

proc fcmp;
  array x[2,2] / nosymbols;
  rc=read_array('account', x, 'price', 'acct');
  put x=;
run;
```

アウトプット 22.16　READ_ARRAY 関数の結果

```
                          The SAS System                        1

                          The FCMP Procedure

x[1, 1]=1 x[1, 2]=2 x[1, 3]=3 x[2, 1]=4 x[2, 2]=5 x[2, 3]=6
```

```
                          The SAS System                        2

                          The FCMP Procedure

x[1, 1]=2 x[1, 2]=1 x[2, 1]=5 x[2, 2]=4
```

RUN_MACRO 関数

事前定義された SAS マクロを実行します。

カテゴリ:　関数内からの SAS コードの呼び出し

注:　この関数の動作は、SAS での%macro_name;の実行に似ています。

構文

rc = **RUN_MACRO** (*'macro_name'* <, *variable_1*, *variable_2*, ...>);

必須引数

rc
関数がマクロをサブミットできる場合は 0 です。リターンコードは、マクロコールが試行されたことのみ示します。マクロが予想どおりに実行するかどうかを決定するためにマクロ自体が PROC FCMP 変数に対応する SAS マクロ変数の値を設定する必要があります。

macro_name
実行するマクロの名前を指定します。

要件　*macro_name* は、引用符で囲まれた文字列か、実行するマクロを含む文字変数にしてください。

オプション引数

variable
同じ名前のマクロ変数によって設定される、任意の PROC FCMP 変数を指定します。これらの引数は、PROC FCMP 倍精度変数、または文字変数にしてください。

マクロの実行前に、SAS マクロ変数が PROC FCMP 変数と同じ名前と値で定義されます。マクロの実行後は、マクロ変数値は対応する PROC FCMP 変数にコピーされます。

例

例 1: PROC FCMP による事前定義されたマクロの実行

この例では、%TESTMACRO というマクロを作成してから、そのマクロを PROC FCMP 内で使用し、2 つのメンバーを減算します。

```
    /* Create a macro called %TESTMACRO. */
%macro testmacro;
  %let p=%sysevalf(&a - &b);
%mend testmacro;

   /* Use %TESTMACRO within a function in PROC FCMP to subtract two numbers. */
proc fcmp outlib=sasuser.ds.functions;
  function subtract_macro(a, b);
    rc=run_macro('testmacro', a, b, p);
    if rc eq 0 then return(p);
    else return(.);
  endsub;
run;

  /* Make a call from the DATA step. */
option cmplib=(sasuser.ds);

data _null_;
  a=5.3;
  b=0.7;
  p=.;
  p=subtract_macro(a, b);
  put p=;
run;
```

例のコード 22.1　PROC FCMP での事前定義されたマクロ実行の結果

```
p=4.6
```

例 2: RUN_MACRO 関数および DOSUBL 関数の結果の比較

この例では、%TESTMACRO を作成し、DOSUBL 関数が RUN_MACRO と同じ%TESTMACRO を呼び出す方法を比較しています。RUN_MACRO 起動時には、すべての変数が、設定できるように引数として渡されます。DOSUBL 起動時には、実行するコードにすべてのマクロ変数がインポートされ、完了時にエクスポートされます。マクロ変数&a および&b は%TESTMACRO に利用できるようになり、マクロ変数&p は DOSUBL 呼び出しに利用できるようになります。詳細については、“DOSUBL 関数” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

```
   /* Create a macro called %TESTMACRO. */
%macro testmacro;
```

```
  %let p=%sysevalf(&a - &b);
%mend testmacro;

   /* Use %TESTMACRO within a function in PROC FCMP to subtract two numbers. */
proc fcmp outlib=sasuser.ds.functions;
  function subtract_macro(a, b);
    rc=run_macro('testmacro', a, b, p);
    if rc eq 0 then return(p);
    else return(.);
  endsub;
run;

  /* Make a call from the DATA step. */
option cmplib=(sasuser.ds);

data _null_;
  a=5.3;
  b=0.7;
  p=.;
  p=subtract_macro(a, b);
  put p= '(RUN_MACRO function and a DATA step)';
run;

%global a b p;
%put p=&p;
   /* The value should not yet be known. */
  %let a=5.3;
  %let b=0.7;
data _null_;
  rc=dosubl('%testmacro');
  run;
%put p=&p (DOSUBL function);
```

例のコード 22.2　RUN_MACRO 関数および DOSUBL 関数の結果

```
p=4.6 (RUN_MACRO function and a DATA step)
p=4.6 (DOSUBL function)
```

例 3: DATA ステップ内の DATA ステップの実行

この例には、別の DATA ステップ内から DATA ステップを実行する方法を示します。プログラムは次のセクションから構成されています。

- プログラムの最初のセクションでは、APPEND_DS というマクロを作成します。このマクロは、データセット間の書き込みと追加を行うことができます。
- プログラムの 2 番目のセクションでは、マクロを PROC FCMP の関数 writeDataset から呼び出します。
- プログラムの 3 番目のセクションでは、SALARIES データセットを作成し、変数 Department の値によってデータセットを 4 つの個別のデータセットに分割します。
- プログラムの 4 番目のセクションでは、結果を出力ウィンドウに書き込みます。

```
  /* Create a macro called APPEND_DS. */
%macro append_ds;
  /* Character values that are passed to RUN_MACRO are put          */
```

```
  /* into their corresponding macro variables inside of quotation     */
  /* marks. The quotation marks are part of the macro variable value. */
  /* The DEQUOTE function is called to remove the quotation marks.    */
  %let dsname=%sysfunc(dequote(&dsname));
  data &dsname;
    %if %sysfunc(exist(&dsname)) %then %do;
      modify &dsname;
    %end;
    Name=&Name;
    WageCategory=&WageCategory;
    WageRate=&WageRate;
    output;
    stop;
  run;
%mend append_ds;

  /* Call the APPEND_DS macro from function writeDataset in PROC FCMP. */
proc fcmp outlib=sasuser.ds.functions;
  function writeDataset (DsName $, Name $, WageCategory $, WageRate);
    rc=run_macro('append_ds', DsName, Name, WageCategory, WageRate);
    return(rc);
  endsub;
run;

  /* Use the DATA step to separate the salaries data set into four separate */
  /* departmental data sets (NAD, DDG, PPD, and STD).                       */
data salaries;
  input Department $ Name $ WageCategory $ WageRate;
  datalines;
BAD Carol Salaried 20000
BAD Beth Salaried 5000
BAD Linda Salaried 7000
BAD Thomas Salaried 9000
BAD Lynne Hourly 230
DDG Jason Hourly 200
DDG Paul Salaried 4000
PPD Kevin Salaried 5500
PPD Amber Hourly 150
PPD Tina Salaried 13000
STD Helen Hourly 200
STD Jim Salaried 8000
;
run;

options cmplib=(sasuser.ds) pageno=1 nodate;
data _null_;
  set salaries;
  by Department;
  length dsName $ 64;
  retain dsName;
  if first.Department then do;
    dsName='work.' || trim(left(Department));
  end;
  rc=writeDataset(dsName, Name, WageCategory, wageRate);
run;
```

```
proc print data=work.BAD; run;
proc print data=work.DDG; run;
proc print data=work.PPD; run;
proc print data=work.STD; run;
```

アウトプット 22.17　DATA ステップ内での DATA ステップの呼び出しの結果

```
            The SAS System
                          1

               Wage      Wage
Obs    Name    Category    Rate

 1   Carol    Salaried   20000
 2   Beth     Salaried    5000
 3   Linda    Salaried    7000
 4   Thomas   Salaried    9000
 5   Lynne    Hourly       230
```

```
            The SAS System
                          2

               Wage     Wage
Obs   Name    Category   Rate

 1   Jason   Hourly      200
 2   Paul    Salaried   4000
```

```
            The SAS System
                          3

               Wage      Wage
Obs   Name    Category    Rate

 1   Kevin   Salaried    5500
 2   Amber   Hourly       150
 3   Tina    Salaried   13000
```

```
            The SAS System
                          4

               Wage     Wage
Obs   Name    Category   Rate

 1   Helen   Hourly      200
 2   Jim     Salaried   8000
```

RUN_SASFILE 関数

指定するファイル参照名で SAS コードを実行します。

カテゴリ: 関数内からの SAS コードの呼び出し

構文

rc = **RUN_SASFILE** (*'fileref_name'* <, *variable-1*, *variable-2*, ...>);

必須引数

rc
関数が SAS ファイルを処理するコードを実行する依頼をサブミットできる場合は 0 です。リターンコードは、呼び出しが試行されたことのみ示します。

fileref_name
SAS コードをポイントする SAS ファイル参照名の名前を指定します。

要件 *fileref_name* は、引用符で囲まれた文字列か、SAS ファイル参照名を含む文字変数にしてください。

オプション引数

variable
同じ名前のマクロ変数によって設定される、任意の PROC FCMP 変数を指定します。これらの引数は、PROC FCMP 倍精度変数、または文字変数にしてください。

SAS ファイルを参照するコードを実行する前、SAS マクロ変数は PROC FCMP 変数と同じ名前と値で定義されます。実行後、これらのマクロ変数値は対応する PROC FCMP 変数にコピーされます。

例

次の例は、RUN_SASFILE が事前定義されているマクロではなく SAS ファイルを使用する点を除いて、RUN_MACRO の最初の例に似ています。この例では、test.sas(a, b, c)が現在のディレクトリに置かれているものとします。

```
    /* test.sas(a,b,c) */
data _null_;
  call symput('p', &a * &b);
run;

  /* Set a SAS fileref to point to the data set. */
filename myfileref "test.sas";
```

```
  /* Set up a function in PROC FCMP and call it from the DATA step. */
proc fcmp outlib=sasuser.ds.functions;
  function subtract_sasfile(a, b);
  rc=run_sasfile('myfileref', a, b,
p);
  if rc=0 then return(p);
    else return(.);
  endsub;
run;

options cmplib=(sasuser.ds);
data _null_;
  a=5.3;
  b=0.7;
  p=.;
  p=subtract_sasfile(a, b);
  put p=;
run;
```

SOLVE 関数

Gauss-Newton 法を使用して、関数の暗黙値を計算します。

カテゴリ: 暗黙値の計算

注: この特殊な目的のある関数は、FCMP プロシジャによって 参考のために自動的に提供されます。

構文

answer = **SOLVE**('*function-name*', *options-array*, *expected-value*, *argument-1* <, *argument-2*, ...>);

必須引数

answer
SOLVE 関数から返される値を指定します。

'*function-name*'
関数の名前を指定します。*function-name* を引用符で囲みます。

options-array
SOLVE 関数と使用するオプションの配列を指定します。*Options-array* は、ルート検索プロセスの制御とモニターに使用されます。*Options-array* には欠損値(.)が可能です。または次の順序の次の要素のうち 5 つまでを含めることができます。

initial-value
暗黙値の開始値を指定します。最初の呼び出しのデフォルトは 0.001 です。同じ命令行が再度実行されると、*options-array* では前に検出された暗黙値が使用されます。

absolute-criterion
収束の値を指定します。期待値と予測値間の差異の絶対値は、収束に対する *absolute-criterion* の値未満にしてください。

デフォルト 1.0e-12

relative-criterion
収束の値を指定します。計算済みの暗黙値の変更が *relative-criterion* 値未満のときは、収束が想定されます。

デフォルト 1.0e-6

maximum-iterations
ソリューションの検索に使用する反復の最大数を指定します。

デフォルト 100

solve-status
は次の値のどれかになります。

0 successful.

1 エラーを減らすことができません。

2 変更ベクトルを計算できません。

3 反復の最大数を超えています。

4 最初の目的関数がありません。

expected-value
対象の関数の期待値を指定します。

argument
最小化中の関数に渡す引数を指定します。

詳細

SOLVE 関数は、次の形式の式をゼロに等しくする指定引数の値を検索します。

expected-value - function-name
(*argument-1*, *argument-2*, ...)

対象の引数を欠損値(.)で指定します。これは、上記のパラメーターリストに引数のかわりに表示されます。SOLVE 関数が値を検索する場合、この関数に対して返される値が暗黙値となります。

次に、オプション配列の例を示します。

```
array opts[5] initial abconv relconv maxiter (.5 .001 1.0e-6 100);
```

これらの値は前の命令行を参照します。

- **initial** (initial-value)=.5
- **abconv** (absolute-criterion)=.001
- **relconv** (relative-criterion)=1.0e-6
- **maxiter** (maximum-iterations)=100

解決ステータスは、配列の 5 番目の要素です。この値は出力リストの opts[5]を指定して表示できます。

例

例 1: 平方根値の計算

次の SOLVE 関数の例は、方程式 y=1/sqrt(x)を満たす x の値を計算します。関数とサブルーチンは、SOLVE 関数で使用できるように、最初に定義する必要があります。この例では、関数 INVERSESQRT が最初に定義され、SOLVE 関数で使用されます。

```
options pageno=1 nodate ls=80 ps=64;

proc fcmp;
    /* define the function */
  function inversesqrt(x);
    return(1/sqrt(x));
  endsub;

  y=20;
  x=solve("inversesqrt", {.}, y, .);
  put x;
run;
```

アウトプット 22.18　平方根値の計算からの結果

```
                                The SAS System                          1

                              The FCMP Procedure

0.0025
```

例 2: Garman-Kohlhagen インプライドボラティリティの 計算

この例で、サブルーチン GKIMPVOL は SOLVE 関数と GARKHPRC 関数を使用して、FX オプションに対する Garman-Kohlhagen インプライドボラティリティを 計算します。

この例では、次の点に注意してください。

- options_array は SOLVOPTS で、初期値が必要です。
- 期待値は FX オプションの価格です。
- サブルーチンの欠損引数はボラティリティ (sigma)です。

```
proc fcmp;
```

```
function garkhprc(type$, buysell$, amount, E, t, S, rd, rf, sig)
  kind=pricing label='FX option pricing';

if buysell='Buy' then sign=1.;
else do;
  if buysell='Sell' then sign=-1.;
  else sign=.;
end;

if type='Call' then
  garkhprc=sign*amount*(E+t+S+rd+rf+sig);
else do;
  if type='Put' then
    garkhprc=sign*amount*(E+t+S+rd+rf+sig);
  else garkhprc=.;
end;
return(garkhprc);
endsub;

subroutine gkimpvol(n, premium[*], typeflag[*], amt_lc[*],
                    strike[*], matdate[*], valudate, xrate,
                    rd, rf, sigma);
outargs sigma;

array solvopts[1] initial (0.20);
sigma=0;
do i=1 to n;
  maturity=(matdate[i] - valudate) / 365.25;
  stk_opt=1./strike[i];
  amt_opt=amt_lc[i] * strike[i];
  price=premium[i] * amt_lc[i];

  if typeflag[i] eq 0 then type="Call";
  if typeflag[i] eq 1 then type="Put";

    /* solve for volatility */
  sigma=sigma + solve("GARKHPRC", solvopts, price,
                      type, "Buy", amt_opt, stk_opt,
                      maturity, xrate, rd, rf, .);
end;
sigma=sigma / n;
endsub;
run;
```

例 3: Black-Scholes インプライドボラティリティの計算

この SOLVE 関数の例では、ビルトイン SAS 関数 BLKSHCLPRC を使用して、関数 BLKSCH を定義します。SOLVE 関数は BLKSCH 関数を使用して、オプションの Black-Scholes インプライドボラティリティを計算します。

この例では、次の点に注意してください。

- options_array は OPTS です。
- 関数の欠損引数はボラティリティ (VOLTY)です。

- PUT ステートメントは、インプライドボラティリティ (BSVOLTY)、初期値、解決ステータスの書き込みに使用されます。

```
options pageno=1 nodate ls=80 ps=64;

proc fcmp;
  opt_price=5;
  strike=50;
  today='20jul2010'd;
  exp='21oct2010'd;
  eq_price=50;
  intrate=.05;
  time=exp - today;
  array opts[5] initial abconv relconv maxiter status
            (.5 .001 1.0e-6 100 -1);
  function blksch(strike, time, eq_price, intrate, volty);
    return(blkshclprc(strike, time/365.25,
                  eq_price, intrate, volty));
  endsub;
  bsvolty=solve("blksch", opts, opt_price, strike,
                      time, eq_price, intrate, .);

  put 'Option Implied Volatility:' bsvolty
     'Initial value: ' opts[1]
     'Solve status: ' opts[5];
run;
```

アウトプット 22.19　Black-Scholes インプライドボラティリティの 計算の結果

```
                                The SAS System                                1

                              The FCMP Procedure

Option Implied Volatility: 0.4687011859 Initial value:  0.5 Solve status:  0
```

注: SAS 関数と外部 C 関数は SOLVE 関数で直接使用できません。これらの関数は、PROC FCMP 関数で囲む必要があります。この例で、ビルトイン SAS 関数 BLKSHCLPRC が PROC FCMP 関数 BLKSCH で囲まれると、BLKSCH が SOLVE 関数で呼び出されます。

WRITE_ARRAY 関数

データを PROC FCMP 配列変数から SAS プログラム、マクロ、プロシジャによって 使用可能なデータセットに書き込みます。

カテゴリ:　配列

注:　配列とデータセット間の変換時に、配列は[row, column]のようにインデックス付けされます。

構文

rc = **WRITE_ARRAY**(*data_set_name*, *array_variable* <, '*column_name_1*', '*column_name_2*', ...>);

必須引数

rc
関数がデータセットを正常に書き込める場合は 0 です。

data_set_name
配列データの書き込み先のデータセットの名前を指定します。*data_set_name* は、作成されるデータセットのメンバー名(libname.memname)が含まれている文字リテラルまたは変数にしてください。

array_variable
コンテンツが *data_set_name* に書き込まれる PROC FCMP 配列または行列変数を指定します。

オプション引数

column_name
作成されるデータセットの列に対して任意の名前を指定します。

column_name を指定した場合、引用符で囲まれたリテラル文字列にしてください。*column_name* は PROC FCMP 変数にはなりません。列名が指定されない場合、列名は接尾辞が数値の配列名になります。

例

例 1: WRITE_ARRAY 関数と PROC FCMP 配列変数の使用

この例では、ARRAY ステートメント、WRITE_ARRAY 関数を PROC FCMP と使用して、出力をデータセットに書き込みます。

```
options nodate pageno=1 ls=80 ps=64;

proc fcmp;
  array x[4,5] (11 12 13 14 15 21 22 23 24 25 31 32 33 34 35 41 42 43 44 45);
  rc=write_array('work.numbers', x);
run;

proc print data=work.numbers;
run;
```

アウトプット 22.20　WRITE_ARRAY 関数を使用した結果

```
                 The SAS System                                    1
             Obs     x1     x2     x3     x4     x5

               1     11     12     13     14     15
               2     21     22     23     24     25
               3     31     32     33     34     35
               4     41     42     43     44     45
```

例 2: WRITE_ARRAY 関数を使用した列名の指定

この例では、任意の *colN* 変数を使用して、列名をデータセットに書き込みます。

```
options pageno=1 nodate ps=64 ls=80;

proc fcmp;
  array x[2,3] (1 2 3 4 5 6);
  rc=write_array('numbers2', x, 'col1', 'col2', 'col3');
run;

proc print data=numbers2;
run;
```

アウトプット 22.21　関数を使用して列名を指定した結果

```
                   The SAS System                                1

                Obs     col1     col2     col3

                  1        1        2        3
                  2        4        5        6
```

23

FEDSQL プロシジャ

概要: FEDSQL プロシジャ

FEDSQL プロシジャの機能

FEDSQL プロシジャを使用すると、SAS FedSQL 言語ステートメントをサブミットできます。FedSQL ステートメントを使用して、データの作成、クエリ、および結合を行うことができます。FedSQL ステートメントを SAS およびサードパーティの DBMS にサブミットできます。または、CAS のデータをクエリする FedSQL ステートメントをサブミットできます。

FedSQL 言語は、ANSI SQL:1999 コアスタンダードの SAS 実装です。DECIMAL、INTEGER、VARCHAR などの拡張データ型、その他の ANSI 1999 コアコンプライアンス機能および専用拡張子に対するサポートを提供します。

FedSQL は、複数のデータソース内のリレーショナルデータにアクセス、管理と共有を行う拡張可能で高性能な方法を導入する、スレッド化されたデータアクセス技術を提供します。可能な場合、FedSQL クエリは、大規模な処理を解決するためのマルチスレッド式アルゴリズムを使用して最適化されます。

FedSQL は、ベンダーに中立な SQL ダイアレクトです。アプリケーションについて、FedSQL は、サポートするすべてのデータソース 全体にわたって共通の SQL 構文を提供します。アプリケーションは、データソースに固有の SQL ダイアレクトでクエリをサブミットする代わりに、すべての FedSQL データソースに同じ FedSQL クエリをサブミットできます。

注: 他のほとんどの SAS プロシジャとは異なり、FEDSQL プロシジャは CAS への接続に CAS エンジンを使用しません。FedSQL 要求を CAS サーバーにサブミットするには、CASLIB ステートメントを使用して割り当てられた caslib を指定する必要があります。詳細については、“CAS サーバーへの接続” (804 ページ)を参照してください。

CAS でサポートされている FedSQL ステートメントは、SAS および DBMS データソースで使用できるステートメントのサブセットです。詳細については、次を参照してください。

- “SAS および DBMS データソースを使用した FedSQL 機能” (806 ページ)
- “SAS Cloud Analytic Services での FedSQL 機能” (806 ページ)。

概念: FEDSQL プロシジャ

FedSQL の利点

FedSQL は、SAS SQL プロシジャでは提供されていない次の機能を提供します。

- FedSQL は SQL 1999 ANSI 規格に準拠します。この規格に準拠することによって、独自の言語はもちろん、ANSI 1999 規格に準拠するその他の DBMS のネイティブ言語でもクエリを処理できます。
- FedSQL は、以前の SAS SQL 実装に比べてより多くのデータ型をサポートします。SAS/ACCESS LIBNAME エンジンを利用する従来の DBMS アクセスでは、ターゲット DBMS データ型と SAS レガシーデータ型(SAS 数値および SAS 文字)間で変換が行われます。FedSQL が DBMS に接続する場合、言語はターゲットデータソースの定義と一致するか、または必要に応じてその定義を FedSQL データ型に変換します。これによって、精度が飛躍的に高まります。

  FedSQL が CAS から DBMS に接続すると、言語はデータソースの定義をネイティブの CAS データ型に変換します。
- FedSQL はフェデレーションクエリをサポートします。フェデレーションクエリを使用すると、複数のデータソースに接続し、そのデータにアクセスして操作を実行し、単一の結果セットを返すことができます。出力データは、接続された任意のデータソースで作成できます。これによって、アプリケーションの要件に最も対応するデータソースにデータを保存することができます。

  (CAS の FedSQL 出力テーブルはインメモリテーブルです。caslib データソースにデータを保持するのに、他の CAS 機能を使用できます。)
- FedSQL は、ANSI 準拠の機能に対して暗黙的な SQL パススルーを提供します。ネイティブ処理のメリットを得るために、各データソースのネイティブ SQL で要求をサブミットする必要はありません。FedSQL の暗黙的なパススルーによって、クエリの応答時間が改善され、セキュリティが強化されます。詳細については、“FedSQL 暗黙的パススルー機能” (*SAS FedSQL 言語リファレンス*)を参照してください。
- FedSQL の明示的なパススルー機能を選択すると、データソースに接続し、ネイティブ SQL ステートメントをそのデータソースに送信して実行できるようになります。

  CAS 接続は、FedSQL SELECT ステートメントの FROM 句の CONNECTION TO コンポーネントを使用してネイティブ SQL 構文を受け入れます。

データソースサポート

FedSQL プロシジャでは、次のデータソースがサポートされています。

- Amazon Redshift
- CAS テーブル
- UNIX 動作環境の DB2
- Google BigQuery
- Greenplum
- Hadoop (Hive)
- Impala
- JDBC 準拠のデータベース
- Microsoft SQL Server
- MongoDB
- MySQL
- Netezza
- ODBC 準拠のデータベース
- Oracle
- PostgreSQL
- Salesforce
- SAP
- SAP HANA
- SAP IQ
- SAS データセット
- SAS Scalable Performance Data Engine データセット
- SAS Scalable Performance Data Server テーブル
- SingleStore
- Snowflake
- Spark
- Teradata
- Vertica
- Yellowbrick

DBMS データソースにアクセスするには、適切な SAS/ACCESS ソフトウェアを構成する必要があります。CAS サーバーにデータをロードするには、適切な SAS データコネクタソフトウェアを構成する必要があります。PROC FEDSQL とのデータソース接続を確立する方法については、次を参照してください。

- “SAS および DBMS データソースへの接続”
- “CAS サーバーへの接続”

構文: FEDSQL プロシジャ

PROC FEDSQL<*connection-options*><*processing-options*>;

...FedSQL statements

QUIT;

ステートメント	タスク	例
“PROC FEDSQL ステートメント”	後続の入力が FedSQL 言語ステートメントであることを指定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8
“QUIT ステートメント”	FEDSQL プロシジャの実行を停止します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8

PROC FEDSQL ステートメント

後続の入力が FedSQL 言語ステートメントであることを指定します。

要件

PROC FEDSQL ステートメントの後に FedSQL 言語ステートメントを続けます。SAS ライブラリでサポートされている FedSQL ステートメントについては、*SAS FedSQL 言語リファレンス*を参照してください。CAS セッションでサポートされている FedSQL ステートメントについては、“SAS Cloud Analytic Services での FedSQL 機能” (806 ページ)を参照してください。

操作:

PROC FEDSQL を使用してテーブルを作成する場合、デフォルトでは既存のテーブルを上書きできないことに注意してください。最初に DROP TABLE ステートメントを使用してテーブルを破棄する必要があります。CREATE TABLE ステートメントと同じプロシジャ実行で DROP TABLE ステートメントを指定する場合は、FORCE ステートメントオプションを指定する必要があります。CAS では、CREATE TABLE ステートメントで REPLACE=テーブルオプションを指定することにより、既存のテーブルを上書きできます。詳細については、“REPLACE= テーブルオプション” (*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*)を参照してください。

デフォルトでは、SAS または DBMS データソースにアクセスする場合、FEDSQL プロシジャは CHAR(n)列と DOUBLE 列に存在しない値を SAS 欠損値として処理します。存在しない値が ANSI SQL null 値として処理されるように要求するには、

ANSIMODE オプションを指定します。CAS では ANSIMODE はサポートされていません。

サポート: スレッド処理

注: 2023 年 9 月の時点で、Microsoft Azure Active Directory (Azure AD)は Microsoft Entra ID に名前変更されました。

例:
“例 1: SAS データセットを作成する” (813 ページ)
“例 2: 既存のテーブルから新しいテーブルを作成” (815 ページ)
“例 3: 相関サブクエリを使用したデータのクエリ” (817 ページ)
“例 4: 結合を実行するためのインデックスの作成および使用” (819 ページ)
“例 5: DS2 パッケージの式での使用” (820 ページ)
“例 6: CAS でのデータのクエリ” (823 ページ)
“例 7: CAS でのテーブルの明示的なロードと結合” (825 ページ)
“例 8: 複数の CAS ライブラリからのテーブルの結合” (830 ページ)

構文

PROC FEDSQL<*connection-options*><*processing-options*>;

オプション引数の要約

Connection

LIBS=*libref* | (*libref1libref2...librefn*)
データソース接続を、指定された 1 つまたは複数のライブラリ参照とデフォルトのライブラリに制限します。

SESSREF=*session-reference*
CAS セッションで FedSQL ステートメントを実行することを指定します。CAS セッションは、そのセッション名によって識別されます。

SESSUUID="*session-uuid*"
CAS セッションで FedSQL ステートメントを実行することを指定します。CAS セッションは、その汎用一意識別子(UUID)によって識別されます。

Execution Control

CNTL=(*instruction(s)*)
CAS における、FedSQL クエリプランニングおよびクエリ実行に対するオプションコントロール手順を指定します。

General Processing

ANSIMODE
CHAR 列と DOUBLE 列に存在しない値が ANSI SQL の null 値として処理されるように指定します。

AUTOCOMMIT

NOAUTOCOMMIT

トランザクションのみをサポートする DBMS で使用するために、デフォルトの数の行が更新された後に更新をテーブルに自動的に保存するか、またはロールバックを使用できるかを指定します。

ERRORSTOP

NOERRORSTOP
プロシジャにエラーが発生した場合、その実行を停止するかどうかを指定します。

LABEL

NOLABEL
列見出しとして列ラベルまたは列名を使用するかどうかを指定します。

MEMSIZE=*n* | *n*M | *n*G
割り当てられたメモリが他の PROC FEDSQL 処理をサポートするために使用できるように、基本クエリ(SELECT ステートメントなど)に使用されるメモリ量を制限するように指定します。

NOPRINT
結果の通常表示を抑制します。

NUMBER
Row という名前の列(行が取得される際のデータの行(オブザベーション)番号)を含めるように指定します。

STIMER
経過時間統計などのシステムパフォーマンス統計のサブセットが SAS ログに書き込まれるように指定します。

XCODE=ERROR | WARNING | IGNORE
NLS トランスコーディングが失敗した場合の SAS セッションの動作を制御します。

Query Planning

_METHOD
FedSQL クエリプランの簡単なテキスト説明が出力されます。

_POSTOPTPLAN
FedSQL クエリプランを示す XML ツリーが出力されます。

EXEC

NOEXEC
構文が正確かを確認した後、ステートメントを実行するかどうかを指定します。

オプション引数

_METHOD
FedSQL クエリプランの簡単なテキスト説明が出力されます。 FedSQL クエリは複数のステージに分割されます。実行の各ステージで、スタンドアロン SQL クエリが必要とされます。このオプションでは、クエリプランのノードやステージの簡単なテキスト説明が生成されます。その情報は SAS ログに書き込まれます。

注 _METHOD は、CAS サーバーでのみサポートされています。

_METHOD オプションは、ステージクエリまたは FedSQL 要求によって使用されるスレッドの数を返しません。この情報は、showStages CNTL=オ

プションで取得できます。詳細については、“SHOWSTAGES” (798 ページ)を参照してください。

参照項目: FedSQL は、CAS でのクエリプランニングとクエリ実行に関する情報を返すいくつかの方法を提供します。方法の比較については、“FedSQL クエリプランオプションの使用: 例” (*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*)を参照してください

_POSTOPTPLAN
FedSQL クエリプランを示す XML ツリーが出力されます。 FedSQL クエリは複数のステージに分割されます。実行の各ステージで、スタンドアロン SQL クエリが必要とされます。このオプションでは、FedSQL クエリプランの各ステージと各実行ステージの結果を示す XML ツリーが生成されます。その情報は SAS ログに書き込まれます。

注 _POSTOPTPLAN は、CAS サーバーでのみサポートされています。

XML ツリーは非常に長くなることがあります。かわりに_METHOD オプションを使用することもできます。

ANSIMODE
CHAR 列と DOUBLE 列に存在しない値が ANSI SQL の null 値として処理されるように指定します。 デフォルトでは、PROC FEDSQL は CHAR 列、DOUBLE 列、と VARCHAR 列に存在しない値を欠損値として処理します。SAS では、このようにして存在しない値を処理します。ANSIMODE オプションは、CHAR 列、DOUBLE 列、と VARCHAR 列に存在しない値が ANSI SQL の null 値として処理されるように指定します。違いを理解することが重要です。でないとデータが失われる可能性があります。

デフォルト SAS モード

制限事項 ANSIMODE は、CAS サーバーではサポートされていません。

注 他のすべてのデータ型は、常に ANSI NULL セマンティクスを使用します。

参照項目: “FedSQL が null および SAS 欠損値を処理する方法” (*SAS FedSQL 言語リファレンス*)

AUTOCOMMIT | NOAUTOCOMMIT
トランザクションのみをサポートする DBMS で使用するために、デフォルトの数の行が更新された後に更新をテーブルに自動的に保存するか、またはロールバックを使用できるかを指定します。

デフォルト AUTOCOMMIT

制限事項 トランザクションをサポートしないデータソースの場合、NOAUTOCOMMIT を指定するとエラーが返されます。トランザクションをサポートするデータソースには、Greenplum、Microsoft SQL Server、MySQL、ODBC に準拠したデータベース、Oracle、SAP HANA、Teradata、および Vertica が含まれます。

NOAUTOCOMMIT は、CAS サーバーではサポートされていません。

CNTL=(*instruction(s)*)
CAS における、FedSQL クエリプランニングおよびクエリ実行に対するオプションコントロール手順を指定します。 複数の手順はかっこ内で使用できます。各手順はスペースで区切ります。次の手順がサポートされています。

DISABLEPASSTHROUGH
CAS で暗黙的 FedSQL パススルーを無効にします。

FedSQL はデフォルトですべての SQL データソースに対して暗黙的パススルーを使用しようとします。FedSQL 要求が CAS の暗黙的パススルーに適格になるためには、すべてのテーブルが同じ caslib に存在し、テーブルが CAS セッションにまだロードされていない必要があります。その他の要件については、“CAS での FedSQL 暗黙的パススルー機能” (*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*)を参照してください。このオプションでは、SAS 固有の関数を含む FedSQL 要求、またはすでに CAS セッションにロードされているテーブルの処理時間を節約できます。

DYNAMICCARDINALITY
クエリプランを選択する前に基数推定を実行するように FedSQL クエリプランナーに指示します。

FedSQL クエリプランナーは、デフォルトでクエリプランを選択する前に基数推定を実行しません。基数推定は、結合条件と WHERE 句述語の選択性推定の精度を向上させることができます。これにより、一部のクエリの結合順序の決定が改善され、クエリの実行時間が短縮されます。ただし、動的基数はすべてのクエリに役立つわけではなく、クエリプランニングプロセスにオーバーヘッドを追加します。

参照項目: “動的カーディナリティ 命令の使用” (*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*)

OPTIMIZEVARBINARYPRECISION
VARBINARY 列に対して精度宣言された精度ではなく、実際のデータに適切な精度を使用して VARBINARY の精度を最適化します。

精度は、テーブルから読み取るときと、結果セットから新しいテーブルを作成するときの両方で最適化されます。このオプションの最大の利点は、宣言された精度が実際のデータの精度よりもはるかに大きい場合に達成されます。次に、このオプションを使用すると、ソーステーブルから精度を伝達するのではなく、パフォーマンスを向上させ、メモリフットプリントを削減し、必要なサイズの新しいテーブルに VARBINARY 列を作成できます。

OPTIMIZEVARCHARPRECISION
VARCHAR 列に対して精度宣言された精度ではなく、実際のデータに適切な精度を使用して VARCHAR の精度を最適化します。

精度は、テーブルから読み取るときと、結果セットから新しいテーブルを作成するときの両方で最適化されます。このオプションの最大の利点は、宣言された精度が実際のデータの精度よりもはるかに大きい場合に達成されます。次に、このオプションを使用すると、ソーステーブルから精度を伝達するのではなく、パフォーマンスを向上させ、メモリフットプリントを削減し、必要なサイズの新しいテーブルに VARCHAR 列を作成できます。

PRESERVEJOINORDER
FedSQL クエリオプティマイザーが選択した順序ではなく、指定した順序でテーブルを結合します。

REQUIREFULLPASSTHROUGH
完全クエリの暗黙的パススルーが達成できない場合、FedSQL 要求の処理を停止します。

SHOWSTAGES
クエリの実行を SAS ログに書き込みます。

これには、_METHOD オプションによって返される情報が含まれます。さらに、showStages は、ステージクエリ、各ステージで使用される SQL スレッドの数、および次のクエリ実行の詳細を出力します。

- 各中間段階の時間
- 各中間ステージからの出力行の数
- 各ステージおよびアクション全体の経過時間
- テーブルがすべてのワーカーにレプリケートされたか、自動パーティション化されたか。

情報はクエリの実行時に収集され、クライアントログに書き込まれます。showStages で NOEXEC を指定しないでください。クエリが実行されていない場合、実行の詳細を印刷することはできません。

参照項目: FedSQL は、CAS でのクエリプランニングとクエリ実行に関する情報を返すいくつかの方法を提供します。方法の比較については、*"要求の FedSQL クエリプランの表示" (SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング)*を参照してください。

注 CNTL=は、CAS サーバーでのみサポートされています。

参照項目: *"FedSQL クエリプランの変更による FedSQL パフォーマンスの最適化" (SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング)*

ERRORSTOP | NOERRORSTOP
プロシジャにエラーが発生した場合、その実行を停止するかどうかを指定します。 ERRORSTOP は、エラーが発生した後に、ステートメントの実行は停止しつつ、構文のチェックを続行するようにプロシジャに指示します。NOERRORSTOP は、プロシジャにステートメントの実行を命令し、エラー発生後も構文のチェックを継続するように命令します。

デフォルト 対話型 SAS セッションでの NOERRORSTOP、バッチまたは非対話型セッションでの ERRORSTOP

制限事項 ERRORSTOP は、SAS がバッチまたは非対話型実行モードで実行している場合にのみ有効です。

ヒント NOERRORSTOP は、エラー発生後もバッチジョブに SQL プロシジャステートメントの実行を継続させる場合に役立ちます。

EXEC | NOEXEC
構文が正確かを確認した後、ステートメントを実行するかどうかを指定します。EXEC は、ステートメントを実行するように指定します。NOEXEC は、このステートメントを実行しないように指定します。

デフォルト EXEC

ヒント NOEXEC は、ステートメントを実行せずに FedSQL ステートメントの構文をチェックする場合に役立ちます。

LABEL | NOLABEL
列見出しとして列ラベルまたは列名を使用するかどうかを指定します。 列にラベルがない場合、プロシジャでは列名が列見出しとして使用されます。

デフォルト LABEL

操作 列の別名を使用すると、列の別名は列見出しとしてラベルまたは列名よりも優先されます。

LIBS=*libref* | (*libref1libref2...librefn*)
データソース接続を、指定された 1 つまたは複数のライブラリ参照とデフォルトのライブラリに制限します。 LIBS=は、複数の LIBNAME ステートメントが SAS セッションで定義されている場合に、FedSQL 要求のスコープを要求適用先の LIBNAME ステートメントに限定する場合に便利です。また、Netezza などのネイティブカタログをサポートするデータソースに 接続するときに、重複したカタログエラーを回避できます。

複数のライブラリ参照名を指定する場合は、ライブラリ参照名をかっこで囲みます。各ライブラリ参照名はスペースで 区切ります。

別名 LIBNAMES=

制限事項 LIBS=は、CAS サーバーではサポートされません。

操作 LIBS=を指定しても、LIBS=オプションに 1 つのライブラリ参照名のみが指定されている場合でも、FedSQL ステートメントでライブラリ参照名を指定する必要性がなくなるわけではありません。テーブル名にライブラリ参照名を指定しない場合は、デフォルトのライブラリが使用されます。詳細については、“SAS および DBMS データソースへの接続” (803 ページ)を参照してください。

プロシジャステートメントに LIBS=と SESSREF= (または SESSUUID=)の両方が指定されている場合、SESSREF=が適用され、LIBS=は無視されます。

プロシジャステートメントで LIBS=を複数回指定した場合、プロシジャオプションの最後のインスタンスのライブラリ参照名のみが適用されます。

ヒント LIBS=がライブラリ割り当てに及ぼす影響を知りたい場合は、LIBS=を指定して PROC FEDSQL 要求を実行する前に MSGLEVEL=i システムオプションを設定します。このオプションで、各 LIBNAME ステートメントに対して Include および Ignore メッセージが生成されます。

例 次の PROC FEDSQL プロシジャステートメントは、SAS セッションで割り当てられた 4 つのライブラリ参照名のうち 2 つとデフォルトのライブラリのみを使用するデータソース接続を作成することを指定します。

```
proc fedsql libs=(mylib3 mylib4);
```

MEMSIZE=*n* | *n*M | *n*G
割り当てられたメモリが他の PROC FEDSQL 処理をサポートするために使用できるように、基本クエリ(SELECT ステートメントなど)に使用されるメモリ量を制限するように指定します。 メモリ制限は 1 (バイト)、1,048,576 (MB)、または 1,073,741,824 (GB)の倍数で指定します。たとえば、23M という値を指定すると、24,117,248 バイトのメモリが指定されます。16G という値を指定すると、17,179,869,184 バイトのメモリが指定されます。

デフォルト この手順では、ホスト上のメモリ量に基づいて設定が最適化されます。

注 CAS サーバーでは、MEMSIZE=で、単一の CAS ワーカーに対してメモリが指定されます。

ヒント 通常、FedSQL でメモリ問題エラーが報告されないかぎり、メモリ制限を指定する必要はありません。

NOPRINT
結果の通常表示を抑制します。

操作 NOPRINT は、ステートメントによって実行される行数が含まれる SQLOBS 自動マクロ変数の値に影響を与えます。

NUMBER
Row という名前の列(行が取得される際のデータの行(オブザベーション)番号)を含めるように指定します。

デフォルト 行番号はありません。

SESSREF=*session-reference*
CAS セッションで FedSQL ステートメントを実行することを指定します。CAS セッションは、そのセッション名によって識別されます。

要件 ご使用のシステム用に CAS サーバーを構成する必要があります。

CAS セッションは、前もって CAS ステートメントを使用して確立しておく必要があります。指定された CAS セッションが存在しない場合、プロシジャは終了します。

操作 CAS セッションに接続するには、SESSREF=または SESSUUID=を使用します。両方のオプションが指定されている場合、プロシジャステートメントの最後のオプションが適用されます。

プロシジャステートメントで SESSREF=と LIBS=の両方が指定されている場合、SESSREF=が適用され、その他のオプションは無視されます。

参照項目: “CAS サーバーへの接続” (804 ページ)

“SAS Cloud Analytic Services での FedSQL 機能” (806 ページ)

“例 7: CAS でのテーブルの明示的なロードと結合” (825 ページ)

SESSUUID="*session-uuid*"
CAS セッションで FedSQL ステートメントを実行することを指定します。CAS セッションは、その汎用一意識別子(UUID)によって識別されます。

要件 ご使用のシステム用に CAS サーバーを構成する必要があります。

CAS セッションは、前もって CAS ステートメントを使用して確立しておく必要があります。CAS ステートメントでは、UUID 値が生成されます。指定された CAS セッションが存在しない場合、プロシジャは終了します。

操作 CAS セッションに接続するには、SESSREF=または SESSUUID=を使用します。両方のオプションが指定されている場合、プロシジャステートメントの最後のオプションが適用されます。

プロシジャステートメントに SESSUUID=と LIBSF=の両方が指定されている場合、SESSUUID=が適用され、その他のオプションは無視されます。

参照項目: “CAS サーバーへの接続” (804 ページ)

“SAS Cloud Analytic Services での FedSQL 機能” (806 ページ)

例 SESSUUID=を指定している PROC FEDSQL プロシジャステートメントを次に示します。

```
proc fedsql sessuuid="76904741-fb09-554d-a8de-6cbce2a0e0e5";
```

UUID 値は、単一引用符または二重引用符で囲むことができます。

STIMER
経過時間統計などのシステムパフォーマンス統計のサブセットが SAS ログに書き込まれるように指定します。 STIMER が有効な場合、このプロシジャは各ステップと SAS セッション全体に使用されるコンピューターリソースのリストを SAS ログに書き込むように指定します。

デフォルト パフォーマンス統計は SAS ログに書き込まれません。

操作 SAS システムオプション FULLSTIMER が有効な場合、コンピューターリソースの詳細リストが SAS ログに書き込まれます。

XCODE=ERROR | WARNING | IGNORE
NLS トランスコーディングが失敗した場合の SAS セッションの動作を制御します。 トランスコーティングの失敗は、行の入力または出力操作中や、ストリングの割り当て中に発生する可能性があります。トランスコーディングは、あるエンコーディングから別のエンコーディングに文字データを変換するプロセスです。

ERROR
実行時エラーが発生し、行処理が停止するように指定します。エラーメッセージが SAS ログに書き込まれます。これはデフォルトの動作です。

WARNING
互換性のない文字が代替文字に設定されるように指定します。この結果、警告メッセージが SAS ログに書き込まれます。

IGNORE
互換性のない文字が代替文字に設定されるように指定します。SAS ログにメッセージは書き込まれません。

デフォルト ERROR

QUIT ステートメント

FEDSQL プロシジャの実行を停止します。

操作: PROC FEDSQL はステップ境界を認識しません。最初に QUIT ステートメントを指定せずに DATA ステップまたは別のプロシジャステップをサブミットすると、FedSQL 言語は構文エラーを発行し、PROC FEDSQL は処理を続行します。QUIT ステートメントは、FEDSQL プロシジャを停止するために必要です。

構文

QUIT;

詳細

FEDSQL プロシジャが QUIT ステートメントに到達すると、プロシジャによって割り当てられたすべてのリソースが 解放されます。プロシジャを再度起動しなければ、FedSQL 言語ステートメントを実行することはできなくなります。ただし、データソースサーバーへの接続は LIBNAME ステートメントを介して確立されているため、その接続が失われることはありません。この結果、LIBNAME エンジンはサーバーにすでに接続されているため、同じ libref を使用するプロシジャをこれ以降に起動すると、ほぼ即時に実行されます。

使用法: FEDSQL プロシジャ

SAS および DBMS データソースへの接続

ライブラリ参照名を指定して、PROC FEDSQL を SAS または DBMS データソースに接続します。

1 まず、接続するデータソースの SAS および SAS/ACCESS エンジンを指定する LIBNAME ステートメントをサブミットします。特定のデータソースの LIBNAME ステートメントを定義する方法については、次のリソースを参照してください。

SAS データセット
SAS グローバルステートメント: リファレンス

リレーショナル DBMS データソース
SAS/ACCESS for Relational Databases: リファレンス

MongoDB と Salesforce
SAS/ACCESS for Relational Databases: リファレンス

SPD Engine データセット
SAS Scalable Performance Data Engine: リファレンス

SPD Server テーブル
SAS Scalable Performance Data Server: ユーザーガイド

注: LIBNAME ステートメントのライブラリパスは、SAS Compute Server に認識されている必要があります。

注: PROC FedSQL で CAS エンジンライブラリ参照名を使用しないでください。FedSQL ステートメントを CAS サーバーに送信する手順については、"CAS サーバーへの接続" (804 ページ)を参照してください。

2 FedSQL 言語ステートメントでは、テーブルを参照するために *libref.table-name* 形式の名前を使用します。libref は、テーブルを作成または検索する場所を FedSQL 言語に指示します。

3 次に、PROC FEDSQL をサブミットします。

この例では、以前割り当てた libref の属性を使用して PROC FEDSQL がデータソースにどのようにアクセスするかを示します。

```
libname mylib 'library-path';
```

```
proc fedsql;
  create table mylib.mytable (col1 char(5), col2 double);
  insert into mylib.mytable values ('one', 1);
  insert into mylib.mytable values ('two', 2);
quit;
```

この例では、LIBNAME ステートメントは libref Mylib を割り当て、次に SAS データセットの物理的な場所を指定します。FedSQL ステートメントは、*libref.table-name* 形式の名前を使用してテーブルを参照します。

FEDSQL プロシジャを使用してプログラムをサブミットすると、プロシジャはすべてのアクティブな libref を含むデータソース接続文字列を内部的に構築します。次に、プロシジャはこの内部接続文字列を FedSQL プログラムに送信します。FedSQL 言語は、プログラム内の libref を使用して、この内部接続文字列内から特定のデータソース接続のプロパティを検索します。

この内部接続文字列には、主にデータソース接続(物理的な場所や認証など)の libref 属性が含まれます。文字列には、動作を定義する libref 属性が含まれていない可能性があります。たとえば、SAS V9 エンジンに以前にサブミットした LIBNAME ステートメントによって、SAS データセットが圧縮されるように指定されている場合、圧縮属性はプロシジャによって 使用されません。圧縮を要求するには、FedSQL プログラムで COMPRESS=テーブルオプションを指定する必要があります。

すべての LIBNAME ステートメントオプションに対応するテーブルオプションがあるわけではありません。いくつかの LIBNAME 動作属性は、プロシジャによって 使用されます。

> ヒント　LIBNAME ステートメントをサブミットする前に MSGLEVEL=i システムオプションを設定することにより、プロシジャが使用する LIBNAME オプションを判別できます。

プロシジャのデフォルト接続動作は次のとおりです。FedSQL プログラムで libref なしでテーブルが参照される場合、または LIBNAME ステートメントからエンジン名が省略されている場合は、デフォルトの SAS V9 エンジンが使用されます。libref なしで参照されるテーブルは、SAS Work ライブラリに作成されます。User ライブラリが定義されている場合は、User ライブラリに作成されます。

CAS サーバーへの接続

FedSQL ステートメントを CAS サーバーにサブミットするには:

1. CAS ステートメントをサブミットして、CAS セッションを確立します。
2. セッションで対話するデータソースに caslib を割り当てます(または事前定義された caslib を使用します)。CASLIB ステートメントを使用して、caslib を割り当て、CAS セッションで使用可能な caslib をリストすることができます。caslib は、SAS Data Connector (または SAS Data Connector Accelerator)を使用してデータにアクセスします。SAS Data Connector については、*SAS Cloud Analytic Services: ユーザーガイド*を参照してください。

3 PROC FEDSQL ステートメントで、CAS セッション名またはセッション識別子を使用して SESSREF=(または SESSUUID=)プロシジャオプションを指定します。SESSREF= (または SESSUUID=)を指定すると、デフォルトの接続メカニズムと LIBS=プロシジャオプションの両方が無視されます。そのかわりに、プロシジャは、指定された CAS セッションに接続します。

4 FedSQL ステートメントで caslib のテーブルを参照できるようになりました。アクティブな caslib にあるテーブルは、名前だけで参照できます。アクティブな caslib にないテーブルにアクセスするには、*caslib*.*table-name* 形式の 2 レベルの名前を使用します。

次の例は、CAS サーバーへの PROC FEDSQL 要求を示しています。

```
cas; 1

caslib castera desc='Teradata Caslib' 2
datasource=(srctype='teradata'
username='myname'
password='mypw'
server='testserver',
db='testdb');

proc fedsql sessref=casauto; 3
 create table newtable as
 select * from employees; 4
quit;
```

1 CAS ステートメントは、CAS セッションを確立します。CAS セッションのデフォルト名は Casauto です。

2 CASLIB ステートメントは、caslib Castera を割り当てます。Castera はアクティブな caslib になります。

3 PROC FEDSQL ステートメントで、SESSREF=プロシジャオプションと CAS セッション名 Casauto が指定されます。

4 FedSQL CREATE TABLE ステートメントは、テーブル Employees のすべての列を選択し、アクティブな caslib (Castera)に Newtable という名前のテーブルを作成するように指定します。アクティブな caslib に存在するテーブルの名前を caslib で修飾する必要はありません。アクティブな caslib では、テーブル名での caslib の使用はオプションです。

FedSQL の場合、テーブルを名前だけで参照することには利点があります。caslib を使用せずにアクティブな caslib でテーブル名を参照すると、CAS で FedSQL の暗黙的なパススルー(IP)が発生する可能性が高くなります。他にも要件があります。詳細については、“CAS での FedSQL 暗黙的パススルー機能” (*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*)を参照してください。

CASLIB=セッションオプションを指定した CAS ステートメントを使用して、CAS セッションのアクティブな caslib を明示的に設定または変更できます。CASLIB=セッションオプションの使用例については、“例 8: 複数の CAS ライブラリからのテーブルの結合” (830 ページ)を参照してください。

CAS ステートメント、CASLIB ステートメント、および CASUTIL プロシジャの詳細については、*SAS Cloud Analytic Services: ユーザーガイド*を参照してください。

前の例では、データコネクタを使用してデータを CAS に自動的にロードします。パスベースの場所からデータを CAS にロードすることもできます。FedSQL で処理するためにデータを CAS セッションに明示的にロードする例については、“例 7: CAS でのテーブルの明示的なロードと結合” (825 ページ)を参照してください。

PROC FEDSQL から作成および操作する CAS テーブルは、インメモリセッションテーブルです。つまり、このテーブルは CAS セッションの間は使用可能であり、現在のセッションにのみアクセス可能です。PROC FEDSQL では、テーブルをデータソースに保持したり、他の CAS セッションとテーブルを共有したりする方法は提供されていません。CAS 出力テーブルを保持または共有するには、CASUTIL プロシジャを使用します。

CAS テーブルはインメモリテーブルですが、PROC FEDSQL で、同名の既存のテーブルを上書きすることはできません。既存のテーブルを置換テーブルで上書きするには、REPLACE=テーブルオプションを指定します。または、DROP TABLE ステートメントを使用して、置換テーブルを作成する前に最初のテーブルを削除します。

SAS および DBMS データソースを使用した FedSQL 機能

SAS および DBMS データソースへの要求では、すべての FedSQL 言語ステートメントがサポートされています。FedSQL ステートメントについての情報は、*SAS FedSQL 言語リファレンス*を参照してください。

すべてのステートメントがすべてのデータソースでサポートされているわけではないことに注意してください。データ型のサポートとテーブルオプションのサポートもデータソース固有です。例については、“SAS データセットの FedSQL データ型のサポート” (810 ページ)を参照してください。

SAS Cloud Analytic Services での FedSQL 機能

PROC FEDSQL ステートメントで SESSREF=(または SESSUUID=)オプションと CAS セッション識別子を指定すると、FedSQL ステートメントは CAS サーバーに送られ、そこで fedSql.execDirect アクションによって処理されます。このアクションの詳細については、*SAS Viya プラットフォーム: システムプログラミングガイド*の fedSql.execDirect アクションを参照してください。SAS Cloud Analytic Services は、FedSQL 要求を処理するための代替環境です。

fedSql.execDirect アクションは、CAS に事前にロードされた CAS テーブル、または参照時に自動的にロードされる CAS テーブルで動作します。

CAS データで使用できる FedSQL ステートメントは、SAS および DBMS データで使用できるステートメントのサブセットです。次の FedSQL ステートメントは CAS でサポートされています。

- CREATE TABLE と AS クエリ式
- SELECT

- DROP TABLE

次の SELECT ステートメント機能は、CAS ではサポートされていません。

- EXCEPT および INTERSECT SET 操作(UNION はサポートされています)
- 相関するサブクエリ
- IN、ANY、および ALL サブクエリ
- 辞書クエリ
- ビュー
- DS2 ユーザー定義関数(DS2 パッケージ式とも呼ばれます)。

FedSQL ステートメントのテキストを CAS サーバーにサブミットする際に最良の結果を得るには、FedSQL プログラム内の各 FedSQL ステートメントと各 SELECT ステートメント句(WHERE を除く)の最初の行をインデントします。

FedSQL 機能の CAS サーバーの詳細については、*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*を参照してください。

FedSQL テーブルオプションの適用

PROC FEDSQL を使用してデータソースにアクセスするときに、後続の FedSQL ステートメントで FedSQL テーブルオプションを適用できます。テーブルオプションは、バッファーページサイズの割り当てやパスワードの指定などの処理をテーブル上で実行できるようにするアクションを指定します。

FedSQL テーブルオプションは、SAS データセットオプションが DATA ステップ処理のオプションを適用するのと同様に、FedSQL 言語処理のオプションを適用するために使用されます。SAS データセットオプションは FedSQL 言語では使用できません。たとえば、次のコードは新しいテーブルの恒久バッファーページのサイズを指定するためにテーブルオプションを SAS データセットに適用します。

```
libname myfiles base 'library-path';

proc fedsql;
  create table myfiles.table1 {options bufsize=16k}(x double) ;
  insert into myfiles.table1 values (1.0);
  insert into myfiles.table1 values (2.0);
  insert into myfiles.table1 values (3.0);
quit;
```

FedSQL は、SAS Compute Server とは異なるテーブルオプションを CAS サーバーでサポートします。CAS でサポートされているテーブルオプションのリストについては、*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*を参照してください。SAS および SAS/ACCESS データソースでサポートされているテーブルオプションのリストについては、*SAS FedSQL 言語リファレンス*を参照してください。

マクロ変数

この情報は、CAS サーバー上での FedSQL の使用には適用されません。

リテラル文字列でのマクロ変数の使用

マクロ変数を指定すると、シンボルの置換によってプログラム内でテキストを動的に変更できます。プログラム内でマクロ変数を参照すると、マクロプロセッサによって参照値は指定したマクロ変数の値に置き換えられます。

PROC FEDSQL では、後続の FedSQL ステートメントでマクロ変数を使用できます。ただし、マクロ変数がリテラル文字列内にある場合は、文字列を二重引用符で囲むことはできません。マクロプロセッサは、二重引用符を使用してマクロ変数参照を解決します。FedSQL は、二重引用符で囲まれた文字列をテーブル名または列名などの区切り識別記号(大文字/小文字を区別する)とみなします。

リテラル文字列でマクロ変数を参照するには、SAS マクロ関数%TSLIT を使用します。%TSLIT は、リテラル文字列を二重引用符で囲む必要性を無効にし、入力値を単一引用符で囲みます。たとえば、次のステートメントには&SYSHOSTNAME マクロ変数を指定するための%TSLIT 関数が含まれており、これによって、それが実行されるコンピューターのホスト名が返されます。

```
select %tslit(&syshostname);
```

%TSLIT マクロ関数は、デフォルトのオートコールマクロライブラリに保存されます。詳細については、*“区切り識別子でのマクロ変数の参照” (SAS FedSQL 言語リファレンス)*を参照してください。

プロシジャによるマクロ変数セットの使用

PROC FEDSQL は、各ステートメントの実行後、特定の値を使用してマクロ変数を設定します。これらのマクロ変数はマクロ内部でテストし、PROC ステップの実行を継続するかどうかを決定できます。各ステートメントの実行後、次のマクロ変数がそれぞれの値を使用して更新されます。

SQLRC
: PROC FEDSQL ステートメントの成功を示す次のステータス値が格納されます。

  0
  : PROC ステートメントはエラーなしで正常に完了しました。

  4
  : PROC ステートメントで警告が発行される状況が発生しました。ステートメントは引き続き実行されます。

  8
  : PROC ステートメントでエラーが発生しました。ステートメントはこの時点で停止されました。

16
PROC ステートメントで実行時エラーが発生しました。たとえば、このエラーコードが使用されるのは、サブクエリ(1 つの値のみを返す)が複数の行に対して評価するときです。このようなエラーが検出されるのは、実行時のみです。

パスワード

この情報は、CAS サーバー上での FedSQL の使用には適用されません。

SAS ソフトウェアを使用して、ファイルに SAS パスワードを割り当てることで、SAS データセットや SPD Engine データセットへのアクセスを制限できます。読み取り、書き込み、変更という、3 つのレベルの保護を 指定できます。

注: SAS パスワードは、SAS 以外での SAS ファイルへのアクセスを制御しません。SAS の外部にある SAS ファイルへのアクセスを制御するには、オペレーティングシステムで提供されているユーティリティやファイルシステムセキュリティを 使用する必要があります。SAS パスワードの詳細については、*SAS FedSQL 言語リファレンス*を参照してください。

PROC FEDSQL では、FedSQL テーブルオプション(ALTER=、PW=、READ=、および WRITE=)を使用してデータソースのパスワードを割り当てるか、指定することができます。たとえば、次のコードでは、READ、WRITE、および ALTER の各パスワードを SAS データセットに割り当てるために FedSQL テーブルオプション PW=を適用します。

```
libname myfiles base 'library-path';

proc fedsql;
  create table myfiles.table1 {options pw=luke}(x double) ;
  insert into myfiles.table1 values (1.0);
  insert into myfiles.table1 values (2.0);
  insert into myfiles.table1 values (3.0);
quit;
```

このコードは、データセットを読み取るためにテーブルオプションを指定する方法を示しています。

```
select * from myfiles.table1 {options pw=luke};
```

CAS テーブルは SAS パスワードをサポートしていません。CAS テーブルにパスワードを割り当てることはできません。CAS からパスワードで保護されたデータにアクセスする場合、パスワードはデータにアクセスするために使用される CAS 言語要素で指定されます。たとえば、データを CAS にロードする CASUTIL プロシジャ、または CASLIB ステートメントではパスワードがサポートされています。詳細については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector ドキュメントを参照してください。

暗号化

SAS ソフトウェアを使用すると、SAS データセット、SPD Engine データセット、および SPD Server テーブルのコンテンツを暗号化できます。SAS では、SAS 独自の暗号化と AES 暗号化がサポートされています。

SAS 独自の暗号化は、ENCRYPT=テーブルオプションを PW=または READ=テーブルオプションで指定することによって実行されます。SAS 独自の暗号化で暗号化されたデータセットまたはテーブルは、適切なパスワードを PW=または READ=テーブルオプションで指定して解読する必要があります。

AES 暗号化は、ENCRYPT=テーブルオプションを ENCRYPTKEY=テーブルオプションと一緒に指定することによって実行されます。AES 暗号化で暗号化されたデータセットまたはテーブルは、後で ENCRYPTKEY=テーブルオプションを適切なキー値で指定することによって解読されます。SAS は、次の 2 つのレベルの AES 暗号化をサポートしています: AES および AES2。AES2 は、AES よりも新しく安全な暗号化規格を満たしています。詳細については、“ENCRYPT= テーブルオプション” (*SAS FedSQL 言語リファレンス*)を参照してください。

FedSQL では現在、CAS テーブルの暗号化属性はサポートされていません。CAS から SAS で暗号化されたデータセットや AES で暗号化されたデータセットにアクセスする際、パスワードおよび暗号化キーは、データにアクセスするために使用される CAS 言語要素で指定されます。たとえば、データを CAS にロードする CASUTIL プロシジャ、または CASLIB ステートメントではパスワードと暗号化キーがサポートされています。詳細については、*SAS Cloud Analytic Services: ユーザーガイ ド*の SAS Data Connector ドキュメントを参照してください。

SAS データセットの FedSQL データ型のサポート

PROC FEDSQL では、FedSQL ステートメントをサブミットするときに、すべての FedSQL 言語データ型がサポートされます。FedSQL データ型の詳細については、*SAS FedSQL 言語リファレンス*を参照してください。ただし、データの読み取りと作成の際、FedSQL データ型はターゲットデータソースのデータ型との間で変換されます。次のテーブルには、FedSQL データ型と、SAS データ型への変換方法および SAS データ型からの変換方法が一覧表示されています。

表 23.1 SAS データセットの FedSQL データ型の変換

FedSQL データ型	SAS のデータ型	説明
BIGINT	SAS 数値	SAS 数値は、正確な数値データ型ではなく近似数値データ型である DOUBLE であるため、精度が失われる可能性があります。
BINARY(*n*)	SAS 文字	SAS 出力形式$*n* を適用します。

FedSQL データ型	SAS のデータ型	説明
CHAR(*n*)	SAS 文字	SAS 出力形式$*n* を適用します。
DATE	SAS 数値 [1]	SAS 出力形式 DATE9 を適用します。
DECIMAL|NUMERIC(*p*,**s**)	SAS 数値	
DOUBLE	SAS 数値	
FLOAT(*p*)	SAS 数値	
INTEGER	SAS 数値	
NCHAR(*n*)	SAS 文字	SAS 出力形式$*n* を適用します。
NVARCHAR(*n*)	SAS 文字	SAS 出力形式$*n* を適用します。
REAL	SAS 数値	
SMALLINT	SAS 数値	
TIME(*p*)	SAS 数値	SAS 出力形式 TIME8 を適用します。
TIMESTAMP(*p*)	SAS 数値	SAS 出力形式 DATETIME19.2 を適用します。
TINYINT	SAS 数値	
VARBINARY(*n*)	SAS 文字	SAS 出力形式$*n* を適用します。
VARCHAR(*n*)	SAS 文字	SAS 出力形式$*n* を適用します。

1 有効な SAS 日付値は、1582-01-01 から 9999-12-31 までです。SAS 日付範囲外の日付はサポートされていないため、無効な日付として処理されます。

DBMS データソースのデータ型のサポートの詳細については、"データ型リファレンス" (*SAS FedSQL 言語リファレンス*)を参照してください。

CAS テーブルの FedSQL データ型のサポート

FedSQL は CAS ネイティブデータ型を CAS でのみサポートしています。次のテーブルは、FedSQL データ型が CAS ネイティブ型にどのようにマップされるかを示しています。

表 23.2 CAS テーブルの FedSQL データ型の変換

FedSQL データ型	CAS のデータ型	説明
BIGINT	INT64	大きな符号付き、正確な整数。
CHAR(*n*)	CHAR	固定長文字列、ここで *n* は文字の最大保存数です。
DOUBLE	DOUBLE	符号付き、近似、倍精度、浮動小数点数。CAS サーバーの場合、これは 64 ビット倍精度浮動小数点数です。
INTEGER	INT32	正規の符号付き、正確な整数。
VARBINARY[1]	VARBINARY	可変長のバイナリ不透明(OPAQUE)データ。このデータ型は、画像データ、音声、ドキュメント、およびその他の非構造化データを格納するために使用されます。
VARCHAR(*n*)	VARCHAR	可変長文字列、ここで *n* は文字の最大保存数です。

1 FedSQL は、CAS テーブルの VARBINARY 列を読み書きできます(CREATE TABLE AS を使用)。ただし、PUT 関数を使用して VARBINARY 列に$HEX.フォーマットを適用しない限り、SELECT ステートメントは一部のクライアントで VARBINARY 列を表示しません。BINARY データを読み取ろうとすると、エラーが発生します。

CAS テーブルの日付、時刻、およびタイムスタンプの値は、SAS 形式が適用された DOUBLE としてサポートされます。SAS エンジンによって作成されたテーブルから CAS に日付、時刻、およびタイムスタンプの値を含む数値列を読み取る場合、FedSQL は既存の形式を保持します。他の入力データソースの場合は、DATE9 が適用されます。日付値への SAS 形式、TIME8。時刻値への SAS 形式、および日時値への DATETIME25.6 SAS 形式。

CAS テーブルは、デフォルトで UTF-8 文字セットを使用します。

FedSQL が CAS の欠損値をどのように処理するかを理解することが重要です。*“存在しないデータの取り扱い” (SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング)*を参照してください。

PROC FEDSQL と PROC SQL: 比較

"PROC SQL と FedSQL の高レベル比較"を参照してください。

例: FEDSQL プロシジャ

例 1: SAS データセットを作成する

要素: PROC FEDSQL ステートメント
FedSQL CREATE TABLE ステートメント
FedSQL INSERT ステートメント
FedSQL SELECT ステートメント
QUIT ステートメント
LIBNAME ステートメント

詳細

注: FedSQL 言語を使用したデータ定義は CAS サーバーではサポートされていません。CAS では、FedSQL は既存のテーブルから列を選択して結合することによってのみ新しいテーブルを作成できます。

この例では、FEDSQL プロシジャで FedSQL ステートメントをサブミットして、SAS データセットを作成します。この例では、FedSQL CREATE TABLE、INSERT、および SELECT ステートメントをサブミットします。

プログラム

```
libname mybase 'library-path';

proc fedsql;

create table mybase.sales (prodid double not null,
                           custid double not null,
                           totals double having format dollar10.,
                           country char(30));

insert into mybase.sales values (3234, 1, 189400, 'United States');
insert into mybase.sales values (1424, 3, 555789, 'Japan');
insert into mybase.sales values (3421, 4, 781183, 'Japan');
insert into mybase.sales values (3422, 2, 2789654, 'United States');
insert into mybase.sales values (3975, 5, 899453, 'Argentina');
```

```
title 'SAS Data Set "Sales"';
select * from mybase.sales;

quit;
```

プログラムの説明

SAS ライブラリの libref を割り当てます。 LIBNAME ステートメントは、libref Mybase を割り当てて、SAS データセットの物理的場所を指定します。LIBNAME ステートメントではエンジンが指定されていないため、データセットの作成にはデフォルトの SAS V9 エンジンが使用されます。

```
libname mybase 'library-path';
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントでは、FedSQL ステートメントをサブミットする環境が設定されます。

```
proc fedsql;
```

FedSQL CREATE TABLE ステートメントを入力します。 このステートメントでは、Sales というテーブルを作成することが指定されます。FedSQL CREATE TABLE ステートメントによって、カタログ識別子 Mybase (割り当てられた libref)が指定されます。変数宣言では、列 ProdId および CustId に対して NOT NULL 一貫性制約も定義され、SAS 出力形式 DOLLARw.が列 Totals に適用されます。FedSQL は、SAS データソースの SAS 拡張機能として SAS 出力形式、入力形式、およびラベルをサポートします。

```
create table mybase.sales (prodid double not null,
                           custid double not null,
                           totals double having format dollar10.,
                           country char(30));
```

INSERT ステートメントを入力して、テーブルにデータを設定します。 テーブル名の前に各ステートメントのカタログ識別子が付いていることに注意してください。各ステートメントのデータ値は、キーワード VALUES が前にあるカンマ区切り文字列でサブミットされます。

```
insert into mybase.sales values (3234, 1, 189400, 'United States');
insert into mybase.sales values (1424, 3, 555789, 'Japan');
insert into mybase.sales values (3421, 4, 781183, 'Japan');
insert into mybase.sales values (3422, 2, 2789654, 'United States');
insert into mybase.sales values (3975, 5, 899453, 'Argentina');
```

FedSQL SELECT ステートメントを入力して、テーブルのコンテンツを表示します。 SELECT ステートメント内のアスタリスクは、FROM 句で指定されたテーブルのすべての列と行の値を表示することを指定します。テーブル名の前にカタログ識別子が付いていることに注意してください。SAS グローバルステートメント TITLE は、表示のタイトルを指定します。

```
title 'SAS Data Set "Sales"';
select * from mybase.sales;
```

プロシジャを停止します。 QUIT ステートメントは、プロシジャを終了します。

```
quit;
```

SAS Data Set "Sales"

Obs	PRODID	CUSTID	TOTALS	COUNTRY
1	3234	1	$189,400	United States
2	1424	3	$555,789	Japan
3	3421	4	$781,183	Japan
4	3422	2	$2,789,654	United States
5	3975	5	$899,453	Argentina

例 2: 既存のテーブルから新しいテーブルを作成

要素:

PROC FEDSQL ステートメント
CREATE TABLE ステートメントと AS 式の使用
LIBNAME ステートメント
3 つのテーブルの結合

詳細

この例では、PROC FEDSQL と、AS クエリ式構文の CREATE TABLE ステートメントを使用して、既存のテーブルから新しい SAS データセットを作成します。クエリ式では、既存の 3 つのテーブル(SAS データセット Sales、SPD Engine データセット Products、および Oracle テーブル Customers)から行が選択され、Results という新しいテーブルが作成されます。

プログラム

```
libname mybase base 'library-path';
libname myspde spde 'library-path';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;

proc fedsql;

  create table mybase.results as
    select products.prodid, products.product, customers.name,
      sales.totals, sales.country
    from myspde.products, mybase.sales, myoracle.customers
    where products.prodid = sales.prodid and
      customers.custid = sales.custid;

  select * from mybase.results;
```

```
quit;
```

プログラムの説明

3 つの libref を割り当てます。 1 つ目の LIBNAME ステートメントは libref Mybase を割り当て、ベースエンジンを指定し、作成する SAS データセットの物理的な場所を指定します。2 番目の LIBNAME ステートメントは、libref Myspde を割り当て、SPDE エンジンを指定し、既存の SPD Engine データセットの物理的な場所を指定します。3 番目の LIBNAME ステートメントは、libref Myoracle を割り当て、ORACLE エンジンを指定し、既存の Oracle テーブルを含む Oracle データベースへの接続情報を指定します。

```
libname mybase base 'library-path';
libname myspde spde 'library-path';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントでは、FedSQL ステートメントをサブミットするための環境が設定されます。

```
proc fedsql;
```

新しいテーブルを作成します。 CREATE TABLE ステートメントは、クエリ式を使用して既存のデータセットから行を選択して、既存の 3 つの SAS データセットから新しい SAS データセットを作成します。SELECT ステートメントは、既存のデータセットから適格な列と行を取得し、新しい SAS データセットを作成します。

```
create table mybase.results as
  select products.prodid, products.product, customers.name,
    sales.totals, sales.country
  from myspde.products, mybase.sales, myoracle.customers
  where products.prodid = sales.prodid and
    customers.custid = sales.custid;
```

SAS データセット内のデータを取得します。 この 2 番目の SELECT ステートメントでは、出力データセットのコンテンツが表示されます。

```
select * from mybase.results;
```

プロシジャを停止します。

```
quit;
```

出力: 既存のテーブルから新しいテーブルを作成

アウトプット 23.1　出力データセット Mybase.Results のコンテンツ

PRODID	PRODUCT	NAME	TOTALS	COUNTRY
3234	Rice	Peter Frank	189,400	United States
3422	Oat	Jim Stew art	2789654	United States
1424	Corn	Janet Chien	555,789	Japan
3421	Wheat	Qing Ziao	781,183	Japan
3975	Barley	Humberto Sertu	899,453	Argentina

例 3: 相関サブクエリを使用したデータのクエリ

要素:

PROC FEDSQL ステートメント
FedSQL SELECT ステートメント
LIBNAME ステートメント

詳細

この例は、相関サブクエリを使用したデータのクエリを示しています。相関サブクエリでは、サブクエリ内の WHERE 句が外部クエリ内のテーブルの値を参照します。相関サブクエリは、外部クエリ内の各行について評価されます。相関サブクエリを使用すると、FedSQL はサブクエリと外部クエリを同時に実行します。FedSQL は、異種相関サブクエリを実行できます。FedSQL は、データソースによって 実行されるサブクエリを送信し、データソースから転送される結果セットを制限します。

注: 相関サブクエリは、CAS サーバーではまだサポートされていません。

プログラム

```
libname myspde spde 'library-path';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;

proc fedsql;

  title 'Result of Correlated Subquery';
  select * from myspde.product
    where exists (select * from myoracle.sales
```

```
    where product.prodid=sales.prodid);
quit;
```

プログラムの説明

2 つの libref を割り当てます。 最初の LIBNAME ステートメントは、libref Myspde を割り当て、SPDE エンジンを指定し、SPD Engine データセットの物理的な場所を指定します。2 つ目の LIBNAME ステートメントは libref Myoracle を割り当て、Oracle エンジンを指定し、Oracle データベースへの接続情報を指定します。

```
libname myspde spde 'library-path';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントは、FedSQL ステートメントをサブミットするための環境を作成します。

```
proc fedsql;
```

相関クエリをサブミットします。 クエリは、Sales テーブルの ProdId 値と一致する ProdId 値を持つ Products テーブルのすべての 列から値を取得するように指定します。グローバルステートメント TITLE は、出力のタイトルを指定します。

```
title 'Result of Correlated Subquery';
select * from myspde.product
  where exists (select * from myoracle.sales
  where product.prodid=sales.prodid);
```

プロシジャを停止します。

```
quit;
```

出力: 相関サブクエリを使用したデータのクエリ

Result of Correlated Subquery

PRODID	PRODUCT
3234	Rice
1424	Corn
3421	Wheat
3422	Oat
3975	Barley

例 4: 結合を実行するためのインデックスの作成および使用

要素: PROC FEDSQL ステートメント
FedSQL 言語ステートメント
LIBNAME ステートメント

詳細

この例では、Oracle テーブルのインデックスを作成する方法を示します。次に、インデックスを使用して、Oracle テーブルと SPD Engine データセットの結合を実行する方法を示します。

注: CREATE INDEX ステートメントは CAS サーバーではサポートされません。

プログラム

```
libname myspde spde 'library-path';
libname myoracle oracle connection-options;

proc fedsql;

  create index prodid on myoracle.sales (prodid);

  title 'Result of Indexed Join';
  select * from myspde.product, myoracle.sales
    where product.prodid=sales.prodid;

quit;
```

プログラムの説明

データソースの libref を割り当てます。 最初の LIBNAME ステートメントは、libref Myspde を割り当て、SPDE エンジンを指定し、SPD Engine データセットの物理的な場所を指定します。2 番目の LIBNAME ステートメントは libref Myoracle を割り当て、ORACLE エンジンを指定し、Oracle データベースへの接続情報を指定します。

```
libname myspde spde 'library-path';
libname myoracle oracle connection-options;
```

PROC FEDSQL ステートメントを実行します。 PROC FEDSQL ステートメントは、FedSQL ステートメントをサブミットするための環境を作成します。

```
proc fedsql;
```

Oracle テーブルのインデックスを作成します。 CREATE INDEX ステートメントによって、Sales という名前の Oracle テーブルの ProdId 列で ProdId という名前のインデックスが作成されます。

```
create index prodid on myoracle.sales (prodid);
```

列と行を取得します。 SELECT ステートメントは、Product という名前の SPD Engine データセットと Sales という名前の Oracle テーブルからデータを取得します。インデックスが Oracle データベース内にある場合でも、FedSQL はインデックスを活用して結合を実行できます。

```
title 'Result of Indexed Join';
select * from myspde.product, myoracle.sales
  where product.prodid=sales.prodid;
```

プロシジャを停止します。

```
quit;
```

出力: 結合を実行するためのインデックスの作成および使用

Result of Indexed Join

PRODID	PRODUCT	Product ID	Customer ID	TOTALS	COUNTRY
3234	Rice	3234	1	$189,400	United States
1424	Corn	1424	3	$555,789	Japan
3421	Wheat	3421	4	$781,183	Japan
3422	Oat	3422	2	$2,789,654	United States
3975	Barley	3975	5	$899,453	Argentina

例 5: DS2 パッケージの式での使用

要素:

FedSQL SELECT ステートメント
パッケージメソッド宣言
DS2 PACKAGE ステートメント
DS2 METHOD ステートメント
DS2 DATA ステートメント

詳細

FedSQL 言語では、ユーザー定義の DS2 パッケージメソッドを SELECT ステートメントの関数として呼び出す機能がサポートされています。この例では、PROC FEDSQL から Numbers というテーブルに Add という DS2 パッケージメソッドを作成してサブミットします。パッケージメソッドとテーブルは、Work ライブラリに作成されます。この機能は、サードパーティのデータベースに接続する場合にも使用できます。

注: PROC FEDSQL から実行されるパッケージメソッドは、メソッド内に入力引数のみを持つことができます。詳細については、*"式での DS2 パッケージの使用" (SAS FedSQL 言語リファレンス)*を参照してください。

注: ユーザー定義のパッケージメソッドは、CAS サーバーではサポートされていません。

プログラム

```
proc ds2;

  package adder / overwrite =yes;

  method add( double x, double y ) returns double;
  return x + y;
 end;
endpackage;

data numbers / overwrite = yes;
  dcl double x y;
  method init();
  dcl int i;
  do i = 1 to 10;
  x = i; y = i * i;
  output;
 end;
end;
enddata;

run;

quit;

proc fedsql;

  select x, y, work.adder.add( x, y ) as z from work.numbers;

quit;
```

プログラムの説明

DS2 プロシジャを呼び出します。

```
proc ds2;
```

パッケージを定義します。 この PACKAGE ステートメントでは、Adder というパッケージを作成することが指定されます。

```
package adder / overwrite =yes;
```

メソッドを定義します。 この METHOD ステートメントでは、Add というメソッドを作成することが指定されます。メソッド Add には、2 つの入力変数 X と Y (両方とも DOUBLE 型)があり、DOUBLE 型の出力が戻されます。出力変数には、変数 X の値を変数 Y に加算した合計が含まれます。END および ENDPACKAGE ステートメントでは、METHOD および PACKAGE 宣言完了のシグナルが送信されます。

```
method add( double x, double y ) returns double;
return x + y;
end;
endpackage;
```

テーブルを作成します。 この DATA ステートメントでは、Numbers というテーブルを作成することが指定されます。ライブラリが指定されていないため、Work ライブラリが使用されます。テーブルには、DOUBLE 型の 2 つの列 X および Y があります。システム INIT メソッドが呼び出され、テーブルに値が設定されます。変数 I は、入力値を保持するように定義されています。DO ステートメントは、変数 I を使用して、1 から 10 までの値を含む行を列 X に挿入します。次に、I の各インスタンスをそれ自体に掛けて、結果を列 Y に挿入します。END および ENDDATA ステートメントでは、DO ステートメント、INIT メソッド、および DATA ステートメント宣言完了のシグナルが送信されます。

```
data numbers / overwrite = yes;
  dcl double x y;
  method init();
  dcl int i;
  do i = 1 to 10;
  x = i; y = i * i;
  output;
 end;
end;
enddata;
```

DS2 ステートメントをサブミットします。

```
run;
```

DS2 プロシジャを停止します。

```
quit;
```

FEDSQL プロシジャを呼び出します。

```
proc fedsql;
```

テーブルのパッケージメソッドを呼び出す SELECT ステートメントをサブミットします。 このステートメントでは、列 X と Y を選択するように指定されており、パッケージメソッド式の結果はテーブル Numbers の列 Z です。パッケージメソッドは、[catalog.][schema.]package.method の形式で 3 部構成の名前を使用して参照されます。

```
select x, y, work.adder.add( x, y ) as z from work.numbers;
```

FEDSQL プロシジャを停止します。

```
quit;
```

SELECT ステートメントの出力を次に示します。

x	y	Z
1	1	2
2	4	6
3	9	12
4	16	20
5	25	30
6	36	42
7	49	56
8	64	72
9	81	90
10	100	110

例 6: CAS でのデータのクエリ

要素:

PROC FEDSQL ステートメント
SESSREF=プロシジャオプション
FedSQL 言語ステートメント
CAS ステートメント
CASLIB ステートメント

詳細

この例は、CAS から Employees という名前のデータベーステーブルをクエリするために必要な手順を示しています。

プログラム

```
cas;

caslib castera desc='Teradata Caslib'
  datasource=(srctype='teradata',
    dataTransferMode='serial',
    username='myname',
    password='mypw',
    server='testserver',
    db='test');

proc fedsql sessref=casauto;

select Pos, count(Pos) as Count_Pos
   from employees
   group by Pos
   having count(Pos) >= 2;

quit;
```

プログラムの説明

CAS サーバー上のセッションを確立します。 CAS ステートメントをサブミットすることによって、CAS セッションを確立します。この CAS セッションは、デフォルト名 Casauto で作成されます。

```
cas;
```

caslib を追加します。 CAS では、ライブラリ参照名の代わりに caslib を使用してターゲットデータソースを識別します。この CASLIB ステートメントは、Teradata データベースへの接続を定義し、caslib Castera を割り当てます。Castera は、CAS セッションでアクティブな caslib になります。

```
caslib castera desc='Teradata Caslib'
  datasource=(srctype='teradata',
    dataTransferMode='serial',
    username='myname',
    password='mypw',
    server='testserver',
    db='test');
```

PROC FEDSQL ステートメントを SESSREF=プロシジャオプション付きで指定します。SESSREF=プロシジャオプションで、CAS セッション名を指定します。 SESSREF=オプションは Casauto を指定します。SESSREF=オプションは、fedSql.execDirect アクションに続く FedSQL 言語ステートメントを渡し、それらを CAS セッション Casauto で処理するようにプロシジャに指示します。

```
proc fedsql sessref=casauto;
```

FEDSQL ステートメントを指定します。 この SELECT ステートメントは、2 人以上の従業員が記載されたデータベーステーブル Employees にすべての役職を一覧表示するよう指定します。テーブル参照は、テーブル Employees がアクティブな caslib に存在することを示しています。FedSQL 言語は、CAS での単一ソースのフルクエリの暗黙的な SQL パススルーをサポートします。ターゲットテーブルがまだ CAS セッションにロードされていない場合、要求は暗黙的なパススルーについて評

価されます。暗黙的なパススルーは、適格な要求を処理のためにデータソースに渡し、結果セットを CAS にロードします。FedSQL の暗黙的なパススルーは、CAS セッションにまだロードされていないテーブルに対してのみ可能です。他にも重要な要件があります。これらの要件の詳細については、*"CAS での FedSQL 暗黙的パススルー機能" (SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング)*を参照してください。.パススルーの対象とならないアンロードされたテーブルは、CAS サーバーで処理するために CAS セッションに自動的にロードされます。CAS セッションにすでに存在するテーブルは、CAS サーバーによって処理されます。

```
select Pos, count(Pos) as Count_Pos
   from employees
   group by Pos
   having count(Pos) >= 2;
```

プロシジャを停止します。

```
quit;
```

出力: CAS からのデータベースクエリの結果

アウトプット 23.2　CAS データベースクエリの出力

Pos	COUNT_POS
Manager	4.000000
Developer	2.000000
Sales Associate	2.000000

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK
72
73             proc fedsql sessref=casauto;
74             select Pos, count(Pos) as Count_Pos
75                from castera.employees
76                groupby Pos
77                having count(Pos) >=2;
NOTE: The SQL statement was fully offloaded to the underlying data source via
full pass-through
78             quit;
```

例 7: CAS でのテーブルの明示的なロードと結合

要素:

PROC FEDSQL ステートメント
SESSREF=プロシジャオプション
CAS ステートメント
CASLIB ステートメント

CAS LIBNAME ステートメント
DATA ステップ
CASUTIL プロシジャ

詳細

場合によっては、データを正常に処理する前に、いくつかのフォーマットが必要になります。この例では、カンマ区切りのデータを含む 3 つのファイルを CAS に明示的にロードします。CAS LIBNAME エンジンと DATA ステップは、テーブルのフォーマットとロードに使用されます。次に、テーブルが CAS に配置された後、PROC FEDSQL を使用してテーブルを結合し、結果セットを含む新しい CAS テーブルを作成します。入力ファイルには、Supplier、Nation および Customer が指定されています。出力 CAS テーブルの名前は Newtable です。すべての CAS テーブルはインメモリテーブルにあります。PROC CASUTIL を使用して保存またはプロモートしない限り、CAS セッションの終了時に消えます。

この例では、caslib とライブラリ参照名を割り当てます。ライブラリ参照名は、CASLIBNAME ステートメントで caslib にマップされます。DATA ステップはそのライブラリ参照名で実行されます。PROC FEDSQL プロシジャステートメントで SESSREF=オプションが指定されている場合、FedSQL ステートメントは caslib で実行されます。

注: FEDSQL 要求をフォーマットする場合、ステートメントや句の前の先頭のスペースが重要であることに注意してください。ステートメントと句の先頭を左余白揃えにしないでください。引用符で囲まれた文字列に改行を入れる場合は、常に改行の後に少なくとも 1 つ空白を入れるようにしてください。

プログラム

```
cas;

caslib casdata path='/r/ge.unx.company.com/vol/vol210/u21/myID/hold';

libname mycas cas sessref=casauto caslib=casdata;

data mycas.supplier;
  infile "/r/ge.unx.company.com/vol/vol210/u21/myID/hold/supplier.tbl" delimiter='|';
  length S_SUPPKEY 8. S_NAME VARCHAR(25) S_ADDRESS VARCHAR(40) S_NATIONKEY 8.
S_PHONE VARCHAR(15) S_ACCTBAL 8. S_COMMENT VARCHAR(101);
  input  S_SUPPKEY S_NAME S_ADDRESS S_NATIONKEY S_PHONE S_ACCTBAL S_COMMENT;
run;

data mycas.nation;
  infile "/r/ge.unx.company.com/vol/vol210/u21/myID/hold/nation.tbl" delimiter='|';
  length N_NATIONKEY 8. N_NAME VARCHAR(25) N_REGIONKEY 8. N_COMMENT
VARCHAR(152);
  input  N_NATIONKEY N_NAME N_REGIONKEY N_COMMENT;
run;
```

```
data mycas.customer;
 infile "/r/ge.unx.company.com/vol/vol210/u21/myID/hold/customer.tbl" delimiter='|';
 length C_CUSTKEY 8. C_NAME VARCHAR(25) C_ADDRESS VARCHAR(40) C_NATIONKEY 8.
C_PHONE VARCHAR(15) C_ACCTBAL 8. C_MKTSEGMENT VARCHAR(10) C_COMMENT
VARCHAR(117);
 input  C_CUSTKEY C_NAME C_ADDRESS C_NATIONKEY C_PHONE C_ACCTBAL
C_MKTSEGMENT
 C_COMMENT;
run;

proc casutil;
 list tables incaslib="casdata";
run;

proc fedsql sessref=casauto;

 create table newtable {options replace=true} as
 select
  s_name, s_acctbal, n_name, sum_c_acctbal
 from
  supplier,
  nation,
  (select c_nationkey, sum(c_acctbal) as sum_c_acctbal from customer group by
c_nationkey) C
  where
  s_nationkey = n_nationkey and
  s_nationkey = c_nationkey
 ;

select * from newtable;

quit;
```

プログラムの説明

CAS サーバー上のセッションを確立します。 この手順は、CAS セッションがまだ存在しない場合、または 2 つ目の CAS セッションを作成する場合にのみ必要です。このセッションでは、デフォルト名 Casauto を使用します。

```
cas;
```

入力ファイルを指す caslib を割り当てます。 この CASLIB ステートメントでは、PATH=パラメーターで指定された場所に caslib Casdata が割り当てられます。パス指定には、絶対パス名を使用する必要があります。

```
caslib casdata path='/r/ge.unx.company.com/vol/vol210/u21/myID/hold';
```

CAS エンジンのライブラリ参照名を割り当てます。 LIBNAME ステートメントは、ライブラリ参照名 Mycas、CAS エンジン、および Casdata caslib を指定します。LIBNAME ステートメントは CAS エンジンを呼び出し、ライブラリ参照名を caslib にマップします。

```
libname mycas cas sessref=casauto caslib=casdata;
```

DATA ステップを使用して、最初のファイルをフォーマットして CAS にロードします。 DATA ステートメントは、Mycas.Supplier というテーブルを作成するように指定します。CAS エンジンは、出力テーブルを CAS テーブルとして作成します。SAS セッションでは、テーブルは Mycas.Supplier と呼ばれます。CAS セッション

(Casauto)では、テーブルは Casdata.Supplier と呼ばれます。INFILE=ステートメントは、|(パイプ記号)を 列区切り文字として使用してファイルの内容を読み取ることを指定します。LENGTH ステートメントは、出力テーブルの列名と長さを指定します。INFILE 指定は、CASLIB ステートメントで指定したパスと相対的になる場合があることに注意してください。CAS エンジンは、caslib Casdata にテーブル Supplier を作成します。

```
data mycas.supplier;
  infile "/r/ge.unx.company.com/vol/vol210/u21/myID/hold/supplier.tbl" delimiter='|';
  length S_SUPPKEY 8. S_NAME VARCHAR(25) S_ADDRESS VARCHAR(40) S_NATIONKEY 8.
S_PHONE VARCHAR(15) S_ACCTBAL 8. S_COMMENT VARCHAR(101);
  input  S_SUPPKEY S_NAME S_ADDRESS S_NATIONKEY S_PHONE S_ACCTBAL S_COMMENT;
run;
```

2 番目のファイルをフォーマットして CAS にロードします。 この DATA ステートメントでは、Mycas.Nation という CAS テーブルを作成することを指定します。INFILE=ステートメントは、パイプ記号を区切り文字として使用してファイルの内容を読み取ります。LENGTH ステートメントは、出力テーブルの列名と長さを指定します。CAS エンジンは、caslib Casdata にテーブル Nation を作成します。

```
data mycas.nation;
  infile "/r/ge.unx.company.com/vol/vol210/u21/myID/hold/nation.tbl" delimiter='|';
  length N_NATIONKEY 8. N_NAME VARCHAR(25) N_REGIONKEY 8. N_COMMENT
VARCHAR(152);
  input  N_NATIONKEY N_NAME N_REGIONKEY N_COMMENT;
run;
```

3 番目のファイルをフォーマットして CAS にロードします。 DATA ステップは、MyCas.Customer という CAS テーブルを作成することを指定します。INFILE=ステートメントは、ファイルの内容を読み取ります。LENGTH ステートメントは、出力テーブルの列名と長さを指定します。CAS エンジンは、caslib Casdata にテーブル Customer を作成します。

```
data mycas.customer;
  infile "/r/ge.unx.company.com/vol/vol210/u21/myID/hold/customer.tbl" delimiter='|';
  length C_CUSTKEY 8. C_NAME VARCHAR(25) C_ADDRESS VARCHAR(40) C_NATIONKEY 8.
 C_PHONE VARCHAR(15) C_ACCTBAL 8. C_MKTSEGMENT VARCHAR(10) C_COMMENT
VARCHAR(117);
  input  C_CUSTKEY C_NAME C_ADDRESS C_NATIONKEY C_PHONE C_ACCTBAL
C_MKTSEGMENT
  C_COMMENT;
run;
```

ファイルが CAS に作成されたことを確認します。 CASUTIL プロシジャをサブミットして、caslib Casdata で使用可能なテーブルをリスト出力します。

```
proc casutil;
  list tables incaslib="casdata";
run;
```

PROC FEDSQL ステートメントを指定します。 PROC FEDSQL ステートメントでは、SESSREF=プロシジャオプションで CAS セッション名を指定します。SESSREF=オプションは Casauto を指定します。CAS セッションでの以前のアクティビティのため、Casdata はアクティブな caslib です。

```
proc fedsql sessref=casauto;
```

FedSQL ステートメントをサブミットします。 CREATE TABLE ステートメントでは、Supplier、Nation および Customer テーブルからの列を使用して Newtable と

いう名前のテーブルを作成することが指定されます。SELECT ステートメントは、サブクエリを発行して、列 S_NAME、S_ACCTBAL、N_NAME をテーブルから取得し、新しい列 SUM_C_ACCTBAL を作成します。テーブルは、S_NATIONKEY、N_NATIONKEY および C_NATIONKEY 列の値に基づいて結合されます。FedSQL は、データを組み合わせて、その結果を新しい CAS テーブルに返します。

```
create table newtable {options replace=true} as
select
  s_name, s_acctbal, n_name, sum_c_acctbal
from
  supplier,
  nation,
  (select c_nationkey, sum(c_acctbal) as sum_c_acctbal from customer group by
c_nationkey) C
  where
  s_nationkey = n_nationkey and
  s_nationkey = c_nationkey
;
```

テーブル Newtable のコンテンツを表示します。 SELECT ステートメントでは、CAS テーブル Newtable のコンテンツを表示することを指定します。Newtable はインメモリテーブルです。それを保持またはプロモートするには、PROC CASUTIL を使用する必要があります。

```
select * from newtable;
```

プロシジャを停止します。

```
quit;
```

出力: CAS に明示的にロードされたテーブルの結合

アウトプット 23.3　CASUTIL プロシジャの LIST ステートメントの出力

The CASUTIL Procedure

Table Information for Caslib CASDATA

Table Name	Number of Rows	Number of Columns	NLS encoding	Created	Last Modified	Promoted Table	Repeated Table	View	Compressed
SUPPLIER	10000	7	utf-8	08Jul2016:13:57:25	08Jul2016:13:57:25	No	No	No	No
NATION	25	4	utf-8	08Jul2016:13:57:43	08Jul2016:13:57:43	No	No	No	No
CUSTOMER	150000	8	utf-8	08Jul2016:13:58:14	08Jul2016:13:58:14	No	No	No	No

アウトプット 23.4　テーブル Newtable の SELECT 結果の一部

S_NAME	S_ACCTBAL	N_NAME	SUM_C_ACCTBAL
Supplier#000000014	9189.820000	MOROCCO	26625512
Supplier#000000153	850.550000	INDONESIA	27930482
Supplier#000000292	9598.620000	VIETNAM	27081998
Supplier#000000431	9477.340000	CANADA	27025344
Supplier#000000570	922.720000	PERU	26556292
Supplier#000000709	8638.350000	SAUDI ARABIA	26455693
Supplier#000000848	4404.290000	CANADA	27025344
Supplier#000000987	-40.300000	CHINA	26740212
Supplier#000001126	7243.050000	JAPAN	26898469
Supplier#000001265	3986.630000	IRAQ	26919597
Supplier#000001404	5223.720000	ALGERIA	26322970
Supplier#000001543	2107.210000	EGYPT	27100354
Supplier#000001682	9413.980000	CANADA	27025344
Supplier#000001821	7431.290000	VIETNAM	27081998
Supplier#000001960	443.600000	SAUDI ARABIA	26455693
Supplier#000002099	7043.940000	ALGERIA	26322970
Supplier#000002238	5244.930000	JORDAN	26891366
Supplier#000002377	7455.660000	MOZAMBIQUE	27022906
Supplier#000002516	2819.790000	EGYPT	27100354
Supplier#000002655	4856.740000	VIETNAM	27081998
Supplier#000002794	4271.280000	KENYA	27406567
Supplier#000002933	3557.950000	BRAZIL	26821676
Supplier#000003072	7279.730000	INDIA	27293627
Supplier#000003211	8315.050000	UNITED STATES	27316299

例 8: 複数の CAS ライブラリからのテーブルの結合

要素:

PROC FEDSQL ステートメント
SESSREF=プロシジャオプション
FedSQL 言語ステートメント
CAS ステートメント
CASUTIL プロシジャ
CASLIB ステートメント
CASLIB セッションオプション

詳細

この例では、CAS サーバー上で“例 2: 既存のテーブルから新しいテーブルを作成” (815 ページ)の結合を実行する方法を示します。違いは、この例では、Oracle テー

ブルの代わりに Teradata テーブルを使用して SAS データセットと SPD Engine データセットを結合する点です。

CAS セッションを開始します。 この手順は、CAS セッションがまだ存在しない場合、または 2 つ目の CAS セッションを作成する場合にのみ必要です。ここで、新しいセッションが作成されます。Mysess という名前でセッションが作成されます。

```
cas mysess;
```

SAS データセット Customers を CAS セッションにロードします。 PROC CASUTIL は、最初にアクティブな caslib である Casuser にデータセットをロードします。これは、データパスを指定することによって行われます。

```
proc casutil;
 load data="path-to-customers-data-set" outcaslib="casuser";
quit;
```

Teradata caslib を追加します。 CASLIB ステートメントは、caslib TDcaslib を追加することを指定します。

```
caslib TDcaslib desc='Teradata Caslib'
  datasource=(srctype='teradata'
    username='myname'
    password='mypw'
    server='testserver',
    db='test')
    notactive;
```

SPD Engine caslib を追加します。 CASLIB ステートメントは、caslib spdeCaslib を追加することを指定します。caslib は SrcType=SPDE を指定し、SPD Engine データセット製品のメタデータファイルを含むディレクトリへのパスを指定します。spdecaslib は最後に割り当てられた caslib であるため、アクティブな caslib です。

```
caslib spdecaslib Desc="SPD Engine caslib"
datasource=(srctype="spde", username="",
mdfpath="path-to-metafile",
dataTransferMode="serial");
run;
```

(オプション)アクティブな caslib を変更します。 この例は、CAS ステートメントと CASLIB セッションオプションを使用して、アクティブな caslib を設定する方法を示しています。この例では、TDcaslib をアクティブな caslib として設定します。

```
cas mysess sessopts=(caslib=TDcaslib);
```

結合要求をサブミットします。 PROC FEDSQL ステートメントは SESSREF=プロシジャオプションを指定します。SESSREF=オプションは、CAS セッションの Mysess を指定します。テーブル Products は、SPD エンジンのデータセットです。テーブル Customers は SAS データセットです。これらは、*caslib.table-name* の形式で 2 部構成のテーブル名を使用して、CREATE TABLE ステートメントで識別されます。Teradata テーブルであるテーブル Sales は、単に名前で識別されます。出力テーブルの結果がアクティブな caslib に作成されます。出力を保存する場合は、別の caslib にテーブルを作成することをお勧めします。SELECT ステートメントは、テーブル結果のコンテンツの印刷を要求します。

```
proc fedsql sessref=mysess;
create table results as
    select products.prodid, products.product, customers.name,
      sales.totals, sales.country
    from spdecaslib.products, sales, casuser.customers
```

```
    where products.prodid = sales.prodid and
      customers.custid = sales.custid;

select * from results;
quit;
```

出力: 複数の CAS ライブラリからのテーブルの結合

アウトプット 23.5　CAS テーブル結果のコンテンツ

PRODID	PRODUCT	NAME	TOTALS	COUNTRY
3421.000000	Wheat	Qing Ziao	781183	Japan
1424.000000	Corn	Janet Chien	555789	Japan
3975.000000	Barley	Humberto Sertu	899453	Argentina
3421.000000	Wheat	Jim Stewart	2789654	United States
3234.000000	Rice	Peter Frank	189400	United States

24

FMTC2ITM プロシジャ

概要: FMTC2ITM プロシジャ

FMTC2ITM プロシジャの機能

FMTC2ITM プロシジャは、1 つまたは複数のフォーマットカタログを、CAS が使用できる単一のアイテムストアに変換します。PROC FMTC2ITM を使用すると、次の内容のアイテムストアを作成できます。

- 1 つのカタログのすべてのフォーマット
- 1 つのカタログのフォーマットのサブセット
- 複数のカタログのすべてのフォーマット
- 複数のカタログの各々のフォーマットのサブセット

PROC FMTC2ITM を使用してアイテムストアを作成した後、addFmtLib アクションで CAS ステートメントを使用して、アイテムストアを CAS で使用できるようにすることができます。詳細については、“CAS ステートメント” (*SAS Cloud Analytic Services: ユーザーガイド*)および addFmtLib アクションを参照してください。

構文: FMTC2ITM プロシジャ

PROC FMTC2ITM <*options*>;
<SELECT *member-list*>;

ステートメント	タスク
“FMTC2ITM ステートメント”	1 つまたは複数の書式カタログを単一のアイテムストアに変換します。
“SELECT ステートメント”	アイテムストアに配置する書式を一覧表示します。

FMTC2ITM ステートメント

1 つまたは複数の書式カタログを単一のアイテムストアに変換します。

注: アイテムストアは新しいファイルとして書き込まれます。ITEMSTORE オプションを使用して既存のアイテムストアの名前を指定すると、既存のアイテムストアの内容が上書きされます。

構文

FMTC2ITM <*options*>;

必須引数

CATALOG=*memname* | *libname.memname*(| *list*)
アイテムストアに変換されるカタログを指定します。CATALOG オプションを指定しない場合、デフォルトのカタログは Work.Formats です。

CATALOG オプションに対して、次の値を指定できます。

- SAS が WORK ライブラリのカタログ名として解釈する単一レベル名。
- SAS がカタログの libname.memname として解釈する 2 レベル名。
- SAS がカタログ名のリストとして解釈するかっこで囲まれたリスト。

SAS はリスト内の各カタログをリストされた順序で開き、カタログのメンバーをアイテムストアに書き込みます。メンバーの最初の出現のみが書き込まれます。たとえば、CATALOG A にメンバー X と Y があり、CATALOG B にこのコード例で指定されているメンバー X と Z があるとします。

```
proc fmtc2itm catalog=(a b) itemstore='itemstoreA'; run;
```

結果のアイテムストアには、CATALOG A のメンバー X と Y、CATALOG B のメンバー Z が含まれます。

ITEMSTORE=*fileref* | *'filename'*
アイテムストアの名前を指定します。ファイル参照名を指定することも、パス名を引用符で囲んで指定することもできます。

注: PROC FMTC2ITM を呼び出すたびに、項目ストアを 1 つだけ指定できます。

ENCODING=*encoding-name*
カタログまたはリスト内のすべてのカタログのエンコーディングを 指定します。

1 つのカタログのエンコーディングを指定するには、このサンプルコードに示すように、カタログ名の後に ENCODING=オプションを指定します。

```
proc fmtc2itm cat=(abc.fmtlib1/encoding=utf8 abc.fmtlib2);
```

SAS は、UTF8 エンコーディングを Abc.Fmtlib1 カタログにのみ適用します。Abc.Fmtlib2 カタログは、セッションエンコーディングを使用します。

フォーマットカタログのリストのエンコーディングを指定するには、カタログのリストの最後に ENCODING=オプションを指定します。

```
proc fmtc2itm cat=(abc.fmtlib1 abc.fmtlib2) encoding=utf8;
```

SAS は、リスト内のすべてのカタログに UTF8 エンコーディングを適用します。

カタログに ENCODING=オプションを指定しない場合、SAS はセッションエンコーディングオプションをカタログに割り当てます。

PROC FMTC2ITM は、カタログ内のすべての 文字データ(すべてのラベルと 文字範囲値)を検証して、指定されたエンコードに対して有効であることを確認します。いずれかの文字が正常にトランスコードされない場合、SAS はエラーを発行します。

オプション引数

PRINT
書かれた各カタログメンバーに関する情報を表示します。

LOCALE
アイテムストアのメンバー名にロケールに敏感な接頭辞を追加します。

LOCALE オプションを指定すると、メンバー名のかっこ内のリストの処理は、リスト内のメンバー名の通常の処理とは異なります。いずれかのカタログにロケール接尾辞(_xx または_xx_yy の形式)がある場合、カタログのメンバーはその接尾辞を接頭辞として持つアイテムストアに書き込まれます。たとえば、カタログ X_EN_US にメンバー ABC、DEF、および GHI があり、カタログ X_FR_FR にもこのコード例で指定されているメンバー ABC、DEF および JKL が含まれているとします

```
proc fmtc2itm catalog=(x_en_us x_fr_fr) itemstore locale; run;
```

結果のアイテムストアには、メンバー EN_US-ABC、EN_US-DEF、EN_US-GHI、FR_FR-ABC、FR_FR-DEF、および FR_FR-JKL が含まれます。その後の CAS でのアイテムストアの使用により、ABC または DEF が EN_US または FR_FR ロケールに

基づいて適切にロードされます。LOCALE オプションが指定されていない場合、アイテムストアには、X_EN_US のメンバー ABC、DEF、および GHI、および X_FR_FR のメンバー JKL が含まれます。

注: PROC FMTC2ITM を指定すると、SAS セッションでフォーマットカタログが作成されたときに使用されたものと同じエンコーディングを使用する必要があります。たとえば、カタログの作成時に EN_US ロケールが使用された場合は、PROC FMTC2ITM を指定するセッションでも EN_US ロケールを使用する必要があります。

SELECT ステートメント

アイテムストアに配置する書式を一覧表示します。

ヒント: SELECT 文を使用してフォーマットカタログのメンバーを指定しないと、カタログ内のすべてのフォーマットがアイテムストアに 書き込まれます。

構文

SELECT <*メンバーリスト*>;

オプション引数

member-list
アイテムストアに配置する書式の名前を含みます。

詳細

SELECT ステートメントを使用すると、フォーマットライブラリからアイテムストアに追加するフォーマットのみを選択できます。たとえば、CHOICE、SINGLE2、TESTFMTA、WHEN、および WHERE の形式を含む書式ライブラリがある場合は、WHEN および WHERE フォーマットを SELECT ステートメントで指定できます。WHEN および WHERE フォーマットはアイテムストアに追加されますが、CHOICE、SINGLE2、および TESTFMTA フォーマットは追加されません。

例: フォーマットの CAS セッションへの移行

要素:

PROC FMTC2ITM ステートメントオプション
- CATALOG
- ITEMSTORE

CAS ステートメントオプション
- LOADFORMATS
- LISTFORMAT
- SAVEFMTLIB

詳細

この例では、FMTC2ITM プロシジャを使用して、1 つ以上の SAS フォーマットカタログに格納されているユーザー定義フォーマットを CAS セッションのフォーマットライブラリに移行します。

プログラム

```
libname orion "path-to-library";

proc format;
  value $codes
    "A" = "Alpha"
    "B" = "Beta"
    "C" = "Charlie"
    "D" = "Delta";
  value response
    1 = "Yes"
    2 = "No"
    3 = "Undecided"
    4 = "No response";
  value MPGrating
    34  - HIGH = "Excellent"
    24  -< 34  = "Good"
    19  -< 24  = "Fair"
    LOW -< 19  = "Poor";
run;

proc format library=orion.mailfmts;
```

```
  value $officeCodes
    "CHI" = "Chicago"
    "NYC" = "New York"
    "ATL" = "Atlanta"
    "CUP" = "Cupertino";
  value $regionCodes
    "E" = "East"
    "W" = "West"
    "N" = "North"
    "S" = "South";
run;

cas casauto;

proc fmtc2itm
  catalog=(work.formats orion.mailfmts)
  itemstore="path-to-item-store-file";
run;

cas casauto addfmtlib fmtlibname="myfmtlib"
  path=path-to-item-store-file
  replacefmtlib;

cas casauto listformat fmtlibname="myfmtlib"
  members;

cas casauto savefmtlib fmtlibname=myfmtlib
  caslib=casuser table=myfmtlib replace;
```

プログラムの説明

Orion ライブラリを作成し、フォーマットを追加します。

```
libname orion "path-to-library";

proc format;
  value $codes
    "A" = "Alpha"
    "B" = "Beta"
    "C" = "Charlie"
    "D" = "Delta";
  value response
    1 = "Yes"
    2 = "No"
    3 = "Undecided"
    4 = "No response";
  value MPGrating
    34  - HIGH = "Excellent"
    24  -< 34  = "Good"
    19  -< 24  = "Fair"
    LOW -< 19  = "Poor";
run;

proc format library=orion.mailfmts;
  value $officeCodes
    "CHI" = "Chicago"
    "NYC" = "New York"
```

```
    "ATL" = "Atlanta"
    "CUP" = "Cupertino";
  value $regionCodes
    "E" = "East"
    "W" = "West"
    "N" = "North"
    "S" = "South";
run;
```

CAS セッションを開始します。

```
cas casauto;
```

それらのフォーマットを含むアイテムストアを作成します。 FMTC2ITM プロシジャは、フォーマットカタログ Work.Formats および Orion.Mailfmts のフォーマットをアイテムストアファイルに書き込みます。CATALOG オプションは、フォーマットカタログ Work.Formats および Orion.Mailfmts を検索するように指定します。ITEMSTORE オプションは、アイテムストアが書き込まれるパスを指定します。指定されたフォーマットカタログ内の各フォーマットのサブセットを選択するには、FMTC2ITM プロシジャステップで SELECT ステートメントを指定します。

```
proc fmtc2itm
  catalog=(work.formats orion.mailfmts)
  itemstore="path-to-item-store-file";
run;
```

フォーマットをロードする CAS ステートメント ADDFMTLIB オプションは、FMTC2ITM プロシジャで作成したアイテムストアファイルを使用して、フォーマットライブラリ Myfmtlib を追加します。

```
cas casauto addfmtlib fmtlibname="myfmtlib"
  path=path-to-item-store-file
  replacefmtlib;
```

フォーマットを一覧表示します。 CAS ステートメントの LISTFORMAT オプションは、フォーマットライブラリ Myfmtlib のフォーマットを SAS ログに一覧表示して検証できるようにします。

```
cas casauto listformat fmtlibname="myfmtlib"
  members;
```

フォーマットライブラリを保存します。 CAS ステートメント SAVEFMTLIB オプションは、フォーマットライブラリを SASHDAT ファイルに保管します。この手順はオプションです。

繰り返し使用する書式ライブラリでは、caslib に保存するのがベストプラクティスです。caslib からフォーマットライブラリを追加するときは、CAS ステートメントの ADDFMTLIB オプションを、パラメーター CASLIB=および TABLE=と共に使用します。

```
cas casauto savefmtlib fmtlibname=myfmtlib
  caslib=casuser table=myfmtlib replace;
```

25

FONTREG プロシジャ

概要: FONTREG プロシジャ

FONTREG プロシジャの機能

FONTREG プロシジャを使用して、SAS レジストリを更新し、SAS 出力で使用できるシステムフォントを含めることができます。PROC FONTREG は FreeType フォン

トレンダリングを使用して、さまざまな種類のフォント定義を認識し、組み込みます。SAS での組み込みと使用が可能なフォントの種類は、このドキュメントでは一括して FreeType ライブラリのフォントと呼ばれます。

注: システムフォントを SAS レジストリに含めるということは、SAS でフォントファイルの検索場所が認識されるということです。フォントファイルは、フォントが SAS プログラムで呼び出されるまで実際に使用されません。そのため、フォントを SAS レジストリに含めた後は、フォントファイルを移動、削除しないでください。

詳細については、次のソースを参照してください。

- “Specifying Fonts in SAS/GRAPH Programs” (*SAS/GRAPH: Reference*)
- “GDEVICE Procedure” (*SAS/GRAPH: Reference*)
- “FONTSLOC= システムオプション” (*SAS システムオプション: リファレンス*)
- www.freetype.org (FreeType プロジェクトの詳細)

概念: FONTREG プロシジャ

サポート対象のフォントの種類とフォントの命名規則

フォントが SAS レジストリに追加されると、フォント名には山かっこ(< >)で囲まれた 3 文字のタグの接頭辞が付きます。この接頭辞は、フォントの種類を示します。たとえば、TrueType フォントの **Arial** を SAS レジストリに追加すると、レジストリでの名前は**<ttf> Arial** となります。この命名規則を使用して、同じ名前で種類が異なるフォントの追加と区別が可能です。

SAS プログラムでフォントを指定するときは、3 文字のタグを使用して、同じ名前のフォントを区別します。

```
proc report data=sashelp.class nowd
        style(header)=[font_face='<ttf> Palatino Linotype'];
run;
```

SAS プログラムでフォントを指定できる場合の例は、TEMPLATE プロシジャまたは PROC REPORT の STYLE=オプションにあります。

タグをフォント指定に含めない場合、レジストリでその名前のフォントが検索されます。その名前のフォントが複数見つかった場合、次のテーブルのランクが最も高いフォントが使用されます。

表 25.1 サポートされるフォントの種類

ランク	タイプ	タグ	ファイル拡張子
1	TrueType	**<ttf>**	**.ttf**
2	Type1	**<at1>**	**.pfa** **.pfb**
3	OpenType	**<cff>**	**.otf**

注: OpenType フォントは TrueType フォントの拡張で、SAS でサポートされています。OpenType には、serif、sans-serif、monospace、symbol フォントのファミリ値が含まれます。OpenType には.otf フォントファイルが登録されます。

注: PDF と PostScript では、2 バイトの Type1 フォントをサポートしていません。

SAS では、FreeType フォントレンダリングが必要な拡張性のないフォントの種類をサポートしていません。有効なフォントとして認識されても、SAS レジストリに追加されません。

主要なベンダーで生成されないフォントファイルは信頼性がない可能性があり、場合によっては SAS で使用できないことがあります。

次の SAS 出力メソッドとデバイスドライバーで、FreeType フォントレンダリングを使用できます。

- SAS/GRAPH GIF、GIFANIM
- SAS/GRAPH JPEG
- SAS/GRAPH PCL
- SAS/GRAPH PNG
- SAS/GRAPH SASEMF
- SAS/GRAPH SASWMF
- SAS/GRAPH TIFFP、TIFFB
- ユニバーサル EMF
- ユニバーサル PNG
- ユニバーサル印刷 GIF
- ユニバーサル印刷 TIFF
- ユニバーサル印刷 PCL
- ユニバーサル印刷 PDF
- ユニバーサル PS
- ユニバーサル SVG

PROC FONTREG でフォントを登録する

PROC FONTREG は、フォントを SAS レジストリに登録するために使用されます。たとえば、Windows フォントディレクトリに Type1 フォントまたは OpenType フォントがある場合、次のコードをサブミットして、そのフォントを登録できます。

```
proc fontreg mode=add;
fontpath '!SYSTEMROOT\fonts';
run;
```

このコードは、他のすべてのフォントファイルを Windows フォントディレクトリに登録します。

SAS レジストリからフォントを削除する

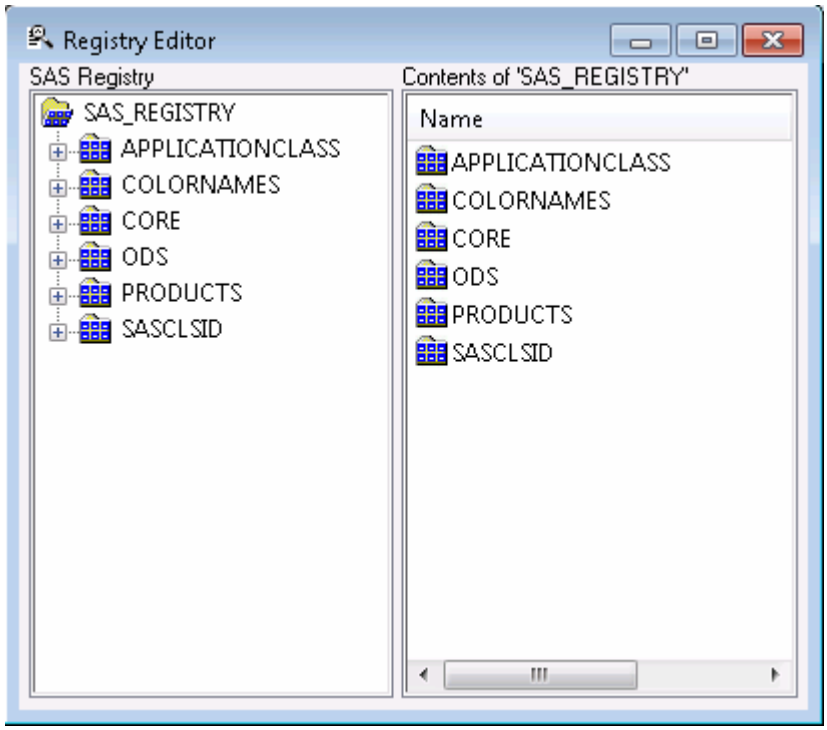
フォントを SAS レジストリから削除するには、次の方法を使用します。

- SAS レジストリエディターを使用する
- PROC REGISTRY を使用する
- REMOVE ステートメントを PROC FONTREG で使用する

SAS レジストリエディターを使用してフォントを削除するには、**ソリューション** ⇨ **アクセサリ** ⇨ **レジストリエディター**を選択します。または、コマンドウィンドウまたは **Command ===>**プロンプトに **regedit** を入力します。

次の画面は、SAS **レジストリエディター**ウィンドウです。

図 25.1　SAS レジストリエディターウィンドウ



レジストリエディターウィンドウの左ペインで、[CORE\PRINTING\FREETYPE\FONTS]キーにナビゲートします。削除するフォントを選択し、次の方法のうちいずれかを使用して削除します。

- フォント名を右クリックして、メニューから**削除**を選択します。
- **削除**ボタン  を選択します。
- **編集** ⇨ **削除** ⇨ **キー**を選択します。

PROC REGISTRY を使用してフォントを削除するには、次の例のようなプログラムをサブミットします。この例では、**<ttf> Arial** フォントを削除します。

```
  /* Write the key name for the font to an external file */
proc registry export='external-filename'
        startat='core\printing\freetype\fonts\<ttf> Arial';
run;

  /* Remove the "<ttf> Arial" font from the SAS registry */
proc registry
uninstall='external-filename' fullstatus;
run;
```

PROC FONTREG で REMOVE ステートメントを使用してフォントを削除するには、“REMOVE ステートメント” (851 ページ)を参照してください。

PROC REGISTRY の詳細については、50 章, “REGISTRY プロシジャ,” (1723 ページ)を参照してください。

PROC FONTREG でファイル参照名を使用する

最初にファイル名を定義している場合は、PROC FONTREG の FONTPATH、TRUETYPE、TYPE1、OPENTYPE の各ステートメントでファイル参照を使用することができます。ファイル参照の使用例を次に示します。

```
filename fonts1 'c:\windows\fonts';
proc fontreg mode=all;
  fontpath fonts1;
run;

proc fontreg mode=all;
  truetype fonts1;
run;
```

ファイル参照を使用できれば、FILENAME ステートメントとその機能を直接使用できます。たとえば、URL を使用して使用可能なフォントを登録できます。ファイル参照のサポートによって、FILENAME ステートメントと PROC FONTREG ステップを使用できます。

フォントの別名とロケール

FONTFILE、FONTPATH、TRUETYPE、OPENTYPE ステートメントでは、別名とロケールがサポートされています。処理中のフォントに現在の SAS セッションと同じロケールのローカライズされた名前が含まれている場合、ローカライズされた名前の別名がフォントファミリを参照するために SAS レジストリに追加されます。

構文: FONTREG プロシジャ

操作: ステートメントが指定されていない場合、PROC FONTREG は FONTSLOC= SAS システムオプションで示されるディレクトリの TrueType フォントファイルを検索します。

ヒント: 2 つ以上のステートメントを指定すると、それらステーメントは出現する順序で実行されます。ただし、REMOVE ステートメントは常に最初に実行されます。単一の PROC FONTREG ステップで同じステートメントを複数回使用できます。

PROC FONTREG <*options*>;

FONTFILE '*file*' <...'*file*'> | '*file-1*, *pfm-file-1*, *afm-file-1*' <...'*file-n*'> | |;

FONTPATH <*fileref*> '*directory*' <...'*directory*'>;

OPENTYPE <*fileref*> '*directory*' <...'*directory*'>

REMOVE '*family-name*' | '*alias*' | *family-type* | _ALL_;

TRUETYPE <*fileref*> '*directory*' <...'*directory*'>;

TYPE1 <*fileref*> '*directory*' <...'*directory*'>;

ステートメント	タスク
“PROC FONTREG ステートメント”	新しいフォントと既存のフォントの処理方法を指定します
“FONTFILE ステートメント”	処理するフォントファイルを識別します
“FONTPATH ステートメント”	処理する有効なフォントファイルを識別するディレクトリを検索します
“REMOVE ステートメント”	SAS レジストリの場所 Core\Printing\Freetype\Fonts からフォントファミリ、特定の種類のすべてのフォント、またはすべてのフォントを削除します
“TRUETYPE ステートメント”	TrueType フォントファイルを識別するディレクトリを検索します
“TYPE1 ステートメント”	有効な Type 1 フォントファイルを識別するディレクトリを検索します
“OPENTYPE ステートメント”	有効な OpenType フォントファイルを識別するディレクトリを検索します

PROC FONTREG ステートメント

SAS レジストリを更新し、SAS 出力で使用できるシステムフォントを含めることができます。

構文

PROC FONTREG <*options*>;

オプション引数の要約

MODE=ADD | REPLACE | ALL
: 新しいフォントと既存のフォントの処理方法を指定します。

MSGLEVEL=VERBOSE | NORMAL | TERSE | NONE
: SAS ログに含める詳細のレベルを指定します。

NOUPDATE
: プロシジャが SAS レジストリを更新せずに実行するように指定します。

USESASHELP
: Sashelp ライブラリの SAS レジストリが更新されるように指定します。

オプション引数

MODE=ADD | REPLACE | ALL
SAS レジストリの新しいフォントと既存のフォントの処理方法を指定します。

ADD
SAS レジストリにまだ存在していないフォントを追加するように指定します。既存のフォントは変更しません。

REPLACE
SAS レジストリにすでに存在するフォントを置き換えるように指定します。新しいフォントは追加しません。

ALL
SAS レジストリに存在していない新しいフォントを追加し、SAS レジストリにすでに存在しているフォントを置き換えるように指定します。

デフォルト ADD

例 “例 3: ディレクトリの既存の TrueType フォントファイルの置換” (857 ページ)

MSGLEVEL=VERBOSE | NORMAL | TERSE | NONE
SAS ログに含める詳細のレベルを指定します。

レベルを次に示します。

VERBOSE
SAS ログメッセージには、追加されたフォント、追加されなかったフォント、認識されなかったフォントが含まれます。ログには、追加されたフォント、追加されなかったフォント、認識されなかったフォントの数を示す概要も含まれています。

NORMAL
SAS ログメッセージには追加されたフォント、および追加されたフォント、追加されなかったフォント、認識されなかったフォントの数を示す概要が含まれます。

TERSE
SAS ログメッセージには、追加されたフォント、追加されなかったフォント、認識されなかったフォントの数を示す概要のみが含まれます。

NONE
エラー以外(発生した場合)のメッセージは SAS ログに書き込まれません。

デフォルト TERSE

例 “例 2: 複数のディレクトリに含まれるすべてのフォントファイルの追加” (855 ページ)

NOUPDATE
プロシジャが実際に SAS レジストリを更新せずに実行するように指定します。このオプションを使用して、SAS レジストリを変更する前に指定フォントでプロシジャをテストできます。

USESASHELP
Sashelp ライブラリの SAS レジストリが更新されるように指定します。このオプションを使用するには、Sashelp ライブラリへの書き込みアクセス権が必要です。USESASHELP オプションが指定されない場合、Sasuser ライブラリの SAS レジストリが更新されます。

FONTFILE ステートメント

処理するフォントファイルを指定します。

参照項目: “例 1: 単一フォントファイルの追加” (854 ページ)

構文

FONTFILE '*file*' <...'*file*'> | '*file-1*, *pfm-file-1*, *afm-file-1*' <...'*file-n*'>;

必須引数

file
フォントファイルへの完全パス名です。ファイルは、有効なフォントファイルとして認識されると処理されます。パス名はそれぞれ引用符で囲む必要があります。複数のパス名を指定する場合、パス名をスペースで区切る必要があります。

pfm-file
フォントメトリックおよび Windows フォント名の値を含む Windows 固有のファイルを指定します。

afm-file
フォントメトリックを含むファイルを指定します。

詳細

Type1 フォントを処理する

有効な Type1 フォントが TYPE1 または FONTPATH ステートメントで処理される場合、フォントファイルを含む同じディレクトリの対応する PFM または AFM フォントメトリックファイルの検索が試行されます。ファントファイル名の接頭辞が.PFM 拡張子と.AFM 拡張子と使用されて、メトリックファイル名が生成されます。これらのファイルは正常に開かれ、有効なメトリックファイルであることが決定されると、SAS レジストリへの追加時にフォントファミリのフォントと関連付けられます。

FONTFILE ステートメントで Type1 フォントを指定して、PFM ファイルまたは AFM ファイルを指定しない場合、PFM ファイルまたは AFM ファイルは検索されません。

PFM ファイルまたは AFM ファイルを指定する

フォントファイルに Type1 フォントが含まれている場合、その対応する PFM ファイルと AFM ファイルも指定できます。この例のように、ファイルに対してそれぞれ完全ホスト名(ディレクトリとファイル名)を指定する必要があり、すべてのファイルはグループ化して、引用符で囲む必要があります。

```
fontfile 'c:\winnt\fonts\alpinerg.pfb,
      c:\winnt\fonts\alpinerg.pfm,
      c:\winnt\fonts\alpinerg.afm';
```

AFM ファイルを指定して PFM ファイルを指定しない場合、この例のように、欠損している PFM ファイルに対するプレースホルダーとしてカンマを使用する必要があります。

```
fontfile 'c:\winnt\fonts\alpinerg.pfb, ,
c:\winnt\fonts\alpinerg.afm';
```

PFM ファイルを指定して AFM ファイルを指定しない場合、この例のように、欠損している AFM ファイルに対するプレースホルダーとしてのカンマは不要です。

```
fontfile 'c:\winnt\fonts\alpinerg.pfb,
c:\winnt\fonts\alpinerg.pfm';
```

PFM ファイルまたは AFM ファイルを指定すると、SAS ではファイルを開き、ファイルが指定されている種類のものかどうかを判別しようとします。指定されている種類のものでない場合、メッセージがログに書き込まれ、ファイルは使用されません。

PFM ファイルは Windows 固有のファイルで、フォントメトリックおよび Windows フォント名フィールドの値が含まれます。有効な PFM ファイルを指定すると、ファイルが開かれ、Windows フォント名の値が取得され、それがフォントとともに SAS レジストリに保存されます。このフィールドはファイル(EMF フォーマットファイルなど)の作成時に使用され、Windows アプリケーションにエクスポートされます。

PFM ファイルまたは AFM ファイルを指定しない

FONTFILE ステートメントで PFM ファイルまたは AFM ファイルを Type1 フォントファイルとともに指定する必要はありません。この場合、メトリックファイル情報は SAS レジストリのフォントファミリのフォントに追加されません。複数のスタイルと重みを含む既存のフォントファミリが SAS レジストリにすでに存在し、FONTFILE ステートメントがそのファミリのフォントのいずれかの置換に使用される場合、そのフォントに関するすべての情報が更新されます。置換によりホストファイル名、PFM 名、AFM 名、Windows フォント名も更新されます。

注: ファミリのフォントを置換し、そのフォントに PFM 名または AFM 名の値が含まれている場合、FONTFILE ステートメントでメトリックに対し欠損値または無効値を指定すると、対応するメトリック値がレジストリのフォントから削除されます。

注: TrueType フォントを指定した場合、PFM ファイルまたは AFM ファイルの指定を使用できません。

FONTPATH ステートメント

処理対象の有効なフォントファイルを検索するディレクトリを指定します。

参照項目: “例 2: 複数のディレクトリに含まれるすべてのフォントファイルの追加” (855 ページ)

構文

FONTPATH <*fileref*> '*directory*' <...'*directory*'>;

必須引数

directory
検索するディレクトリを指定します。有効なフォントファイルとして認識されるすべてのファイルが処理されます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

動作環境の情報: Windows 動作環境でのみ、フォルダーの場所が不明な場合は fonts フォルダーを検索します。また、フォントの場所がわからなくても、システムフォントを登録できます。この情報を見つけるには、次のプログラムをサブミットします。

```
proc fontreg;
   fontpath "%sysget(systemroot)\fonts";
run;
```

%SYSGET マクロはウィンドウ環境変数 SYSTEMROOT の値を取得し、システムディレクトリの場所に解決します。fonts サブディレクトリは、システムディレクトリの 1 レベル下に置かれます。

オプション引数

fileref
FONTPATH ステートメントで使用するファイル参照を指定します。

REMOVE ステートメント

フォントファミリ、特定の種類(TrueType、Type1 など)のすべてのフォント、またはすべてのフォントを SAS レジストリの場所 Core\Printing\Freetype\Fonts から削除します。

構文

REMOVE '*family-name*' | '*alias*' | *family-type* | _ALL_;

必須引数

family-name
SAS レジストリの Core\Printing\Freetype\Fonts キーから削除するフォントのファミリ名を指定します。値にスペースが含まれている場合は、*family-name* を引用符で囲みます。

alias
family-name に対する代替名(通常は短縮形式)を指定します。値にスペースが含まれている場合は、別名を引用符で囲みます。

注 別名として指定可能な有効値は、SAS レジストリの Core\Printing\Alias\Fonts\Freetype キーにリストされます。

family-type
SAS でサポートされ、SAS レジストリから削除するフォントの種類(TrueType、Type1 など)の名前を指定します。

注: フォントの種類は存在する動作システムの場所からは削除されません。SAS レジストリからのフォントの種類の登録は、SAS でそのフォントが認識されないように削除されます。

ALL
SAS レジストリの Core\Printing\Freetype\Fonts キーのすべてのフォントファミリが削除されるように指定します。

詳細

レジストリからフォントを削除する

REMOVE ステートメントは、SAS レジストリの場所 **Core\Printing\Freetype\Fonts** からフォントファミリ、特定の種類のすべてのフォント、またはすべてのフォントを削除します。USESASHELP プロシジャオプションを指定すると、フォントはレジストリの Sashelp 部分から削除されます。USESASHELP を指定しない場合、フォントはレジストリの Sasuser 部分から削除されます。デフォルトでは、レジストリの Sasuser 部分から削除されます。

REMOVE ステートメントで *family-name* 引数を指定すると、ファミリ内の個別のフォントではなく、フォントファミリが削除されます。たとえば、**Arial** ファミリ内で複数のフォントを登録するとします。REMOVE Arial;ステートメントを使用すると、**Arial** ファミリのすべてのフォントがレジストリから削除されます。同様に、*family-type* 引数を指定し、REMOVE Type1;ステートメントを使用すると、Type1 フォントファミリがすべてレジストリから削除されます。

フォントの追加順序または削除順序

フォントは、その他のプロシジャステートメントを使用してレジストリで追加または置換される前に、SAS レジストリから削除されます。REMOVE ステートメントは、ステートメントが処理されるとすぐにフォントファミリをレジストリから削除します。FONTFILE、FONTPATH、TRUETYPE、TYPE1 などの他のフォントステートメントは、受信した順序で処理されます。フォント情報は、すべてのステートメントが処理されるまで保存されます。次に、レジストリが更新されます。

REMOVE ステートメントで指定されるフォントを検索する

REMOVE ステートメントで指定する名前が存在しない場合、フォントタグ接頭辞(<ttf>など)が指定された名前に追加され、それが SAS レジストリに存在するかどうかが決定されます。たとえば、**Arial** を指定すると、SAS は<ttf>接頭辞タグを使用し、最初に TrueType フォントの種類を検索し、レジストリから削除できるようにします。検索が失敗した場合、<at1>接頭辞タグが使用され、レジストリから削除できるように Type1 フォントの種類が検索されます。

フォントファミリを削除できない場合

ALL、*family-type* または *family-name* 引数の情報を処理後、フォントファミリを削除できない場合、SAS レジストリの Core\Printing\Alias\Fonts\Freetype キーが検索され、指定された値が別名かどうかが決定されます。指定された値がこのキーに別名として存在する場合、その別名に対応するフォントファミリが削除され、その別名も削除されます。たとえば、Test の別名が **Arial** フォントファミリを参照し、ユーザーが REMOVE test;ステートメントを PROC FONTREG で指定すると、SAS は、Test が **Arial** の別名であると判断します。**Arial** フォントファミリが SAS レジストリの **Core\Printing\Freetype\Fonts** キーから、Test 別名が SAS レジストリの **Core\Printing\Alias\Fonts\Freetype** キーからそれぞれ削除されます。

この時点でフォントファミリを削除できない場合、REMOVE ステートメントで指定されている値が無効であることを示すメッセージがログに書き込まれます。

TRUETYPE ステートメント

TrueType フォントファイルを検索するディレクトリを指定します。

参照項目: “例 3: ディレクトリの既存の TrueType フォントファイルの置換” (857 ページ)

構文

TRUETYPE *<fileref>* '*directory*' <...'*directory*'>;

必須引数

directory
検索するディレクトリを指定します。有効な TrueType フォントファイルとして認識されるファイルのみが処理されます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

オプション引数

fileref
TRUETYPE ステートメントで使用するファイル参照を指定します。

TYPE1 ステートメント

有効な Type1 フォントファイルを検索するディレクトリを指定します。

構文

TYPE1 *<fileref>* '*directory*' <...'*directory*'>;

必須引数

directory
検索するディレクトリを指定します。有効な Type1 フォントファイルとして認識されるファイルのみ処理されます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

オプション引数

fileref
TYPE1 ステートメントで使用するファイル参照を指定します。

OPENTYPE ステートメント

有効な OpenType フォントファイルを検索するディレクトリを指定します。

構文

OPENTYPE <*fileref*> '*directory*' <...'*directory*'>;

必須引数

directory
検索するディレクトリを指定します。有効な OpenType フォントファイルとして認識されるファイルのみ処理されます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

オプション引数

fileref
OPENTYPE ステートメントで使用するファイル参照を指定します。

例: FONTREG プロシジャ

例 1: 単一フォントファイルの追加

要素: FONTFILE statement

詳細

この例では、単一フォントファイルを SAS レジストリに追加する方法を示します。FONTFILE ステートメントは、単一フォントファイルへの完全パスを指定します。

プログラム

```
proc fontreg;
  fontfile '<ttf> Arial';
run;
```

ログ

例のコード 25.1　単一フォントファイルの SAS レジストリへの追加

```
SUMMARY:
  Files processed: 1
  Unusable files: 0
  Files identified as fonts: 1
  Fonts that were processed: 1
  Fonts replaced in the SAS registry: 0
  Fonts added to the SAS registry: 1
  Fonts that could not be used: 0
  Font Families removed from SAS registry: 0
```

例 2: 複数のディレクトリに含まれるすべてのフォントファイルの追加

要素:
MSGLEVEL=オプション
FONTPATH ステートメント

詳細

この例では、すべての有効なフォントファイルを 2 つの異なるディレクトリから追加する方法と、詳細情報を SAS ログに書き込む方法を示します。

プログラム

```
proc fontreg msglevel=verbose;

fontpath 'your-font-directory-1' 'your-font-directory-2';
run;
```

プログラムの説明

すべての詳細を SAS ログに書き込みます。 MSGLEVEL=VERBOSE オプションは、追加されたフォント、追加されなかったフォント、および認識されなかったフォントファイルに関するすべての詳細を書き込みます。

```
proc fontreg msglevel=verbose;
```

有効なフォントを検索するディレクトリを指定します。 FONTPATH ステートメントでは複数のディレクトリを指定できます。ディレクトリはそれぞれ引用符で囲む必要があります。複数のディレクトリを指定する場合、ディレクトリをスペースで区切る必要があります。

```
fontpath 'your-font-directory-1' 'your-font-directory-2';
run;
```

ログ

例のコード 25.2　複数のディレクトリに含まれるすべてのフォントファイルの追加機能からのメッセージ

```
1   proc fontreg msglevel=verbose;
2   fontpath 'your-font-directory-1'
3            'your-font-directory-2';
4   run;

ERROR: FreeType base module FT_New_Face -- unknown file format.
ERROR: A problem was encountered with file
"your-font-directory-2\MODERN.FON".

...  more log entries ...

WARNING: The "Sasfont" font in file
       "your-font-directory-2\SAS1252.FON" is non-scalable.    Only scalable fonts are supported.

...  ...more log entries .....

NOTE: The font "Albertus Medium" (Style: Regular, Weight: Normal) has been
     added to the SAS Registry at
     [CORE\PRINTING\FREETYPE\FONTS\<ttf>Albertus Medium].Because it is a
     TRUETYPE font, it can be referenced as "Albertus Medium" or
     "<ttf>Albertus Medium" in SAS.The font resides in file
     "your-font-directory-1\albr55w.ttf".

...  more log entries ...

WARNING: The font "Georgia" (Style: Regular, Weight: Normal) will not be added
because it already exists in the "<ttf>Georgia" font family of the SAS Registry.

...  more log entries ...

SUMMARY:
  Files processed: 138
  Unusable files: 3
  Files identified as fonts: 135
  Fonts that were processed: 135
  Fonts replaced in the SAS registry: 0
  Fonts added to the SAS registry: 91
  Fonts that could not be used: 44
  Font Families removed from SAS registry: 0

NOTE: PROCEDURE FONTREG used (Total process time):
     real time          27.81 seconds
     cpu time           1.18 seconds
```

例 3: ディレクトリの既存の TrueType フォントファイルの置換

要素:　MODE=オプション
TRUETYPE ステートメント

詳細

この例では、指定されているディレクトリのすべての TrueType フォントを読み込み、SAS レジストリにすでに存在するものを置き換えます。

プログラム

```
proc fontreg mode=replace;
  truetype 'your-font-directory';
run;
```

プログラムの説明

既存するフォントのみ置き換えます。 MODE=REPLACE オプションは、プロシジャのアクションを SAS レジストリですでに定義されているフォントの置換に制限します。新しいフォントは追加されません。

```
proc fontreg mode=replace;
```

TrueType フォントファイルを含むディレクトリを指定します。 TrueType フォントファイルとして認識されないディレクトリのファイルは無視されます。

```
  truetype 'your-font-directory';
run;
```

ログ

例のコード 25.3　ディレクトリの既存の TrueType フォントファイルの置換

```
SUMMARY:
  Files processed: 49
  Unusable files: 3
  Files identified as fonts: 46
  Fonts that were processed: 40
  Fonts replaced in the SAS registry: 40
  Fonts added to the SAS registry: 0
  Fonts that could not be used: 0
  Font Families removed from SAS registry: 0
```

26

FORMAT プロシジャ

概要: FORMAT プロシジャ

FORMAT プロシジャの機能

FORMAT プロシジャを使用して、変数に対し独自の入力形式と出力形式を定義できます。さらに、次のアクションを実行できます。

- 入力形式または出力形式を含むカタログの一部を出力する
- 入力形式または出力形式の説明を SAS データセットに保存する
- SAS データセットを使用して入力形式または出力形式を作成する

SAS Viya プラットフォームでのフォーマットのエンコーディング

SAS Viya プラットフォームでは、UTF-8 エンコーディングのみがサポートされています。

フォーマットをライブラリに保存する場合、SAS はフォーマットが作成されたセッションエンコーディングを使用します。元のエンコーディングが UTF-8 でない場合、フォーマットライブラリを、文字を表すためにより多くのバイトを必要とするエンコーディングに変換すると、文字が切り捨てられる可能性があります。

注: SAS および CAS の以前のバージョン間でのフォーマットライブラリの移動には、ある程度のリスクが伴う可能性があります。SAS では、このリスクを軽減するために CNTLOUT データセットを使用することをお勧めします。

SAS Viya プラットフォームでのフォーマットライブラリの詳細については、*SAS Viya プラットフォームでの UTF-8 へのデータの移行*および *UTF-8 データの処理における SAS Viya FAQ* を参照してください。

入力形式と出力形式について

入力形式によって、生データ値の読み込み方法と保存方法が決まります。出力形式によって、変数値の印刷方法が決まります。簡単にするため、このセクションでは入力形式変換、出力形式印刷という用語を使用します。

入力形式と出力形式によって、データの種類(文字または数値)と形式(占めるバイト数、数字の小数点配置、先頭、末尾、埋め込みのブランクやゼロの処理方法など)が認識されます。SAS では、変数の読み込みと書き込みを行うための入力形式と出力形式を提供しています。SAS が提供する入力形式と出力形式の詳細な説明については、*SAS 出力形式と入力形式: リファレンス*を参照してください。

入力形式を使用して、次のことができます。

- 文字列を数字に変換(1 を **YES** に変換するなど)。
- 文字列を異なる文字列に変換(**'YES'**を**'OUI'**に変換するなど)。
- 文字列を数字に変換(**YES** を 1 に変換するなど)。
- 数字を別の数字に変換(0–9 を 1 に、10–100 を 2 に変換するなど)。

注: ユーザー定義の入力形式は文字データ読み込み専用です。文字値を実数値に変換できますが、実数を文字に変換することはできません。

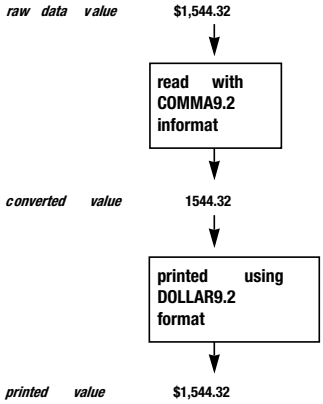
出力形式を使用して、次のことができます。

- 数値を文字値として印刷(1 を **MALE** として、2 を **FEMALE** として変換するなど)。
- 文字列を異なる文字列として印刷(**YES** を **OUI** として変換するなど)。
- テンプレートを使用して数値を印刷(9458763450 を **945-876-3450** として印刷するなど)。

変数への出力形式と入力形式の関連付け法

次の図に、入力形式および出力形式と変数を関連付けた場合を要約します。COMMA*w.d* 入力形式と DOLLAR*w.d* 出力形式は、SAS によって提供されています。

図 26.1　変数への入力形式と出力形式の関連付け



図では、ドル記号とカンマを含む生データ値が読み込まれます。COMMA9.2 入力形式はドル記号とカンマを無視して、値を 1544.32 に変換します。DOLLAR9.2 出力形式は、ドル記号とカンマを含めて値を印刷します。変数への入力形式と出力形式の関連付けの詳細については、“変数に入力形式と出力形式を関連付ける ” (862 ページ)を参照してください。

概念: FORMAT プロシジャ

変数に入力形式と出力形式を関連付ける

変数への入力形式と出力形式の関連付け法

次のテーブルに、変数に入力形式と出力形式を関連付けるためのさまざまな方法を要約します。

表 26.1 変数に入力形式と出力形式を関連付ける

ステップ	入力形式	出力形式
DATA ステップ内	ATTRIB ステートメントまたは INFORMAT ステートメントを使用して、入力形式と変数を永久に関連付けます。INPUT 関数または INPUT ステートメントを使用して、DATA ステップの期間に対してのみ入力形式と変数を関連付けます。	ATTRIB ステートメントまたは FORMAT ステートメントを使用して、出力形式と変数を永久に関連付けます。PUT 関数または PUT ステートメントを使用して、DATA ステップの期間に対してのみ出力形式と変数を関連付けます。
PROC ステップ内	ATTRIB ステートメントと INFORMAT ステートメントは、Base SAS プロシジャで有効です。ただし Base SAS ソフトウェアでは、データがすでに SAS 変数に読み込まれているため、通常は PROC ステップで入力形式を割り当てません。	ATTRIB ステートメントまたは FORMAT ステートメントを使用して、出力形式と変数を関連付けます。出力データセットを作成するプロシジャでいずれかのステートメントを使用する場合、出力形式は入力データセットの変数と永久に関連付けられます。出力データセットを作成しない、または既存データセットを変更しないプロシジャでいずれかのステートメントを使用する場合、ステートメントは PROC ステップの期間に対してのみ出力形式と変数を関連付けます。

FORMAT ステートメントと PROC FORMAT の違い

FORMAT ステートメントと FORMAT プロシジャを混同しないようにしてください。FORMAT ステートメントと INFORMAT ステートメントは既存する出力形式と入力形式(標準 SAS またはユーザー定義)を 1 つ以上の変数と関連付けます。PROC FORMAT は、ユーザー定義の出力形式または入力形式を作成します。

変数への出力形式と入力形式の割り当て

変数への独自の出力形式または入力形式の割り当ては、次の 2 ステップのプロセスからなります。

1 出力形式または入力形式を FORMAT プロシジャを使用して作成

2 出力形式または入力形式を ATTRIB ステートメント、FORMAT ステートメント、INFORMAT ステートメント、または INPUT 関数、PUT 関数を使用して割り当て

ATTRIB、INFORMAT、FORMAT ステートメントに関する完全ドキュメントについては、*SAS DATA ステップステートメント: リファレンス*を参照してください。INPUT および PUT 関数に関する完全なドキュメントについては、*SAS 関数と CALL ルーチ*

*ン: リファレンス*を参照してください。Base SAS プロシジャでの出力形式の使用に関する詳細および例については、"プロシジャの概念"を参照してください。

入力形式と出力形式の保存

フォーマットカタログ

PROC FORMAT は、ユーザー定義の入力形式と出力形式を SAS カタログのエントリとして保存します。[1] カタログを指定するには、PROC FORMAT ステートメントで LIBRARY=オプションを使用します。LIBRARY=オプションを省略すると、出力形式と入力形式は Work.Formats カタログに保存されます。LIBRARY=*libref* を指定し、カタログ名を指定しない場合、出力形式と入力形式は *libref*.FORMATS カタログに保存されます。この 1 レベル名の使用は、SAS 以外での 1 レベル名の使用とは異なります。LIBRARY=オプションでは 1 ラベル名はライブラリを示し、SAS 以外では 1 ラベル名は WORK ライブラリのファイルを示します。

カタログエントリの名前は、出力形式または入力形式の名前です。エントリの種類は、次のとおりです。

- FORMAT (数値出力形式)
- FORMATC (文字出力形式)
- INFMT (数値入力形式)
- INFMTC (文字入力形式)

一時入力形式と一時出力式

入力形式と出力形式は、WORK ライブラリのカタログに保存されているときは一時のものです。LIBRARY=オプションを省略すると、PROC FORMAT は入力形式と出力形式を一時カタログ Work.Formats に保存します。一時入力形式と一時出力形式は、それらが作成されたのと同じ SAS セッションまたはジョブでのみ取得できます。一時出力形式と一時入力形式を取得するには、適切な SAS ステートメントに出力形式または入力形式の名前を含めるだけです。SAS では自動的に Work.Formats カタログの出力形式または入力形式が検索されます。

1. カタログとは、SAS ファイルの一種であり、SAS ライブラリ内に存在します。この種の SAS ファイルまたは SAS ライブラリ構造に不慣れな場合は、*SAS プログラマガイド: エッセンシャル*の SAS ファイルに関するセクションを参照してください。

永久入力形式と永久出力形式

ある SAS ジョブまたはセッションで作成された出力形式または入力形式を後続のジョブまたはセッションで使用する場合、その出力形式または入力形式を永久に SAS カタログに保存する必要があります。

PROC FORMAT ステートメントの LIBRARY=オプションを使用して、入力形式と出力形式を永久に保存します。PROC FORMAT ステートメント (869 ページ)の LIBRARY=オプションの説明を参照してください。

永久入力形式と永久出力形式へのアクセス

入力形式または出力形式を永久に保存した後、それを後の SAS セッションまたはジョブで使用できます。後の SAS セッションまたはジョブで永久入力形式または永久出力形式と変数を関連付ける場合、SAS でその入力形式と出力形式にアクセスできる必要があります。そのため、LIBNAME ステートメントを使用して、その入力形式または出力形式を保存するカタログを保存するライブラリにライブラリ参照名を割り当てる必要があります。

SAS では、ユーザー定義の出力形式と入力形式の検索時に 2 つのうち 1 つの方法が使用されます。

- デフォルトで、FORMATS カタログに対し Library ライブラリ参照名によって参照されるライブラリが常に検索されます。フォーマットカタログが 1 つしかない場合、次を行います。
  1. PROC FORMAT ステップを実行している SAS セッションで Library ライブラリ参照名を SAS ライブラリに割り当てます。
  2. PROC FORMAT ステートメントでオプション library=library を指定します。PROC FORMAT は、そのステップで定義される入力形式と出力形式を Library.Formats カタログに保存します。
  3. ユーザー定義の出力形式と入力形式を使用する SAS プログラムでは、LIBNAME ステートメントを含め、Library ライブラリ参照名を永久フォーマットカタログを含むライブラリに割り当てます。
- 複数のフォーマットカタログがある場合、またはフォーマットカタログの名前が Formats 以外の場合、次を行います。
  1. PROC FORMAT ステップを実行している SAS セッションで SAS ライブラリにライブラリ参照名を割り当てます。
  2. PROC FORMAT ステップで、オプション library=*libref* または library=*libref.catalog* を指定します。*libref* は、ステップ 1 で割り当てたライブラリ参照名です。
  3. ユーザー定義の出力形式と入力形式を使用する SAS プログラムで、FMTSEARCH=オプションを OPTIONS ステートメントで使用し、*libref* または *libref.catalog* をフォーマットカタログリストに含めます。

検索対象となるフォーマットカタログリストを指定するための構文は、次のとおりです。

OPTIONS FMTSEARCH=(*catalog-specification-1<catalog-specification-2 ...>*);

catalog-specification にはそれぞれ *libref* または *libref.catalog* が可能です。*libref* だけが指定される場合、カタログ名が Formats とみなされます。

出力形式または入力形式の検索時は、最初に Work.Formats、次に Library.Formats で検索が行われます。これは、これらのうちいずれも FMTSEARCH=リストに表示されない場合です。SAS は、出力形式または入力形式が見つかるまで FMTSEARCH=リストのカタログをリストされている順序で検索します。

詳細については、*"FMTSEARCH = システムオプション" (SAS システムオプション: リファレンス)*を参照してください。LIBRARY=オプションと FMTSEARCH=オプションを一緒に使用する例については、"例 17: 文字列の範囲指定" (971 ページ)を参照してください。

入力形式と出力形式の欠損

SAS で見つからなかった入力形式または出力形式を参照する場合、エラーメッセージが表示され、SAS システムオプション NOFMTERR が有効でない限り処理は停止します。NOFMTERR が有効な場合、SAS では *w.* または *$w.* デフォルト出力形式を使用して、見つからない出力形式で変数の値を印刷します。たとえば、NOFMTERR を使用するには、次の OPTIONS ステートメントを使用します。

```
options nofmterr;
```

詳細については、*"FMTERR システムオプション" (SAS システムオプション: リファレンス)*を参照してください。

SAS でユーザー定義出力形式を使用してフォーマットする欠損変数が見つかり、MISSING=システムオプションによって欠損値に対して印刷する文字を定義する場合、欠損値は次のように決定されます。

- ユーザー定義の出力形式または入力形式に欠損値に対する value-range-set がある場合、欠損値はユーザー定義の出力形式によって定義されます。
- ユーザー定義の出力形式に欠損値に対する value-range-set が定義されていない場合、欠損値は MISSING=システムオプションによって定義されます。MISSING=システムオプションのデフォルト値は、(ピリオド)です。

入力形式と出力形式の印刷

FMTLIB オプション使用時に提供される出力は、入力形式値と出力形式値の簡単なビューを表すためのものです。

FMTLIB オプションを使用する代わりに、CNTLOUT=オプションを使用して、入力形式と出力形式に関する情報を保存する出力データセットを作成します。次に PROC PRINT または PROC REPORT を使用して、データセットを印刷します。この場合、ラベルは切り捨てられません。

注: データセットオプションを使用して、参照を保持するか、CNTLOUT=オプションを使用して追加された追加変数にドロップすることができます。

CAS セッションでの出力形式の使用

PROC FORMAT では、SAS クライアントセッションでのカタログのフォーマットライブラリの作成と、フォーマットライブラリの SAS Cloud Analytic Services (CAS)セッションへのロードがサポートされています。SESSREF システムオプションを使用して CAS セッションを指定した場合、PROC FORMAT では、フォーマットライブラリが指定した CAS セッションにロードされます。それ以外の場合は、PROC FORMAT では、SAS クライアントセッションが実行されているフォーマットライブラリが保存されます。詳細については、“CASFMTLIB='*name*'” (870 ページ)を参照してください。

SAS クライアントセッションと CAS のセッションは、SESSREF システムオプションまたは CASLIB ステートメントの SESSREF 引数を使用して確立したセッション参照を介して対話することができます。CAS での処理を利用できる SAS 言語要素を使用すると、セッション参照で、その処理が発生する場所が識別されます。セッション参照を指定しない場合、処理はクライアントセッションで発生します。CAS で言語要素がサポートされていない場合、処理はクライアントセッションで発生します。

SESSREF の詳細については、次を参照してください。

- “SESSREF= システムオプション” (*SAS システムオプション: リファレンス*)
- “CASLIB ステートメント” (*SAS Cloud Analytic Services: ユーザーガイド*)

CAS セッションでの出力形式の使用については、次を参照してください。

- “例 1: CAS セッションでのフォーマットライブラリの作成” (927 ページ)
- “SAS Cloud Analytic Services でのユーザー定義出力形式の使用” (*SAS Cloud Analytic Services: ユーザー定義フォーマット*)

バイナリ検索による値に対するユーザー定義出力形式または入力形式の決定

PROC FORMAT は、バイナリ検索を使用して、値に使用するユーザー定義出力形式または入力形式を決定します。それに対し、IF-THEN/ELSE ステートメントの使用は、基本的に値の順次検索になります。バイナリ検索を使用する検索方法であるため、PROC FORMAT を使用すると、多数の比較を効率的に行うことができます。

次に示すのは、PROC FORMAT を使用して作成できるいくつかのユーザー定義出力形式です。

```
1='Yes'
2='No'
3='Possibly'
```

これらの IF-THEN / ELSE ステートメントを使用すると、SAS は範囲内の最初の値、x=1、で検索を開始し 、一致する値が見つかるまで値を順に実行します。このタイプの検索は、必要な値が値の範囲の先頭に近い場合に、より効率的になります。

```
if x=1 then label='Yes';
else if x=2 then label='No';
else if x=3 then label='Possibly';
```

値 3 を検索している場合、一致が見つかる前に 3 回の比較が行われます。値 1 を検索している場合は、比較が 1 回だけ行われます。

PROC FORMAT の VALUE ステートメントを使用すると、SAS はその範囲の中間値で検索を開始します。この例では、SAS は値を中間範囲の値、2='No'、と比較することから始まります。次に、値をより高い範囲の値、3="Possibly"、と比較し、次に、値をより低い範囲の値、1="Yes" 、と比較します。

```
value
  1="Yes"
  2="No"
  3="Possibly";
```

PROC FORMAT が使うようなバイナリ検索では、範囲に検索対象の値が多数あると効率が高まります。

構文: FORMAT プロシジャ

制限事項:

SELECT ステートメントと EXCLUDE ステートメントを同じ PROC FORMAT ステップ内で使用することはできません。

CASFMTLIB オプションを指定すると、SELECT および EXCLUDE ステートメントでは、SAS Cloud Analytics Services (CAS)セッションのフォーマットライブラリが無視され、カタログのみが参照されます。

スレッドカーネル処理に対応していない出力形式は、CAS に書き込まれません。

入力形式は CAS セッションに書き込めきません。SAS コードに含まれた入力形式は無視されます。

ラベルとしての関数やラベルとしての出力形式を使用する出力形式は CAS に書き込めません。

ヒント:

ユーザー定義の出力形式名の最後を数字にすることはできません。詳細については、“ユーザー定義の出力形式” (*SAS 出力形式と入力形式: リファレンス*)と“SAS 名” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

適切なグローバルステートメントをこのプロシジャと使用することができます。リストについては、“SAS 名” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

SAS ライブラリに関連するユーザー定義フォーマットのリストの生成については、*SAS Cloud Analytic Services: ユーザー定義出力形式*の %UDFSEL マクロに関するドキュメントを参照してください。

参照項目:

“CAS セッションでの出力形式の使用” (867 ページ)

PROC FORMAT <*options*>;

EXCLUDE*entry(s)*;

INVALUE <$>*name* <(*informat-options*)> <*value-range-set(s)*>;

PICTURE *name* <(*format-options*)>
<*value-range-set-1* <(*picture-1-options*)>
<*value-range-set-2* <(*picture-2-options*)>> ...>;

SELECT*entry(s)*;

VALUE <$>*name* <(*format-options*)> <*value-range-set(s)*>;

ステートメント	タスク	例
“PROC FORMAT ステートメント”	変数に対して出力形式と入力形式を定義する	Ex. 3, Ex. 13, Ex. 15, Ex. 16
“EXCLUDE ステートメント”	カタログエントリを FMTLIB オプションと CNTLOUT=オプションによる処理から除外する	
“INVALUE ステートメント”	生データ値を読み込み、変換するための入力形式を作成する	Ex. 12
“PICTURE ステートメント”	数字を印刷するためのテンプレートを作成する	Ex. 3, Ex. 5, Ex. 18
“SELECT ステートメント”	FMTLIB オプションと CNTLOUT=オプションによって処理するためのカタログエントリを選択する	Ex. 15
“VALUE ステートメント”	変数値の印刷に使用する文字列を指定する出力形式を作成する	Ex. 8, Ex. 11, Ex. 17

PROC FORMAT ステートメント

変数に対するユーザー指定の出力形式と入力形式を作成します。

ヒント:

ユーザー定義の出力形式名の最後を数字にすることはできません。詳細については、“ユーザー定義の出力形式” (*SAS 出力形式と入力形式: リファレンス*)と“SAS 名” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

データセットオプションを CNTLIN=および CNTLOUT=データセットオプションと使用できます。リストについては、“言語の概念”を参照してください。

SAS および CAS の以前のバージョン間でのカタログ移動には、ある程度のリスクが伴う可能性があります。SAS では、このリスクを軽減するために CNTLOUT データセットを使用することをお勧めします。

例:

“例 3: ピクチャ出力形式の作成” (934 ページ)

“例 13: CNTLIN=データセットからの出力形式の作成” (957 ページ)

“例 15: 入力形式と出力形式の説明の印刷” (967 ページ)

“例 16: 永久出力形式の取得” (968 ページ)

構文

PROC FORMAT <*options*>;

オプション引数の要約

CASFMTLIB='*name*'
: CAS セッションにフォーマットライブラリを追加します。

CNTLIN=*input-control-SAS-data-set*
: PROC FORMAT が入力形式または出力形式を作成する SAS データセットを指定します。

CNTLOUT=*output-control-SAS-data-set*
: LIBRARY=オプションで指定したカタログに含まれる入力形式または出力形式に関する情報を保存する SAS データセットを作成します。

FMTLIB
: LIBRARY=オプションで指定されるカタログの入力形式または出力形式に関する情報を印刷します。

LIBRARY=*libref<.catalog>*
: PROC FORMAT ステップで作成中の入力形式または出力形式を含む SAS ライブラリまたはカタログを指定します。

LOCALE
: 現在の SAS ロケールに対応するフォーマットカタログの作成を指定します。

MAXLABLEN=*number-of-characters*
: PROC FORMAT 出力で表示される入力形式が適用された値または出力形式が適用された値の文字数を指定します。

MAXSELEN=*number-of-characters*
: PROC FORMAT 出力で表示される開始値と終了値の文字数を指定します。

NOREPLACE
: 新しい入力形式または出力形式が同じ名前の既存する入力形式または出力形式を置き換えないようにします。

PAGE
: カタログ内のそれぞれの出力形式と入力形式に関する情報を印刷します。

オプション引数

CASFMTLIB='*name*'
: CAS セッションにフォーマットライブラリを追加します。

  アクティブな SAS Cloud Analytic Services (CAS)セッションでのみ CASFMTLIB オプションを指定できます。PROC FORMAT は、CAS セッションに接続し、フォーマットライブラリをロードします。フォーマットライブラリがすでに CAS セッションに存在する場合は、SAS でそれが更新されます。また、SAS では、CAS のそのセッションで動作しているプロシジャによる後続の参照のために、検索リストにフォーマットライブラリが追加されます。つまり、フォーマットライブラリがすでに存在しており、さらに CASFMTLIB で新しいライブラリを作成すると、検索順序に新しいライブラリが追加されます。ライブラリ名は、スラッシュを含まない 1 レベル名にする必要があります。

注: CAS セッションには、複数のフォーマットライブラリを含めることができます。それらのライブラリは、その CAS セッションでのみ使用できます。CAS アクションを使用してフォーマットライブラリをグローバルスコープにプロモートする方法については、*"フォーマットライブラリのプロモート" (SAS Viya プラットフォーム: システムプログラミングガイド)*を参照してください。

SESSREF=オプションで追加の CAS セッションを指定できます。SESSREF=オプションとその他の CAS 言語要素の詳細については、*SAS Cloud Analytic Services: ユーザーガイド*を参照してください。

SAS 出力形式は、CAS でフォーマットライブラリに追加するかどうかに関係なく、ローカル SAS クライアントで使用可能です。CASFMTLIB オプションを指定すると、EXCLUDE および SELECT ステートメントは、CAS セッションのフォーマットライブラリではなく、ローカル SAS セッションのフォーマットカタログに適用されます。入力形式は CAS セッションにロードできません。CASFMTLIB で INVALUE ステートメントを指定すると、ログに NOTE が書き込まれ、CAS のフォーマットライブラリには何も書き込まれません。

制限事項 CASFMTLIB オプションで指定するフォーマットライブラリ名の長さは、必ず 63 文字以下にしてください。

注 CASFMTLIB オプションの指定値は、単一引用符または二重引用符で囲む必要があります。

指定したライブラリ名は、大文字と小文字が区別されます。これは有効な SAS 名である必要があり、空白スペースは含められきません。

VALUE または PICTURE ステートメントで定義された出力形式も、LIBRARY=オプションで指定されたフォーマットカタログに書き込まれます。ライブラリを指定しなかった場合、SAS では、WORK.FORMATS ライブラリが使用されます。

ヒント CAS アクション addFmtLib=fmtsearch を使用して SAS のフォーマットライブラリの検索順序を制御することができます。詳細については、*SAS Cloud Analytic Services: ユーザーガイド*の"Manage User-Defined Formats with CAS Actions"を参照してください。

参照項目: "CAS セッションでの出力形式の使用" (867 ページ)

CNTLIN=*input-control-SAS-data-set*
PROC FORMAT が入力形式または出力形式を作成する SAS データセットを指定します。

CNTLIN=は、VALUE、PICTURE または INVALUE ステートメントを使用せずに出力形式と入力形式を作成します。1 レベル名を指定する場合、LIBRARY=オプションを指定するかどうかに関係なく、プロシジャはデータセットに対しデフォルトのライブラリ(WORK ライブラリまたは USER ライブラリ)だけを検索します。

注 CNTLIN データセットで PROC FORMAT を使用するときは、START 列と END 列の長さは同じである必要があります。これらの長さが異なっていると、エラーが発生する場合があります。

LIBRARY=は、ライブラリまたはカタログを示します。ライブラリ参照名だけが指定されると、カタログ名 FORMATS が使用されます。

ヒント 入力制御データセットの共通ソースは、別の PROC FORMAT ステップの CNTLOUT=オプションからの出力です。

参照項目: “入力制御データセット” (923 ページ)

例 “例 13: CNTLIN=データセットからの出力形式の作成” (957 ページ)

CNTLOUT=*output-control-SAS-data-set*
LIBRARY=オプションで指定したカタログに含まれる入力形式または出力形式に関する情報を保存する SAS データセットを作成します。 CNTLOUT=オプションが記述されるのと同じステップで入力形式または出力形式を作成している場合、作成している入力形式または出力形式は CNTLOUT=データセットに含まれます。

1 レベル名を指定する場合、LIBRARY=オプションを指定するかどうかに関係なく、プロシジャはデータセットをデフォルトのライブラリ(WORK ライブラリまたは USER ライブラリ)に保存します。

異なるエンコーディングを持つ出力データセットを作成するために ENCODING オプションを指定して CNTLOUT=を発行すると、クロス環境データアクセス(CEDA)が切り捨てエラーを出し、SAS が処理を停止し、CNTLOUT=データセットが 0 回の観測で作成されます。SAS は、データの切り捨てについて次のようなエラーをログに書き込みます。

```
ERROR: Some character data was lost during transcoding in the dataset
    WORK.TEST. Either the data contains characters that are not
    representable in the new encoding or truncation occurred during
    transcoding.
```

ユーロの金額値を含むデータセットを使用しているとします。あなたは Wlatin1 セッションエンコーディングを使用しており、CNTLOUT=データセットに対して UTF-8 エンコーディングを指定しています。Wlatin1 では、LABEL 変数の長さは 5 バイトで、値は 1234 ユーロ(16 進表現では '803132334'x)になるように事前に決定されています。この変数を UTF-8 エンコーディングの CNTLOUT=データセットに格納しようとすると、その文字列の長さは 7 バイトである必要があります(16 進表現では 'E282AC3132334'x)。2 つの追加バイトはユーロ記号の UTF-8 エンコーディングで必要です。追加の 2 バイトがなければ、文字列は切り捨てられます。%COPY_TO_NEW_ENCODING マクロを使用すると、このエラーを回避できます。%COPY_TO_NEW_ENCODING マクロの詳細については、“%COPY_TO_NEW_ENCODING マクロを使用した文字の切り捨ての回避” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

マクロは CNTLOUT データセットを調べ、それを必要な長さの新しいエンコーディングで再作成します。関連する書式の一部として幅が格納されている場合、その値は CVP エンジンによって展開されません。フォーマットされた値が表示されると、フォーマットが切り捨てられることがあります。

CNTLOUT データセット内の書式幅の DEFAULT 値は、その書式のデフォルトの幅(バイト単位)です。ユーザーは、書式を作成するときに DEFAULT=、MIN=、および MAX=を指定できます。そうしない場合、デフォルトが最大のラベルに基づいて計算されます。UTF-8 エンコーディングで START、MIN、MAX、または LABEL 変数の文字数がより大きい場合、これらの幅は CVP エンジンによって拡

張されません。代わりに%COPY_TO_NEW_ENCODING マクロを使用してください。カタログを UTF-8 に変換するために PROC FORMAT で CNTLOUT=を使用する方法の詳細については、“Converting Format Catalogs to UTF-8 Encoding” (*Moving and Accessing SAS Files*)を参照してください。

注 LIBRARY=は、ライブラリまたはカタログを示します。ライブラリ参照名だけが指定されると、カタログ名 FORMATS が使用されます。

ヒント 後続の PROC FORMAT ステップで出力制御データセットを入力制御データセットとして使用できます。

参照項目: “出力制御データセット” (919 ページ)

SAS Viya プラットフォームでは、UTF-8 エンコーディングのみがサポートされています。SAS Viya プラットフォームでのフォーマットカタログのエンコーディングの詳細については、*SAS Viya プラットフォームでの UTF-8 へのデータの移行*および *UTF-8 データの処理における SAS Viya FAQ* を参照してください。

FMTLIB

LIBRARY=オプションで指定されるカタログの入力形式または出力形式に関する情報を印刷します。 特定の入力形式または出力形式に関する情報を取得するには、SELECT ステートメントまたは EXCLUDE ステートメントを使用してカタログをサブセットします。

注: FMTLIB では CASFMTLIB はサポートされていません。

操作 PAGE オプションにより FMTLIB が起動します。

ヒント FMTLIB からの出力が ODS LISTING 出力先で適切にフォーマットされていない場合、LINESIZE=システムオプションの値を増やします。

CNTLOUT=オプションを含めずにコード内で SELECT ステートメントまたは EXCLUDE ステートメントを使用すると、FMTLIB オプションが自動的に呼び出され、FMTLIB レポートが生成されます。必要に応じて、PROC FORMAT ステートメントの直前に次の ODS EXCLUDE ステートメントを置くことで、このレポートを抑制できます。

```
ods exclude FORMAT;
```

例 “例 15: 入力形式と出力形式の説明の印刷” (967 ページ)

LIBRARY=*libref*<.*catalog*>

PROC FORMAT ステップで作成中の入力形式または出力形式を含む SAS ライブラリまたはカタログを指定します。 プロシジャは、これらの入力形式と出力形式を後続の SAS セッションまたはジョブで使用できるように、指定するカタログに保存します。

別名 LIB=

デフォルト LIBRARY=オプションを省略すると、出力形式と入力形式は Work.Formats カタログに保存されます。LIBRARY=オプションを指定し、*catalog* に名前を指定しない場合、出力形式と入力形式は *libref*.FORMATS カタログに保存されます。

注 LIBRARY=は、ライブラリまたはカタログを示します。ライブラリ参照名だけが指定されると、カタログ名 FORMATS が使用されます。

ヒント SAS では Library.Formats が自動的に検索されます。フォーマットカタログに対し LIBRARY ライブラリ参照名を定義し使用することがあります。

FMTSEARCH=システムオプションを使用して、フォーマットカタログの検索順序を制御できます。詳細については、"FMTSEARCH = システムオプション" (*SAS システムオプション: リファレンス*)を参照してください。

参照項目: "入力形式と出力形式の保存 " (864 ページ)

例 "例 3: ピクチャ出力形式の作成" (934 ページ)

LOCALE
現在の SAS ロケールに対応するフォーマットカタログの作成を指定します。SAS が作成するカタログの名前は、LIBRARY=オプションで指定された SAS ライブラリまたはカタログに、現在の SAS のロケールを表す 5 文字の POSIX ロケールが追加された名前です。

参照項目: POSIX ロケール値のリストについては、"ロケールからエンコーディングへのマッピング" (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

例 SAS ロケールが German_Germany の場合、POSIX ロケール値は de_DE です。次の PROC FORMAT ステートメントを使用して、SAS はカタログ mylib.formats_de_DE を作成し、このプロシジャによって 作成した出力形式と入力形式を保存します。

```
proc format locale lib=mylib.formats;
```

MAXLABLEN=*number-of-characters*
CNTLOUT=データセットまたは FMTLIB オプションの出力で表示する入力形式が適用された値または出力形式が適用された値の文字数を指定します。FMTLIB オプションは、入力形式が適用された値または出力形式が適用された値に対し最大 40 文字を印刷します。

MAXSELEN=*number-of-characters*
CNTLOUT=データセットまたは FMTLIB オプションの出力で表示する開始値と終了値の文字数を指定します。FMTLIB オプションは、開始値と終了値に対し最大 16 文字を印刷します。

NOREPLACE
新しい入力形式または出力形式が同じ名前の既存する入力形式または出力形式を置き換えないようにします。 NOREPLACE を省略すると、その入力形式または出力形式がすでに存在することがプロシジャによって 警告され、それらが置き換えられます。

注 出力形式と入力形式を同じ名前にすることができます。

PAGE
カタログ内のそれぞれの出力形式と入力形式に関する情報を印刷します。

操作 PAGE オプションは、FMTLIB オプションをアクティブにします。

ヒント ODS LISTING 出力先で、それぞれの出力形式と入力形式に関する情報が**アウトプット**ウィンドウの別のページに表示されます。

EXCLUDE ステートメント

エントリを FMTLIB オプションと CNTLOUT=オプションによる処理から除外します。

制限事項:

1 つの EXCLUDE ステートメントだけを PROC FORMAT ステップで表示できます。

SELECT ステートメントと EXCLUDE ステートメントを同じ PROC FORMAT ステップ内で使用することはできません。

CASFMTLIB オプションを指定すると、EXCLUDE ステートメントでは、CAS セッションのフォーマットライブラリが無視され、SAS セッションのカタログのみが参照されます。

構文

EXCLUDE *entry(s)*;

必須引数

entry(s)
処理から除外する 1 つ以上のカタログエントリを指定します。カタログエントリの名前は、保存する入力形式または出力形式の名前と同じです。入力形式と出力形式の名前、文字と数値の入力形式と出力形式の名前をそれぞれ同じにすることができるため、EXCLUDE ステートメントで入力形式と出力形式を指定する際は特定の接頭辞を使用する必要があります。EXCLUDE ステートメントでエントリを指定する際は、次の規則に従います。

- ドル記号($)の付いた文字出力形式を含むエントリ名を前に置きます。
- アットマークとドル記号の付いた文字出力形式を含むエントリ名(@$*entry-name* など)を前に置きます。
- アットマーク(@)の付いた数値入力形式を含むエントリ名を前に置きます。
- 接頭辞の付かない数値出力形式を含むエントリ名を指定します。

詳細

名前指定のショートカット

コロン(:)とハイフン(-)ワイルドカード文字を使用して、エントリを除外できます。たとえば、次の EXCLUDE ステートメントは、文字 **a** で始まるすべての出力形式または入力形式を除外します。

```
exclude a:;
```

また、次の EXCLUDE ステートメントは、**apple** から **pear** までにアルファベット順に発生するすべての出力形式または入力形式を除外します。

```
exclude apple-pear;
```

FMTLIB 出力

PROC FORMAT ステートメントの FMTLIB または CNTLOUT なしで EXCLUDE ステートメントを使用する場合、プロシジャは FMTLIB オプションを起動し、FMTLIB オプション出力が行われます。

INVALUE ステートメント

生データ値を読み込み、変換するための入力形式を作成します。

参照項目: SAS によって提供されている入力形式に関するドキュメントについては、*SAS 出力形式と入力形式: リファレンス*を参照してください。

例: “例 12: 生の文字データを数値に変換する” (954 ページ)

構文

INVALUE <$>*name* <(*informat-options*)> <*value-range-set(s)*>;

オプション引数の要約

DEFAULT=*length*
: 入力形式のデフォルト長を指定します。

FUZZ=*fuzz-factor*
: 値を範囲に一致させるためのあいまい度を指定します。

JUST
: すべての入力文字列を範囲と比較する前に左寄せにします。

MAX=*length*
: 入力形式に対し最大長を指定します。

MIN=*length*
: 入力形式に対し最小長を指定します。

NOTSORTED
定義順で値または範囲を保存します。

REGEXP

REGEXPE
前述の範囲を Perl 正規表現として扱うように指定します。

UPCASE
すべての入力文字列を範囲と比較する前に大文字にします。

Control the input template.

value-range-set(s)
データを読み込むための変数テンプレートを指定します。

必須引数

name
作成中の入力形式に名前を付けます。

制限事項
ユーザー定義の入力形式の名前は、SAS によって提供されている入力形式の名前と同じにすることはできません。

"_C"と"_B" (どちらも大文字と小文字を区別しない)は、フォーマット名の予約キーです。これらのキーは、フォーマット名の末尾には表示できません。これらの接尾辞を持つフォーマットカタログを SAS 9 から移行すると、エラーが表示されます。回避策は、_C または_B 接尾辞を含まない名前でフォーマットエントリを再作成することです。

要件
名前は有効な SAS 名である必要があります。数値入力形式名の長さは最大 31 文字が可能です。文字入力形式名の長さは最大 30 文字が可能で、数字で終わらせることはできません。文字入力形式を作成している場合、ドル記号($)を最初の文字として使用します。ドル記号を名前に追加するのは、文字入力形式が 30 文字に制限されているためです。

操作
入力形式名の最大長は、VALIDFMTNAME=システムオプションによって制御されます。詳細については、*SAS システムオプション: リファレンス*を参照してください。

ヒント
名前の後にピリオドを使用して、入力形式を後で参照します。ただし、INVALUE ステートメントでは入力形式名の後にピリオドを使用しないでください。

ユーザー作成入力形式を参照するメッセージの印刷時に、名前にはアットマーク(@)が接頭辞として付けられます。入力形式の保存時に、アットマークが入力形式に指定する名前に接頭辞として付けられます。アットマークを名前に追加するのは、名前が 31 文字または 30 文字に制限されているためです。アットマークを使用する必要があるのは、名前を EXCLUDE ステートメントまたは SELECT ステートメントで使用している場合*だけ*です。入力形式を変数と関連付ける場合は、名前に接頭辞としてアットマークを付けないでください。

オプション引数

DEFAULT=*length*
入力形式のデフォルト長を指定します。 入力形式と変数を関連付ける際に特定の長さを指定しない場合、DEFAULT=の値が入力形式の長さになります。

デフォルト
文字入力形式の場合、最も長いラベルの長さ

数値入力形式の場合、等号の左側に数値データがあれば 12

引用符で囲まれた文字列の場合、最も長い文字列の長さ

注
DEFAULT=に無効な値を指定すると、SAS は値を無視してエラーをログに書き込みます。

ヒント
最良の方法として、既存の入力形式を value-range set で指定している場合は、常に DEFAULT=オプションを指定してください。

FUZZ=*fuzz-factor*
値を範囲に一致させるためのあいまい度を指定します。 数字が範囲に完全には一致しない、または厳密には範囲内になくても *fuzz-factor* 内にあれば、入力形式はそれを一致とみなします。たとえば、次の INVALUE ステートメントでは、LEVELS.入力形式が作成され、そこであいまい度 2 が使用されます。

```
invalue levels (fuzz=.2) 1='A'
               2='B'
               3='C';
```

FUZZ=.2 は、変数値が範囲のどちらかの端の値.2 内にある場合、入力形式が変数値の保存に対応するフォーマットされた値を使用することを意味します。そのため、LEVELS.入力形式は値 2.1 を **B** として保存します。

ヒント
INVALUE ステートメントを使用して数値入力形式を作成するとき、FUZZ=0 を指定して記憶域を保存します。

ゼロ以外のあいまい度は、非常に近いが完全には一致しない数値でのみ使用します。バイナリ検索を実行するために、範囲は並べ替え順に内部保存されます(NOTSORTED オプションが使用されていない場合)。あいまい度が 1 つの範囲の終わりに追加され、次の範囲の先頭から削除れた場合に、これらの範囲が重なると、結果は予測不能なものになることがあります。値は、バイナリ検索で一致した最初の範囲に置かれます。除外演算子は、このバイナリ検索アルゴリズムを無効にするには不十分です。最良の方法として、除外演算子を使用する場合は、FUZZ=0 または NOTSORTED オプションを設定します。

数値入力形式で<除外演算子を使用する場合は、FUZZ=0 を使用するのが最もよい方法です。

JUST
すべての入力文字列を範囲と比較する前に左寄せにします。

MAX=*length*
入力形式に対し最大長を指定します。 入力形式と変数を関連付ける際に、MAX=値を超える幅を指定できません。

デフォルト 40

範囲 1–32767

注 MAX=に無効な値を指定すると、SAS は値を無視してエラーをログに書き込みます。

MIN=*length*
入力形式に対し最小長を指定します。 CNTLIN=データセットに MIN の値として 0 以下になる値が含まれている場合、SAS はその無効な値を無視してそれに 1 を代入します。SAS コードで CNTLIN=データセットを指定すると、SAS はコードステップの処理を続行し、ログにエラーを書き込みません。

次の例では、MIN 値が-1 のデータセットを指定し、CNTLIN=を使用して後続の PROC FORMAT ステートメントでそのデータセットを呼び出します。

```
data temp;
  fmtname='abc';
  start=1;
  label='xyz';
  min=-1;
run;

proc format cntlin=temp;
run;
```

コードで MIN に無効な値を指定した場合、SAS はその無効な値を無視することも、それを別の値に置き換えることもしません。SAS はそのコードステップの処理を停止し、ログにエラーを書き込みます。

次の例では、NEWABC データセットに min=-1 を指定し、その結果、SAS は処理を停止し、エラーを発行します。

```
proc format;
   value newabc (min=-1) 1='yes';
run;
```

デフォルト 1

範囲 1–32767

NOTSORTED
定義順で値または範囲を保存します。

NOTSORTED を指定しない場合、値または範囲はデフォルトで並び替え順序で保存され、SAS では特定の値が含まれる範囲を位置付けるためのバイナリ検索アルゴリズムが使用されます。NOTSORTED を指定する場合、SAS は各範囲を一致するまで定義順に検索します。

次のうちいずれかが true の場合、NOTSORTED を使用します。

- 発生する特定範囲の尤度がわかっており、処理時間を削減するために、入力形式を使用してそれらの範囲を最初に検索する。

- FMTLIB オプションを使用して入力形式の説明を出力する際に、範囲を定義する順序を保持する。
- ORDER=DATA オプションと PRELOADFMT オプションを使用して PROC MEANS、PROC SUMMARY または PROC TABULATE のクラス変数を分析する際に、範囲を定義する順序を保持する。

値の分布が一様または不明な場合、または値の数が比較的小さい場合は、NOTSORTED を使用しないでください。NOTSORTED が指定されていないときに SAS が使用するバイナリ検索アルゴリズムにより、これらの条件下の検索のパフォーマンスが最適化されます。

CPORT プロシジャと CIMPORT プロシジャを使用して標準照合順序が異なる動作環境間で入力形式または出力形式を転送する場合、NOTSORTED が自動的に使用されます。NOTSORTED の自動設定は、ASCII と EBCDIC の動作環境間で入力形式または出力形式を転送する場合に発生します。この状況に問題がある場合、次を行います。

- PROC FORMAT ステートメントの CNTLOUT=オプションを使用し、出力制御データセットを作成します。
- CPORT プロシジャを使用して、制御データセットに対し転送ファイルを作成します。
- 対象の動作環境で CIMPORT プロシジャを使用して、転送ファイルをインポートします。
- 対象の動作環境で、PROC FORMAT を CNTLIN=オプションと使用し、インポートされた制御データセットから出力形式と入力形式を作成します。

REGEXP
REGEXPE
前述の範囲を Perl 正規表現として扱うように指定します。 REGEXPE を指定した場合、正規表現では、代替アクションの使用時のように、変更された結果をもたらすことが期待されます。

実行中に、すべての正規表現がコンパイルされ、最初の表現に入力データが渡されて、一致が確認されます。一致がある場合は、対応するラベルが使用されます。一致がない場合は、次の範囲が比較されます。範囲は並べ替えられず、INVALUE ステートメントで定義された順序か、または CNTLIN=データセットに表示される順序で処理されます。

REGEXP オプションを使用する正規表現のルールは、DATA ステップの PRXPARSE 関数のルールと同じです。REGEXP オプションのルールは、PRXCHANGE 関数のルールと同じです。

操作　CNTLIN=データセットを使用している場合、HLO 変数には、REGEXP を示す P と REGEXPE を示す E が含まれます。

参照項目: “例 10: Perl 正規表現を使用した入力形式の作成” (950 ページ)

UPCASE
すべての生データ値を可能な範囲と比較する前に大文字に変換します。UPCASE を使用する場合、指定する値または範囲が大文字であることを確認してください。

value-range-set(s)
生データと、生データがなる値を指定します。*value-range-set(s)*には、次のうち 1 つ以上が可能です。

value-or-range-1<, *value-or-range-2 ...*>=*informatted-value* | [*existing-informat*]

入力形式は生データを等号の右側の *informatted-value* の値に変換します。

value-or-range
“値や範囲の指定” (914 ページ)を参照してください。

informatted-value
value-or-range の生データがなる値です。*informatted-value* に対し、次の形式のうちいずれかを使用します。

'character-string'
最大 32,767 文字長の文字列です。通常、入力形式を使用して生データを変換する場合に *character-string* は文字変数の値になります。*character-string* を *informatted-value* に使用するのは、文字入力形式を作成しているときだけです。*character-string* を単一引用符または二重引用符で囲まない場合、INVALUE ステートメントは引用符がそこにあるとみなします

16 進リテラルの場合、使用できるのは、最大 32,767 入力文字または最大 16,382 表現文字(表現文字ごとに 2 つの 16 進文字)です。

number
入力形式適用値になる数値です。通常、入力形式を使用して生データを変換する場合に *number* は、数値変数の値になります。*number* を *informatted-value* に使用するのは、数値入力形式を作成しているときだけです。*number* の最大値は、ホストの動作環境によって異なります。

ERROR
指定範囲のデータ値を無効なデータとして扱います。SAS は欠損値を変数に割り当て、データ行を SAS ログに印刷し、警告メッセージを発行します。

SAME
入力形式で生データがその他の値として変換されないようにします。たとえば、次の GROUP.入力形式は値 01 から 20 までを変換し、結果として数字 1 から 20 までを割り当てます。その他すべての値には、欠損値が割り当てられます。

```
invalue group 01-20= _same_
        other= .;
```

existing-informat
SAS によって提供されている入力形式または既存するユーザー定義の入力形式です。作成している入力形式では既存する入力形式を使用して、等号の左側の *value-or-range* に一致する生データを変換します。既存する入力形式を使用する場合、その入力形式名を角かっこ([date9.]など)、またはかっこと縦棒((|date9.|)など)で囲みます。既存する入力形式の名前を単一引用符で囲まないでください。

ヒント 最良の方法として、value-range-set の既存する入力形式を指定する場合、DEFAULT=オプションを使用して常にデフォルト値を指定します。

例

例 1: 生データ値に対する文字入力形式の作成

$GENDER.文字入力形式は、生データ値 **F** と **M** を文字値**'1'**と**'2'**にそれぞれ交換します。

```
invalue $gender 'F'='1'
                'M'='2';
```

ドル記号の接頭辞は、入力形式が文字データを変換することを示します。

例 2: 文字値および数値または値範囲の作成

数値入力形式の作成時に、*value-or-range* に対して文字列または数字を指定できます。たとえば、TRIAL.入力形式は、**A** から **M** までを並び替える文字列を数字 1 に、**N** から **Z** までを並び替える文字列を数字 2 に変換します。入力形式は引用符なしの範囲 1 から 3000 を数値範囲として扱います。これには 1 から 3000 までのすべての数字が含まれます。

```
invalue trial 'A'-'M'=1
              'N'-'Z'=2
               1-3000=3;
```

例 3: 入力文字列を大文字および左揃えに変換する

EVALUATION.入力形式は UPCASE および JUST オプションを使用して、すべての入力文字列を大文字および左揃えにしてから、番号に変換します。

```
invalue evaluation (upcase just) 'O'=4
                                 'S'=3
                                 'E'=2
                                 'C'=1
                                 'N'=0;
```

注: INVALUE ステートメントでのオプションは、かっこで囲んで指定する必要があります。

例 4: _ERROR_と_SAME_を使用した入力形式の作成

CHECK.入力形式は_ERROR_と_SAME_を使用して、値 1 から 4、99 を変換します。その他すべての値は無効です。

```
invalue check 1-4=_same_
              99=.
             other=_error_;
```

数値入力形式を使用して値または範囲に対応しない文字列を変換する場合、エラーメッセージが表示されます。

PICTURE ステートメント

数字を印刷するためのテンプレートを作成します。

ヒント:

最良の方法としては、value-range-set に既存の出力形式を指定する場合、常に DEFAULT=オプション (908 ページ)を使用してデフォルト値を指定します。

DATATYPE=オプションを使用している場合、DEFAULT=オプションを使用して、これらの文字のフォーマットに十分な大きさのデフォルト出力形式幅を設定します。DEFAULT=オプションを設定しない場合、出力形式のデフォルト幅は等号の右側の最大値の幅となります。

DEFAULT、FUZZ、MAX、MIN、MULTILABEL、NOTSORTED、および ROUND オプションは、値範囲の指定前は有効です。

DATATYPE、DECSEP、DIG3SEP、FILL、LANGUAGE、MULT、NOEDIT、および PREFIX オプションは、ユーザー指定の値ラベルの後のかっこ内で有効です。

DATATYPE、DECSEP、DIG3SEP、FILL、LANGUAGE、MULT、NOEDIT、PREFIX、および ROUND オプションは、PICTURE ステートメントでのみ有効です。

参照項目:

SAS によって提供されている出力形式に関するドキュメントは、*SAS 出力形式と入力形式: リファレンス*および *SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

例:

“例 3: ピクチャ出力形式の作成” (934 ページ)
“例 5: ピクチャ出力形式の埋め込み” (939 ページ)
“例 18: 英語以外の言語での出力形式の作成” (974 ページ)

構文

PICTURE *name* <(*format-options*)>
<*value-range-set-1* <(*picture-1-options*)>
<*value-range-set-2* <(*picture-2-options*)>> ...>;

オプション引数の要約

Control the attributes of each picture in the format

FILL='*character*'
: フォーマットされた値を完了する文字を指定します。

MULTIPLIER=*n*
: フォーマットされる前に変数の値に掛ける数字を指定します。

NOEDIT
: 数字が数値セレクターではなく、メッセージ文字になるように指定します。

PREFIX='*prefix*'
: フォーマットされた値の前に置く文字接頭辞を指定します。

Control the attributes of the format

DATATYPE=DATE | TIME | DATETIME | DATETIME_UTIL
: ピクチャでディレクトリをテンプレートとして使用して、日付値、時間値、日時値をフォーマットできます。

DECSEP='*character*'
: 数字の小数部分に対して区切り文字を指定します。

DEFAULT=*length*
: ピクチャのデフォルト長を指定します。

DIG3SEP='*character*'
: 数字に対して 3 桁の区切り文字を指定します。

FUZZ=*fuzz-factor*
: 値を範囲に一致させるためのあいまい度を指定します。

LANGUAGE=
: 日付、時間、日時のピクチャで置き換えられる平日、月に使用される言語を指定します。

MAX=*length*
: 出力形式に対し最大長を指定します。

MIN=*length*
: 出力形式に対し最小長を指定します。

MULTILABEL
: 同一の値または重複した値を含む可能性のある複数の *values-or-range* 値へのラベルの割り当てが可能になります。

NOTSORTED
: 定義順で値または範囲を保存します。

ROUND
: フォーマットする前に、値を四捨五入して整数にします。

Control the template for printing

value-range-set
: 1 つ以上の変数値と、それらの値を印刷するためのテンプレートを指定します。

必須引数

name
作成中の出力形式に名前を付けます。

制限事項
: ユーザー定義の出力形式の名前を SAS によって提供されている出力形式の名前にすることはできません。

"_C"と"_B" (どちらも大文字と小文字を区別しない)は、フォーマット名の予約キーです。これらのキーは、フォーマット名の末尾には表示できません。これらの接尾辞を持つフォーマットカタログを SAS 9 から移行すると、エラーが表示されます。回避策は、_C または_B 接尾辞を含まない名前でフォーマットエントリを再作成することです。

要件
: 名前は有効な SAS 名である必要があります。数値出力形式名の長さには最大 32 文字が可能です。文字出力形式名の長さには最大 31 文字が可能で、

最後を数字にすることはできません。文字出力形式を作成している場合、最初の文字としてドル記号($)を使用します。このため、文字入力形式は 31 文字に制限されています。SAS 名に関する情報については、“ほとんどの SAS 名の規則” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

操作 出力形式名の最大長は、VALIDFMTNAME=システムオプションによって制御されます。詳細については、*SAS システムオプション: リファレンス*を参照してください。

ヒント 名前の後にピリオドを使用して、出力形式を後で参照します。ただし、VALUE ステートメントでは出力形式名の後にピリオドを置かないでください。

オプション引数

DATATYPE=DATE | TIME | DATETIME | DATETIME_UTIL
ピクチャでディレクトリをテンプレートとして使用して、日付値、時間値、日時値をフォーマットできます。 ピクチャ出力形式で使用するディレクティブに基づいて、DATE、TIME、DATETIME、DATETIME_UTIL のいずれかを指定します。ディレクティブ (892 ページ)の定義とリストについては、*picture* の説明を参照してください。

操作 DATATYPE=DATETIME に設定すると、日時の時間が 00:00:00 から 23:59:59 になります。DATATYPE=DATETIME_UTIL に設定すると、日時の時間が 00:00:01 から 24:00:00 の間になります。

ヒント 数値欠損値をフォーマットする場合、結果のラベルは ERROR となります。プログラムに欠損値をチェックする句を追加することにより、ERROR ラベルを削除できます。

DEFAULT=*length*
ピクチャのデフォルト長を指定します。 出力形式と変数を関連付ける際に特定の長さを指定しない場合、DEFAULT=の値が *picture* の長さになります。

デフォルト 最も長いピクチャ値の長さ

範囲 1–32767

ヒント DATATYPE=オプションを使用している場合は、DEFAULT=オプションを使用して、これらの文字のフォーマットに十分な大きさのデフォルト出力形式幅を設定します。

DECSEP='*character*'
数字の小数部分に対して区切り文字を指定します。

デフォルト .(小数点)

DIG3SEP='*character*'
数字に対して 3 桁の区切り文字を指定します。

デフォルト , (カンマ)

FILL='*character*'
フォーマットされた値を完了する文字を指定します。

有効桁数が出力形式の長さより少ない場合、出力形式はフォーマットされた値を完了つまり幅に合わせる必要があります。

- 数値セレクターとしてゼロを指定すると、出力形式は *character* を使用して、フォーマットされた値を幅に合わせます。
- ゼロ以外の数値セレクターを指定すると、出力形式はゼロを使用してフォーマットされた値を幅に合わせます。FILL=オプションは影響しません。

配置されている最終有効桁にマップする数値セレクターの左側に表示されるカンマなどのその他の文字がピクチャに含まれている場合、文字は埋め文字または先頭のゼロによって置き換えられます。

デフォルト ' ' (ブランク)

制限事項 FILL=オプションは、値に出力形式を設定する関数を使用する場合は、有効になりません。

FILL オプションでは 1 バイト文字のみがサポートされます。

操作 同一のピクチャで FILL=オプションと PREFIX=オプションを使用する場合、出力形式は接頭辞を置いてから埋め字を入力します。

例 “例 5: ピクチャ出力形式の埋め込み” (939 ページ)

FUZZ=*fuzz-factor*
値を範囲に一致させるためのあいまい度を指定します。 数字が範囲に完全には一致しない、または厳密には範囲内になくてもどちらかの端の *fuzz-factor* 内にあれば、出力形式はそれを一致とみなします。たとえば、次の VALUE ステートメントでは、LEVELS.出力形式が作成され、そこであいまい度 2 が使用されます。

```
value levels (fuzz=.2) 1='A'
                       2='B'
                       3='C';
```

FUZZ=.2 は、変数値が範囲のどちらかの端の値.2 内にある場合、出力形式が変数値の印刷に対応するフォーマットされた値を使用することを意味します。そのため、LEVELS.形式出力は値 2.1 を **B** としてフォーマットします。

デフォルト 1E−12 (数値出力形式)

ヒント VALUE ステートメントを使用して数値出力形式を作成するとき、FUZZ=0 を指定して記憶域を保存します。

ゼロ以外のあいまい度は、非常に近いが完全には一致しない数値でのみ使用します。*あいまい度*が 1 つの範囲の終わりに追加され、次の範囲の先頭から削除された場合に、これらの範囲が重なると、結果は予測不能なものになることがあります。値は、バイナリ検索で一致した最初の範囲におかれます。

最良の方法は、数値出力形式で<除外演算子を使用する場合は、FUZZ=0 を使用します。

LANGUAGE=
日付、時間、日時のピクチャで置き換えられる平日、月に使用される言語を指定します。 サポートされていない、または無効な言語を指定すると、英語が使用されます。

LANGUAGE=オプションの有効値は次のとおりです。

Afrikaans	English	Macedonian	Spanish
Catalan	Finnish	Norwegian	Swedish
Croatian	French	Polish	Swiss_French
Czech	German	Portuguese	Swiss_German
Danish	Hungarian	Russian	
Dutch	Italian	Slovenian	

デフォルト シングルバイトの文字セットの場合、DFLANG=システムオプションで指定される言語

ダブルイトおよび UTF-8 の文字セットの場合、LOCALE=システムオプションで指定される言語

ヒント ユーザー定義の出力形式を、LANGUAGE=オプションでサポートされる言語以外の言語で使用するには、LOCALE=システムオプションをその言語のロケールに設定します。PROC FORMAT では、LANGUAGE=オプションを指定しないでください。ピクチャ出力形式の言語は、ロケール設定によって決定されます。ロケールのリストについては、*"ロケールからエンコーディングへのマッピング" (SAS 各国語サポート(NLS): リファレンスガイド)*を参照してください。

参照項目: *"DFLANG= システムオプション" (SAS 各国語サポート(NLS): リファレンスガイド)*

MAX=*length*
出力形式に対し最大長を指定します。 出力形式と変数を関連付ける際に、MAX=値を超える幅を指定できません。

デフォルト 40

範囲 1–32767

MIN=*length*
出力形式に対し最小長を指定します。

デフォルト 1

範囲 1–32767

MULTILABEL
同一の値または重複した値を含む可能性のある複数の *values-or-range* 値へのラベルの割り当てが可能になります。 ラベルは value-range-set の等号の右側の

ピクチャ定義によって決定されるフォーマットされた値です。ここでは、MULTILABEL の使用方法の例を示します。

次の PICTURE ステートメントでは、MULTILABEL オプションの 2 つの使用を示します。いずれの場合も、数値出力形式は、ラベルとして割り当てられます。最初の PICTURE ステートメントは複数のラベルを単一の値に割り当てます。複数のラベルは単一の値範囲に割り当てることもできます。2 番目の PICTURE ステートメントは、ラベルを重複した値範囲に割り当てます。MULTILABEL オプションにより、複数のラベルを重複した値に割り当てることができます。

```
picture abc (multilabel)
  1000='9,999'
  1000='9999';

 picture overlap (multilabel)
  /* without decimals */
  0-999='999'
  1000-9999='9,999'

   /* with decimals */
  0-9='9.999'
  10-99='99.99'
  100-999='999.9';
```

PROC MEANS、PROC SUMMARY、PROC TABULATE などのマルチラベル有効プロシジャだけが、複数のラベルを使用できます。その他すべてのプロシジャと DATA ステップは、1 次ラベルのみ認識します。

指定エントリに対する *primary label* は、最初の値に割り当てられるフォーマットされた値(ピクチャに基づく)、またはすべての値(等号の左側の)が順番に並べ替えられるときにエントリに一致する、または含まれる range-of-values (等号の左側)です。次に例を示します。

- 最初の PICTURE ステートメントでは 1000 の 1 次ラベルは 1,000 です。ピクチャ 9,999 が 1000 に割り当てられている最初の値であるためです。1000 の第 2 ラベルは 1000 で、これは 9999 ピクチャに基づいています。
- 2 番目の PICTURE ステートメントでは、5 の 1 次ラベルは範囲 0–9 に割り当てられている 9.999 ピクチャに基づく 5.000 です。0–9 が 5 を含む値の最初の範囲であるためです。5 の 2 次ラベルは 005 です。範囲 0–999 が範囲 0–9 の後に順に発生するためです。

複数のラベルを 1 つの値に割り当てる際は、注意が必要です。

value-range-sets の割り当て時に NOTSORTED オプションを使用しない限り、value-range-sets は並べ替え順に保存されます。この順序により、MULTILABEL 出力形式の value-range-sets の処理時に予期せぬ結果が発生する可能性があります。次に例を示します。

2 番目の PICTURE ステートメントでは、15 の 1 次ラベルは 015 で、15 の 2 次ラベルは 15.00 です。範囲 0–999 が範囲 10–99 の前に順に発生するためです。15 の 1 次ラベルで 99.99 出力形式を使用する場合、PICTURE ステートメントで範囲 10–99 を 0–99 に変更します。範囲 0–99 は範囲 0–999 の前に順に発生し、必要な結果が生成されます。

制限事項　単一の出力形式または入力形式に対して作成できるラベルの最大数は、255 です。

MULTIPLIER=*n*
フォーマットされる前に変数の値に掛ける数字を指定します。 MULTIPLIER=オプションの値は乗算の結果と *value-range-set* の *picture* 部分の数値セレクターによって異なります。たとえば、次の PICTURE ステートメントでは、MILLION.出力形式が作成されます。この出力形式では、変数値 1600000 が$1.6M としてフォーマットされます。

```
picture million low-high='09.9M'
    (prefix='$' mult=.00001);
```

1600000 に最初に.00001 を掛けると、16 になります。小数点の後に数値セレクターがあることに注意してください。値 16 は、右で始まるピクチャに置かれます。値 16 は 09.9 を上書きし、結果は 01.6 となります。先頭のゼロが削除され、最終結果は 1.6M となります。

低-高の値が'000M'である場合、結果は 16M となります。

別名 MULT=

デフォルト 10^n.*n* はピクチャの最初の小数点の後の桁数です。たとえば、データに値 123.456 が含まれていて、それを'999.999'のピクチャを使用して印刷するとします。この出力形式は、123.456 に 10^3 を掛けて求め値 123456 が、フォーマットされた値 123.456 になります。

制限事項 MULT=オプションは、値に出力形式を設定する関数を使用する場合は、有効になりません。

DATATYPE=オプションを指定すると、MULT=オプションは無視されます。

例 “例 3: ピクチャ出力形式の作成” (934 ページ)

“例 4: 大きなドル金額のピクチャ出力形式の作成” (936 ページ)

NOEDIT
数字が数値セレクターではなく、メッセージ文字になるように指定します。 つまり、出力形式は数字をピクチャで表示されるように印刷します。たとえば、次の PICTURE ステートメントでは、MILES.出力形式が作成されます。この出力形式では、1000 より大きい変数値はいずれも**>1000 miles** としてフォーマットされます。

```
picture miles 1-1000='0000'
      1000<-high='>1000 miles'(noedit);
```

制限事項 NOEDIT=オプションは、値に出力形式を設定する関数を使用する場合は、有効になりません。

NOTSORTED
定義順で値または範囲を保存します。 NOTSORTED を指定しない場合、値または範囲はデフォルトで並び替え順序で保存され、SAS では特定の値が含まれる範囲を位置付けるためのバイナリ検索アルゴリズムが使用されます。
NOTSORTED を指定する場合、SAS は各範囲を一致するまで定義順に検索します。

次のうちいずれかが true の場合、NOTSORTED を使用します。

- 発生する特定範囲の尤度がわかっており、処理時間を削減するために、入力形式を使用してそれらの範囲を最初に検索する。
- FMTLIB オプションを使用して出力形式の説明を出力する際に、範囲を定義する順序を保持する。
- ORDER=DATA オプションと PRELOADFMT オプションを使用して PROC MEANS、PROC SUMMARY または PROC TABULATE のクラス変数を分析する際に、範囲を定義する順序を保持する。

値の分布が一様または不明な場合、または値の数が比較的小さい場合は、NOTSORTED を使用しないでください。NOTSORTED が指定されていないときに SAS が使用するバイナリ検索アルゴリズムにより、これらの条件下の検索のパフォーマンスが最適化されます。

CPORT プロシジャと CIMPORT プロシジャを使用して標準照合順序が異なる動作環境間で入力形式または出力形式を転送する場合、NOTSORTED が自動的に使用されます。NOTSORTED の自動設定は、ASCII と EBCDIC の動作環境間で入力形式または出力形式を転送する場合に発生します。この状況に問題がある場合、次を行います。

- PROC FORMAT ステートメントの CNTLOUT=オプションを使用し、出力制御データセットを作成します。
- CPORT プロシジャを使用して、制御データセットに対し転送ファイルを作成します。
- 対象の動作環境で CIMPORT プロシジャを使用して、転送ファイルをインポートします。
- 対象の動作環境で、PROC FORMAT を CNTLIN=オプションと使用し、インポートされた制御データセットから出力形式と入力形式を作成します。

PREFIX='*prefix*'
フォーマットされた値の前に置く文字接頭辞を指定します。 接頭辞は、値の最初の有効桁の前に置かれます。ゼロ数値セレクターを使用する必要があります。使用しない場合、接頭辞は使用されません。

PREFIX=は通常、先頭の通貨記号とマイナス記号の印刷に使用します。たとえば、PAY.出力形式は、変数値 25500 を**$25,500.00** と印刷します。

```
picture pay
  low-high='000,009.99' (prefix='$');
```

デフォルト　no prefix

制限事項　PREFIX=オプションは、値に出力形式を設定する関数を使用する場合は、有効になりません。

操作　同一のピクチャで FILL=オプションと PREFIX=オプションを使用する場合、出力形式は接頭辞を置いてから埋め字を入力します。

例　“例 3: ピクチャ出力形式の作成” (934 ページ)

“例 5: ピクチャ出力形式の埋め込み” (939 ページ)

注意 **ピクチャの幅が値と接頭辞の両方を含めるのに十分でない場合、出力形式は接頭辞を切り捨て、または省略し、これによりデータが不正確になります。**

ROUND
フォーマットする前に、値を四捨五入して整数にします。 ROUND オプションを指定しない場合、出力形式は変数値に乗数を掛け、小数部(ある場合)を切り捨て、定義するテンプレートに従って結果を印刷します。ROUND オプションを指定する場合、出力形式は変数値に乗数を掛け、その結果を四捨五入して整数にしてから、その値にテンプレートに従ってフォーマットします。FUZZ=オプションも指定されている場合、四捨五入は、値が属する範囲の決定にあいまい度が使用された後に発生します。

ヒント ROUND オプションは、値.5 を次に最も大きい整数に四捨五入します。

注意 **数字を四捨五入することにより桁が数字に追加される場合、ピクチャが追加される桁に対し十分広い必要があります。** たとえば、数字.996 のピクチャは'99' (prefix '.' mult=100)になります。数字を四捨五入し、それに 100 を掛けると、結果の数字は 100 になります。ピクチャが適用されると、結果は不正確な数字、.00 となります。四捨五入するときに数字が正しいものになるように、ピクチャ幅をより大きな数字を表示するのに十分なものにしてください。

value-range-set
1 つ以上の変数値と、それらの値を印刷するためのテンプレートを指定します。*value-range-set* には、次の形式があります。

value-or-range-1 <, value-or-range-2, ...>='picture'

value-or-range
"値や範囲の指定" (914 ページ)を参照してください。

picture
数値変数の値をフォーマットするためのテンプレートを指定します。ピクチャは、単一引用符で囲まれた一連の文字です。ピクチャの最大長は 40 文字です。ピクチャは、数値セレクター、メッセージ文字、ディレクティブの 3 つの種類の文字で指定されます。1 つのピクチャに可能な数値セレクターは最大 16 です。

digit selectors
数値に対し位置を定義する数値文字(0 から 9 まで)です。数値セレクターがゼロ以外のピクチャ出力形式は変数値の先頭のゼロを印刷します。ピクチャ数値セレクターが 0 の場合は変数値の先頭のゼロを印刷しません。ピクチャ出力形式に数値セレクターが含まれている場合、数値セレクターはピクチャの最初の文字である必要があります。

注 このセクションでは 9 をゼロ以外の数値セレクターとして使用します。

message characters
ピクチャで指定されているように印刷する非数値文字です。次の PICTURE ステートメントには、数値セレクター(99)とメッセージ文字 (**illegal day value**)が含まれています。DAYS.出力形式に非ゼロ数値セレク

ターがあるため、値は先頭のゼロとともに印刷されます。特殊範囲 OTHER は、指定範囲(1 から 31 まで)内にない値に対しメッセージ文字を印刷します。

```
picture days
      01-31='99'
      other='99-illegal day value';
```

例　値 02 と 67 は、のように印刷されます

```
02
67-illegal day value
```

directives

ピクチャで日付値、時間値、日時値をフォーマットするために使用できる特殊文字です。

注: ディレクティブを 使用できるのは、PICTURE ステートメントで DATATYPE=オプション (885 ページ)を指定するときだけです。DATATYPE=オプションの値が、使用するディレクティブのタイプに適していることを確認してください。不適切な値を使用すると、データはフォーマットされません。たとえば、%a ディレクティブの 場合、DATATYPE=DATE を使用します。

DFLANG 日時ハンドラーと各国語(NL)日時ハンドラーは、ユーザー形式の処理で文字の大文字小文字を制御する 2 つの方法です。DFLANG 日時ハンドラーは、非 UTF8 および非 DBCS セッションでのデフォルトのメソッドです。NL 日時ハンドラーは、UTF8 および DBCS セッションでのデフォルトのメソッドです。

注: DFLANG 日時ハンドラーは、UTF8 および DBCS セッションに指定した場合に使用されます。それ以外の場合は、NL 日時ハンドラーが使用されます。

DFLANG 日時ハンドラーを指定する場合、英語(JAN など)を指定すると月の名前はすべて大文字になり、フランス語(jan など)を指定すると月の名前はすべて小文字になります。ドイツ語(たとえば、Jan)を指定すると、月の名前はすべて大文字と小文字が混在します。

NL 日時ハンドラーを指定する場合、英語またはドイツ語(Jan)を指定すると月の名前は大文字と小文字が混在し、フランス語(jan など)を指定すると小文字になります。

DFLANG 日時ハンドラーは、サポートされている言語が DFLANG=システムオプションで指定され、また指定されたすべての日時要素が DFLANG 形式ハンドラーによってサポートされている場合にのみ使用されます。それ以外の場合は、NL 日時ハンドラーが使用されます。

月名の大文字と小文字は、DFLANG=システムオプションと PICTURE ステートメントの LANGUAGE=引数によって決定されます。それ以外の場合、大文字と小文字はロケールデータの言語要素によって決定されます。たとえば、LOCALE=オプションで指定された言語が英語の場合、ロケールデータでは大文字と小文字が混在する文字列として定義されているため、最初の文字のみが大文字になります。詳細については、“DFLANG= システム

オプション" (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

注: DFLANG オプションは、22 の言語のみをサポートします。DFLANG=オプションは、互換性の目的でのみ使用してください。DFLANG=オプションを指定すると、日時ディレクティブのサポートが制限され、%I などの一部の要素はサポートされません。

FORMAT プロシジャで使用する必要がある場合、または月名が英語などのロケールですべて大文字で表記される必要がある場合は、LANGUAGE=引数を指定します。

主に EUR *日時形式を使用し、互換性のためにのみヨーロッパ言語のサポートが必要な場合は、DFLANG=オプションを使用して言語を指定できます。

LOCALE=システムオプションを使用して、2 バイトおよび UTF-8 文字セットのロケールを設定します。ロケールが LOCALE=で設定されている場合、%b ディレクティブは、月の名前を最初の文字のみ大文字にしてフォーマットします。詳細については、"LOCALE システムオプション" (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

NL 日時ハンドラーのみでサポートされるフォーマット要素を指定すると、DFLANG=オプションが使用されているかどうかに関係なく、NL フォーマットハンドラーが使用されます。%b は、%I などのサポートされていないディレクティブがある 場合、英語の場合に大文字と小文字を混在する省略形の月名も生成します。

UTF-8 セッションの LANGUAGE=引数のデフォルト値は LOCALE です。UTF-8 セッションで月の名前をすべて大文字でフォーマットするには、LANGUAGE=引数でロケールを指定します。詳細については、"LANGUAGE=" (887 ページ)を参照してください。

使用できるディレクティブは次のとおりです。

%a
曜日の省略名(Wed など)。

%A
曜日の完全名(Wednesday など)。

%b
月の省略名(JAN や Jan など)。

ヒント 英語の場合、ディレクティブ %3B を使用して、先頭文字のみを大文字(Jan など)にして月の省略名を作成します。

%<*n*>B
n がディレクティブに 含まれていない場合は、完全な月の名前(たとえば、1 月)。 *n* は、月の名前に表示される文字数を指定します。それに対し、一部のロケールでは、%b ディレクティブで、大文字の 3 文字の月の省略名が作成されます。

制限事項 *n* は、DBCS および Unicode SAS セッションでサポートされません。

例 %3B と指定すると、October の月を表す Oct が書き込まれます。

%C
ブランクを埋め込んだ長い月名(January から December まで)(December など)。

%d
月の日付。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0d など)。

%e
先頭のスペースを 含む 2 文字の 10 進数の月の日付(" 1"- "31") (“ 2”など)。

%F
ブランクを埋め込んだ曜日の完全名。

%G
4 桁の 10 進数の年(2008 など)。1 月 1 日を含む週の 4 日以上が新年にかかっている場合、新年の第 1 週とみなされます。その他の場合は前年の最終週となり、年は前年とみなされます。

%H
時間(24 時間)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0H など)。

ヒント DATETYPE=DATETIME に設定すると、SAS は日時の時間に 00:00–23:59 を使用します。DATETYPE=DATETIME_UTIL に設定すると、SAS は日時の時間に 00:00:01–24:00:00 を使用します。24:00:00 はその日の終わりの午前 0 時です。00:00:00 という時刻は時間の範囲に含まれていないため、この値を使用すると前の日の 24:00:00 に変換されます。DATETIME を指定すると、00:00:00 は新しい日の午前 0 時になり、24:00:00 は次の日の午前 0 時になります。

例 “例 7: 24 時間形式から 00:00:01–24:00:00 形式への変更” (943 ページ)

%I
時間(12 時間)。

別名 %i

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0I など)。

%j
先頭のゼロを含む 10 進数の年の日付(1–366)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0j など)。

%m
月(1–12)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0m など)。

%M
分(0–59)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0M など)。

%n
10 進数での期間の日数(最大 10 桁) (25 など)。

制限事項 このディレクティブは、DBCS と Unicode の SAS セッションには対応していません。

%o
ブランクが埋め込まれた月(1-12) (" 2"など)。

%p
a.m.または p.m.と同じです

%q
1、2、3、4 など、四半期の省略文字列。

制限事項 "_C"と"_B" (どちらも大文字と小文字を区別しない)は、フォーマット名の予約キーです。これらのキーは、フォーマット名の末尾には表示できません。これらの接尾辞を持つフォーマットカタログを SAS 9 から移行すると、エラーが表示されます。回避策は、_C または_B 接尾辞を含まない名前でフォーマットエントリを再作成することです。

%Q
Quarter1、Quarter2、Quarter3、Quarter4 などの四半期の文字列。

%s
10 進数での小数点以下の秒数(.39555 など)。フォーマットされた桁数は、出力形式使用時に指定される小数点の右側の桁数です。小数点以下の秒数は四捨五入され、小数点以下の秒数に指定された桁数に調整されます。

制限事項 このディレクティブは、DBCS と Unicode の SAS セッションには対応していません。

操作 %s ディレクティブと %S ディレクティブを一緒に使用することはできません。

ヒント %s ディレクティブは、秒全体と秒の小数部の両方を表示します。

1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0s など)。

%S
可能なうるう秒に使用可能な秒数(0–59)。

操作 %s ディレクティブと %S ディレクティブを一緒に使用することはできません。

ヒント %S ディレクティブは秒全体を表示します。小数点以下の秒数は表示されません。

1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0S など)。

例 58 と 59.07

%u
1 桁の 10 進数の曜日(1–7 (Monday - Sunday)) (Sunday=7 など)。

%U
10 進数の年の週数(0–53)。日曜日が週の第 1 日とみなされます。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0U など)。

%V
第 1 週の開始日が第 1 月曜日の週数(01–53)。第 1 週の最小日数は 4 です。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0SV など)。

%w
1 桁の 10 進数の曜日(0–6 (Sunday から Saturday)) (Sunday=0 など)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0w など)。

%W
第 1 週の開始日が第 1 月曜日の週数(0–53)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0W など)。

%y
世紀なしの年(0–99) (93 など)。

注 1 桁の数字の前に先頭のゼロを追加するには、ディレクティブの前に 0 を挿入します(%0y など)。

%Y
4 桁の 10 進数の世紀を含む年(1970–2069) (1994 など)。

%z
UTC タイムゾーンオフセット。

%Z
タイムゾーン名。

%%
%文字。

ヒント　コードをプログラムに追加して、欠損値の表示方法を指示します。

操作　出力形式定義で LANGUAGE=と PICTURE=を指定すると、出力形式では英語とヨーロッパ言語のみサポートされます。LANGUAGE=オプションによってサポートされているもの以外の言語のユーザー定義出力形式を使用するには、PICTURE=ステートメントを使用します。LANGUAGE=オプションを指定しないでください。ピクチャ出力形式の言語は、ロケール設定によって決定されます。

詳細

ピクチャ出力形式の作成: ステップバイステップ

このセクションでは、先頭にゼロを付けないピクチャ出力形式の作成方法を説明します。サンプルデータセットでは、変数 Amount のデフォルト印刷で 1 から-1 までの数字に先頭のゼロが含まれています。PICTURE ステートメントで、1 から–1 までの数値の先頭のゼロを削除する 2 つの似たような出力形式を定義します。2 つの出力形式の違いは、NOZEROSR.出力形式では ROUND オプションを指定して数字を四捨五入し、NOZEROS.出力形式では四捨五入しないという点です。

次のプログラムでは、サンプルデータセットを作成、並べ替え、印刷します。

```
data sample;
  input Amount;
  datalines;
-2.051
-.05
-.017
 0
.093
.54
.556
6.6
14.63
0.996
-0.999
-45.00
;
run;


proc sort data=sample;
  by amount;
run;
```

```
proc print data=sample;
  title 'Default Printing of the Variable Amount';
run;
```

Default Printing of the Variable Amount

Obs	Amount
1	-45.000
2	-2.051
3	-0.999
4	-0.050
5	-0.017
6	0.000
7	0.093
8	0.540
9	0.556
10	0.996
11	6.600
12	14.630

次は、NOZEROSR.および NOZEROS.出力形式を指定する PROC FORMAT ステップです。どちらの出力形式でも、フォーマットされた値で先頭のゼロは削除されます。NOZEROSR.出力形式では、ROUND オプションが指定されて数値が四捨五入されます。NOZEROS.出力形式では、四捨五入は実行されません。

```
libname library 'SAS-library';

proc format;
  picture nozerosR (round fuzz=0)
   low -  -1      =  '000.00' (prefix='-')
   -1 < - < -.99  =  '0.99'  (prefix='-' mult=100)
   -0.99 <-<  0   =  '99'    (prefix='-.' mult=100)
            0   =  '9.99'
    0 <  -< .99   =  '99'    (prefix='.' mult=100)
    0.99 - < 1    =  '0.99'  (mult=100)
    1  - high     =  '09.99';

  picture nozeros (fuzz=0)
   low -  -1      =  '000.00' (prefix='-')
   -1 < - < -.99  =  '0.99'  (prefix='-.' mult=100)
   -0.99 <-<  0   =  '99'    (prefix='-.' mult=100)
            0   =  '9.99'
    0 <  -< .99   =  '99'    (prefix='.' mult=100)
    0.99 - < 1    =  '0.99'  (prefix='.' mult=100)
    1  - high     =  '09.99';

run;
```

次のテーブルに、各範囲からの値をフォーマットする方法について説明します。各手順の詳細については、表 26.69 (902 ページ)を参照してください。

表 26.2　ピクチャ出力形式の作成

ステップ	PICTURE ステートメントの処理のルール	例
1	値の範囲を決定し、ピクチャを使用します。	2 番目の範囲で、除外演算子<がハイフンの両側に表示され、範囲から−1 と-.99 が削除されます。3 番目の範囲では、0 と.99 が削除されます。4 番目の範囲では、1 が削除されます。 除外演算子が使用されるため、FUZZ=0 オプションが指定されています。
2	数値の絶対値を取得します。	絶対値が使用されているため、マイナス記号に接頭辞を付けるために負数には別の範囲とピクチャが必要です。
3	数字に MULT=値を掛けます。MULT=オプションを指定しない場合、PICTURE ステートメントはデフォルト値を使用します。デフォルトは 10^n です。*n* は、ピクチャの小数点の右側の数値セレクターの数です(ステップ 6 で数値セレクターについてさらに説明しています)。[1]	MULT=値の指定は、0 から 1 までの数字と 0 から-1 までの数字に必要です。小数点がこれらの範囲のピクチャに表示されないためです。MULT=のデフォルトは 1 になるため、有効桁の切り捨てにより MULT=値が指定されません(切り捨てについては次のステップで説明します)。MULT=値が指定されていない 3 つの範囲に対し、MULT=値のデフォルトは 100 になります。対応するピクチャに小数点の右側に 2 つの数値セレクターが含まれているためです。MULT=値が適用された後、すべての 有効桁は小数点の左側に移動されます。
4	数値が大きい整数の 10^{-8} 内である場合、数値は切り上げられます。この演算は、ROUND オプションが実行される前に実行されます。 ROUND オプションが有効になっています。この出力形式では、小数点の後の数字が.5 以上の場合、小数点の後の数字は次に大きい整数まで切り上げられます。	例ではすべての有効桁が小数点の左に移動されるようにする MULT=値を使用するため、有効桁は失われません。ゼロは切り捨てられます。 205.1 は、205 まで切り捨てられます。 55.6 は、56 に切り上げられます。 99.6 は、100 に切り上げられます。 四捨五入は 5 と 660 で実行されません。
4a	ROUND オプションが実行されない場合、小数点の後の数字が切り捨てられます。	205.1 は、205 に切り捨てられます。 55.6 は、55 に切り捨てられます。 99.6 は、99 に切り捨てられます。

ステップ	PICTURE ステートメントの処理のルール	例
5	数字を文字列に変換します。数字がピクチャよりも短い場合、文字列の長さはピクチャの数値セレクターの数と同じです。文字列に先頭のゼロを埋め込みます(結果は Zw.出力形式の使用と同じになります。Zw.については、*SAS 出力形式と入力形式: リファレンス*の SAS 出力形式に関するセクションを参照してください。)	205 は、文字列 00205 になります。 5 は、文字列 05 になります。 56 は、文字列 56 になります。 100 は、文字列 100 になります。 660 は、文字列 0660 になります。 ピクチャが数字より長い場合、出力形式は先頭のゼロを各値に追加します。出力形式は先頭のゼロを文字列 56 と 100 に追加しません。これは、対応するピクチャに同じ数の数値セレクターが含まれているためです。
5a		205 は、文字列 00205 になります。 5 は、文字列 05 になります。 55 は、文字列 55 になります。 99 は、文字列 099 になります。 660 は、文字列 0660 になります。 ピクチャが数字より長い場合、出力形式は先頭のゼロを各値に追加します。出力形式は先頭のゼロを文字列 55 に追加しません。これは、対応するピクチャに同じ数の数値セレクターが含まれているためです。
6	文字列をピクチャに適用します。出力形式は文字列の最右の *n* 文字のみマップします。*n* はピクチャの数値セレクターの数です。そのため、ピクチャに文字列の文字に合うだけの十分な数値セレクターが含まれていることを確認することは重要です。 出力形式は最右の *n* 文字を取得した後、これらの文字をピクチャへ左から右にマップします。文字列に先頭のゼロが含まれている場合は、ゼロまたは非ゼロの数値セレクターを選択することが重要です。文字列の先頭のゼロのうちいずれかが非ゼロ数値セレクターにマップする場合、そのゼロとすべての後続の先頭のゼロおよびメッセージ文字は、フォーマットされた値の一部	00205 は、2.05 にマップされます。 05 は、05 にマップされます。 56 は、56 にマップされます。 100 は、1.00 にマップされます。 0660 は、6.60 にマップされます。 先頭のゼロは各文字列 **00205** と **0660** から削除されます。これは先頭ゼロがピクチャのゼロ数値セレクターにマップするためです。

ステップ	PICTURE ステートメントの処理のルール	例
	になります。先頭のゼロがすべてゼロ数値セレクターにマップする場合、先頭のゼロまたはメッセージ文字のいずれもフォーマットされた値の一部にはなりません。出力形式は文字列の先頭のゼロをブランクと置き換えます。[2]	
6a		00205 は、2.05 にマップされます。 05 は、05 にマップされます。 55 は、55 にマップされます。 099 は、99 にマップされます。 0660 は、6.60 にマップされます。 先頭のゼロは各文字列 **00205**、**099**、**0660** から削除されます。これは先頭ゼロがピクチャのゼロ数値セレクターにマップするためです。 0.99 ピクチャのピリオド(.)メッセージ文字は削除されます。これは、先頭ゼロがゼロ数値セレクターにマップするためです。 ピリオドメッセージ文字が削除されるため、範囲 0.99 – < 1 の出力形式定義では、NOZEROS.出力形式で 10 進数をフォーマットするために'.'の接頭辞が必要です。
7	PREFIX=オプションで指定される文字に接頭辞を付けます。ピクチャに数値セレクターが含まれている場合、ピクチャは数値セレクターで始まる必要があるため、PREFIX=オプションは必要です。そのため、ピクチャを小数点、マイナス記号、または数値セレクターではない、その他の文字で開始することはできません。	PREFIX=オプションは、フォーマットされた値**-2.05**、**-.05** および**.56** で表わされているように、小数点とマイナス記号を回復します。
7a		PREFIX=オプションは、フォーマットされた値**-2.05**、**-.05**、**.55**、**.99** で表されているように、小数点とマイナス記号を回復します。

1 PREFIX=オプションの小数点は、ピクチャの一部ではありません。
2 FILL=オプションを使用して、フォーマットされた値の一部となるブランク以外の文字を指定できます。

表 26.3　さまざまな値をフォーマットするためのステップ

ステップ	アクション	-2.051	-.05	.556	.996	6.6
1	範囲	low – –1	–0.99 < – < 0	0 < – < .99	0.99 – < 1	1 – high
	ピクチャ	000.00	99	99	0.99	09.99
2	絶対値	2.051	.05	.556	.996	6.6
3	MULT=	2.051×10^{2}=205.1	.05×100=5	.556×100=55.6	.996×100=99.6	6.6×10^{2}=660
4	四捨五入	205	5	56	100	660
4a	四捨五入なし	205	5	55	99	660
5	文字列、四捨五入	00205	05	56	100	0660
5a	文字列、四捨五入なし	00205	05	55	099	0660
6	テンプレート、四捨五入	2.05	05	56	1.00	6.60
6a	テンプレート、四捨五入なし	2.05	05	55	99	6.60
7	接頭辞、四捨五入	prefix='–'	prefix='–.'	prefix='.'	なし	なし
7a	接頭辞、四捨五入なし	prefix='–'	prefix='–.'	prefix='.'	prefix='.'	なし
	フォーマットされた結果、四捨五入	-2.05	–.05	.56	1.00	6.60
	フォーマットされた結果、四捨五入なし	-2.05	–.05	.55	.99	6.60

次の PROC PRINT ステップは、NOZEROSR.出力形式と NOZEROS.出力形式を SAMPLE の AMOUNT 変数と関連付けます。最初の出力には、四捨五入の結果が表示されます。

```
proc print data=sample;
  format amount nozerosr.;
  title 'Formatting the Variable Amount';
  title2 'with the NOZEROSR. Format Using Rounding';
run;

proc print data=sample;
  format amount nozeros.;
  title 'Formatting the Variable Amount';
  title2 'with the NOZEROS. Format, No Rounding';
run;
```

Formatting the Variable Amount with the NOZEROSR. Format Using Rounding

Obs	Amount
1	-45.00
2	-2.05
3	-1.00
4	-.05
5	-.02
6	00
7	.09
8	.54
9	.56
10	1.00
11	6.60
12	14.63

Formatting the Variable Amount with the NOZEROS. Format, No Rounding

Obs	Amount
1	-45.00
2	-2.05
3	-.99
4	-.05
5	-.01
6	00
7	.09
8	.54
9	.55
10	.99
11	6.60
12	14.63

注意
ピクチャには、接頭辞と数字に対して十分な幅がある必要があります。 この例では、値-45.00 に NOZEROS.が適用された場合、最初の範囲 low - -1 内にあり、その範囲のピクチャは接頭辞の付いたマイナス記号と数字を表示するには広さが十分でないため、結果は 45.00 になります。

注意
数字を四捨五入することにより桁が数字に追加される場合、ピクチャが追加される桁に対し十分広い必要があります。 たとえば、数字.996 のピクチャは'99' (prefix '.' mult=100)になります。数字を四捨五入し、それに 100 を掛けると、結果の数字は 100 になります。ピクチャが適用されると、結果は不正確な数字、.00 となります。四捨五入するときに数字が正しいものになるように、ピクチャ幅をより大きな数字を表示するのに十分なものにしてください。

ピクチャなしの指定

この PICTURE ステートメントは、ピクチャを含まない *picture-name* 出力形式を作成します。

picture *picture-name*;

この出力形式を使用すると、デフォルト SAS 出力形式を値に適用する効果があります。

SELECT ステートメント

FMTLIB オプションと CNTLOUT=オプションによって処理するためのエントリを選択します。

制限事項: 1 つの SELECT ステートメントだけを PROC FORMAT ステップに表示できます。
SELECT ステートメントと EXCLUDE ステートメントを同じ PROC FORMAT ステップ内で使用することはできません。

例: "例 15: 入力形式と出力形式の説明の印刷" (967 ページ)

構文

SELECT *entry(s)*;

必須引数

entry(s)
処理対象のカタログエントリを 1 つ以上指定します。カタログエントリの名前は、保存する入力形式または出力形式の名前と同じです。入力形式と出力形式の名前、文字と数値の入力形式と出力形式の名前をそれぞれ同じにすることができるため、SELECT ステートメントで入力形式と出力形式を指定する際は特定の接頭辞を使用する必要があります。SELECT ステートメントでエントリを指定する際は、次の規則に従います。

- ドル記号($)の付いた文字出力形式を含むエントリ名を前に置きます。
- アットマークとドル記号の付いた文字出力形式を含むエントリ名(@$*entry-name* など)を前に置きます。
- アットマーク(@)の付いた数値入力形式を含むエントリ名を前に置きます。
- 接頭辞の付かない数値出力形式を含むエントリ名を指定します。

詳細

名前指定のショートカット

コロン(:)とハイフン(-)ワイルドカード文字を使用して、エントリを選択できます。たとえば、次の SELECT ステートメントは、文字 **a** で始まるすべての出力形式または入力形式を選択します。

```
select a:;
```

また、次の SELECT ステートメントは、**apple** から **pear** まで(両端の値を含む)にアルファベット順に発生するすべての出力形式または入力形式を選択します。

```
select apple-pear;
```

カタログが読み取り/更新モードで開かれる場合の FMTLIB と CNTLOUT=オプションの影響

SELECT ステートメントで FMTLIB オプションと CNTLOUT=オプションを使用し、カタログが読み込みモードまたは更新モードのいずれかで開かれるかを示します。次の規則が適用されます。

- SELECT ステートメントを使用し、FMTLIB オプションまたは CNTLOUT=オプションを指定しない場合、PROC FORMAT はカタログが更新モードで開かれているとみなします。
- SELECT ステートメントを使用し、FMTLIB オプションまたは CNTLOUT=オプションを指定する場合、カタログは読み取りアクセス向けに開かれています。
- SELECT ステートメントを FMTLIB オプションまたは CNTLOUT=オプションを指定せずに使用し、SAS プログラムにカタログへの書き込みアクセス権限がない場合、次のエラーが SAS ログに書き込まれます。

  ```
  ERROR: User does not have appropriate authorization level
      for file libref.FORMATS.CATALOG.
  ```

VALUE ステートメント

変数値の印刷に使用する文字列を指定する出力形式を作成します。

参照項目: SAS 出力形式に関するドキュメントについては、*SAS 出力形式と入力形式: リファレンス*を参照してください。

例: “例 8: 文字値の出力形式の作成” (944 ページ)

“例 11: 標準 SAS 出力形式とカラー背景を使用した、日付の出力形式の書き出し” (951 ページ)
“例 17: 文字列の範囲指定” (971 ページ)

構文

VALUE <$>*name* <(*format-options*)> <*value-range-set(s)*>;

オプション引数の要約

DEFAULT=*length*
出力形式のデフォルト長を指定します。

FUZZ=*fuzz-factor*
値を範囲に一致させるためのあいまい度を指定します。

MAX=*length*
出力形式に対し最大長を指定します。

MIN=*length*
出力形式に対し最小長を指定します。

MULTILABEL
複数のラベルまたは外部値を内部値に割り当てることができます。

NOTSORTED
定義順で値または範囲を保存します。

value-range-set(s)
フォーマットされた値への値または値範囲の割り当てを指定します。

必須引数

name
作成中の出力形式に名前を付けます。出力形式として使用するために FCMP プロシジャを使用して関数を作成した場合、*name* は、かっこのない関数名になります。

制限事項 ユーザー定義の出力形式の名前は、SAS によって提供されている出力形式の名前と同じにすることはできません。

出力形式名の最後を数字にすることはできません。詳細については、“ユーザー定義の出力形式” (*SAS 出力形式と入力形式: リファレンス*)と“SAS 名” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

"_C"と"_B" (どちらも大文字と小文字を区別しない)は、フォーマット名の予約キーです。これらのキーは、フォーマット名の末尾には表示できません。これらの接尾辞を持つフォーマットカタログを SAS 9 から移行すると、エラーが表示されます。回避策は、_C または_B 接尾辞を含まない名前でフォーマットエントリを再作成することです。

要件 名前は有効な SAS 名である必要があります。数値出力形式名の長さは最大 32 文字です。文字出力形式名の長さは最大 31 文字です。文字出力形式を作成している場合、ドル記号($)を最初の文字として使用します。

操作 出力形式名の最大長は、VALIDFMTNAME=システムオプションによって制御されます。詳細については、*SAS システムオプション: リファレンス*を参照してください。

ヒント 名前の後にピリオドを使用して、出力形式を後で参照します。ただし、VALUE ステートメントでは出力形式名の後ろにピリオドを使用しないでください。

参照項目: “関数を使用した値のフォーマッティング” (917 ページ)

オプション引数

DEFAULT=*length*
出力形式のデフォルト長を指定します。 出力形式と変数を関連付ける際に特定の長さを指定しない場合、DEFAULT=の値が出力形式の長さになります。

デフォルト 等号の右側に割り当てられる最も長いラベルの長さ

範囲 1–32767

ヒント 最良の方法として、出力形式をラベルとして指定する場合は、常にDEFAULT=オプションを指定します。

FUZZ=*fuzz-factor*
値を範囲に一致させるためのあいまい度を指定します。 数字が範囲に完全には一致しない、または厳密には範囲内になくても *fuzz-factor* 内にあれば、出力形式はそれを一致とみなします。たとえば、次の VALUE ステートメントでは、LEVELS.出力形式が作成され、そこであいまい度 2 が使用されます。

```
value levels (fuzz=.2) 1='A'
                       2='B'
                       3='C';
```

FUZZ=.2 は、変数値が範囲のどちらかの端の値.2 内にある場合、出力形式が変数値の印刷に対応するフォーマットされた値を使用することを意味します。そのため、LEVELS.出力形式は値 2.1 を **B** としてフォーマットします。

デフォルト 1E−12 (数値出力形式)と 0 (文字出力形式)

ヒント VALUE ステートメントを使用して数値出力形式を作成するとき、FUZZ=0 を指定して記憶域を保存します。

ゼロ以外のあいまい度は、非常に近いが完全には一致しない数値でのみ使用します。バイナリ検索を実行するために、範囲は並べ替え順に内部保存

されます(NOTSORTED オプションが使用されていない場合)。あいまい度が 1 つの範囲の終わりに追加され、次の範囲の先頭から削除れた場合に、これらの範囲が重なると、結果は予測不能なものになることがあります。値は、バイナリ検索で一致した最初の範囲に置かれます。除外演算子は、このバイナリ検索アルゴリズムを無効にするには不十分です。最良の方法として、除外演算子を使用する場合は、FUZZ=0 または NOTSORTED オプションを設定します。

最良の方法は、数値出力形式で<除外演算子を使用する場合は、FUZZ=0 を使用します。

MAX=*length*
出力形式に対し最大長を指定します。 出力形式と変数を関連付ける際に、MAX=値を超える幅を指定できません。

デフォルト 40

範囲 1–32767

MIN=*length*
出力形式に対し最小長を指定します。

デフォルト 1

範囲 1–32767

MULTILABEL
複数のラベルまたは外部値を内部値に割り当てることができます。 次の VALUE ステートメントでは、MULTILABEL オプションの 2 つの使用を示します。最初の VALUE ステートメントは複数のラベルを単一の内部値に割り当てます。複数のラベルは単一の内部値範囲に割り当てることもできます。2 番目の VALUE ステートメントは、ラベルを重複した内部値範囲に割り当てます。MULTILABEL オプションにより、複数のラベルを重複した内部値に割り当てることができます。

```
value one (multilabel)
  1='ONE'
  1='UNO'
  1='UN';

 value agefmt (multilabel)
  15-29='below 30 years'
  30-50='between 30 and 50'
  51-high='over 50 years'
  15-19='15 to 19'
  20-25='20 to 25'
  25-39='25 to 39'
  40-55='40 to 55'
  56-high='56 and above';
```

PROC MEANS、PROC SUMMARY、PROC TABULATE などのマルチラベル有効プロシジャだけが、複数のラベルを使用できます。その他すべてのプロシジャと DATA ステップは、1 次ラベルのみ認識します。

指定エントリに対する 1 次ラベルは、最初の内部値に割り当てられる外部値、またはすべての 内部値が順に並べ替えられるときにそのエントリに一致する、またはそのエントリを含む内部値の範囲です。次に例を示します。

- 最初の VALUE ステートメントでは、1 に対する 1 次ラベルは ONE です。ONE が 1 に割り当てられている最初の外部値であるためです。1 に対する 2 次ラベルは UNO と UN です。
- 2 番目の VALUE ステートメントでは、33 に対する 1 次ラベルは **25 to 39** です。範囲 25–39 が 33 を含む内部値の連続した最初の範囲であるためです。33 に対する 2 次ラベルは **between 30 and 50** です。範囲 30–50 が範囲 25–39 の後に連続して発生するためです。

制限事項　単一の出力形式に対して作成できるラベルの最大数は 255 です。

NOTSORTED
定義順で値または範囲を保存します。 NOTSORTED を指定しない場合、値または範囲はデフォルトで並び替え順序で保存され、SAS では特定の値が含まれる範囲を位置付けるためのバイナリ検索アルゴリズムが使用されます。
NOTSORTED を指定する場合、SAS は各範囲を一致するまで定義順に検索します。

次のうちいずれかが true の場合、NOTSORTED を使用します。

- 発生する特定範囲の尤度がわかっており、処理時間を削減するために、入力形式を使用してそれらの範囲を最初に検索する。
- FMTLIB オプションを使用して出力形式の説明を出力する際に、範囲を定義する順序を保持する。
- ORDER=DATA オプションと PRELOADFMT オプションを使用して PROC MEANS、PROC SUMMARY または PROC TABULATE のクラス変数を分析する際に、範囲を定義する順序を保持する。

値の分布が一様または不明な場合、または値の数が比較的小さい場合は、NOTSORTED を使用しないでください。NOTSORTED が指定されていないときに SAS が使用するバイナリ検索アルゴリズムにより、これらの条件下の検索のパフォーマンスが最適化されます。

CPORT プロシジャと CIMPORT プロシジャを使用して標準照合順序が異なる動作環境間で出力形式を転送する場合、NOTSORTED が自動的に使用されます。NOTSORTED の自動設定は、ASCII と EBCDIC の動作環境間で出力形式を転送する場合に発生します。この状況に問題がある場合、次を行います。

- PROC FORMAT ステートメントの CNTLOUT=オプションを使用し、出力制御データセットを作成します。
- CPORT プロシジャを使用して、制御データセットに対し転送ファイルを作成します。
- 対象の動作環境で CIMPORT プロシジャを使用して、転送ファイルをインポートします。
- 対象の動作環境で、PROC FORMAT を CNTLIN=オプションと使用し、インポートされた制御データセットから出力形式を作成します。

value-range-set(s)
フォーマットされた値への値または値範囲の割り当てを指定します。 *value-range-set(s)*には、次の形式があります。

value-or-range-1 <, *value-or-range-2*, ...>=*formatted-value* | [*existing-format*] | [*functionname*]

等号の左側の変数値は、等号の右側に文字列として印刷します。等号の左側の各 *value-or-range* の最大長は 32,767 文字です。

value-or-range
value-or-range の指定方法については、“値や範囲の指定” (914 ページ)を参照してください。

formatted-value
等号の左側に表示される変数値の印刷値になる文字列を指定します。フォーマットされた値は、文字出力形式または数値出力形式を作成しているかどうかに関係なく、常に文字列です。

フォーマットされた値には、最大 32,767 文字可能です。16 進リテラルの場合、使用できるのは、最大 32,767 入力文字または最大 16,382 表現文字(表現文字ごとに 2 つの 16 進文字)です。ただし、一部のプロシジャではフォーマットされた値の最初の 8 文字または 16 文字だけを使用します。

要件 フォーマットされた値を単一引用符か二重引用符で囲む必要があります。次の例に、二重引用符で囲まれたフォーマットされた値を示します。

```
value $ score
  'M'="Male"
  'F'="Female";
```

フォーマットされた値に単一引用符が含まれている場合、値を二重引用符で囲みます。

```
value sect
  1="Smith's class"
  2="Leung's class";
```

ヒント 数値変数をフォーマットすると、これらの変数の使用は算術演算子に含まれません。SAS では、算術演算に対して保存された値が使用されます。

existing-format
SAS によって提供されている出力形式または既存するユーザー定義の出力形式を指定します。作成している出力形式では既存する出力形式を使用して、等号の左側の *value-or-range* に一致する生データを変換します。

既存する出力形式を使用することは、出力形式をネストすることと考えられます。ネストされたレベルとは、出力形式 B をフォーマットされた値として使用して出力形式 A を作成している場合、プロシジャは A を作成するために既存する出力形式を 1 つだけ使用する必要があるということです。

要件 既存する出力形式を使用する場合、その出力形式名を角かっこ([date9.]など)、またはかっこと縦棒((|date9.|)など)で囲みます。既存する出力形式の名前を単一引用符で囲まないでください。

ヒント 出力形式の複数レベルのネストは避けます。レベルの追加ごとにリソース要件が非常に増加する可能性があります。

最良の方法としては、*value-range-set* に既存の出力形式を指定する場合、常に DEFAULT=オプション (908 ページ)を使用してデフォルト値を指定します。

functionname
SAS によって提供されている関数または既存するユーザー定義の関数を指定します。

ヒント　ベストプラクティスとして、デフォルトの長さを定義する必要があります。

出力形式として使用するために FCMP プロシジャを使用して関数を作成した場合、*name* は、かっこのない関数名になります。

参照項目:　“関数を使用した値のフォーマッティング”

例

例 1: 選択した州の郵便番号を印刷するための出力形式の作成

$STATE.文字出力形式は、選択した州の郵便番号を印刷します。

```
value $state 'Delaware'='DE'
             'Florida'='FL'
             'Ohio'='OH';
```

変数値 **Delaware** は **DE**、変数値 **Florida** は **FL**、変数値 **Ohio** は **OH** と印刷されます。$STATE.出力形式は、ドル記号で始まります。

注: 範囲指定は、大文字と小文字を区別します。上記の$STATE.出力形式では、値 **OHIO** は指定範囲のいずれにも一致しません。データ値が大文字、小文字であるかがわからない場合、1 つの解決策としてデータ値で UPCASE 関数を使用し、範囲にすべての大文字を指定します。

例 2: 数値を文字値として書き出す

数値出力形式 ANSWER.は、値 1 と 2 を yes と no にそれぞれ書き出します。

```
value answer 1='yes'
             2='no';
```

例 3: 範囲なしを指定

この VALUE ステートメントは、範囲が指定されていない *format-name* 出力形式を作成します。

```
value format-name;
```

この出力形式を使用すると、デフォルト SAS 出力形式を値に適用する効果があります。

例 4: ラベルとしての SAS 出力形式を使用する形式の作成

このプログラムでは、MYfmt.形式を作成し、該当する年に基づいて日付がフォーマットされます。

注: format-as-label または function-as-label が角かっこで囲まれている場合、それはネストされたものと呼ばれます。角かっこで囲まれた書式指定の後には、たとえば[year.]のように少なくとも 1 つのピリオドが続きます。角かっこで囲まれた関数の指定の後には、たとえば[year()]のように、かっこが続きます。SAS はネスト形式を使用して範囲内の値をフォーマットします。たとえば、 次の例の low-'31DEC2011'd=[year4.]を参照してください。ネストされた形式を使用する別の例については、“例 5: 既存の SAS フォーマットをラベルとして使用して欠損値と非欠損値のフォーマットを作成する” (913 ページ)を参照してください。

```
data test;
  do Date='01jan2006'd to '31dec2013'd;
    do j=1 to rannor(0)*100;
      output;
    end;
  end;
run;
proc format;
   value MYfmt
       /* Format dates prior to 31DEC2011 using only a year. */
       low-'31DEC2011'd=[year4.]

       /* Format 2012 dates using the month and year. */
       '01jan2012'd-'31DEC12'd=[monyy7.]

       /* Format dates 01JAN2013 and beyond using the day, month, and year. */
       '01JAN2013'd-high=[date9.]

       /* Catch missing values. */
       other='n/a';
run;

proc freq data=test;
     table date /missing;
     format date myfmt.;
run;
```

例 5: 既存の SAS フォーマットをラベルとして使用して欠損値と非欠損値のフォーマットを作成する

このプログラムは、N/A というラベルが付いた欠損数値をフォーマットし、既存の SAS フォーマット 12.1 で非欠損値をフォーマットします。

```
proc format;
     value myfmt .='N/A' other=[12.1];
run;
```

使用法: FORMAT プロシジャ

値や範囲の指定

ステートメント INVALUE、PICTURE、VALUE の構文で示されているように、値を *value-range-sets* として指定する必要があります。等号の左側で、その他の値に変換する値を指定します。等号の右側で、左側の値になる値を指定します。このセクションでは、等号の左側の値を表す *value-or-range* に使用できるさまざまな出力形式について説明します。等号の右側の値を指定する方法については、該当するステートメントの“必須引数”セクションを参照してください。

ステートメント INVALUE、PICTURE、VALUE では、等号の左側の数値を使用できます。文字入力形式では、数値範囲は文字列として扱われます。INVALUE と VALUE では、等号の左側の文字列を使用することもできます。

構文で示されているように、*value-or-range* の複数の発生を *value-range-set* ごとに、それぞれをカンマで区切ることができます。*value-or-range* の発生は、それぞれ次のうちいずれかです。

value
単一値(12、**'CA'**など)。文字出力形式および文字入力形式の場合、文字値を単一引用符で囲みます。

キーワード OTHER=を単一値として使用できます。OTHER=は、その他の値または範囲に一致しないすべての値に一致します。OTHER=には、数値と文字両方のユーザー定義の出力形式の欠損値が含まれます。出力形式が値をフォーマットする関数でない限り、出力形式を OTHER=の値として使用してユーザー定義の出力形式をネストできません。

値を表すには狭すぎる出力形式を指定した場合、SAS は長いラベルを利用可能なスペースに収めようとします。SAS は右側の文字値を切り捨て、数値を BEST*wd* 出力形式に戻す場合があります。出力形式に十分な幅を指定しない場合、SAS は出力にアスタリスクを出力します。この例では、指定された値は DK の 2 文字です。DK の幅は 5 文字の 99999 値には十分でなく、SAS は出力に 2 つのアスタリスクを出力します。

```
proc format;
  value fmtzip 99999="DK";

data testzip;
  zip=27609;
  output;
  zip=99999;
  output;
run;
```

```
proc print data=testzip;
  format Zip fmtzip.;
run;
```

zip=99999 の結果として、出力テーブルの Zip というラベルの付いた列の最初の観測には**が表示されます。

この問題の発生を防ぐひとつの方法は、この例のように、出力形式を適用するときに幅を割り当てることです。

```
proc format;
  value fmtzip 99999="DK";

data testzip;
  zip=27609;
  output;
  zip=99999;
  output;
run;

proc print data=testzip;
  format Zip fmtzip5.;
run;
```

format Zip fmtzip5.;の結果として、27609 が出力テーブルの Zip というラベルの付いた列の最初の観測で正しく表示されるのに十分なスペースが指定されます。

この例のように、出力形式を作成するときに DEFAULT または MIN オプションを指定することでも、この問題を防ぐことができます。

```
proc format;
  value fmtzip (default=5) 99999="DK";

data testzip;
  zip=27609;
  output;
  zip=99999;
  output;
run;

proc print data=testzip;
  format Zip fmtzip5.;
run;
```

値の指定に(default=5)を含めると、27609 が出力テーブルの Zip というラベルの付いた列の最初の観測で正しく表示されるのに十分なスペースが指定されます。

指定された値の末尾に空白を追加すると、値の幅が広がります。たとえば、指定された値 DK は 99999 に対して十分な幅ではありません。DK を指定するときに値に 3 つの空白スペースを追加すると、99999 値に 5 つのスペースが提供されます。

注: 指定した値に空白を追加する場合は、その出力形式で処理する可能性のある最長の値に対応するのに十分な空白を追加してください。

SAS が幅を設定する方法の詳細については、“構文 ” (*SAS 出力形式と入力形式: リファレンス*)を参照してください。

出力形式の値の詳細については、“関数を使用した値のフォーマッティング” (917 ページ)を参照してください。例については、“例 8: 文字値の出力形式の作成” (944 ページ)と“例 20: 出力形式として使用する関数の作成” (980 ページ)を参照してください。

範囲
値のリスト(12–68、**'A'-'Z'**)など)。文字列を含む範囲の場合、各文字列を単一引用符で囲む必要があります。たとえば、A から Z までの文字列を含む範囲を指定する場合、**A** と **Z** を単一引用符で囲んで、**'A'-'Z'**を範囲として指定します。

'A-Z'を指定すると、プロシジャはそれを最初の文字が **A**、2 番目の文字がハイフン(-)、3 番目の文字が **Z** の 3 文字の文字列として解釈します。

数値ユーザー定義入力形式では、プロシジャは *value-range-set* の左側の引用符で囲まない数値範囲を数値範囲として解釈します。文字ユーザー定義入力形式では、プロシジャは **value-range-set** の左側の引用符で囲まない数値範囲を文字列として解釈します。たとえば、文字入力形式では、範囲 **12–86** は**'12'–'86'**と解釈されます。

範囲の 1 つの値として LOW または HIGH を使用できます。範囲 LOW-HIGH を使用してすべての 値を含めることができます。たとえば、次は有効な範囲です。

```
low-'ZZ'
35-high
low-high
other
```

数値範囲で、LOW には、欠損値を除く、最小の数値が含まれます。HIGH には、範囲内で最大値が含まれます。文字範囲では、LOW に欠損値が含まれます。OTHER には、数値形式と文字形式の両方の欠損値が含まれます。

未満(<)記号を使用して、値を範囲から除外できます。範囲の最初の値を除外する場合は、その値の後に<除外演算子を挿入します。範囲の最後の値を除外する場合は、その値の前に<除外演算子を挿入します。たとえば、次の範囲に 0 は含まれません。

```
0<-100
```

同様に、次の範囲に 100 は含まれません。

```
0-<100
```

> ヒント　<除外演算子を使用して範囲に値を挿入する場合、数値出力形式では VALUE ステートメントで FUZZ=0 オプションを使用します。文字出力形式では、FUZZ=0 がデフォルトであるためこの設定は不要です。

範囲の上限の値が別の範囲の下限に表示されていて、<除外演算子を使用しない場合、PROC FORMA はその値を最初の範囲に割り当てます。たとえば、次の範囲では、値 **AJ** は最初の範囲の一部です。

```
'AA'-'AJ'=1 'AJ'-'AZ'=2
```

この例では、値 **AJ** を 2 番目の範囲に含めるため、最初の範囲で<除外演算子を使用します。

```
'AA'-<'AJ'=1 'AJ'-'AZ'=2
```

範囲に値が重複する場合、VALUE ステートメントに MULTILABEL オプションが指定されていない限り、PROC FORMAT はエラーメッセージを返します。たとえば、範囲が次の場合はエラーとなります: 'AA'-'AK'=1 'AJ'-'AZ'=2。

value-or-range にはそれぞれ最大 32,767 文字が可能です。*value-or-range* が 32,767 文字を超える場合、プロシジャは最初の 32,767 文字を処理した後にその値を切り捨てます。

注: 等号の左側のすべての値を表示する必要はありません。これらの値は、デフォルトの入力形式または出力形式を使用して変換されます。たとえば、次の VALUE ステートメントでは、TEMP.出力形式が作成されます。この出力形式では、98.6 のすべての発生が **NORMAL** として印刷されます。

```
value temp 98.6='NORMAL';
```

値が 96.9 の場合、印刷される結果は **96.9** となります。

関数を使用した値のフォーマッティング

SAS では、値で関数を最初に実行することにより、値をフォーマットする方法を提供しています。値をフォーマットする関数を使用して、カスタマイズされた出力形式を作成できます。たとえば、SAS では日付を四半期を使用してフォーマットする YYQ*w*.、YYQ*xw*.、YYQR*w*.、YYQR*xw*.の 4 つの出力形式を提供しています。これらの出力形式のいずれも Q1、Q2、Q3、Q4 を使用する要件を満たしていません。日付と SAS 出力形式に基づいて Q1、Q2、Q3、Q4 値を作成する FCMP プロシジャを使用して関数を作成してから、その関数を FORMAT プロシジャで使用して出力形式を作成できます。

関数を作成し、それを使用して値をフォーマットするためのステップは、次のとおりです。

1. FCMP プロシジャを使用して、関数を作成します。
2. OPTIONS ステートメントを使用して、CMPLIB=システムオプションで関数の場所を指定することにより、関数を利用可能にします。
3. FORMAT プロシジャを使用して、新しい出力形式を作成します。
4. その新しい出力形式を SAS プログラムで使用します。

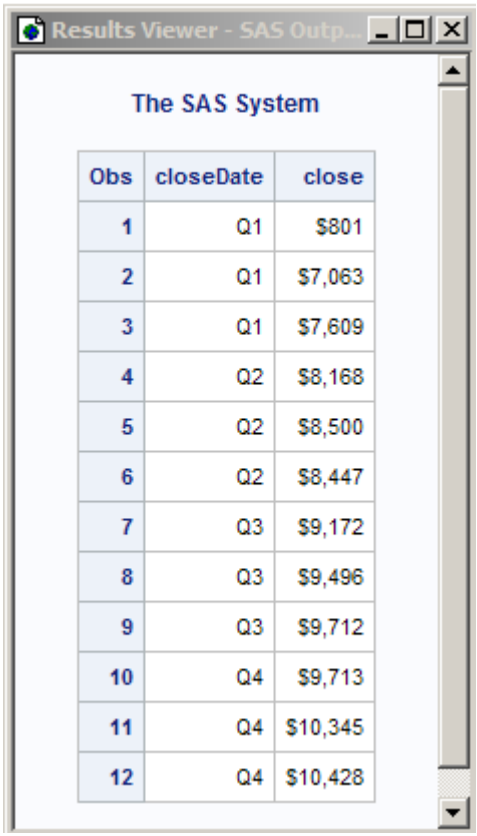
次に例を示します。

```
/* Create a function that creates the value Qx from a formatted value. */

proc fcmp outlib=work.functions.smd;

  function qfmt(date) $;
    length qnum $4;
    qnum=put(date,yyq4.);
    if substr(qnum,3,1)='Q'
      then return(substr(qnum,3,2));
    else return(qnum);
  endsub;
```

```
run;

/* Make the function available to SAS. */

options cmplib=(work.functions);

/* Create a format using the function created by the FCMP procedure. */

proc format;
  value qfmt
       other=[qfmt()]; run;

/* Use the format in a SAS program.  */

data djia2013;
     input closeDate date7. close;
     datalines;
01jan13 800.86
02feb13 7062.93
02mar13 7608.92
01apr13 8168.12
01may13 8500.33
01jun13 8447.00
01jul13 9171.61
03aug13 9496.28
01sep13 9712.28
01oct13 9712.73
02nov13 10344.84
02dec13 10428.05
run;

proc print data=djia2013;
format closedate qfmt. close dollar9.;
run;
```

出力は次のとおりです。

アウトプット 26.1　四半期ごとのダウジョーンズ終値



Obs	closeDate	close
1	Q1	$801
2	Q1	$7,063
3	Q1	$7,609
4	Q2	$8,168
5	Q2	$8,500
6	Q2	$8,447
7	Q3	$9,172
8	Q3	$9,496
9	Q3	$9,712
10	Q4	$9,713
11	Q4	$10,345
12	Q4	$10,428

関数出力形式を OTHER=に対する値として使用できます。

結果: FORMAT プロシジャ

出力制御データセット

出力制御データセットには、入力形式または出力形式を説明する情報が含まれています。出力制御データセットには、多くの使用があります。たとえば、出力制御データセットは、DATA ステップを使用して編集し、値範囲をプログラムで変更できます。または DATA ステップを使用してサブセット化し、新しい出力形式と入力形式を作成できます。また、出力制御データセットを作成し、CPORT プロシジャを使用してデータセットの転送ファイルを作成して、出力形式と入力形式を別の動作環境に移動できます。次に、移動先の動作環境で CIMPORT および FORMAT プロシジャを使用して、その環境に出力形式と入力形式を作成します。

PROC FORMAT ステートメントの CNTLOUT=オプションを使用して、出力制御データセットを作成します。出力制御データット、または出力制御データセットからの

一連のオブザベーションを、CNTLIN=オプションを指定した後続の PROC FORMAT ステップの入力制御データセットとして使用できます。

出力制御データセットには、LIBRARY=カタログの各入力形式または各出力形式の各値または各範囲に対するオブザベーションが含まれています。データセットは、PROC FORMAT ステップで作成された各出力形式と各入力形式に関するグローバル情報、または各範囲と各値に関する特定情報を提供する変数から構成されています。

出力制御データセットの変数は、次のとおりです。

DATATYPE
ピクチャでディレクトリをテンプレートとして使用して、日付値、時間値、日時値をフォーマットできます。

DECP
数字の小数部分に対して区切り文字を指定します。

DEFAULT
出力形式または入力形式のデフォルト長を示す数値変数を指定します。

DIG3SEP
数字に対して 3 桁の区切り文字を指定します。

END
範囲の終了値を提供する文字変数を指定します。

EEXCL
範囲の終了値が除外されるかどうかを示す文字変数を指定します。有効な値は次のとおりです。

Y
範囲の終了値が除外されるように指定します。

N
範囲の終了値が除外されないように指定します。

FILL
ピクチャ出力形式に対し、値が FILL=オプションの値である数値変数を指定します。

FMTNAME
値が出力形式名または入力形式名である文字変数を指定します。

FUZZ
値が FUZZ=オプションの値である数値変数を指定します。

HLO
出力形式または入力形式に関する範囲情報を含む文字変数を指定します。次の有効値は、組み合わせで表示できます。

F
値と使用される標準 SAS 出力形式または入力形式を指定します。

H
範囲の終了値が HIGH になるように指定します。

I
数値入力形式範囲を指定します。

J
入力形式の位置合わせを指定します。

L
範囲の開始値が LOW になるように指定します。

M
MULTILABEL オプションが有効になるように指定します。

N
出力形式または入力形式に範囲が含まれないように指定します。OTHER=範囲なしを含みます。

O
範囲が OTHER になるように指定します。

R
ROUND オプションが有効になるように指定します。

S
NOTSORTED オプションが有効になるように指定します。

U
UPCASE オプションが入力形式に使用されるように指定します。

LABEL
値が出力形式または入力形式と関連付けられている文字変数を指定します。

LANGUAGE
日付、時間、日時のピクチャで置き換えられる平日、月に使用される言語を指定します。サポートされていない、または無効な言語を指定すると、英語が使用されます。

LENGTH
値が LENGTH=オプションの値である数値変数を指定します。

MAX
値が MAX=オプションの値である数値変数を指定します。

MIN
値が MIN=オプションの値である数値変数を指定します。

MULT
値が MULT=オプションの値である数値変数を指定します。

NOEDIT
ピクチャ出力形式に対し、値が NOEDIT オプションが有効であるかどうかを示す数値変数を指定します。有効な値は次のとおりです。

1
NOEDIT オプションが有効になるように指定します。

0
NOEDIT オプションが有効にならないように指定します。

PREFIX
ピクチャ出力形式に対し、値が PREFIX=オプションの値である文字変数を指定します。

SEXCL
範囲の開始値が除外されるかどうかを示す文字変数を指定します。有効な値は次のとおりです。

Y
範囲の開始値が除外されるように指定します。

N
範囲の開始値が除外されないように指定します。

START
範囲の開始値を提供する文字変数を指定します。

TYPE
出力形式の種類を示す文字変数を指定します。可能な値は次のとおりです。

C
文字出力形式を指定します。

I
数値入力形式を指定します。

J
文字入力形式を指定します。

N
数値出力形式(ピクチャを除く)を指定します。

P
ピクチャ出力形式を指定します。

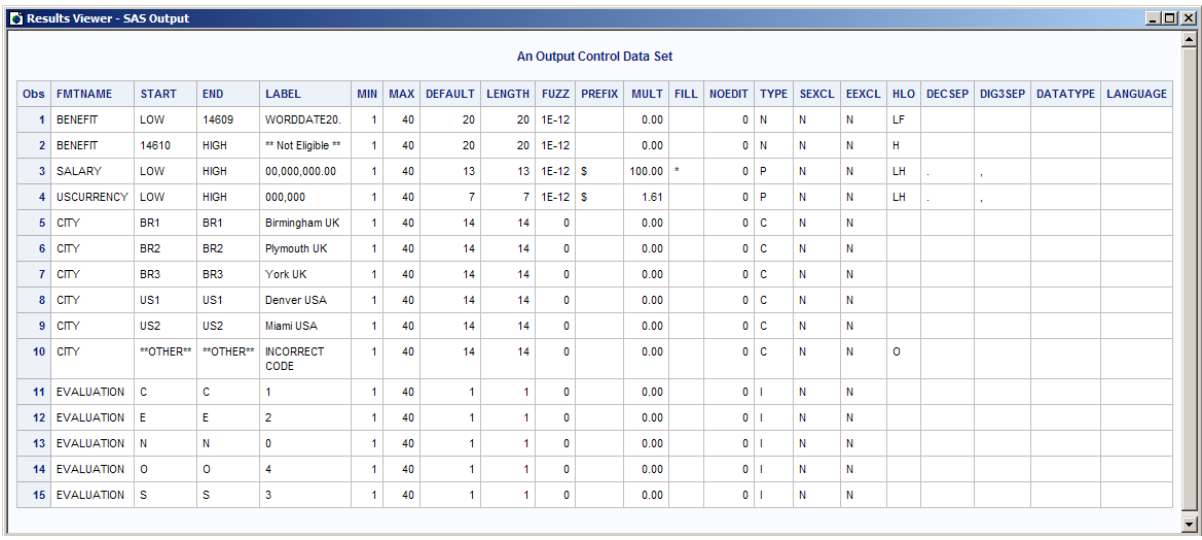
このテーブルは、PROC FORMAT および CNTLIN オプションを使用して出力形式と入力形式を作成するための TYPE 値を指定します。たとえば、シナリオ 1 では、START=Numeric および LABEL=Numeric の場合には TYPE=I であり、INPUT 関数を使用して数値列を作成できます。PROC FORMAT と CNTLIN オプションを使用して出力形式を作成する例については、“例 13: CNTLIN=データセットからの出力形式の作成”を参照してください。

表 26.4 TYPE 値

シナリオ	出力形式/入力形式	START	LABEL	TYPE
1	出力形式	数値	文字	N
2	出力形式	文字	文字	C
3	入力形式	文字	数値	I
4	入力形式	文字	文字	J

次の出力に、FORMAT プロシジャの例で作成されたすべての入力形式と出力形式に関する情報を含む出力制御データセットを示します。

アウトプット 26.2　PROC FORMAT 例の出力制御データセット



Results Viewer - SAS Output

An Output Control Data Set

Obs	FMTNAME	START	END	LABEL	MIN	MAX	DEFAULT	LENGTH	FUZZ	PREFIX	MULT	FILL	NOEDIT	TYPE	SEXCL	EEXCL	HLO	DECSEP	DIG3SEP	DATATYPE	LANGUAGE
1	BENEFIT	LOW	14609	WORDDATE20.	1	40	20	20	1E-12		0.00		0	N	N	N	LF				
2	BENEFIT	14610	HIGH	** Not Eligible **	1	40	20	20	1E-12		0.00		0	N	N	N	H				
3	SALARY	LOW	HIGH	00,000,000.00	1	40	13	13	1E-12	$	100.00	*	0	P	N	N	LH	.	,		
4	USCURRENCY	LOW	HIGH	000,000	1	40	7	7	1E-12	$	1.61		0	P	N	N	LH	.	,		
5	CITY	BR1	BR1	Birmingham UK	1	40	14	14	0		0.00		0	C	N	N					
6	CITY	BR2	BR2	Plymouth UK	1	40	14	14	0		0.00		0	C	N	N					
7	CITY	BR3	BR3	York UK	1	40	14	14	0		0.00		0	C	N	N					
8	CITY	US1	US1	Denver USA	1	40	14	14	0		0.00		0	C	N	N					
9	CITY	US2	US2	Miami USA	1	40	14	14	0		0.00		0	C	N	N					
10	CITY	**OTHER**	**OTHER**	INCORRECT CODE	1	40	14	14	0		0.00		0	C	N	N	O				
11	EVALUATION	C	C	1	1	40	1	1	0		0.00		0	I	N	N					
12	EVALUATION	E	E	2	1	40	1	1	0		0.00		0	I	N	N					
13	EVALUATION	N	N	0	1	40	1	1	0		0.00		0	I	N	N					
14	EVALUATION	O	O	4	1	40	1	1	0		0.00		0	I	N	N					
15	EVALUATION	S	S	3	1	40	1	1	0		0.00		0	I	N	N					

SELECT ステートメントまたは EXCLUDE ステートメントを使用して、出力制御データセットで表される出力形式または入力形式を制御できます。詳細については、“SELECT ステートメント” (905 ページ)および “EXCLUDE ステートメント” (875 ページ)を参照してください。

入力制御データセット

PROC FORMAT ステートメントの CNTLIN=オプションを使用して、入力制御データセットを指定します。FORMAT プロシジャは入力制御データセットのデータを使用して、入力形式と出力形式を作成します。そのため、INVALUE ステートメント、PICTURE ステートメント、VALUE ステートメントを記述せずに入力形式と出力形式を作成できます。

入力制御データセットに、次の特性が含まれている必要があります。

- 数値出力形式と文字出力形式の両方に対し、データセットに変数 FMTNAME、START、LABEL が含まれている必要があります。これらの変数については、“出力制御データセット” (919 ページ)を参照してください。その他の変数は、常に必要であるというわけではありません。
- 文字出力形式と文字入力形式を作成している場合、出力形式名または入力形式名のいずれかをドル記号($)で始めるか、値が **C** の TYPE 変数を指定する必要があります。

  複数のフォーマットが CNTLIN テーブルに定義されていて、それらが異なるタイプ(文字と数値)である場合は、TYPE 列を追加する必要があります。
- PICTURE ステートメント出力形式を作成している場合、値 **P** の TYPE 変数を指定する必要があります。
- 入力値の範囲を含む出力形式を作成する場合、END 変数を指定する必要があります。範囲値が非包含の場合、変数 SEXCL と EEXCL の値はそれぞれ **Y** である必要があります。包含がデフォルトです。

- CNTLIN=オプションを使用して出力形式を作成する場合、START 変数の LOW または OTHER の値と END 変数の HIGH の値は、それらに対応する、LOW、OTHER、または HIGH のキーワードとして解釈されます。この解釈は、値を大文字で指定するか小文字で指定するかに関わりなく発生します。LOW、OTHER、HIGH の明示的な値を使用する場合は、以下で説明する値を持つ HLO 変数を指定します。

  START='LOW'を指定した場合で、HLO 変数に'L'が含まれていない場合、リテラル値 LOW が使用されます。START='OTHER'を指定した場合で、HLO 変数に'O'が含まれていない場合、リテラル値 OTHER が使用されます。END='HIGH'を指定した場合で、HLO 変数に'H'が含まれていない場合、リテラル値 HIGH が使用されます。

各出力形式に対するオブザベーションがグループ化されると、入力制御データセットから複数の出力形式を作成できます。

CNTLIN=オプションが指定されている同じ PROC FORMAT ステップで、VALUE ステートメント、INVALUE ステートメント、または PICTURE ステートメントを使用できます。VALUE ステートメント、INVALUE ステートメント、または PICTURE ステートメントは CNTLIN=オプションが作成しているものと同じ入力形式または出力形式を作成している場合、入力形式または出力形式を作成し、CNTLIN=データセットは使用されません。ただし、VALUE、INVALUE または PICTURE を使用して入力形式または出力形式を作成し、同じ PROC FORMAT ステップで CNTLIN=を使用して異なる入力形式または出力形式を作成できます。

入力制御データセットに関する例については、“例 13: CNTLIN=データセットからの出力形式の作成” (957 ページ)を参照してください。

プロシジャの出力

FORMAT プロシジャは、PROC FORMAT ステートメントで FMTLIB オプションまたは PAGE オプションを指定するときだけ、出力を印刷します。出力は、LIBRARY=オプションで指定されるカタログの各出力形式エントリまたは各入力形式エントリに対するテーブルです。出力には、グローバル情報、出力形式または入力形式に対して定義される各値または範囲に関する詳細も含まれています。SELECT ステートメントまたは EXCLUDE ステートメントを使用して、FMTLIB 出力で表される出力形式と入力形式を制御できます。詳細については、“SELECT ステートメント” (905 ページ)と“EXCLUDE ステートメント” (875 ページ)を参照してください。例については、“例 15: 入力形式と出力形式の説明の印刷” (967 ページ)を参照してください。

次の出力で表示される FMTLIB 出力には、“例 8: 文字値の出力形式の作成” (944 ページ)で作成される$CITY.出力形式と、“例 12: 生の文字データを数値に変換する” (954 ページ)で作成される EVALUATION.入力形式の説明が含まれています。

アウトプット 26.3　FMTLIB オプションによる PROC FORMAT からの出力

FORMAT NAME: $CITY　LENGTH: 14　NUMBER OF VALUES: 6
MIN LENGTH: 1　MAX LENGTH: 40　DEFAULT LENGTH: 14　FUZZ: 0

START	END	LABEL (VER. V7|V8 12OCT2012:11:33:26)
BR1	BR1	Birmingham UK
BR2	BR2	Plymouth UK
BR3	BR3	York UK
US1	US1	Denver USA
US2	US2	Miami USA
OTHER	**OTHER**	INCORRECT CODE

INFORMAT NAME: @EVALUATION LENGTH: 1
MIN LENGTH: 1　MAX LENGTH: 40　DEFAULT LENGTH: 1　FUZZ: 0

START	END	INVALUE(VER. 9.4 12OCT2012:11:34:18)
C	C	1
E	E	2

INFORMAT NAME: @EVALUATION LENGTH: 1
MIN LENGTH: 1　MAX LENGTH: 40　DEFAULT LENGTH: 1　FUZZ: 0

START	END	INVALUE (CONT'D)
N	N	0
O	O	4
S	S	3

フィールドは次に、出力に表示される順序で左から右に説明されます。

INFORMAT NAME または **FORMAT NAME**
入力形式または出力形式の名前です。入力形式名はアットマーク(@)で始まります。

LENGTH
入力形式または出力形式の長さです。PROC FORMAT は長さを次の方法で決定します。

- 文字入力形式の場合、LENGTH の値は、等号の左側の最も長い生データ値の長さです。
- 数値入力形式の場合、次が true です。
  - 等号の左側のすべての値が数値の場合、LENGTH は 12 です。
  - LENGTH は、等号の左側の最も長い生データ値と同じになります。

- 出力形式の場合、LENGTH の値は等号の右側の最も長い値の長さになります。

  $CITY.の出力では LENGTH は 14 です。最も長いピクチャが 14 文字であるためです。

  @EVALUATION.の出力では長さは 1 です。1 が等号の左側の最も長い生データ値であるためです。

NUMBER OF VALUES
入力形式または出力形式と関連付けられている値または範囲の数。NOZEROS.には 4 つの範囲、EVAL.には 5 が設定されています。

MIN LENGTH
入力形式または出力形式の最小長。MIN=オプションで異なる最小長を指定しない限り、MIN LENGTH の値は 1 です。

MAX LENGTH
入力形式または出力形式の最大長。MAX=オプションで異なる最大長を指定しない限り、MAX LENGTH の値は 40 です。

DEFAULT LENGTH
INVALUE フィールドまたは LABEL フィールドの最長値、または DEFAULT=オプションの値。

FUZZ
あいまい度。入力形式の場合、FUZZ は常に 0 です。出力形式の場合、FUZZ=オプションを使用しない場合はこのフィールドの値は STD です。STD はデフォルトの誤差値を表します。

START
範囲の開始値。FMTLIB は、START 列と END 列の値の最初の 16 文字だけを印刷します。

END
範囲の終了値。値が範囲から除外されると、除外記号(<)が START と END の値の後に表示されます。

INVALUE
入力形式に対してのみ表示され、入力形式のある値が含まれます。SAS バージョンでは、入力形式に互換性があるバージョンが指定されます。日付は、入力形式が作成された日付を示します。

注: バージョン番号 V7 | V8 が表示される場合、入力形式はこれらのバージョンと互換性があります。以前のリリースと互換性がない場合、入力形式を作成したリリースが表示されます。バージョン V9 では長い(8 文字を超える)入力形式名をサポートしていますが、V7 | V8 ではサポートしていません。

LABEL
LABEL は出力形式に対してのみ表示され、フォーマットされた値またはピクチャのいずれかが含まれます。SAS バージョンでは、出力形式に互換性があるバージョンが指定されます。日付は、出力形式が作成された日付を示します。

注: バージョン番号 V7 | V8 が表示される場合、出力形式はこれらのバージョンと互換性があります。以前のリリースと互換性がない場合、出力形式を作成したリリースが表示されます。バージョン V9 では長い(8 文字を超える)出力形式名をサポートしていますが、V7 | V8 ではサポートしていません。

NOZEROS.などのピクチャ出力形式の場合、LABEL セクションには PREFIX=、FILL=および MULT=の値が含まれています。これらの値を表記するため、FMTLIB は各オプションを表すための文字 **P**、**F**、および **M** の後に値を印刷します。たとえば、LABEL セクションでは、**P-.**は、接頭辞値はハイフンで、その後にピリオドが続くことを示します。

FMTLIB は、LABEL 列の 40 文字のみ印刷します。

例: FORMAT プロシジャ

例 1: CAS セッションでのフォーマットライブラリの作成

要素: PROC FORMAT ステートメントオプション
CASFMTLIB
CAS ステートメント

詳細

この例では、CASFMTLIB オプションを使用して、CAS セッションでフォーマットライブラリを作成します。ここでは、フォーマットライブラリが WORK ディレクトリ内のテーブルに関連付けられ、CAS Engine の libref が割り当てられます。

プログラム

```
cas casauto sessopts=(caslib="casuser");
caslib _all_ assign;

libname proclib cas;
proc format casfmtlib='myformats';
   value hospx
     1='New_York          '
     2='Massachusetts_General'
     3='Los_Angeles'
     4='Mary_Fletcher';
run;

data clinicalTrial;
```

```
 input hospital treatment $ @@;
 severity=rannor(1323)*5 + 10;
 format hospital hospx.;
 cards;
3 B  3 B  3 C   3 C
1 A  1 A  1 A   1 B
1 B  1 B  1 C   1 C
1 C  1 D  1 D   1 D
2 A  2 A  2 A   2 B
2 B  2 B  2 C   2 C
2 C  2 D  2 D   2 D
3 A  3 A  3 A   3 B
3 C  3 D  3 D   3 D
4 A  4 A  4 A   4 B
4 B  4 B  4 C   4 C
4 C  4 D  4 D   4 D
;

data proclib.clinicalTrial;
      set work.clinicalTrial;
run;

proc regselect data=proclib.clinicalTrial;   class treatment hospital;
  model severity=treatment hospital;
run;
```

プログラムの説明

CAS セッションでフォーマットライブラリを作成します。 LIBNAME ステートメントでライブラリを割り当てます。PROC FORMAT では、*hospx* という名前の出力形式が作成されます。CASFMTLIB オプションでは、CAS セッションのフォーマットライブラリの名前 *myformats* が指定されます。

```
cas casauto sessopts=(caslib="casuser");
caslib _all_ assign;

libname proclib cas;
proc format casfmtlib='myformats';
   value hospx
     1='New_York          '
     2='Massachusetts_General'
     3='Los_Angeles'
     4='Mary_Fletcher';
run;
```

HOSPX 出力形式を列または変数に関連付けます。

```
data clinicalTrial;
 input hospital treatment $ @@;
 severity=rannor(1323)*5 + 10;
 format hospital hospx.;
 cards;
3 B  3 B  3 C   3 C
1 A  1 A  1 A   1 B
1 B  1 B  1 C   1 C
```

```
1 C  1 D  1 D   1 D
2 A  2 A  2 A   2 B
2 B  2 B  2 C   2 C
2 C  2 D  2 D   2 D
3 A  3 A  3 A   3 B
3 C  3 D  3 D   3 D
4 A  4 A  4 A   4 B
4 B  4 B  4 C   4 C
4 C  4 D  4 D   4 D
;
```

CAS セッションにアクションを送信します。 LIBNAME ステートメントでは、REGSELECT プロシジャステップでテーブルを 識別するために使用される CAS Engine の libref が割り当てられます。

```
data proclib.clinicalTrial;
      set work.clinicalTrial;
run;

proc regselect data=proclib.clinicalTrial;   class treatment hospital;
  model severity=treatment hospital;
run;
```

ログ

例のコード 26.1　CAS セッションでのフォーマットライブラリの作成(その 1)

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
 56
 57         cas casauto sessopts=(caslib="casuser");
 NOTE: 'CASUSER(userid)' is now the active caslib.
 NOTE: The CAS statement request to update one or more session options       for session CASAUTO
completed.
 58         caslib _all_ assign;
 NOTE: A SAS Library associated with a caslib can only reference library member
       names that conform to SAS Library naming conventions.
 NOTE: CASLIB CASUSER(userid) for session CASAUTO will be mapped to SAS Library CASUSER.
 NOTE: CASLIB Formats for session CASAUTO will be mapped to SAS Library FORMATS.
 NOTE: CASLIB Models for session CASAUTO will be mapped to SAS Library MODELS.
 NOTE: CASLIB Public for session CASAUTO will be mapped to SAS Library PUBLIC.
 59
 60         libname proclib cas;
 NOTE: Libref PROCLIB was successfully assigned as follows:
       Engine:       CAS
       Physical Name: c0d2aee4-4628-5b42-b740-513df8d08070
 61         proc format casfmtlib='myformats';
 NOTE: Both CAS based formats and catalog-based formats will be written.
       The CAS based formats will be written to the session
       CASAUTO.
 62             value hospx
 63               1='New_York          '
 64               2='Massachusetts_General'
 65               3='Los_Angeles'
 66               4='Mary_Fletcher';
 NOTE: Format library MYFORMATS added.Format search update using parameter
       APPEND completed.
 NOTE: Format HOSPX has been output.
 67         run;

 NOTE: PROCEDURE FORMAT used (Total process time):
       real time          0.04 seconds
       cpu time           0.00 seconds

 68
 69         data clinicalTrial;
 70           input hospital treatment $ @@;
 71           severity=rannor(1323)*5 + 10;
 72           format hospital hospx.;
 73           datalines;

 NOTE: SAS went to a new line when INPUT statement reached past the end
       of a line.
 NOTE: The data set WORK.CLINICALTRIAL has 48 observations and 3 variables.
 NOTE: DATA statement used (Total process time):
       real time          0.00 seconds
       cpu time           0.01 seconds
```

例のコード 26.2　CAS セッションでのフォーマットライブラリの作成(その 2)

```
88        data proclib.clinicalTrial;
89              set work.clinicalTrial;
90        run;

NOTE: There were 48 observations read from the data set WORK.CLINICALTRIAL.
NOTE: The data set PROCLIB.CLINICALTRIAL has 48 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time          0.08 seconds
      cpu time           0.01 seconds

91
92        proc regselect data=proclib.clinicalTrial;
93          class treatment hospital;
94          model severity=treatment hospital;
95        run;

NOTE: The Cloud Analytic Services server processed the request in
      0.065319 seconds.
NOTE: PROCEDURE REGSELECT used (Total process time):
      real time          0.25 seconds
      cpu time           0.10 seconds

96
97
98        OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
```

CAS セッションでのフォーマットライブラリの作成

出力は複数のセクションに分かれていますが、これはドキュメント内での見やすさを考慮しただけです。

The REGSELECT Procedure

Number of Observations Read	48
Number of Observations Used	48

Class Level Information		
Class	Levels	Values
treatment	4	A B C D
hospital	4	Los_Angeles Mary_Fletcher Massachusetts_General New_York

Dimensions	
Number of Effects	3
Number of Parameters	9

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	251.03651	41.83942	2.02	0.0845
Error	41	848.39550	20.69257		
Corrected Total	47	1099.43201			

Root MSE	4.54891
R-Square	0.22833
Adj R-Sq	0.11541
AIC	201.86300
AICC	205.55531
SBC	164.96141
ASE	17.67491

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	13.240102	1.737143	7.62	<.0001
treatment A	1	1.825309	1.857084	0.98	0.3314
treatment B	1	-1.711518	1.857084	-0.92	0.3621
treatment C	1	-2.357254	1.857084	-1.27	0.2115
treatment D	0	0	.	.	.
hospital Los_Angeles	1	-2.844992	1.857084	-1.53	0.1332
hospital Mary_Fletcher	1	-2.539727	1.857084	-1.37	0.1789
hospital Massachusetts_General	1	-4.498280	1.857084	-2.42	0.0199
hospital New_York	0	0	.	.	.

Task Timing		
Task	Seconds	Percent
Setup and Parsing	0.00	6.29%
Levelization	0.01	59.60%
Model Initialization	0.00	0.29%
SSCP Computation	0.00	16.01%
Model Fitting	0.00	1.69%
Cleanup	0.00	16.00%
Total	0.02	100.00%

例 2: サンプルデータセットの作成

詳細

このセクションのいくつかの例では、PROCLIB.STAFF データセットを使用します。また、これらの例で作成される入力形式と出力形式の多くは Library.Formats に保存されます。“出力制御データセット” (919 ページ)で示される出力データセットには、これらの入力形式と出力形式の説明が含まれています。

変数は、アメリカ合衆国とイギリスにサイトのある企業に勤務する従業員の小さなサブセットに関するものです。データには、各従業員に対する名前、ID 番号、給与(イギリスポンド)、場所、雇用日が含まれています。

プログラム

```
libname proclib 'SAS-library';

data proclib.staff;
  infile datalines dlm='#';
  input Name & $16. IdNumber $ Salary
      Site $ HireDate date8.;
  format hiredate date8.;
  datalines;
Capalleti, Jimmy#  2355# 21163# BR1# 30JAN13
Chen, Len#         5889# 20976# BR1# 18JUN06
Davis, Brad#       3878# 19571# BR2# 20MAR04
Leung, Brenda#     4409# 34321# BR2# 18SEP94
Martinez, Maria#   3985# 49056# US2# 10JAN93
Orfali, Philip#    0740# 50092# US2# 16FEB03
Patel, Mary#       2398# 35182# BR3# 02FEB90
Smith, Robert#     5162# 40100# BR5# 15APR06
Sorrell, Joseph#   4421# 38760# US1# 19JUN11
Zook, Carla#       7385# 22988# BR3# 18DEC10
;
```

プログラムの説明

```
libname proclib 'SAS-library';
```

データセット PROCLIB.STAFF を作成します。 INPUT ステートメントは、名前 Name、IdNumber、Salary、Site、HireDate を DATALINES ステートメントの後に表示される変数に割り当てます。FORMAT ステートメントは、標準 SAS 出力形式 DATE7.を変数 HireDate に割り当てます。

```
data proclib.staff;
  infile datalines dlm='#';
  input Name & $16. IdNumber $ Salary
      Site $ HireDate date8.;
  format hiredate date8.;
  datalines;
Capalleti, Jimmy#  2355# 21163# BR1# 30JAN13
Chen, Len#         5889# 20976# BR1# 18JUN06
Davis, Brad#       3878# 19571# BR2# 20MAR04
Leung, Brenda#     4409# 34321# BR2# 18SEP94
Martinez, Maria#   3985# 49056# US2# 10JAN93
Orfali, Philip#    0740# 50092# US2# 16FEB03
Patel, Mary#       2398# 35182# BR3# 02FEB90
Smith, Robert#     5162# 40100# BR5# 15APR06
Sorrell, Joseph#   4421# 38760# US1# 19JUN11
Zook, Carla#       7385# 22988# BR3# 18DEC10
;
```

例 3: ピクチャ出力形式の作成

要素:

PROC FORMAT ステートメントオプション
- LIBRARY=

PICTURE ステートメントオプション
- MULT=
- PREFIX=

LIBRARY ライブラリ参照名

LOW キーワードと HIGH キーワード

データセット: 例 1 の PROCIB.STAFF

詳細

この例では、PICTURE ステートメントを使用して、データセット PROCLIB.STAFF の変数 Salary の値を U.S.ドルで印刷する出力形式を作成します。

プログラム

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';

options nodate pageno=1 linesize=80 pagesize=40;

  proc format library=library;

    picture uscurrency low-high='000,000' (mult=1.61 prefix='$');
  run;
```

```
proc print data=proclib.staff noobs label;

  label salary='Salary in U.S. Dollars';
  format salary uscurrency.;

  title 'PROCLIB.STAFF with a Format for the Variable Salary';
run;
```

プログラムの説明

2 つの SAS ライブラリ参照(PROCLIB と LIBRARY)を割り当てます。 この場合、ライブラリ参照 LIBRARY を割り当てると便利です。PROC FORMAT を使用すると、LIBRARY ライブラリ参照名で参照されるライブラリの入力形式と出力形式が自動的に検索されるためです。

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';
```

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

ユーザー定義の出力形式がカタログ Library.Formats に保存されるように指定します。 LIBRARY=オプションは、PROC FORMAT で作成する出力形式または入力形式を含む SAS カタログを指定します。LIBRARY という名前のライブラリを作成する際、LIBRARY 内に FORMATS という名前のカタログが自動的に作成されます。

```
proc format library=library;
```

USCURRENCY.ピクチャ出力形式を定義します。 PICTURE ステートメントは、数字を印刷するためのテンプレートを作成します。LOW-HIGH によりすべての 値が範囲に含まれます。MULT=ステートメントオプションは、各値に 1.61 を掛けるように指定します。PREFIX=ステートメントは、US ドル記号をフォーマットする数字に追加します。ピクチャには数値セレクターが給与に 5 つ、ドル記号接頭辞に 1 つ、合計 6 つ含まれています。

```
  picture uscurrency low-high='000,000' (mult=1.61 prefix='$');
run;
```

PROCLIB.STAFF データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。LABEL オプションは列ヘッダーの変数名ではなく、変数ラベルを使用します。

```
proc print data=proclib.staff noobs label;
```

Salary 変数に対しラベルと出力形式を指定します。 LABEL ステートメントは、レポートの変数に対する特定のラベルを置き換えます。この場合、“Salary in US Dollars”はこの印刷ジョブに対してのみ変数 Salary に置き換えられます。FORMAT ステートメントは、このプロシジャステップの期間に対し USCurrency.出力形式と変数名 Salary を関連付けます。

```
label salary='Salary in U.S. Dollars';
format salary uscurrency.;
```

タイトルを指定します。

```
  title 'PROCLIB.STAFF with a Format for the Variable Salary';
run;
```

出力

アウトプット 26.4　変数 Salary の出力形式を含む PROCLIB.STAFF

PROCLIB.STAFF with a Format for the Variable Salary

Name	IdNumber	Salary in U.S. Dollars	Site	HireDate
Capalleti, Jimmy	2355	$34,072	BR1	30JAN09
Chen, Len	5889	$33,771	BR1	18JUN06
Davis, Brad	3878	$31,509	BR2	20MAR04
Leung, Brenda	4409	$55,256	BR2	18SEP94
Martinez, Maria	3985	$78,980	US2	10JAN93
Orfali, Philip	0740	$80,648	US2	16FEB03
Patel, Mary	2398	$56,643	BR3	02FEB90
Smith, Robert	5162	$64,561	BR5	15APR06
Sorrell, Joseph	4421	$62,403	US1	19JUN11
Zook, Carla	7385	$37,010	BR3	18DEC10

例 4: 大きなドル金額のピクチャ出力形式の作成

要素: PICTURE ステートメントオプション
MULT

出力形式: BIGMONEY.

詳細

この例では、PICTURE ステートメントの MULT オプションを使用して、それぞれ 100 万ドル、億ドル、または兆ドルを示す M、B、T を表示するドルをフォーマットします。この例では、出力形式定義で指数表記と小数点表記を使用します。

ヒント　セントを含まないドル金額を使用するため、四捨五入は不要です。ドル金額にセントが含まれる場合は、PICTURE ステートメントで ROUND オプションを使用して金額を四捨五入します。詳細については、“ROUND” (891 ページ)を参照してください。

プログラム

```
proc format;
  picture bigmoney (fuzz=0)
    1E06-<1000000000='0000 M' (prefix='$' mult=.000001)
    1E09-<1000000000000='0000 B' (prefix='$' mult=1E-09)
    1E12-<1000000000000000='0000 T' (prefix='$' mult=1E-012);
run;

data mult;
  do i=5 to 12;
    x=16**i;
    put x=comma20. x= bigmoney.;
  end;
run;
```

プログラムの説明

BIGMONEY 出力形式を作成します。 BIGMONEY.出力形式では、100 万ドル、億ドル、兆ドルをフォーマットする 3 つの value-range sets を定義します。1E06 は 100 万、1E09 は 1 億、1E12 は 1 兆です。<除外演算子で、範囲に後続の数値を含めるかどうかを指定します。除外演算子を使用して正確な数値になるようには、FUZZ=0 を使用するのが最もよい方法です。100 万ドルの場合、範囲は 1,000,000 から 999,999,999 です。等号の右側で指定されるラベルでは、4 つのゼロと数値セレクターを使用します。ゼロの数値セレクターでは、先頭のゼロを印刷しないことを指定します。最初の数値セレクターは、値が 3 桁の場合に$接頭辞記号を印刷するために必要です。MULT=オプションの値を.000001 に設定して、1E-06 (100 万分の 1)を作成することもできます。値に、100 万分の 1、1 億分の 1、1 兆分の 1 の乗数を掛けると、100 万ドル、億ドル、兆ドルの数値が返されます。

```
proc format;
  picture bigmoney (fuzz=0)
    1E06-<1000000000='0000 M' (prefix='$' mult=.000001)
    1E09-<1000000000000='0000 B' (prefix='$' mult=1E-09)
    1E12-<1000000000000000='0000 T' (prefix='$' mult=1E-012);
run;
```

ドルとしてフォーマットする大きな数字を生成します。

```
data mult;
  do i=5 to 12;
    x=16**i;
    put x=comma20. x= bigmoney.;
  end;
run;
```

ログ

例のコード 26.3　フォーマットされた 100 万ドル、10 億ドル、兆ドル金額

```
x=1,048,576 x=$1 M
x=16,777,216 x=$16 M
x=268,435,456 x=$268 M
x=4,294,967,296 x=$4 B
x=68,719,476,736 x=$68 B
x=1,099,511,627,776 x=$1 T
x=17,592,186,044,416 x=$17 T
x=281,474,976,710,656 x=$281 T
```

プログラム

```
proc format;
  picture bigmoney (fuzz=0)
    1E06-<1000000000='0000.99 M' (prefix='$' mult=.0001)
    1E09-<1000000000000='0000.99 B' (prefix='$' mult=1E-07)
    1E12-<1000000000000000='0000.99 T' (prefix='$' mult=1E-010);
run;

data mult;
  do i=5 to 12;
    x=16**i;
    put x=comma20. x= bigmoney.;
  end;
run;
```

プログラムの説明

このプログラムでは、BIGMONEY.出力形式を変更し、小数点値を追加してより精度の高い数値を表示します。

BIGMONEY 出力形式を変更します。 より精度の高い数値を表示するには、ピクチャ値および MULT=値を変更します。2 桁の小数点値を表示するには、ピクチャに.99 を追加します。 2 桁の小数点値を計算するには、MULT=オプションの値を 100 万分の 1 から 1 万分の 1 に減らします。16^5 に.0001 を掛けると、結果は 104.8576 になります。小数点値は切り捨てられ、104 がピクチャの右側から配置されます。 フォーマットされた結果の値は 1.04 M です。

```
proc format;
  picture bigmoney (fuzz=0)
    1E06-<1000000000='0000.99 M' (prefix='$' mult=.0001)
    1E09-<1000000000000='0000.99 B' (prefix='$' mult=1E-07)
    1E12-<1000000000000000='0000.99 T' (prefix='$' mult=1E-010);
run;
```

ドルとしてフォーマットする大きな数字を生成します。

```
data mult;
  do i=5 to 12;
    x=16**i;
```

```
    put x=comma20. x= bigmoney.;
  end;
run;
```

ログ

例のコード 26.4　より高い精度でフォーマットされた大きなドル金額

```
x=1,048,576 x=$1.04 M
x=16,777,216 x=$16.77 M
x=268,435,456 x=$268.43 M
x=4,294,967,296 x=$4.29 B
x=68,719,476,736 x=$68.71 B
x=1,099,511,627,776 x=$1.09 T
x=17,592,186,044,416 x=$17.59 T
x=281,474,976,710,656 x=$281.47 T
```

例 5: ピクチャ出力形式の埋め込み

要素:

PICTURE ステートメントオプション
FILL=
PREFIX=

詳細

この例では、次のタスクを実行します。

- フォーマットされた値への指定文字の接頭辞指定
- 先頭のブランクへの指定文字の入力
- FILL=オプションと PREFIX=オプションの間の相互作用の表示

プログラム

```
data pay;
  input Name $ MonthlySalary;
  datalines;
Liu   1259.45
Lars  1289.33
Kim   1439.02
Wendy 1675.21
Alex  1623.73
;
```

```
proc format;
  picture salary low-high='00,000,000.00' (fill='*' prefix='$');
run;

proc print data=pay noobs;
  format monthlysalary salary.;

  title 'Printing Salaries for a Check';
run;
```

プログラムの説明

PAY データセットを作成します。 PAY データセットには、各従業員に対する月次給与が含まれます。

```
data pay;
  input Name $ MonthlySalary;
  datalines;
Liu   1259.45
Lars  1289.33
Kim   1439.02
Wendy 1675.21
Alex  1623.73
;
```

SALARY.ピクチャ出力形式を定義し、ピクチャの入力方法を指定します。 FILL=および PREFIX= PICTURE ステートメントオプションが同一のピクチャに表示されるとき、出力形式は接頭辞を置いてから、埋め字を入力します。SALARY.出力形式はピクチャに埋め字を入力しますが、これはピクチャに数値セレクターとしてゼロが含まれているためです。ピクチャの最左のカンマは埋め字と置き換えられます。

```
proc format;
  picture salary low-high='00,000,000.00' (fill='*' prefix='$');
run;
```

PAY データセットを印刷します。 NOOBS オプションはオブザベーション番号の印刷を行いません。FORMAT ステートメントは、SALARY.出力形式と変数 MonthlySalary を一時的に関連付けます。

```
proc print data=pay noobs;
  format monthlysalary salary.;
```

タイトルを指定します。

```
  title 'Printing Salaries for a Check';
run;
```

出力

アウトプット 26.5　給与の確認用印刷

Printing Salaries for a Check

Name	Monthly Salary
Liu	****$1,259.45
Lars	****$1,289.33
Kim	****$1,439.02
Wendy	****$1,675.21
Alex	****$1,623.73

例 6: ディレクティブを 使用して日付または時刻の形式を作成する

要素:　PICTURE ステートメントオプション
　DATATYPE=TIME
ディレクティブ

詳細

この例では、ディレクティブを 使用して日付、時刻、および日時の値をフォーマットします。また、複数の value-range ペアにディレクティブを 使用します。

プログラム

```
proc format;
  picture mytime  (round)
    low-<86400='%H hours, %M minutes'        (datatype=time)  1
    86400-high='%n days, %H hours, %M minutes' (datatype=time) ; 2
    /*  86400=number of seconds in one day */
run;

data test;
  input xtime;
  newtime=put(xtime,mytime.);
datalines;
```

```
12345
46987
86400
99999
172800
1012345
3333333
;

run;

proc print data=test;
run;
```

プログラムの説明

キーワード LOW、HIGH、および LOW–HIGH を指定してディレクティブを 使用します。 これらのディレクティブは、日付、時刻、および日時の値をフォーマットします。

```
proc format;
  picture mytime  (round)
    low-<86400='%H hours, %M minutes'         (datatype=time)   1
    86400-high='%n days, %H hours, %M minutes' (datatype=time) ; 2
    /*  86400=number of seconds in one day */
run;
```

1 %H および%M ディレクティブでは、出力の中で時間と分が識別されることを指定します。DATATYPE オプションを使用すると、ピクチャでディレクティブを 使用できます。

2 %n、%H、%M ディレクティブは、出力の中で日、時間、および分が識別されることを指定します。

データセットを作成します。

```
data test;
  input xtime;
  newtime=put(xtime,mytime.);
datalines;
12345
46987
86400
99999
172800
1012345
3333333
;

run;
```

データセットのコンテンツが印刷されます。

```
proc print data=test;
run;
```

アウトプット 26.6　フォーマットされた値を含むテストデータセット

The SAS System

Obs	xtime	newtime
1	12345	3 hours, 25 minutes
2	46987	13 hours, 3 minutes
3	86400	1 days, 0 hours, 0 minutes
4	99999	1 days, 3 hours, 46 minutes
5	172800	2 days, 0 hours, 0 minutes
6	1012345	11 days, 17 hours, 12 minutes
7	3333333	38 days, 13 hours, 55 minutes

例 7: 24 時間形式から 00:00:01–24:00:00 形式への変更

要素:　PICTURE ステートメントオプション
　　DATATYPE=DATETIME_UTIL

詳細

場合によっては、午前 0 時を 24:00 と表現することが必要になったり、日時の時間範囲 00:00:01–24:00:00 を使用することが必要になったりします。DATATYPE=DATETIME と設定した場合の時間値の範囲は 00:00:00 から 23:59:59 です。この例では、オプション DATATYPE=DATETIME_UTIL を使用して 00:00:01 から 24:00:00 の範囲で時間を表現し、00:00:00 を使用した場合日付が変更される様子を示します。

プログラム

```
proc format;
  picture  hours (default=19)
        other='%Y-%0m-%0d %0H:%0M:%0S' (datatype=datetime_util);
run;

data _null_;
  x = '01jul2015:00:00:01'dt; put x=hours.;
  x = '01jul2015:00:00:00'dt; put x=hours.;
run;
```

プログラムの説明

DATATYPE=DATETIME_UTIL オプションを設定して、時間範囲 00:00:01–24:00:00 を使用します。

```
proc format;
  picture  hours (default=19)
        other='%Y-%0m-%0d %0H:%0M:%0S' (datatype=datetime_util);
run;
```

日付の値を比較します。 最初の日時値は 00:00:01 の範囲内にあり、日付は July 1 と示されます。2 番目の日時値は 00:00:01 から 24:00:00 の範囲内にないため、前の日の午前 0 時として結果が表示されます。

```
data _null_;
  x = '01jul2015:00:00:01'dt; put x=hours.;
  x = '01jul2015:00:00:00'dt; put x=hours.;
run;
```

ログ

例のコード 26.5　日付範囲 00:00:01–24:00:00 の使用

```
x=2015-07-01 00:01:00
x=2015-06-30 24:00:00
```

例 8: 文字値の出力形式の作成

要素:	VALUE ステートメントオプション: OTHER
データセット:	PROCLIB.STAFF
出力形式:	例 2 の USCURRENCY.

詳細

この例では、VALUE ステートメントを使用して、文字変数の値を異なる文字列として印刷する文字出力形式を作成します。

プログラム

```
libname proclib 'SAS-library-1';
```

```
libname library 'SAS-library-2';


  proc format library=library;
    value  $city 'BR1'='Birmingham UK'
                 'BR2'='Plymouth UK'
                 'BR3'='York UK'
                 'US1'='Denver USA'
                 'US2'='Miami USA'
                 other='INCORRECT CODE';
  run;
proc print data=proclib.staff noobs label;
  label salary='Salary in U.S. Dollars';
  format salary uscurrency. site $city.;
  title 'PROCLIB.STAFF with a Format for the Variables';
  title2 'Salary and Site';
run;
```

プログラムの説明

2 つの SAS ライブラリ参照(PROCLIB と LIBRARY)を割り当てます。 この場合、ライブラリ参照 LIBRARY を割り当てると便利です。PROC FORMAT を使用すると、LIBRARY ライブラリ参照名で参照されるライブラリの入力形式と出力形式が自動的に検索されるためです。

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';
```

ユーザー定義の出力形式が保存される Library.Formats という名前のカタログを作成します。 LIBRARY=オプションは作成する出力形式に対し永久保存場所を指定します。指定されたライブラリに FORMAT という名前のカタログも作成します。LIBRARY=を使用しない場合、作成する出力形式と入力形式は Work.Formats というカタログに一時的に保存されます。

```
  proc format library=library;
```

$CITY.出力形式を定義します。 特殊コード BR1、BR2 などは、対応する市の名前に変換されます。キーワード OTHER は、リストされている市コード値のいずれにも一致しないデータセットの値が値 INCORRECT CODE に変換されるように指定します。

```
    value  $city 'BR1'='Birmingham UK'
                 'BR2'='Plymouth UK'
                 'BR3'='York UK'
                 'US1'='Denver USA'
                 'US2'='Miami USA'
                 other='INCORRECT CODE';
  run;
```

PROCLIB.STAFF データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。LABEL オプションは列ヘッダーの変数名ではなく、変数ラベルを使用します。

```
proc print data=proclib.staff noobs label;
```

Salary 変数に対しラベルを指定します。 LABEL ステートメントは、名前 SALARY にラベル“Salary in U.S. Dollars”を代入します。

```
label salary='Salary in U.S. Dollars';
```

Salary と Site に対し出力形式を指定します。 FORMAT ステートメントは、USCURRENCY.出力形式を変数 SALARY に一時的に関連付け、また出力形式$CITY.を変数 SITE に一時的に関連付けます。

```
format salary uscurrency. site $city.;
```

タイトルを指定します。

```
title 'PROCLIB.STAFF with a Format for the Variables';
title2 'Salary and Site';
run;
```

出力

アウトプット 26.7　Salary と Site に対してフォーマットされた変数を含む PROCLIB.STAFF

PROCLIB.STAFF with a Format for the Variables
Salary and Site

Name	IdNumber	Salary in U.S. Dollars	Site	HireDate
Capalleti, Jimmy	2355	$34,072	Birmingham UK	30JAN09
Chen, Len	5889	$33,771	Birmingham UK	18JUN06
Davis, Brad	3878	$31,509	Plymouth UK	20MAR04
Leung, Brenda	4409	$55,256	Plymouth UK	18SEP94
Martinez, Maria	3985	$78,980	Miami USA	10JAN93
Orfali, Philip	0740	$80,648	Miami USA	16FEB03
Patel, Mary	2398	$56,643	York UK	02FEB90
Smith, Robert	5162	$64,561	INCORRECT CODE	15APR06
Sorrell, Joseph	4421	$62,403	Denver USA	19JUN11
Zook, Carla	7385	$37,010	York UK	18DEC10

例 9: 欠損した変数値と欠損していない変数値の出力形式の作成

要素: VALUE ステートメント
VALUE ステートメントオプション
OTHER

データセット: EDUCATION

詳細

EDUCATION データセットは、複数の州に対して中退率と数学のスコアをレポートし、州ごとに地域を示します。

この例では、VALUE ステートメントを使用して、計算スコアの欠損値すべてに対してテキスト値 n/a が作成されます。欠損していない数学のスコア値はすべて、5.1 出力形式を使用してフォーマットされます。

次の例では、地域ごとに各州の中退率と数学のスコアが出力されます。

プログラム

```
options obs=20;


proc format;
  value myfmt .='n/a' other=[5.1];
run;

proc sort data=education;
  by region;
run;

proc print data=education;
  by region;
  var state dropOutRate mathScore;
  format mathScore myfmt.;
run;
```

プログラムの説明

出力するオブザベーションの数を設定します。

```
options obs=20;
```

Mathscore 変数値の出力形式を作成します。 VALUE ステートメントを使用して、Mathscore 変数用の出力形式 MYFMT.を作成します。プログラムで欠損した Mathscore 値が検出されると、その値は n/a としてフォーマットされます。Mathscore のその他のすべての値は 5.1 出力形式を使用してフォーマットされます。

```
proc format;
  value myfmt .='n/a' other=[5.1];
run;
```

データを並べ替えて出力します。 PROC SORT を使用して、地域ごとにデータセットを並べ替えます。地域ごとにデータを出力するには、PROC PRINT BY ステートメントで region 変数を指定します。州、中退率、および数学のスコアをレポートするには、VAR ステートメントを使用して、state、dropOutRate、mathScore の各変数を指定します。最後に、FORMAT ステートメントを使用して、MYFMT.出力形式で mathScore 変数をフォーマットするように SAS に指示します。

```
proc sort data=education;
  by region;
run;

proc print data=education;
  by region;
  var state dropOutRate mathScore;
  format mathScore myfmt.;
run;
```

出力

アウトプット 26.8　地域内の州ごとの中退率と数学のスコア

The SAS System

Region=MW

Obs	State	DropoutRate	Math Score
1	Illinois	21.5	260.0
2	Indiana	13.8	267.0
3	Iowa	13.6	278.0
4	Kansas	17.9	n/a

Region=NE

Obs	State	DropoutRate	Math Score
5	Connecticut	16.8	270.0
6	Delaware	28.5	261.0
7	Maine	22.5	n/a
8	Maryland	26.0	260.0

Region=SE

Obs	State	DropoutRate	Math Score
9	Alabama	22.3	252.0
10	Arkansas	11.5	256.0
11	Florida	38.5	255.0
12	Georgia	27.9	258.0
13	Kentucky	32.7	256.0
14	Louisiana	43.1	246.0

Region=W

Obs	State	DropoutRate	Math Score
15	Alaska	35.8	n/a
16	Arizona	31.2	259.0
17	California	32.7	256.0
18	Colorado	24.7	267.0
19	Hawaii	18.3	251.0
20	Idaho	21.8	272.0

例 10: Perl 正規表現を使用した入力形式の作成

要素: INVALUE ステートメントオプション
REGEXP
REGEXPE

詳細

この例では、2 つの Perl 正規表現を使用して入力形式を作成します。最初の表現を使用した入力形式は、入力が整数であることを確認して、その整数を読み取ります。2 番目の入力形式は、置換を呼び出す正規表現を使用して、入力値とは異なる数を読み取ります。

プログラム

```
proc format;
  invalue isnum (default=5) '/[0-9]/' (regexp) = _same_ other=_error_;
  invalue x1to2x(default=5) 's/1/2/' (regexpe) = _same_ other=_same_;
run;


data _null_;
  input x:isnum. y:x1to2x.;
  put x= y=;
  datalines;
1 121
2 145
a 232
run;
```

プログラムの説明

新しい入力形式を作成します。 入力値が 10 進整数の場合、ISNUM.入力形式はその数を読み取ります。それ以外の場合は、SAS でログにエラーが書き込まれます。X1TO2X.入力形式は、入力値のすべての 1 を 2 に置換します。

```
proc format;
  invalue isnum (default=5) '/[0-9]/' (regexp) = _same_ other=_error_;
  invalue x1to2x(default=5) 's/1/2/' (regexpe) = _same_ other=_same_;
run;
```

データを読み取ります。 データの最初の 2 行は有効です。1 が 2 に置換されるため、最初の入力値 121 は 222 としてフォーマットされます。入力値の 145 は、同じ置換ルールを使用して 245 としてフォーマットされます。3 行目は、x の値が文字であるため、エラーが発生します。

```
data _null_;
  input x:isnum. y:x1to2x.;
  put x= y=;
  datalines;
1 121
2 145
a 232
run;
```

ログ

```
1    proc format;
2         invalue isnum (default=5) '/[0-9]/' (regexp) = _same_ other=_error_;
NOTE: Informat ISNUM has been output.
3         invalue x1to2x(default=5) 's/1/2/' (regexpe) = _same_ other=_same_;
NOTE: Informat X1TO2X has been output.
4         run;

NOTE: PROCEDURE FORMAT used (Total process time):
      real time           0.32 seconds
      cpu time            0.07 seconds


5
6    data _null_;
7         input x:isnum. y:x1to2x.;
8         put x= y=;
9         datalines;

x=1 y=222
x=2 y=245
NOTE: Invalid data for x in line 12 1-1.
x=. y=232
RULE:      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+--
12         a 232
x=. y=232 _ERROR_=1 _N_=3
```

例 11: 標準 SAS 出力形式とカラー背景を使用した、日付の出力形式の書き出し

要素: VALUE ステートメントオプション
HIGH
PROC PRINT ステートメント
VAR ステートメントの STYLE オプション

データセット: PROCLIB.STAFF

出力形式: 例 3 の USCURRENCY.

出力形式: 例 7 の$CITY.

詳細

この例では、フォーマットされた値として SAS によって提供される既存の出力形式と、日付に基づくカラーコード値を使用します。

タスクには、次が含まれます。

- 数値出力形式の作成
- 出力形式のネスト
- 標準 SAS 出力形式を使用した出力形式の書き出し
- カラースキームを使用した日付のフォーマット

プログラム

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';

proc format library=library;

  value benefit
    low-'31DEC2008'd=[worddate20.]
    '01JAN2009'd-high='  ** Not Eligible **';

  value color
    low-'31DEC2008'd='light green'
    '01JAN2009'd-high='light red';
run;

proc print data=proclib.staff noobs label;
  var name idnumber salary site;
  var hiredate /style=[background=color.];

  label salary='Salary in U.S. Dollars';

  format salary uscurrency. site $city. hiredate benefit.;

  title 'PROCLIB.STAFF with a Format for the Variables';
  title2 'Salary, Site, and HireDate';
run;
```

プログラムの説明

このプログラムでは、BENEFIT.という出力形式を定義します。これにより 31DEC2008 以前に雇用された従業員を区別します。このプログラムの目的は、福利厚生を受ける資格のある従業員を 2008 年 12 月 31 日以前の雇用日に基づいて示す

ことです。雇用日がそれより後のその他すべての従業員は福利厚生の資格なしとしてリストされます。

2 つの SAS ライブラリ参照(PROCLIB と LIBRARY)を割り当てます。 この場合、ライブラリ参照 LIBRARY を割り当てると便利です。PROC FORMAT を使用すると、LIBRARY ライブラリ参照名で参照されるライブラリの入力形式と出力形式が自動的に検索されるためです。

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';
```

BENEFIT.出力形式をカタログ LIBRARY.FORMATS に保存します。 LIBRARY=オプションは作成する出力形式に対し永久保存場所 LIBRARY を指定します。ILIBRARY=を使用しない場合、作成する出力形式と入力形式は Work.Formats というカタログに一時的に保存されます。

```
proc format library=library;
```

BENEFIT.出力形式の最初の範囲を定義します。 最初の範囲では、31DEC2008 以前に雇用された従業員とその日より後に雇用された従業員を区別します。キーワード LOW と SAS 日付定数'31DEC2008'd は、2008 年 12 月 31 日以前に発生するすべての日付値を含む最初の範囲を作成します。この範囲内の値に対し、WORDDATE*w*.出力形式が適用されます。SAS 日付定数の詳細については、“日付、時間、および間隔” (*SAS プログラマガイド: エッセンシャル*)を参照してください。WORDDATE 出力形式の詳細については、“WORDDATEw. 出力形式” (*SAS 出力形式と入力形式: リファレンス*)を参照してください。

```
  value benefit
    low-'31DEC2008'd=[worddate20.]
    '01JAN2009'd-high='  ** Not Eligible **';
```

範囲の色を定義します。 同じ日付範囲を使用して、日付に基づいて福利厚生を受ける資格のある従業員が、薄緑でカラーコーディングされます。福利厚生を受ける資格のない従業員は、明るい赤でカラーコーディングされます。

```
  value color
    low-'31DEC2008'd='light green'
    '01JAN2009'd-high='light red';
run;
```

PROCLIB.STAFF データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。LABEL オプションは列ヘッダーの変数名ではなく、変数ラベルを使用します。VAR ステートメントで、印刷する変数を指定します。2 番目の VAR ステートメントでは、STYLE=オプションを使用して、Hiredate 変数の背景色として color.出力形式を指定します。

```
proc print data=proclib.staff noobs label;
  var name idnumber salary site;
  var hiredate /style=[background=color.];
```

Salary 変数に対しラベルを指定します。 LABEL ステートメントは、名前 SALARY にラベル“Salary in U.S. Dollars”を代入します。

```
  label salary='Salary in U.S. Dollars';
```

Salary、Site、Hiredate に対し出力形式を指定します。 FORMAT ステートメントは、USCURRENCY.出力形式(“例 3: ピクチャ出力形式の作成” (934 ページ)で作成)と SALARY、$CITY.出力形式(“例 8: 文字値の出力形式の作成” (944 ページ)で作成)と SITE、BENEFIT.出力形式と HIREDATE を関連付けます。

```
    format salary uscurrency. site $city. hiredate benefit.;
```

タイトルを指定します。

```
    title 'PROCLIB.STAFF with a Format for the Variables';
    title2 'Salary, Site, and HireDate';
  run;
```

出力

アウトプット 26.9　変数 Salary、Site、HireDate に対する出力形式を含む PROCLIB.STAFF

PROCLIB.STAFF with a Format for the Variables Salary, Site, and HireDate

Name	IdNumber	Salary in U.S. Dollars	Site	HireDate
Capalleti, Jimmy	2355	$34,072	Birmingham UK	** Not Eligible **
Chen, Len	5889	$33,771	Birmingham UK	June 18, 2006
Davis, Brad	3878	$31,509	Plymouth UK	March 20, 2004
Leung, Brenda	4409	$55,256	Plymouth UK	September 18, 1994
Martinez, Maria	3985	$78,980	Miami USA	January 10, 1993
Orfali, Philip	0740	$80,648	Miami USA	February 16, 2003
Patel, Mary	2398	$56,643	York UK	February 2, 1990
Smith, Robert	5162	$64,561	INCORRECT CODE	April 15, 2006
Sorrell, Joseph	4421	$62,403	Denver USA	** Not Eligible **
Zook, Carla	7385	$37,010	York UK	** Not Eligible **

例 12: 生の文字データを数値に変換する

要素:　INVALUE ステートメント

詳細

この例では、INVALUE ステートメントを使用して、数値と文字の生データを数値データに変換する数値入力形式を作成します。

プログラム

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';

proc format library=library;

  invalue evaluation 'O'=4
                     'S'=3
                     'E'=2
                     'C'=1
                     'N'=0;
run;

data proclib.points;
  input EmployeeId $ (Q1-Q4) (evaluation.,+1);
  TotalPoints=sum(of q1-q4);
  datalines;
2355 S O O S
5889 2 2 2 2
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E C C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;

proc print data=proclib.points noobs;

  title 'The PROCLIB.POINTS Data Set';
run;
```

プログラムの説明

このプログラムは、グレードをポイントとして合計するレポートを生成できるように、アルファベット順の従業員評価グレードを年に 4 回数値に変換します。

PROCLIB と LIBRARY の 2 つの SAS ライブラリ参照を設定します。

```
libname proclib 'SAS-library-1';
libname library 'SAS-library-2';
```

EVALUATION.入力形式をカタログ Library.Formats に保存します。

```
proc format library=library;
```

数値入力形式 EVALUATION を作成します。 INVALUE ステートメントは指定値を変換します。文字 O (Outstanding)、S (Superior)、E (Excellent)、C (Commendable)、および N (None)はそれぞれ数字 4、3、2、1、0 に対応しています。

```
  invalue evaluation 'O'=4
                     'S'=3
                     'E'=2
                     'C'=1
```

```
                'N'=0;
run;
```

PROCLIB.POINTS データセットを作成します。 DATALINES ステートメントの直後に続くインストリームデータには、四半期(Q1–Q4)の各従業員に対する一意の ID 番号(EmployeeId)と賞与評価が含まれています。データ行にリストされている賞与評価値の一部は数字です。その他は文字値です。文字値がデータ行にリストされている場合、EVALUATION.入力形式は値 O を 4 に、値 S を 3 などに変換します。生データ値 0 から 4 は、入力形式の定義で参照されないため、そのまま読み込まれます。文字値を数字に変換することにより、年間の各従業員に対する賞与ポイントの合計数を計算できるようになります。TotalPoints は、賞与ポイントの合計数です。

```
data proclib.points;
  input EmployeeId $ (Q1-Q4) (evaluation.,+1);
  TotalPoints=sum(of q1-q4);
  datalines;
2355 S O O S
5889 2 2 2 2
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E C C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;
```

PROCLIB.POINTS データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=proclib.points noobs;
```

タイトルを指定します。

```
  title 'The PROCLIB.POINTS Data Set';
run;
```

出力

アウトプット 26.10　PROCLIB.POINT データセット

The PROCLIB.POINTS Data Set

EmployeeId	Q1	Q2	Q3	Q4	TotalPoints
2355	3	4	4	3	14
5889	2	2	2	2	8
3878	1	2	2	2	7
4409	0	1	1	1	3
3985	3	3	3	2	11
0740	3	2	2	3	10
2398	2	2	1	1	6
5162	1	1	1	2	5
4421	3	2	2	2	9
7385	1	1	1	0	3

例 13: CNTLIN=データセットからの出力形式の作成

要素:

PROC FORMAT ステートメントオプション
　CNTLIN=
入力制御データセット

詳細

この例では、SAS データセットから出力形式を作成する方法を示します。

手順は次のとおりです。

- 入力制御データセトからの出力形式の作成
- 既存する SAS データセットからの入力制御データセットの作成

一時データセットを作成するためのプログラム

```
data scale;
```

```
  input begin: $char2. end: $char2. amount: $char2.;
  datalines;
0  3   0%
4  6   3%
7  8   6%
9  10  8%
11 16  10%
;

data ctrl;
  length label $ 11;

  set scale(rename=(begin=start amount=label)) end=last;

  retain fmtname 'PercentageFormat' type 'n';

  output;

  if last then do;
    hlo='O';
    label='***ERROR***';
    output;
  end;
run;

proc print data=ctrl noobs;

  title 'The CTRL Data Set';
run;
```

プログラムの説明

scale という名前の一時データセットを作成します。 データ行の BEGIN、END という最初の 2 つの変数は、出力形式の範囲を指定するために使用されます。AMOUNT という 3 つ目の変数には、出力形式のフォーマットされた値として使用されるパーセントが含まれます。この 3 つの変数はすべて、PROC FORMAT 入力制御データセットに必要な文字変数です。

```
data scale;
  input begin: $char2. end: $char2. amount: $char2.;
  datalines;
0  3   0%
4  6   3%
7  8   6%
9  10  8%
11 16  10%
;
```

入力制御データセット CTRL を作成し、LABEL 変数の長さを設定します。 LENGTH ステートメントにより、LABEL 変数がラベル***ERROR***に合う十分な長さになります。

```
data ctrl;
  length label $ 11;
```

変数名を変更し、ファイルの終わりフラグを作成します。 データセット CTRL は WORK.SCALE から派生します。RENAME=は、BEGIN と AMOUNT をそれぞれ

START と LABEL に名前変更します。END=オプションは、変数 LAST を作成します。この変数の値は、最後のオブザベーションの処理時に 1 に設定されます。

```
set scale(rename=(begin=start amount=label)) end=last;
```

固定値を使用して、変数 Fmtname と Type を作成します。 RETAIN ステートメントはこの場合、割り当てステートメントよりも効率的です。RETAIN はプログラムデータベクトルの Fmtname と Type の値を保持し、DATA ステップの反復ごとに値を書き込む必要をなくします。Fmtname は名前 PercentageFormat を指定します。これは、入力制御データセットが作成する出力形式です。Type 変数は、入力制御データセットが数値出力形式を作成するように指定します。

```
retain fmtname 'PercentageFormat' type 'n';
```

出力データセットにオブザベーションを書き込みます。

```
output;
```

“その他”カテゴリを作成します。 このアプリケーションに有効な値は 0–16 だけであるため、その他の値(欠損値など)はエラーとしてユーザーに示される必要があります。IF ステートメントは DATA ステップが入力データセットからの最後のオブザベーションを処理した後にのみ実行します。IF の実行時には、HLO は範囲が OTHER であることを示す値 0 を受け取ります。LABEL は値***ERROR***を受け取ります。OUTPUT ステートメントはこれらの値をデータセットの最後のオブザベーションとして書き込みます。HLO には、その他すべてのオブザベーションに対する欠損値が含まれます。

```
  if last then do;
    hlo='O';
    label='***ERROR***';
    output;
  end;
run;
```

制御データセット CTRL を印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=ctrl noobs;
```

タイトルを指定します。

```
  title 'The CTRL Data Set';
run;
```

出力

アウトプット 26.11　CTRL データセット

The CTRL Data Set

label	start	end	fmtname	type	hlo
0%	0	3	PercentageFormat	n	
3%	4	6	PercentageFormat	n	
6%	7	8	PercentageFormat	n	
8%	9	10	PercentageFormat	n	
10%	11	16	PercentageFormat	n	
ERROR	11	16	PercentageFormat	n	O

出力形式を作成するプログラム

```
proc format library=work cntlin=ctrl;
run;

proc format library=work;
  invalue evaluation 'O'=4
                     'S'=3
                     'E'=2
                     'C'=1
                     'N'=0;
run;

data points;
  input EmployeeId $ (Q1-Q4) (evaluation.,+1);
  TotalPoints=q1+q2+q3+q4;
  datalines;
2355 S O O S
5889 2 . 2 2
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E   C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;

proc report data=work.points nowd headskip split='#';
  column employeeid totalpoints totalpoints=Pctage;
  define employeeid / right;
  define totalpoints / 'Total#Points' right;
  define pctage / format=PercentageFormat12. 'Percentage' left;
```

```
  title 'The Percentage of Salary for Calculating Bonus';
run;
```

プログラムの説明

作成した出力形式をカタログ Work.Formats に保存し、出力形式のソースを指定します。 CNTLIN=オプションは、データセット CTRL が出力形式 PercentageFormat のソースになるように指定します。

```
proc format library=work cntlin=ctrl;
run;
```

数値入力形式 EVALUATION を作成します。 INVALUE ステートメントは指定値を変換します。文字 O (Outstanding)、S (Superior)、E (Excellent)、C (Commendable)、および N (None)はそれぞれ数字 4、3、2、1、0 に対応しています。

```
proc format library=work;
  invalue evaluation 'O'=4
                     'S'=3
                     'E'=2
                     'C'=1
                     'N'=0;
run;
```

WORK.POINTS データセットを作成します。 DATALINES ステートメントの直後に続くインストリームデータには、四半期(Q1–Q4)の各従業員に対する一意の ID 番号(EmployeeId)と賞与評価が含まれています。データ行にリストされている賞与評価値の一部は数字です。その他は文字値です。文字値がデータ行にリストされている場合、Evaluation.入力形式は値 O を 4 に、値 S を 3 などに変換します。生データ値 0 から 4 は、入力形式の定義で参照されないため、そのまま読み込まれます。文字値を数字に変換することにより、年間の各従業員に対する賞与ポイントの合計数を計算できるようになります。TotalPoints は、賞与ポイントの合計数です。欠損値が TotalPoints の欠損値になるように、加算演算子が SUM 関数の代わりに使用されます。

```
data points;
  input EmployeeId $ (Q1-Q4) (evaluation.,+1);
  TotalPoints=q1+q2+q3+q4;
  datalines;
2355 S O O S
5889 2 . 2 2
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E   C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;
```

WORK.POINTS に対しレポートを生成し、PERCENTAGEFORMAT.出力形式と TotalPoints 変数を関連付けます。 DEFINE ステートメントはその関連付けを実行します。TotalPoints のフォーマットされた値を含む列では、別名 Pctage を使用し

ています。別名を使用することにより、変数を出力形式とデフォルト出力形式を使用してそれぞれ 1 回ずつ、合計 2 回印刷できます。REPORT プロシジャの詳細については、51 章, “REPORT プロシジャ”を参照してください。

```
proc report data=work.points nowd headskip split='#';
  column employeeid totalpoints totalpoints=Pctage;
  define employeeid / right;
  define totalpoints / 'Total#Points' right;
  define pctage / format=PercentageFormat12. 'Percentage' left;
  title 'The Percentage of Salary for Calculating Bonus';
run;
```

出力

アウトプット 26.12　賞与を計算するための給与のパーセント

The Percentage of Salary for Calculating Bonus

EmployeeId	Total Points	Percentage
2355	14	10%
5889	.	***ERROR***
3878	7	6%
4409	3	0%
3985	11	10%
0740	10	8%
2398	.	***ERROR***
5162	5	3%
4421	9	8%
7385	3	0%

例 14: CNTLIN=データセットからの入力形式の作成

要素:

PROC FORMAT ステートメントオプション
　CNTLIN=
入力制御データセット

詳細

この例では、CNTLIN=データセットから入力形式を作成する方法を示します。

手順は次のとおりです。

- 入力制御データセットからの入力形式の作成
- 既存する SAS データセットからの入力制御データセットの作成

プログラム

```
proc format;
   invalue mytest
    'abc'=1
    'xyz'=2
    other=3;
   invalue $chrtest
    'abc'='xyz'
    other='else';
   run;

data _null_;
   input value:mytest. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;

data _null_;
   input value:$chrtest. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;

data temp;
   length start $8 type $1 hlo $1;
   fmtname='newtest'; type='i';
   start='abc'; label=1; hlo=' '; output;
   start='xyz'; label=2; hlo=' '; output;
   start=' ';   label=3; hlo='O'; output;
   run;

proc format cntlin=temp; run;

data temp;
   length start label $8 type $1 hlo $1;
   fmtname='$newchr'; type='j';
   start='abc'; label='xyz'; hlo=' '; output;
   start=' ';   label='else'; hlo='O'; output;
   run;

proc format cntlin=temp; run;
```

```
data _null_;
   input value:newtest. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;

data _null_;
   input value:$newchr. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;

data temp;
   length start label $8 hlo $1;
   fmtname='@new2test';
   start='abc'; label='1'; hlo=' '; output;
   start='xyz'; label='2'; hlo=' '; output;
   start=' ';   label='3'; hlo='O'; output;
   fmtname='@$new2chr';
   start='abc'; label='xyz'; hlo=' '; output;
   start=' ';   label='else'; hlo='O'; output;
   run;

proc format cntlin=temp; run;

data _null_;
   input value:new2test. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;

data _null_;
   input value:$new2chr. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;


proc print data=temp noobs;
title 'The CTRL Data Set';
run;
```

プログラムの説明

INVALUE ステートメントを使用して入力形式を作成します。 数値入力形式と文字入力形式を作成します。

```
proc format;
   invalue mytest
    'abc'=1
    'xyz'=2
```

```
  other=3;
invalue $chrtest
  'abc'='xyz'
  other='else';
run;
```

数値入力形式をインストリームデータと共に使用します。 このコードでは、VALUE の結果として 1、2、3、および 3 が再度、生成されるはずです。

```
data _null_;
   input value:mytest. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;
```

文字入力形式をインストリームデータと共に使用します。 このコードでは、VALUE の結果として、xyz、else、else、および else が生成されるはずです。

```
data _null_;
   input value:$chrtest. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;
```

CNTLIN=データセットを使用して MYTEST 数値入力形式と同等のものを作成し、入力形式名 NEWTEST を使用します。 FMTNAME 変数の値が NEWTEST であることを指定します。TYPE 変数に値「i」または「I」を指定して、数値入力形式であることを示します。その他の場合は O の値を指定し、HLO 変数の場合は空白値を指定します。

```
data temp;
   length start $8 type $1 hlo $1;
   fmtname='newtest'; type='i';
   start='abc'; label=1; hlo=' '; output;
   start='xyz'; label=2; hlo=' '; output;
   start=' ';   label=3; hlo='O'; output;
   run;
```

文字入力形式をインストリームデータと共に使用します。 このコードでは、VALUE の結果として、xyz、else、else、および else が生成されるはずです。

```
proc format cntlin=temp; run;
```

文字入力形式には CNTLIN=データセットを作成します。 この入力形式は、上記で作成された$CHRTEST 入力形式と同等です。それには$NEWCHR という名前を指定します。TYPE の値は、それが文字入力形式であることを示すために'j'または'J'でなければなりません。

```
data temp;
   length start label $8 type $1 hlo $1;
   fmtname='$newchr'; type='j';
   start='abc'; label='xyz'; hlo=' '; output;
   start=' ';   label='else'; hlo='O'; output;
   run;
```

文字入力形式を作成するには、CNTLIN=データセットを読み込みます。 この例では、 *temp* を CNTLIN の値として使用しています。コード例を保存する場合は、よりわかりやすい名前を指定してください。

```
proc format cntlin=temp; run;
```

以前作成したバージョンと同じ 2 つの DATA ステップを作成します。 これらのバージョンには、入力形式名 NEWTEST と$NEWCHR を使用します。

```
data _null_;
   input value:newtest. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;

data _null_;
   input value:$newchr. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;
```

FMTNAME 値を@で始めることでそれが入力形式であることを示し、またタイプ変数が必要でないことを示します。 同じ CNTLIN=データセットで数値と文字の入力形式を作成することができます。文字出力形式が定義されているため、ラベル変数は文字でなければなりません。数値は、数値を含む文字列として保存されます。

```
data temp;
   length start label $8 hlo $1;
   fmtname='@new2test';
   start='abc'; label='1'; hlo=' '; output;
   start='xyz'; label='2'; hlo=' '; output;
   start=' ';   label='3'; hlo='O'; output;
   fmtname='@$new2chr';
   start='abc'; label='xyz'; hlo=' '; output;
   start=' ';   label='else'; hlo='O'; output;
   run;

proc format cntlin=temp; run;

data _null_;
   input value:new2test. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;

data _null_;
   input value:$new2chr. @@;
   put value=;
   datalines;
abc xyz ghi 4
   run;
```

CNTLIN=データセットのコンテンツがリストで出力されます。 テーブルに "CTRL Data Set"というラベルを付けます。

```
proc print data=temp noobs;
title 'The CTRL Data Set';
run;
```

アウトプット 26.13　CRTL データセットのコンテンツ

The CTRL Data Set

start	label	hlo	fmtname
abc	1		@new2test
xyz	2		@new2test
	3	O	@new2test
abc	xyz		@$new2chr
	else	O	@$new2chr

例 15: 入力形式と出力形式の説明の印刷

要素: PROC FORMAT ステートメントオプション
FMTLIB
SELECT ステートメント

出力形式: 例 4 の BENEFIT.

入力形式: 例 6 の EVALUATION.

詳細

この例では、入力形式と出力形式の説明を印刷する方法を示します。説明には、読み書きされる値が示されます。

プログラム

```
libname library 'SAS-library';
proc format library=library fmtlib;
  select @evaluation benefit;
  title 'FMTLIB Output for the BENEFIT. Format and the';
  title2 'EVALUATION. Informat';
run;
```

プログラムの説明

LIBRARY という名前の SAS ライブラリ参照を設定します。

```
libname library 'SAS-library';
```

EVALUATION.と BENEFIT の説明を印刷します。 FMTLIB オプションは、LIBRARY=オプションが指定するカタログの出力形式と入力形式に関する情報を印刷します。LIBRARY=LIBRARY は Library.Formats カタログを指します。

```
proc format library=library fmtlib;
```

入力形式と出力形式を選択します。 SELECT ステートメントは前の例で作成された EVALUATION.と BENEFIT.を選択します。EVALUATION.の前のアットマーク(@)は、EVALUATION.が入力形式であることを示します。

```
  select @evaluation benefit;
```

タイトルを指定します。

```
  title 'FMTLIB Output for the BENEFIT. Format and the';
  title2 'EVALUATION. Informat';
run;
```

出力

アウトプット 26.14　利点出力形式と評価入力形式の FMTLIB 出力

FMTLIB Output for the BENEFIT. Format and the EVALUATION. Informat

FORMAT NAME: BENEFIT LENGTH: 20 NUMBER OF VALUES: 2 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 20 FUZZ: STD		
START	END	LABEL (VER. V7|V8 17SEP2012:15:53:27)
LOW	17897	[WORDDATE20.]
17898	HIGH	** Not Eligible **

INFORMAT NAME: @EVALUATION LENGTH: 1 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 1 FUZZ: 0		
START	END	INVALUE(VER. 9.4 17SEP2012:16:10:24)
C	C	1
E	E	2
N	N	0
O	O	4
S	S	3

例 16: 永久出力形式の取得

要素:　PROC FORMAT ステートメントオプション
　LIBRARY=

FMTSEARCH=システムオプション

データセット: SAMPLE

この例では、LIBRARY=オプションと FMTSEARCH=システムオプションを使用して、Work.Formats または Library.Formats 以外のカタログに保存されている出力形式を保存し、取得します。

プログラム

```
libname proclib 'SAS-library';

proc format library=proclib;

picture nozeros (fuzz=0)
         low -  -1  = '000.00'(prefix='-')
      -1 < - < -.99  = '0.99' (prefix='-.' mult=100)
      -0.99 < - < 0  = '99'    (prefix='-.' mult=100)
                0  = '0.99'
       0 < - < .99   = '99'    (prefix='.'  mult=100)
         0.99 - <1   = '0.99'  (prefix='.'  mult=100)
          1  - high = '00.99';
run;

options  fmtsearch=(proclib);

data sample;
  input Amount;
   datalines;
-2.051
-.05
-.017
  0
.093
.54
.556
6.6
14.63
0.996
-0.999
-45.00
;
run;

proc print data=sample;
   format amount nozeros.;

   title1 'Retrieving the NOZEROS. Format from PROCLIB.FORMATS';
   title2 'The SAMPLE Data Set';
run;
```

プログラムの説明

PROCLIB という名前の SAS ライブラリ参照を設定します。

```
libname proclib 'SAS-library';
```

NOZEROS.出力形式を PROCLIB.FORMATS カタログに保存します。

```
proc format library=proclib;
```

NOZEROS.出力形式を作成します。 PICTURE ステートメントはピクチャ出力形式 NOZEROS.を定義します。“詳細” (897 ページ)を参照してください。

```
picture nozeros (fuzz=0)
        low -  -1  = '000.00'(prefix='-')
      -1 < - < -.99  = '0.99' (prefix='-.' mult=100)
      -0.99 < - < 0  = '99'    (prefix='-.' mult=100)
               0  = '0.99'
       0 < - < .99   = '99'    (prefix='.'  mult=100)
         0.99 - <1   = '0.99'  (prefix='.'  mult=100)
          1  - high = '00.99';
run;
```

PROCLIB.FORMATS カタログをユーザー定義の出力形式の検索に使用される検索パスに追加します。 FMTSEARCH=システムオプションは、検索パスを定義します。FMTSEARCH=システムオプションにはライブラリ参照名のみ必要です。FMTSEARCH=は、カタログ名が表示されていない場合はカタログ名を FORMATS とみなします。FMTSEARCH=オプションがないと、NOZEROS.出力形式は見つかりません。詳細については、“FMTSEARCH = システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

```
options  fmtsearch=(proclib);
```

サンプルデータセットを作成します。

```
data sample;
  input Amount;
   datalines;
-2.051
-.05
-.017
  0
.093
.54
.556
6.6
14.63
0.996
-0.999
-45.00
;
run;
```

SAMPLE データセットを印刷します。 FORMAT ステートメントは、NOZEROS.出力形式と Amount 変数を関連付けます。

```
proc print data=sample;
   format amount nozeros.;
```

タイトルを指定します。

```
   title1 'Retrieving the NOZEROS. Format from PROCLIB.FORMATS';
   title2 'The SAMPLE Data Set';
run;
```

出力

アウトプット 26.15　NOZEROS.の取得 PROCLIB.FORMATS からの出力形式

Retrieving the NOZEROS. Format from PROCLIB.FORMATS
The SAMPLE Data Set

Obs	Amount
1	-2.05
2	-.05
3	-.01
4	.00
5	.09
6	.54
7	.55
8	6.60
9	14.63
10	.99
11	-.99
12	45.00

例 17: 文字列の範囲指定

要素: VALUE ステートメント

データセット: PROCLIB.STAFF

この例では、出力形式を作成し、文字列を含む範囲を使用する方法を示します。

プログラム

```
libname proclib'SAS-library';

data train;
  set proclib.staff(keep=name idnumber);
run;

proc print data=train noobs;

  title 'The TRAIN Data Set without a Format';
```

```
run;
```

プログラムの説明

```
libname proclib'SAS-library';
```

PROCLIB.STAFF データセットから TRAIN データセットを作成します。
PROCLIB.STAFF は“例 2: サンプルデータセットの作成” (933 ページ)で作成されました。

```
data train;
  set proclib.staff(keep=name idnumber);
run;
```

データセット TRAIN を出力形式なしで印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=train noobs;
```

タイトルを指定します。

```
  title 'The TRAIN Data Set without a Format';
run;
```

出力

アウトプット 26.16　出力形式のない TRAIN データセット

The TRAIN Data Set without a Format

Name	IdNumber
Capalleti, Jimmy	2355
Chen, Len	5889
Davis, Brad	3878
Leung, Brenda	4409
Martinez, Maria	3985
Orfali, Philip	0740
Patel, Mary	2398
Smith, Robert	5162
Sorrell, Joseph	4421
Zook, Carla	7385

出力形式を Work.Formats に保存します。 LIBRARY=オプションが表示されないため、出力形式は Work.Formats に保存され、現在の SAS セッションにのみ利用可能です。

```
proc format;
```

$SKILLTEST.出力形式を作成します。 $SKILLTEST.出力形式は各従業員の ID 番号と割り当てられているスキルテストを出力します。従業員は、姓によって TEST A、TEST B、TEST C のいずれかを受ける必要があります。除外演算子(<)は、範囲の最終値を除外します。そのため、最初の範囲には姓が A から D で始まる従業員が、2 番目の範囲には姓が E から M で始まる従業員が含まれます。最後の範囲のチルダ(~)は、Z で始まる文字列全体を含めるために必要です。

```
  value $skilltest  'a'-<'e','A'-<'E'='Test A'
            'e'-<'m','E'-<'M'='Test B'
            'm'-'z~','M'-'Z~'='Test C';
run;
```

TRAIN データセットのレポートを生成します。 DEFINE ステートメントの FORMAT=オプションは、$SKILLTEST.出力形式と Name 変数を関連付けます。Name のフォーマットされた値を含む列は、別名 Test を使用しています。別名を使用することにより、変数を出力形式とデフォルト出力形式を使用してそれぞれ 1 回ずつ、合計 2 回印刷できます。詳細については、51 章, “REPORT プロシジャ”を参照してください。

```
proc report data=train nowd headskip;
  column name name=test idnumber;
  define test / display format=$skilltest. 'Test';
  define idnumber / center;
  title 'Test Assignment for Each Employee';
run;
```

出力

アウトプット 26.17　各従業員のテスト割り当て

Test Assignment for Each Employee

Name	Test	IdNumber
Capalleti, Jimmy	Test A	2355
Chen, Len	Test A	5889
Davis, Brad	Test A	3878
Leung, Brenda	Test B	4409
Martinez, Maria	Test C	3985
Orfali, Philip	Test C	0740
Patel, Mary	Test C	2398
Smith, Robert	Test C	5162
Sorrell, Joseph	Test C	4421
Zook, Carla	Test C	7385

例 18: 英語以外の言語での出力形式の作成

要素:

PICTURE ステートメントオプション
　DATATYPE=
　LANGUAGE=
LOCALE=システムオプション

詳細

この例では、次のタスクを実行します。

- DATATYPE=ステートメントオプションを使用して日付値と日時値をフォーマットするためのディレクティブを 使用してピクチャ出力形式を作成します。
- LOCALE=システムオプションを使用して、ドイツ語のロケールを指定します。
- ピクチャ出力形式を使用して、日付値と日時値をドイツ語のログに印刷します。
- LANGUAGE=French を指定するピクチャ出力形式を使用して、フランス語の日時値をログに印刷します。

プログラム

```
proc format;
  picture mdy(default=8) other='%0d%0m%Y' (datatype=date);
  picture langtsda (default=50) other='%A, %d %B, %Y' (datatype=date);
  picture langtsdt (default=50) other='%A, %d,%B, %Y %H %M %S'
                    (datatype=datetime);
  picture langtsfr (default=50) other='%A, %d %B, %Y %H %M %S'
                    (datatype=datetime language=french);
  picture alltest (default=100)
       other='%a %A %b %B %d %H %I %j %m %M %p %S %w %U %y %%'
       (datatype=datetime);
run;

option locale = de_DE;

data _null_ ;
  a= 18903;
  b = 1633239000;
  put a= mdy.;
  put a= langtsda.;
  put b= langtsdt.;
  put b= langtsfr.;
  put b= alltest.;
run ;
```

プログラムの説明

PICTURE ステートメントを使用して、出力形式を作成します。 各 PICTURE ステートメントは、ディレクティブを 使用してフォーマットする日付値と日時値を指定します。%A は週の完全名を印刷します。%B は月の完全名を印刷します。%d は月の日を印刷します。%Y は年を印刷します。%H は時間(24 時間)を印刷します。%M は分を印刷します。%S は秒を印刷します。最初の 3 つの出力形式は、日付と日時を LOCALE=システムオプションの現在値によって指定される言語で印刷します。LANGTSFT.出力形式は、日時をフランス語で印刷します。その他のディレクティブについては、PICTURE ステートメント (883 ページ)を参照してください。

```
proc format;
  picture mdy(default=8) other='%0d%0m%Y' (datatype=date);
  picture langtsda (default=50) other='%A, %d %B, %Y' (datatype=date);
  picture langtsdt (default=50) other='%A, %d,%B, %Y %H %M %S'
                    (datatype=datetime);
  picture langtsfr (default=50) other='%A, %d %B, %Y %H %M %S'
                    (datatype=datetime language=french);
  picture alltest (default=100)
       other='%a %A %b %B %d %H %I %j %m %M %p %S %w %U %y %%'
       (datatype=datetime);
run;
```

LOCALE=システムオプションを設定します。 de_DE は、ドイツ語のロケール値です。

```
option locale = de_DE;
```

日付値と日時値をドイツ語とフランス語で印刷します。 DATA ステップは、SAS ログに 2011 年 10 月 3 日、05:30:00 AM に関する日付および日時の情報を印刷します。LANGTSFR.出力形式を使用してフォーマットされている b の値を除くすべての値がドイツ語で書き出されます。この出力形式は、日時値をフランス語で印刷します。

```
data _null_ ;
  a= 18903;
  b = 1633239000;
  put a= mdy.;
  put a= langtsda.;
  put b= langtsdt.;
  put b= langtsfr.;
  put b= alltest.;
run ;
```

ドイツ語、フランス語でピクチャ出力形式の出力を表示する SAS ログ

```
1    proc format;
2      picture mdy(default=8) other='%0d%0m%Y' (datatype=date);
NOTE: Format MDY has been output.
3      picture langtsda (default=50) other='%A, %d %B, %Y' (datatype=date);
NOTE: Format LANGTSDA has been output.
4      picture langtsdt (default=50) other='%A, %d,%B, %Y %H %M %S'
5                               (datatype=datetime);
NOTE: Format LANGTSDT has been output.
6      picture langtsfr (default=50) other='%A, %d %B, %Y %H %M %S'
7                               (datatype=datetime language=french);
NOTE: Format LANGTSFR has been output.
8      picture alltest (default=100)
9              other='%a %A %b %B %d %H %I %j %m %M %p %S %w %U %y %%'
10             (datatype=datetime);
NOTE: Format ALLTEST has been output.
11   run;

NOTE: PROCEDURE FORMAT used (Total process time):
      real time           0.03 seconds
      cpu time            0.03 seconds


12
13   option locale = de_DE;
14
15   data _null_ ;
16     a= 18903;
17     b = 1633239000;
18     put a= mdy.;
19     put a= langtsda.;
20     put b= langtsdt.;
21     put b= langtsfr.;
22     put b= alltest.;
23   run ;

a=03102011
a=Montag, 3 Oktober, 2011
b=Montag, 3,Oktober, 2011 5 30 0
b=Lundi, 3 octobre, 2011 5 30 0
b=Mo  Montag Okt Oktober 3 5 5 276 10 30 AM 0 2 40 11 %
```

例 19: ロケール固有のフォーマットカタログの作成

要素: PROC FORMAT LOCALE オプション
FMTSEARCH=システムオプション

詳細

この例では、英語とルーマニア語の 2 つの言語で出力形式を作成する方法と、英語とルーマニアのフォーマットカタログにアクセスして、これらの 2 つの言語でデータセットを印刷する方法を示します。この例は、SAS セッションのエンコードが、ルーマニアロケールをサポートする latin 2 エンコードである場合に最も適切に動作します。

プログラム

```
/*no locale information*/
proc format lib=work.formats;
value age low - 5 = 'baby'
6 - 12 = 'child'
13 - 15 = 'teen'
16 - 30 = 'youth'
31 - 50 = 'midlife'
51 - high = 'older';
run;

options locale=ro_RO;

proc format lib = work.formats locale;
value age low - 5 = 'Copil'
6 - 12 = 'Copil'
13 - 15 = 'Adolescent'
16 - 30 = 'Tineretului'
31 - 50 = 'Asta vrei'
51 - high = 'Mai vechi';
run;


options fmtsearch=(work/locale);

/* Set the locale back to English(US) */
options locale=en_US;

data datatst;
input age sex $;
attrib age format= age.;
cards;
```

```
5 M
6 F
12 M
13 F
15 M
16 F
30 M
35 F
51 M
100 F
;
run;
/* Use the English format catalog*/
title "Locale is English, Use the Original Format Catalog";
proc print data=datatst; run;

/* Use the Romanian format catalog*/
options locale=ro_RO;
title 'Locale is ro_RO, Use the Romanian Format Catalog';
proc print data=datatst;run;
```

プログラムの説明

AGE.出力形式を英語で作成します。

```
/*no locale information*/
proc format lib=work.formats;
value age low - 5 = 'baby'
6 - 12 = 'child'
13 - 15 = 'teen'
16 - 30 = 'youth'
31 - 50 = 'midlife'
51 - high = 'older';
run;
```

ロケールを変更し、AGE.出力形式をロケール固有のフォーマットカタログに作成します。 LOCALE=システムオプションを使用して、ロケールをルーマニアロケールに変更します。PROC FORMAT ステートメントの LOCALE オプションで、ルーマニア言語に対応する、現在のロケールを表すフォーマットカタログを作成することを指定します。

```
options locale=ro_RO;

proc format lib = work.formats locale;
value age low - 5 = 'Copil'
6 - 12 = 'Copil'
13 - 15 = 'Adolescent'
16 - 30 = 'Tineretului'
31 - 50 = 'Asta vrei'
51 - high = 'Mai vechi';
run;
```

ロケール固有のフォーマットカタログを出力形式検索パスに追加します。
FMTSEARCH=システムオプションでは、検索するフォーマットカタログを指定しま

す。複数のロケール固有カタログを作成できるため、検索リストの libref に/LOCALE を追加すると、現在のロケールに関連付けられたカタログが検索されます。

```
options fmtsearch=(work/locale);
```

データセットを作成し、英語のフォーマットカタログを使用して印刷します。
LOCALE=システムオプションで、ロケールを英語に設定します。

```
/* Set the locale back to English(US) */
options locale=en_US;

data datatst;
input age sex $;
attrib age format= age.;
cards;
5 M
6 F
12 M
13 F
15 M
16 F
30 M
35 F
51 M
100 F
;
run;
/* Use the English format catalog*/
title "Locale is English, Use the Original Format Catalog";
proc print data=datatst; run;
```

ルーマニア語のフォーマットカタログを使用して、データセットを印刷します。
LOCALE=システムオプションを使用して、ロケールをルーマニアロケールに設定します。

```
/* Use the Romanian format catalog*/
options locale=ro_RO;
title 'Locale is ro_RO, Use the Romanian Format Catalog';
proc print data=datatst;run;
```

次は、英語とルーマニア語のフォーマットカタログを使用して印刷されたデータセットを示しています。

アウトプット 26.18　英語とルーマニア語のフォーマットカタログを使用して印刷されたデータセット

Locale is English, Use the Original Format Catalog

Obs	age	sex
1	baby	M
2	child	F
3	child	M
4	teen	F
5	teen	M
6	youth	F
7	youth	M
8	midlife	F
9	older	M
10	older	F

Locale is ro_RO, Use the Romanian Format Catalog

Obs	age	sex
1	Copil	M
2	Copil	F
3	Copil	M
4	Adolescent	F
5	Adolescent	M
6	Tineretului	F
7	Tineretului	M
8	Asta vrei	F
9	Mai vechi	M
10	Mai vechi	F

例 20: 出力形式として使用する関数の作成

要素:　PROC FCMP ステートメント

CMPLIB=システムオプション
PROC FORMAT ステートメント
出力形式機能としての関数

詳細

この例では、気温をセ氏からカ氏、カ氏からセ氏に変換する関数を作成します。プログラムでは、関数を 1 つの DATA ステップで関数として、別の DATA ステップで出力形式として使用します。

プログラム

```
proc fcmp outlib=library.functions.smd;
  function ctof(c) $;
    return(cats(((9*c)/5)+32,'F'));
  endsub;

  function ftoc(f) $;
    return(cats((f-32)*5/9,'C'));
  endsub;

run;

options cmplib=(library.functions);

data _null_;
  f=ctof(100);
  put f=;
run;

proc format;
  value ctof (default=10) other=[ctof()];
  value ftoc (default=10) other=[ftoc()];
 run;

data _null_;
  c=100;
  put c=ctof.;
  f=212;
  put f=ftoc.;
run;
```

プログラムの説明

気温をセ氏からカ氏、カ氏からセ氏に変更する関数を作成します。 FCMP プロシジャは、セ氏気温をカ氏、カ氏気温をセ氏に変換する CTOF 関数を作成します。

```
proc fcmp outlib=library.functions.smd;
```

```
  function ctof(c) $;
    return(cats(((9*c)/5)+32,'F'));
  endsub;

  function ftoc(f) $;
    return(cats((f-32)*5/9,'C'));
  endsub;

run;
```

関数ライブラリにアクセスします。 CMPLIB システムオプションにより、プログラムのコンパイラ中に関数を含めることができます。

```
options cmplib=(library.functions);
```

その関数を SAS プログラムの関数として使用します。

```
data _null_;
  f=ctof(100);
  put f=;
run;
```

関数を使用して、ユーザー定義の出力形式を作成します。 出力形式の名前は、関数の名前です。関数を出力形式として使用する際に、OTHER キーワードによって示されるように出力形式をネストできます。

```
proc format;
  value ctof (default=10) other=[ctof()];
  value ftoc (default=10) other=[ftoc()];
 run;
```

関数を出力形式として使用します。 この DATA ステップは、PUT ステートメントを使用して気温をフォーマットします。ここで出力形式を PUT ステートメントの変数に割り当てます。

```
data _null_;
  c=100;
  put c=ctof.;
  f=212;
  put f=ftoc.;
run;
```

出力: ログ

例のコード 26.6　出力形式として使用する関数を作成した後の SAS ログ

```
323  proc fcmp outlib=library.functions.smd;
324    function ctof(c) $;
325      return(cats(((9*c)/5)+32,'F'));
326    endsub;
327
328    function ftoc(f) $;
329      return(cats((f-32)*5/9,'C'));
330    endsub;
331
332  run;

NOTE: Function ftoc saved to library.functions.smd.
NOTE: Function ctof saved to library.functions.smd.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          17.59 seconds
      cpu time           1.26 seconds


333
334  options cmplib=(library.functions);
335
336  data _null_;
337    f=ctof(100);
338    put f=;
339  run;

f=212F
NOTE: DATA statement used (Total process time):
      real time          0.50 seconds
      cpu time           0.01 seconds


340
341  proc format;
342      value ctof (default=10) other=[ctof()];
NOTE: Format CTOF has been output.
343      value ftoc (default=10) other=[ftoc()];
NOTE: Format FTOC has been output.
344      run;

NOTE: PROCEDURE FORMAT used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds


345
346  data _null_;
347    c=100;
348    put c=ctof.;
349    f=212;
350    put f=ftoc.;
351  run;

c=212F
f=100C
```

例 21: 出力形式を使用してドリルダウンテーブルを作成する

要素: VALUE ステートメント
PROC PRINT FORMAT ステートメント

詳細

この例では、アメリカ合衆国の 5 つの州に関する人口情報を含む HTML テーブルを作成します。州の名前は、州の Web サイトへのリンクになっています。リンクは、州名をフォーマットするためのユーザー定義の出力形式を使用して作成されます。この例では、次を行います。

- 州の人口情報を含むデータセットの作成
- 値が HTML リンク(<a>) 要素である VALUE ステートメントを使用したユーザー定義の出力形式の作成
- HTML ファイルの名前と HTML ファイルのタイトルの定義
- ユーザー定義の出力形式を使用した HTML テーブルの印刷

プログラム

```
data mydata;
  format population comma12.0;
  label st='State';
  label population='Population';
  input st $ 1-2 population;
  year=2000;
  datalines;
VA  7078515
NC  8049313
SC  4012012
GA  8186453
FL 15982378
;
run;

proc format;
value $COMPND
'VA'='<a href=http://www.va.gov>VA</a>'
'NC'='<a href=http://www.nc.gov>NC</a>'
'SC'='<a href=http://www.sc.gov>SC</a>'
'GA'='<a href=http://www.ga.gov>GA</a>'
```

```
'FL'='<a href=http://www.fl.gov>FL</a>';
run;

ods html file="c:\mySAS\html\Drilldown.htm"
 (title="An ODS HTML Drill-down Table Using a User-defined Format in the PRINT
 Procedure");

title h=.25in "Year 2000 U.S. Census Population";
title2 color=gray "An ODS HTML Drill-down Table Using a User-defined Format in
the PRINT Procedure";
footnote color=gray "(Click the underlined text to drill down.)";

options nodate;
proc print data=mydata label noobs;
 var st population;
 format st $compnd. ;
run;

ods html close;
ods html;
```

プログラムの説明

データセットを作成します。 mydata DATA ステップは、2000 年に行われた国勢調査に基づいたアメリカ合衆国の 5 つの州の人口に関する情報を含むデータセットを作成します。作成される変数は、国勢調査の年、州の略称、州の人口に関するデータを割り当てます。

```
data mydata;
  format population comma12.0;
  label st='State';
  label population='Population';
  input st $ 1-2 population;
  year=2000;
  datalines;
VA  7078515
NC  8049313
SC  4012012
GA  8186453
FL 15982378
;
run;
```

$COMPND.出力形式を作成します。 $COMPND.出力形式は、各州を州の各 Web サイトへのリンクとしてフォーマットします。

```
proc format;
value $COMPND
'VA'='<a href=http://www.va.gov>VA</a>'
'NC'='<a href=http://www.nc.gov>NC</a>'
'SC'='<a href=http://www.sc.gov>SC</a>'
'GA'='<a href=http://www.ga.gov>GA</a>'
'FL'='<a href=http://www.fl.gov>FL</a>';
run;
```

テーブルファイル名とテーブルタイトルを設定します。 ODS HTML FILE=オプションは、HTML 出力が保存されるディレクトリとファイル名を指定します。

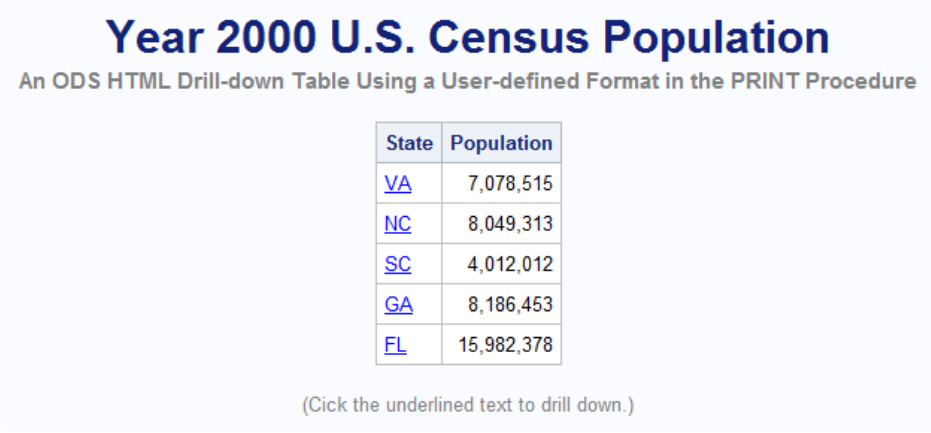
```
ods html file="c:\mySAS\html\Drilldown.htm"
 (title="An ODS HTML Drill-down Table Using a User-defined Format in the PRINT
 Procedure");

title h=.25in "Year 2000 U.S. Census Population";
title2 color=gray "An ODS HTML Drill-down Table Using a User-defined Format in
the PRINT Procedure";
footnote color=gray "(Click the underlined text to drill down.)";
```

テーブルを印刷して、HTML 出力先を閉じ、再度開きます。 PRINT プロシジャは出力形式$COMPND.を使用して、州名をフォーマットします。フォーマットされた名前は、州の各 Web サイトへのリンクとなります。ODS HTML ステートメントは、後の出力が作成したばかりの HTML ファイルを上書きしないように、HTML 出力先を閉じ、再度開きます。

```
options nodate;
proc print data=mydata label noobs;
 var st population;
 format st $compnd. ;
run;

ods html close;
ods html;
```

出力

アウトプット 26.19　出力形式を使用した HTML テーブルでのドリルダウンテキストの作成



Year 2000 U.S. Census Population

An ODS HTML Drill-down Table Using a User-defined Format in the PRINT Procedure

State	Population
VA	7,078,515
NC	8,049,313
SC	4,012,012
GA	8,186,453
FL	15,982,378

(Cick the underlined text to drill down.)

27

GATEWAY プロシジャ

概要: GATEWAY プロシジャ

GATEWAY プロシジャの機能

> 重要　PROC ゲートウェイは、2024.09 リリースから利用可能です。

GATEWAY プロシジャを使用すると、サポートされているオープンソースエンジンの複数のセッションを実行して、SAS とオープンソース間の並列処理と高効率のデータ転送を行うことができます。ユーザーは Python と R を使用して、SAS に安全に保存されている大きな SAS テーブルを操作できます。

PROC GATEWAY の機能

PROC GATEWAY の主な機能は次のとおりです。

- CAS ワーカー内で複数の Python または R を起動します。
- CAS サーバーからデータを効率的に読み書きします。

概念: GATEWAY プロシジャ

外部言語アクセスと必要なライブラリ

PROC GATEWAY は、環境変数の代わりに外部言語アクセスコントロールの構成設定を使用して、環境での Python および R インタープリターの使用を承認します。

PROC GATEWAY を使用するには、次のライブラリを Python 環境および R 環境にインストールする必要があります。

- Python: pyrow (8 以降)および pandas ライブラリがインストールされている必要があります。

- R: arrow(8 以上)、jsonliter、および logger ライブラリがインストールされている必要があります

コンピューターリソース

計算での使用は、オープンソースコードで使用されているものによって異なります。Python セッションと R セッションは CAS ポッド内で起動されるため、リソースを CAS サーバーと共有します。通常、CAS テーブル全体を 1 つのオープンソースセッションに書き込むのは効率的ではありません。1 つのポッドでは、テーブル全体を保持するための十分なリソースがない場合があるからです。メモリ不足やクラッシュを回避するために、テーブルをバッチ処理で読み書きする方法があります(“new_table_reader” Python と“new_table_reader”R で対応可能)。

表示される出力

PROC GATEWAY を使用している場合、オープンソースクライアントによって 印刷されるメッセージはすべて SAS ログに印刷されます。“return_result”関数(Python)または“return_result”関数(R)を使用して、オープンソースクライアントから SAS ログに結果を印刷できます。

ODS テーブル

デフォルトでは、PROC GATEWAY は ODS テーブルを生成しません。PROC GATEWAY で任意のテーブルをクライアントに返す場合は、“return_table”関数(Python)または“return_table”関数(R)を使用します。

構文: GATEWAY プロシジャ

PROC GATEWAY<*proc-options*>;

ARGS<*options*>;

OPTS<*options*>;

SUBMIT;

ENDSUBMIT;

PROC GATEWAY ステートメント

GATEWAY プロシジャは、Python や R などの別の言語で記述されたコードを実行します。

制限事項: PROC ゲートウェイは、2024.09 リリースから利用可能です。

構文

PROC GATEWAY<*options*>;

オプション引数

CODE_FILE=*"string"*
解析され Python クライアントまたは R クライアントに送信されるファイルへのパスを指定します。この引数は、SUBMIT および ENDSUBMIT ステートメントを使用した場合は無視されます。

LANG=PYTHON | PY | R
デフォルト PYTHON

呼び出す言語エンジンを指定します。

SINGLE
コード実行を 1 つのスレッド内の 1 つのセッションに制限するように指定します。

NTHREADS=*integer*
プログラムの実行に使用されるスレッドの数を指定します。分散サーバーの場合、この値は、プログラムの実行に使用する各ワーカーのスレッド数を指定します。使用可能なすべてのスレッドを 使用するには、この値を指定しないか、-1 に設定します。

ARGS ステートメント

オープンソース言語に渡される引数のキーペアリストを指定します。

構文

ARGS<*options*>

オプション引数

KEY=*"string"*

別名 K

注 ARGS ステートメントの各引数に対して一意の KEY=値を常に指定することをお勧めします。

引数リストのキー名を指定します。

NUMVAL=*number*

別名 NVAL

NUM

N

数値引数の値を指定します。

STRVAL=*"string"*

別名 SVAL

STR

S

文字列引数の値を指定します。

詳細

指定された引数には、args(Python)および gw$args(R)からアクセスできます。

OPTS ステートメント

Python や R などの別の言語で記述されたコードを実行します。

構文

OPTS <options> ;

オプション引数

JITTER_TIMEOUT_MILLIS=*integer*

デフォルト 60000

スレッド同期間の最大分散を指定します。1 つのスレッドやセッションが他よりも多くの作業を抱えている場合は、より高い値を設定することが推奨されます。同期タイムアウトなしの場合は、 *-1* に設定します。

TIMEOUT_MILLIS=*integer*

デフォルト 60000

外部言語で記述されたコードが実行されるのを待機する際の最大アイドルタイムアウトを指定します。

LOG_LEVEL=DEBUG | FATAL | WARN | INFO | NOTSET | ERROR

デフォルト INFO

ログレベルを指定します。

SUBMIT ステートメント

Python または R コードのブロックの先頭を特定します。SUBMIT ステートメントと ENDSUBMIT ステートメントの間に、PYTHON または R ステートメントを入力します。

注: 各 SUBMIT ステートメントには、対応する ENDSUBMIT ステートメントが必要です。

構文

SUBMIT;

引数なし

PROC GATEWAY の SUBMIT ステートメントに引数はありません。

ENDSUBMIT ステートメント

Python または R ステートメントのブロックの末尾を特定します。

重要: ENDSUBMIT ステートメントは、コードブロックの後の新しい行で指定する必要があります。

構文

ENDSUBMIT;

引数なし

PROC GATEWAY の ENDSUBMIT ステートメントに引数はありません。

Python 構文

次の Python 関数と変数は、ゲートウェイモジュールを使用した PROC GATEWAY で使用できます。

詳細

read_table

CAS サーバーで利用可能なテーブルを Python クライアントにデータフレームまたは arrow テーブルとして読み取ります。

read_table(table: *dict*, repeat: *boolean*, as_data_frame: *boolean*, batch_size: *integer*)

table: *dict*
castable に渡すことができるテーブル名、caslib、その他のパラメーターをリストした辞書を指定します。

repeat: *boolean*

デフォルト FALSE

すべての Python スレッドとセッションでテーブルを読み取るかを指定します。

as_data_frame: *boolean*

デフォルト TRUE

入力テーブルを pd.DataFrame として返すかを指定します。false に設定すると、入力テーブルは Apache Arrow テーブル pa.table として返されます。

batch_size: *integer*

デフォルト default_read_batch_row_count

バッチごとに返す行数を指定します。

write_table

Python クライアントから CAS サーバーにテーブルを書き込みます。

write_table(df: *pd.DataFrame* | pa.Table, table: *dict*)

df: *pd.DataFrame* | *pa.Table*
入力 pandas データフレームまたは Arrow テーブルを指定します。

table: *dict*
castable に渡すことができるテーブル名、caslib、その他のパラメーターをリストした辞書を指定します。

注 テーブルに書き込むときは、テーブルが各オープンソースセッションにどのように分散されているかを常に確認することをお勧めします。

new_table_reader

CAS テーブルリーダーオブジェクトを作成します。

new_table_reader(table: *dict*, repeat: *boolean*, batch_size: *integer*)

table: *dict*
castable に渡すことができるテーブル名、caslib、その他のパラメーターをリストした辞書を指定します

repeat: *boolean*

デフォルト FALSE

すべての Python セッションでテーブルを読み取るかを指定します。

注: 返されるオブジェクトには、read_batch()、read_all()、read_all_pa()というメソッドがあります。

new_table_writer

CAS バッチライターオブジェクトを作成します。

new_table_writer(table: *dict*)

table: *dict*
castable に渡すことができるテーブル名、caslib、その他のパラメーターをリストした辞書を指定します。

注: バッチライターオブジェクトは、write_batch(pd.DataFrame | pa.Table)メソッドを利用してバッチテーブルを書き込むことができます。バッチライターオブジェクトは、close()メソッドを使用して閉じることができます。

return_table

Python クライアントからのテーブルを ODS テーブルとして返します。すべての Python スレッドからのテーブルは一緒にバインドされます。

return_table(name: *str*, df: *pd.DataFrame* | *pa.Table*, label: *str*, title: *str*)

name: *str*
テーブル名を指定します。

df: *pd.DataFrame* | *pa.Table*
入力 pandas データフレームまたは Arrow テーブルを指定します。

label: *str*

デフォルト " "

テーブルラベルを指定します。

title: *str*

デフォルト " "

テーブルタイトルを指定します。

return_result

引数のリストを SAS ログに出力します。

return_result(name: *str*, **kwargs)

name: *str*
リスト名を指定します。

**kwargs
返される引数を指定します。たとえば、arg1='123'、arg2=.2、arg3=[0, 2]などです。

Python 変数

次の変数は、PROC GATEWAY の使用中に Python セッションで使用できます。

表 27.1 Python 変数

変数名	変数説明
gateway.num_threads	スレッドの総数。
gateway.num_workers	ワーカーの総数。
gateway.thread_id	現在のセッションスレッド ID(0 で始まる)。
gateway.worker_id	現在のセッションのワーカー ID(0 で始まる)。
gateway.job_id	現在のジョブ ID。
gateway.user_id	現在のユーザー ID。
gateway.args	“ARGS ステートメント”によって渡される引数のリスト。

R 構文

ゲートウェイパッケージを使用する PROC GATEWAY の R セッションで次の R 関数と変数を使用できます。R のゲートウェイパッケージをマスクするパッケージをロードする場合は、gateway::function_name(...)を使用して関数にアクセスでき、gateway::gw を使用して変数にアクセスできます。

詳細

read_table

CAS サーバー上で利用可能なテーブルをデータフレームまたは arrow テーブルとして R クライアントに読み取ります。

read_table(table= *list*, rep = *boolean*, as_data_frame = *boolean*, batch_size = *integer*)

tabe= *list*
: castable に渡すことができるテーブル名、caslib、その他のパラメーターを含むリストを指定します。

repeat= *boolean*

デフォルト FALSE

: 各 R セッションでテーブル全体を読み取るかどうかを指定します。それ以外の場合は、CAS ルールに従って分散されます。

as_data_frame= *boolean*

デフォルト TRUE

: 入力テーブルを data.frame として返すかを指定します。false に設定すると、入力テーブルが Arrow Table オブジェクトとして返されます。

batch_size= *integer*

デフォルト 8192

: バッチごとに返す行数を指定します。

write_table

R クライアントから CAS サーバーにテーブルを書き込みます。

write_table(df= *data.frame* | arrowTable, table= *list*)

df= *data.frame* | | table
: 入力 pandas データフレームまたは Arrow テーブルを指定します。

tabe= *list*
: castable に渡すことができるテーブル名、caslib、その他のパラメーターを含むリストを指定します。

new_table_reader

CAS テーブルリーダーオブジェクトを作成します。

注: 返されるオブジェクトには、read_batch()および read_table()というメソッドがあります。

new_table_reader(table= *list*, repeat= *boolean*, batch_size= *integer*)

tabe= *list*
castable に渡すことができるテーブル名、caslib、その他のパラメーターを含むリストを指定します。

repeat= *boolean*

デフォルト FALSE

すべての R セッションでテーブルを読み取るかを指定します。

batch_size= *integer*

デフォルト 8192

バッチごとに返す行数を指定します。

new_table_writer

CAS バッチライターオブジェクトを作成します。

注: バッチライターオブジェクトは、write_batch(data.frame | Table)メソッドを利用してバッチテーブルを書き込むことができます。バッチライターオブジェクトは、close()メソッドを使用して閉じることができます。

new_table_writer(table= *list*)

tabe= *list*
castable に渡すことができるテーブル名、caslib、その他のパラメーターを含むリストを指定します。

return_table

R クライアントからテーブルを ODS テーブルとして返します。すべての R スレッドからのテーブルは一緒にバインドされます。

return_table(df= data.frame | arrowTable, name= *str*, label= *str*, title= *str*)

df= *data.frame* | | table
入力 pandas データフレームまたは Arrow テーブルを指定します。

name= *str*
テーブル名を指定します。

label= *str*

デフォルト " "

テーブルラベルを指定します。

title= *str*

デフォルト " "

テーブルタイトルを指定します。

return_result

引数のリストを SAS ログに出力します。

return_result(name= *str*, *arg-1*, ..., arg-n)

list= *str*
リストの名前を指定します。

arg-1, ..., *arg-n*
返される引数を指定します。たとえば、arg1='123'、arg2=.2、arg3=[0, 2]などです。

R 変数

次の変数は、PROC GATEWAY の使用中に R セッションで使用できます。

表 27.2 R 変数

変数名	変数説明
gw$num_threads	スレッドの総数。
gw$num_workers	ワーカーの総数。
gw$thread_id	現在のセッションスレッド ID(0 で始まる)。
gw$worker_id	現在のセッションのワーカー ID(0 で始まる)。
gw$job_id	現在のジョブ ID。
gw$user_id	現在のユーザー ID。
gw$args	“ARGS ステートメント”によって渡される引数のリスト。

使用法: GATEWAY プロシジャ

PROC GATEWAY と他の SAS プロシジャの比較

PROC PYTHON とは異なり、PROC GATEWAY は対話型 Python または R セッションを起動しません。これは、すべてのセッションでコードを一斉に実行し、その後セッションを終了する、バッチ並列処理用に設計されています。セッション終了後にデータは復元できません。各オープンソースセッションは互いに対話することはなく、CAS サーバーとのみ対話します。

データ型の変換

PROC GATEWAY は、データが arrow テーブルから CAS テーブルに、または CAS テーブルから arrow テーブルに移動されるときに、特定のデータ型を自動的に変換します。現在、CAS テーブルでサポートされていない arrow データ型もあります。

次の表は、現在サポートされているデータ型とサポートされていないデータ型をまとめたものです。

表 27.3 PROC GATEWAY データ型の変換

サポート	arrow データ型	CAS データ型
はい	BINARY	VARBINARY
はい	BOOL	DOUBLE
はい	DATE32	DATE
はい	DATE64	DATETIME
はい	DOUBLE	DOUBLE
はい	FLOAT	DOUBLE
はい	UINT8	INT32
はい	UINT16	INT32
はい	UINT32	INT64
はい	UINT64	INT64
はい	INT8	INT32
はい	INT16	INT32
はい	INT32	INT32
はい	INT64	INT64
はい	LARGE_BINARY	VARBINARY
はい	LARGE_STRING	VARCHAR
はい	STRING	VARCHAR

サポート	arrow データ型	CAS データ型
はい	TIME32	TIME
はい	TIME64	TIME
はい	TIMESTAMP	DATETIME
いいえ	DECIMAL128	
いいえ	DECIMAL256	
いいえ	DENSE_UNION	
いいえ	DICTIONARY	
いいえ	DURATION	
いいえ	EXTENSION	
いいえ	FIXED_SIZE_BINARY	
いいえ	FIXED_SIZE_LIST	
いいえ	HALF_FLOAT	
いいえ	LARGE_LIST	
いいえ	LIST	
いいえ	MAP	
いいえ	NA	
いいえ	SPARSE_UNION	
いいえ	STRUCT	

オープンソースステートメントのサブミット

PROC GATEWAY は常に CAS セッションを必要とします。PROC GATEWAY は、オープンソース言語ステートメントをサブミットする 2 つの方法を提供します。

PROC GATEWAY 起動内の“SUBMIT ステートメント” および“ENDSUBMIT ステートメント”の間でオープンソースステートメントをサブミットできます。PROC

GATEWAY のデフォルト言語は Python です。R コードをサブミットするには、LANG オプションに R を指定する必要があります。次のコードは、1 つの R print ステートメントを実行します。

```
proc gateway lang=R;
submit;
print(paste('hello world', gw$worker_id))
endsubmit;
run;
```

CODE_FILE 引数を使用して、コードを含むファイルへのパスを使用してオープンソースステートメントをサブミットすることもできます。

注: CODE_FILE 引数は、SUBMIT および ENDSUBMIT ステートメントを使用した場合は無視されます。

```
PROC GATEWAY
code_file ="path/to/file/code.py";
run;
```

セッション数の選択

NTHREADS オプションを使用して、各 CAS ワーカーで実行する Python または R セッションの数を設定できます。

```
PROC GATEWAY nthreads=2;
submit;
print('Hello World from thread_id: {}'.format(thread_id))
endsubmit;
run;
```

SINGLE オプションを使用して、単一の Python または R セッションを実行するように指定することもできます。

```
PROC GATEWAY single;
submit;
print('Hello World from thread_id: {}'.format(thread_id))
endsubmit;
run
```

CAS テーブル読み取りと書き込み

PROC GATEWAY は、ゲートウェイ埋め込みモジュールを使用して、各 Python セッションから CAS と直接やり取りし、テーブルの読み書きを可能にします。複数のセッションを使用する場合、クライアントがデータを読み取る際に CAS がデータをセッション間で自動的に分散し、指定されたテーブルにテーブルを 結合して追加します。

```
PROC GATEWAY single;
submit;
```

```
import pandas as pd
df = pd.read_csv('https://support.sas.com/documentation/onlinedoc/viya/
exampledatasets/hmeq.csv')
write_table(df, {'caslib': 'casuser', 'name': 'hmeq', 'replace':True})
df2 = read_table({'caslib': 'casuser', 'name': 'hmeq'})
print(df2.head())
endsubmit;
run;
```

大きなテーブルの場合、“new_table_reader”Python または“new_table_reader”R を使用してテーブルをバッチで 読み取り、書き込むことができます。

```
PROC GATEWAY single;
submit;
inspec = {'name': 'hmeq', 'caslib': 'casuser'}
outspec = {'name': 'hmeq_b', 'caslib': 'casuser', 'replace': True}
with new_table_writer(outspec) as w:
   with new_table_reader(inspec, batch_size = 500) as r:
      for df in r:
      lastbatchsize = df.shape[0]
      print(f'reading batch of size {lastbatchsize} and writing')
      w.write_batch(df)
w.close()
endsubmit;
run;
```

28

GROOVY プロシジャ

概要: GROOVY プロシジャ

GROOVY プロシジャの機能

Groovy は、Java Virtual Machine (JVM)で実行する動的な言語です。PROC GROOVY により、SAS コードが Groovy コードを JVM で実行できるようになります。

PROC GROOVY は、SAS コードの一部として作成された Groovy ステートメントを実行できます。また、PROC GROOVY コマンドで指定したファイル内のステートメントを実行できます。Groovy ステートメントを解析して Groovy Class オブジェク

トを作成し、そのオブジェクトを実行したり、他の PROC GROOVY ステートメントや Java DATA Step オブジェクトで利用できるようにすることができます。PROC GROOVY を使用して、CLASSPATH 環境変数を追加の CLASSPATH 文字列、または jar ファイルへのファイル参照名で更新することもできます。

特記事項

PROC GROOVY とサブミットされる Groovy コードはプロセスの所有者として実行し、プロセスの所有者と同じリソース(ファイルシステム、ネットワークなど)へアクセス権があります。リソースへの Groovy コードアクセスにより、SAS コードが Stored Process Server などのマルチユーザーサーバー内で実行中のときに問題が発生する可能性があります。この機能に対する一部の制御を管理者に付与するため、PROC GROOVY は NOXCMD オプションがオフの場合にのみ実行します。すべての SAS サーバーでは、NOXCMD オプションがオンに設定されています。

SAS ログに Java によって書かれるテキストの最初のバイトのパーセント文字(%)の使用は、SAS によって予約されます。Java テキスト行の最初のバイトでパーセント文字を書く必要がある場合、別のパーセント文字を直後に続ける必要があります(%%)。

PROC GROOVY は、SAS システムオプション THREADS | NOTHREADS はサポートしていません。ただし、PROC GROOVY でサブミットする Groovy コードにより、JVM のスレッド処理を使用できます。

構文: GROOVY プロシジャ

制限事項: NOXCMD オプションがオンの場合は PROC GROOVY は機能しません。

操作: SAS サーバーがロック状態のときは、PROC GROOVY は実行されません。詳細については、“LOCKDOWN” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

```
PROC GROOVY <CLASSPATH=string-or-fileref> <SASJAR= | version= | range=>;
   ADD CLASSPATH=string-or-fileref<SASJAR=<version=>range=>;
   EVALUATE <(LOAD)>
     "Groovy statement string" <argument(s)>;
   EXECUTE <(LOAD)>
     Groovy filename | fileref <argument(s)>;
   SUBMIT <(LOAD)> <argument(s)>;
     Groovy statement(s)
   ENDSUBMIT;
   CLEAR;
QUIT;
```

ステートメント	タスク	例
“PROC GROOVY ステートメント”	SAS コードが Groovy コードを JVM で実行できるようにします	Ex. 1
“ADD ステートメント”	現行の CLASSPATH 環境変数に、一定のクラスパスを追加します	
“EVALUATE ステートメント”	引用符で囲まれた Groovy ステートメントを解析し、Script で Run メソッドを呼び出します	
“EXECUTE ステートメント”	引用符で囲まれたパスまたはファイル参照名として指定されるファイルのコンテンツを読み込み、Script で Run メソッドを呼び出します	
“SUBMIT ステートメント”	SUBMIT コマンドと ENDSUBMIT コマンド間の Groovy ステートメントを解析し、Script で Run メソッドを呼び出します	
“ENDSUBMIT ステートメント”	SUBMIT コマンドで始まる Groovy ステートメントを終了します。	
“CLEAR ステートメント”	バインドを空にし、Groovy クラスローダーをアンロードします。	

PROC GROOVY ステートメント

SAS コードが Groovy コードを JVM で実行できるようにします。

構文

PROC GROOVY <CLASSPATH=*string-or-fileref*> <SASJAR= | *version*= | *range*=>;

オプション引数

CLASSPATH=*string-or-fileref*
引用符で囲まれた CLASSPATH 文字列、または現在のクラスパスに追加される特定の JAR ファイルへのファイル参照名を指定します。このパスは、ユーザーの CLASSPATH 環境変数にあるパスの後に検索されます。

別名 PATH=

SASJAR=<*version*=> | <*range*=>
現在のクラスパスに追加する必要がある Versioned Jar Repository (VJR)の jar を識別する引用符で囲まれた文字列を指定します。VERSION 値と RANGE 値は任意です。次の例のように、RANGE は VERSION よりも優先されます。

```
ADD SASJAR="sas.core";
ADD SASJAR="sas.core" version="903000.9.0.20100810190000_v930";
```

```
ADD SASJAR="sas.core" range="[0,909000]";
```

注: SAS JAR ファイルには、SAS のバージョンにまたがるソース互換性の保証はありません。この jar の将来のバージョンは、通知なしに変わる可能性があります。続行される機能については、SAS テクニカルサポートに連絡してください。

詳細

PROC GROOVY は、現在のユーザーの CLASSPATH 環境変数をそのクラスパスを構築するための基盤として使用します。CLASSPATH オプションと SASJAR オプションを使用して、パスを現在のクラスパスに追加できます。

クラスのロード時は、パスが次の順序で検索されます。

1. プロセス開始時の CLASSPATH 環境変数
2. ADD CLASSPATH ステートメントと ADD SASJAR ステートメントで実行順に追加されたパス

ADD ステートメント

現行の CLASSPATH 環境変数に一定のクラスパスを追加します。

構文

ADD CLASSPATH=*string-or-fileref*<SASJAR=<*version*=>*range*=>;

必須引数

CLASSPATH=*string-or-fileref*
引用符で囲まれた CLASSPATH 文字列、または現在のクラスパスに追加される特定の JAR ファイルへのファイル参照名を指定します。このパスは、ユーザーの CLASSPATH 環境変数にあるパスの後に検索されます。

別名 PATH=

SASJAR=<*version*=> | <*range*=>
現在のクラスパスに追加する必要がある Versioned Jar Repository (VJR)の jar を識別する引用符で囲まれた文字列を指定します。VERSION 値と RANGE 値は任意です。次の例のように、RANGE は VERSION よりも優先されます。

```
ADD SASJAR="sas.core";
ADD SASJAR="sas.core" version="903000.9.0.20100810190000_v930";
ADD SASJAR="sas.core" range="[0,909000]";
```

注: SAS JAR ファイルには、SAS のバージョンにまたがるソース互換性の保証はありません。この jar の将来のバージョンは、通知なしに変わる可能性があります。続行される機能については、SAS テクニカルサポートに連絡してください。

詳細

ADD ステートメントは、指定のクラスパスを現在の CLASSPATH 環境変数に追加します。

CLASSPATH または SASJAR を少なくとも 1 つ指定する必要があります。複数の CLASSPATH または SASJAR を指定できます。

EVALUATE ステートメント

引用符で囲まれた文字列で groovy.lang.Script オブジェクトに付与される Groovy ステートメントを解析し、Script で Run メソッドを呼び出します。

別名: EVAL

構文

EVALUATE <(LOAD)>
"*Groovy statement string*" <*argument(s)*>;

必須引数

Groovy statement string
EVALUATE コマンドによって解析される Groovy ステートメント文字列を指定します。

オプション引数

(LOAD)
Groovy ステートメントを解析して groovy.lang.Script オブジェクトにしますが、実行しません。引数は相互に別名となります。

別名 (PARSEONLY)

(NORUN)

argument(s)
評価中のコードに渡される引数を指定します。

詳細

EVALUATE ステートメントは、引用符で囲まれた文字列で付与される Groovy ステートメントを解析して groovy.lang.Script オブジェクトにし、Script で Run メソッドを呼び出します。LOAD、PARSEONLY、NORUN オプションのいずれかが存在する場合、このステートメントは Groovy ステートメントを解析して Class オブジェクトにしますが、実行しません。Groovy コードによって定義されるクラスは、PROC GROOVY ステートメントまたは Java DATA Step Objects で使用できます。

EVAL は、EVALUATE ステートメントの別名です。

EXECUTE ステートメント

引用符で囲まれた文字列パスまたはファイル参照名として指定されるファイルのコンテンツを読み込みます。

別名: EXEC

構文

EXECUTE <(LOAD)>
Groovy filename | fileref <*argument(s)*>;

必須引数

Groovy filename
EXECUTE ステートメントによって解析される Groovy ファイルの名前を指定します。

fileref
EXECUTE ステートメントによって解析されるファイル参照名の名前を指定します。

オプション引数

(LOAD)
指定された Groovy ファイルまたはファイル参照名の Groovy ステートメントを解析して groovy.lang.Script オブジェクトにしますが、実行しません。引数は相互に別名となります。

別名 (PARSEONLY)

(NORUN)

argument(s)
実行中のコードに渡される引数を指定します。

詳細

EXECUTE ステートメントは、引用符で囲まれた文字列パスまたはファイル参照名として指定されるファイルのコンテンツを読み込みます。コンテンツは解析されて groovy.lang.Script オブジェクトになり、Run メソッドが Script で呼び出されます。LOAD、PARSEONLY、NORUN オプションのいずれかが存在する場合、このステートメントはファイルコンテンツを Class オブジェクトに解析しますが、実行しません。Groovy コードによって定義されるクラスは、PROC GROOVY ステートメントまたは Java DATA Step Objects で使用できます。

EXEC は、EXECUTE ステートメントの別名です。

注: EXEC PARSEONLY ステートメントを使用してファイルを Class にコンパイルした場合、そのファイルへの変更が将来の EXEC PARSEONLY コマンドによって有効化されるように、CLASS ステートメントをサブミットする必要があります。CLEAR ステートメントをサブミットしない場合、EXEC PARSEONLY ステートメントの発行後にファイルに行った変更は、EXEC PARSEONLY ステートメントの後続サブミットによって含まれません。再ロード可能なスクリプトを使用する必要がない場合、GroovyScriptEngine Class を使用できます。

SUBMIT ステートメント

SUBMIT コマンドは SUBMIT コマンドと ENDSUBMIT コマンドの間の Groovy ステートメントを解析して groovy.lang.Script オブジェクトにし、Script で Run メソッドを呼び出します。

構文

SUBMIT <(LOAD)> <*argument(s)*>;
Groovy statement(s)

ENDSUBMIT;

必須引数

Groovy statement(s)
SUBMIT ステートメントによって解析されて groovy.lang.Script オブジェクトになる Groovy ステートメントを指定します。

オプション引数

(LOAD)
Groovy ステートメントを解析して groovy.lang.Script オブジェクトにしますが、実行しません。引数は相互に別名となります。

別名 (PARSEONLY)

(NORUN)

argument(s)
サブミット中のコードに渡される引数を指定します。

詳細

SUBMIT ステートメントは SUBMIT コマンドと ENDSUBMIT コマンドの間の Groovy ステートメントを解析して groovy.lang.Script オブジェクトにし、Script で Run メソッドを呼び出します。LOAD、PARSEONLY、NORUN オプションのいずれかが存在する場合、このステートメントは Groovy ステートメントを解析して Class オブジェクトにしますが、実行しません。Groovy コードによって定義されるクラスは、PROC GROOVY ステートメントまたは Java DATA Step Objects で使用できます。

注:

- ENDSUBMIT ステートメントのみが 1 行にあり、その前にはブランクスペースのみがある必要があります。
- マクロ代替は、SUBMIT コマンドと NDSUBMIT コマンド間で無効化されます。
- 複数行サブミットコマンドを含む PROC GROOVY は、マクロ内では使用できません。

ENDSUBMIT ステートメント

SUBMIT コマンドで始まる Groovy ステートメントを終了します。

構文

ENDSUBMIT;

詳細

SUBMIT コマンドで始まる Groovy ステートメントを終了します。

注: ENDSUBMIT ステートメントのみが 1 行にあり、その前にはブランクスペースのみがある必要があります。

CLEAR ステートメント

バインドを空にし、Groovy クラスローダーをアンロードします。

別名: RESET

構文

CLEAR;

詳細

CLEAR ステートメントはバインドを空にし、Groovy クラスローダーをアンロードします。このステートメントの実行時は、バインドに保存されている変数が使用できなくなります。Groovy クラスローダーにロードされるクラスも使用できなくなります。

RESET は、CLEAR ステートメントの別名です。

注: CLEAR ステートメントと RESET ステートメントのいずれも、System.Properties コレクションまたは CLASSPATH をリセットしません。

使用法: GROOVY プロシジャ

特殊変数

PROC GROOVY には、BINDING、ARGS、EXPORTS、SHELL という 4 つの特殊変数があります。これらの変数を実行中の Groovy コードで使用できるようにします。

BINDING

BINDING 特殊変数は、PROC GROOVY の実行間のオブジェクトの状態の共有に使用されます。スコープなしで作成される変数、またはバインドに明示的に保存される変数によって生成されます。BINDING は、このセクションで説明されているその他すべての特殊変数も保持します。バインドは、CLEAR コマンドによってクリアできます。

注: BINDING 特殊変数は、PROC GROOVY が実行しているすべての Groovy コードで使用できます。

```
proc groovy;
  eval "a = 42";
  eval "binding.b = 84";
  eval "binding.setProperty( 'c', 168 )";
quit;
proc groovy;
  eval "println ""----> ${binding.getProperty('a')}""";
  eval "println ""----> ${b}""";
  eval "println ""----> ${binding.c}""";
quit;
```

ARGS

引数は、バインドの ARGS 特殊変数の Groovy コードに渡されます。

注: ARGS 特殊変数は、PROC GROOVY が実行しているすべての Groovy コードで使用できます。

```
proc groovy;
  eval "args.each{ println ""----> ev ${it}"" }" "arg1" "arg2" "arg3";

  exec "args.groovy" "arg1" "arg2" "arg3";

  submit "arg1" "arg2" "arg3";
    args.each{
      println "----> su ${it}"
    }
  endsubmit;
quit;
```

EXPORTS

EXPORTS 特殊変数には、バインドにマップが含まれます。キーまたは値のペアをこのマップに追加すると、PROC GROOVY の終了時に SAS マクロ変数が作成されま

す。Groovy は大文字と小文字を区別しますが、マクロは区別しません。ケースのみ異なる 2 つのキーがマップに存在する場合、SAS マクロにエクスポートされるキーは決定されません。java.util.Map から継承するオブジェクトを含むバインドの EXPORTS 変数を置き換えることもできます。変数を置き換えると、そのオブジェクトのすべてのキーまたは値のペアがエクスポートされます。

注: EXPORTS 特殊変数は、PROC GROOVY が実行しているすべての Groovy コードで使用できます。

```
proc groovy;
  eval "exports.fname = ""first name""";
  eval "binding.exports.lname = ""last name""";
  eval "exports.put('state', 'NC')";
quit;

data _NULL_;
  put "----> &fname &lname: &state";
run;

proc groovy;
  submit;
    exports = [fname:"first name", lname: "last name", state: "NC"]
  endsubmit;
quit;

data _NULL_;
  put "----> &fname &lname: &state";
run;
```

SHELL

バインドの SHELL 特殊変数は、現在のスクリプトのコンパイルに使用された groovy.lang.GroovyShell に設定されます。この例で execution.groovy ファイルに行われた変更がコードの後続実行で反映されるようにするには、CLEAR ステートメントをサブミットする必要があります。

注: SHELL 特殊変数は、PROC GROOVY が実行しているすべての Groovy コードで使用できます。

```
proc groovy;
  eval "shell.run(
            new File(""execution.groovy""),
            [] as String[] )";
quit;
```

注: 変更時に自動的にリロードされる Groovy スクリプトが必要な場合、GroovyScriptEngine クラスの新しいインスタンスを作成します。

例: GROOVY プロシジャ

例 1: 分類の定義

要素:

PROC GROOVY ステートメントオプション
    CLASSPATH
SUBMIT ステートメントオプション
    PARSEONLY
SUBMIT ステートメント
ENDSUBMIT ステートメント

次の例に、PROC GROOVY を使用したクラスの定義方法を示します。

プログラム

Groovy コードがデフォルトで実行されます。スクリプトに実行可能なコードが含まれていない場合、エラーが返されます。次の例ではクラスを定義しますが、実行可能なコードがなく、エラーが返されます。

```
proc groovy classpath=cp;
 submit;
  class Speaker {
   def say( word ) {
    println "----> \"${word}\""
   }
  }
 endsubmit;
quit;
```

プログラム

次の例では、main メソッドを含めることによる実行可能なクラスの定義方法を示します。

```
proc groovy classpath=cp;
 submit;
  class Speaker {
   def Speaker() {
    println "----> ctor"
```

```
    }
    def main( args ) {
      println "----> main"
    }
  }
 endsubmit;
quit;
```

プログラム

次の例では、PARSEONLY オプションを使用した、実行呼び出しを回避する方法を示します。この新しいクラスは、PROC GROOVY の別の実行で使用できます。

```
proc groovy classpath=cp;
 submit parseonly;
  class Speaker {
    def say( word ) {
      println "----> \"${word}\""
    }
  }
 endsubmit;
quit;

proc groovy classpath=cp;
 eval "s = new Speaker(); s.say( ""Hi"" )";
quit;
```

例 2: マクロ変数を PROC GROOVY に渡す

要素:

SUBMIT ステートメント
ENDSUBMIT ステートメント

次の例は、マクロ変数を定義して PROC GROOVY に渡す方法を示しています。

```
%let _inzip = C:/path/example.zip;
proc groovy;
  submit "&_inzip.";
    def zipFile = new java.util.zip.ZipFile(new File(args[0]))
      zipFile.entries().each {
        println zipFile.getInputStream(it).text
      }

  endsubmit;
quit;
```

29

HADOOP プロシジャ

概要: HADOOP プロシジャ

HADOOP プロシジャの機能

HADOOP プロシジャを使用すると、Apache Hadoop コードを実行することで SAS が Hadoop データとやりとりできます。Apache Hadoop は、Java で作成されたオ

ープンソースフレームワークで、データ記憶域と大量データの分散処理を提供します。

PROC HADOOP は、Hadoop JobTracker とインターフェイスでつながっています。Hadoop JobTracker は、クラスター内の特定のノードに対するタスクを管理する、Hadoop 内のサービスです。PROC HADOOP を使用すると、次をサブミットできます。

- Hadoop Distributed File System (HDFS)コマンド
- MapReduce プログラム(非推奨)
- Pig 言語コード(非推奨)

構文: HADOOP プロシジャ

制限事項: このプロシジャは、CAS サーバーではサポートされません。

SAS がロック状態のときは、PROC HADOOP は使用できません。サーバー管理者は、ロックダウン状態でもこのプロシジャにアクセスできるように再有効化できます。LOCKDOWN ENABLE_AMS=ステートメントを使用して FILENAME Hadoop アクセスメソッドが再有効化されると、PROC HADOOP は自動的に再有効化されます。詳細については、“LOCKDOWN” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

要件 通常、Hadoop クラスターに接続するには、構成が必要です。WebHDFS 経由で Hadoop にアクセスすると、構成が簡素化されます。詳細については、“Hadoop への Java アクセスまたは WebHDFS アクセスの構成” (1031 ページ)を参照してください。

ヒント: サポートされている Hadoop のディストリビューションとバージョンは、SAS/ACCESS Interface to Hadoop と同じです。システム要件については、“Requirements for SAS/ACCESS Interface to Hadoop” (*System Requirements for the SAS Viya Platform*)を参照してください。

PROC HADOOP<*hadoop-server-options*>;

HDFS<*hadoop-server-options*> <*hdfs-command-options*>;

MAPREDUCE<*hadoop-server-options*> <*mapreduce-options*>;

PIG<*hadoop-server-options*> <*pig-code-options*>;

PROPERTIES<*configuration-properties*>;

ステートメント	タスク	例
“PROC HADOOP ステートメント”	Hadoop サーバーへのアクセスを制御します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“HDFS ステートメント”	Hadoop Distributed File System (HDFS)コマンドをサブミットします。	Ex. 1, Ex. 2

ステートメント	タスク	例
“MAPREDUCE ステートメント”	MapReduce プログラムを Hadoop クラスターにサブミットします。	Ex. 3
“PIG ステートメント”	Pig 言語コードを Hadoop クラスターにサブミットします。	Ex. 4

PROC HADOOP ステートメント

Hadoop サーバーへのアクセスを制御します。

例:
“例 1: HDFS コマンドのサブミット” (1032 ページ)
“例 2: ワイルドカード文字を使用した HDFS コマンドのサブミット” (1034 ページ)
“例 3: MapReduce プログラムのサブミット” (1035 ページ)
“例 4: Pig 言語コードのサブミット” (1037 ページ)

構文

PROC HADOOP<*hadoop-server-options*>;

オプション引数の要約

MAXWAIT=*wait-interval*
: WebHDFS を使用する場合の HTTP ステータス応答時間を指定します。

PASSWORD='*password*'
: Hadoop サーバーのユーザー ID のパスワードです。

USERNAME='*ID*'
: Hadoop サーバーの認証ユーザー ID です。

VERBOSE
: 追加的なメッセージが SAS ログに表示されるようにします。

WEBHDFSURL=*web-HDFS-URL*
: REST API を介して Hadoop 分散ファイルシステム(HDFS)にアクセスするための URL を指定します。

Hadoop サーバーオプション

次のオプションで Hadoop サーバーへのアクセスを制御します。次のオプションはすべての HADOOP プロシジャステートメントで指定できます。

MAXWAIT=*wait-interval*
: WebHDFS を使用する場合の HTTP ステータス応答時間を指定します。

デフォルト　40000 ミリ秒

要件　環境変数 SAS_HADOOP_RESTFUL は 1 に設定する必要があります。

ヒント　ログにタイムアウトメッセージが報告されている場合は、MAXWAIT=を使用して待機時間を増やします。

PASSWORD='*password*'
Hadoop サーバーのユーザー ID のパスワードです。 ユーザー ID とパスワードは、CFG=で指定された一連のオプションに追加されます。

別名　PASS=

操作　PASSWORD=を指定するには、USERNAME=も指定する必要があります。

USERNAME='*ID*'
Hadoop サーバーの認証ユーザー ID です。 ユーザー ID とパスワードは、CFG=で指定された一連のオプションに追加されます。

別名　USER=

VERBOSE
追加的なメッセージが SAS ログに表示されるようにします。 VERBOSE は、優れたエラー診断ツールです。SAS を呼び出すときにエラーメッセージが表示された場合は、このオプションを使用して、システムオプションの指定にエラーがあるかどうか確認できます。

WEBHDFSURL=*web-HDFS-URL*
REST API を介して Hadoop 分散ファイルシステム(HDFS)にアクセスするための URL を指定します。 詳細については、“Hadoop への Java アクセスまたは WebHDFS アクセスの構成” (1031 ページ)を参照してください。

注　このオプションは、 2025.08 に追加されました。

例　webhdfsurl="https://my-gateway-host/my-gateway/my-api"

HDFS ステートメント

Hadoop Distributed File System (HDFS)コマンドをサブミットします。

制限事項:　HDFS ステートメントは、1 回の呼び出しで 1 つの操作のみ実行できます。

CAT、CHMOD、および LS コマンドは、webHDFS を介して HDFS コマンドをサブミットする場合にのみ使用できます。

RECURSE オプションとワイルドカードの使用は、webHDFS を介して HDFS コマンドをサブミットする場合にのみ許可されます。

要件　構成要件については、“Hadoop への Java アクセスまたは WebHDFS アクセスの構成” (1031 ページ)を参照してください。

例:　“例 1: HDFS コマンドのサブミット” (1032 ページ)

“例 2: ワイルドカード文字を使用した HDFS コマンドのサブミット” (1034 ページ)

構文

HDFS<*hadoop-server-options*> <*hdfs-command-options*>;

オプション引数の要約

CAT='*HDFS-file*' < **ONLY=***n* > < **OUT='***output-location***'** > < **RECURSE** > < **SHOW_FILENAME** >
: 指定した 1 つ以上のファイルの内容を表示します。

CHMOD='*HDFS-file*' PERMISSION=< ' >*value*< ' > < **RECURSE** >
: 1 つ以上の HDFS ファイルに対するファイルアクセス権限を変更します。

COPYFROMLOCAL='*local-file*' OUT='*output-location*' < **DELETESOURCE** > < **OVERWRITE** > < **RECURSE** >
: 指定されたローカルファイルを HDFS パスの出力先にコピーします。

COPYTOLOCAL='*HDFS-file*' OUT='*output-location*' < **DELETESOURCE** > < **KEEPCRC** > < **OVERWRITE** > < **RECURSE** >
: 指定された HDFS ファイルをローカルファイルの出力先にコピーします。

DELETE='*HDFS-file*' < **NOWARN** >
: 指定された HDFS ファイルを削除します。

LS='*HDFS-pathname*' < **OUT=***output-location* >< **RECURSE** >
: 指定された HDFS パス名にあるファイルをリストします。

MKDIR='*HDFS-pathname*'
: 指定された HDFS パス名を作成します。完全な HDFS パス名を指定します。

RENAME='*HDFS-file*' OUT='*output-location*'
: 指定された HDFS ファイルの名前を変更します。

HDFS コマンドオプション

次のオプションは、HDFS とやりとりするコマンドをサポートします。1 つの HDFS ステートメントに 1 つの操作のみを含めます。

CAT='*HDFS-file*' < **ONLY=***n* > < **OUT='***output-location***'** > < **RECURSE** > < **SHOW_FILENAME** >
: 指定した 1 つ以上のファイルの内容を表示します。

  '***HDFS-file***'
  : パス名、またはパス名とファイル名を指定します。ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して 1 つ以上の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

ONLY=*n*
ファイルの最初から、指定した行数のみを表示します。たとえば、only=10 を指定するとファイルの最初の 10 行が表示されます。このオプションは、ファイルの内容を判別するときに役立ちます。

OUT='*output-location*'
コンテンツの出力先を指定します。マシンの外部にあるファイルにすることも、FILENAME ステートメントで割り当てられたファイル参照名にすることもできます。デフォルトで、出力先は SAS log になります。

RECURSE
指定されたパス名にあるすべてのファイル、およびサブディレクトリにあるすべてのファイルの内容が表示されるように指定します。RECURSE は、指定された HDFS ファイルがディレクトリの場合のみ有効です。

SHOW_FILENAME
出力にファイルの名前を含めます。たとえば、 hdfs cat='/tmp/*.txt' show_filename only=10 recurse;のように指定すると、SAS ログにそのファイルの名前と、/tmp ディレクトリとそのすべてのサブディレクトリで 見つかったすべての.txt ファイルの最初の 10 行が表示されます。

CHMOD='*HDFS-file*' PERMISSION=< ' >*value*< ' > < **RECURSE** >
1 つ以上の HDFS ファイルに対するファイルアクセス権限を変更します。

'*HDFS-file*'
パス名、またはパス名とファイル名を指定します。ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

PERMISSION=*value*
owner、group、および user という 3 つのレベルの権限を表す値を指定します。3 つすべての権限レベルが必要です。read、write、および execute (rwx) という記号表記または 8 進数表記で権限を指定できます。

- rwx 記号表記の場合は、9 文字を使用します。最初の 3 文字のセットは所有者が実行できる事柄を表し、2 番目のセットはグループが実行できる事柄を表し、3 番目のセットはユーザーが実行できる事柄を表します。3 文字のそれぞれのセットについて、最初の位置には r または- (読み取り権限を示す)、2 番目の位置には w または- (書き込み権限を示す)、3 番目の位置には x または- (実行権限を示す)を指定する必要があります。たとえば permission=rwxr-xr-x のように指定すると、所有者には読み取り、書き込み、および実行権限があり、グループメンバーには読み取りおよび実行権限があり、ユーザーには読み取りおよび実行権限があることになります。
- 8 進法表記では、3 つの数字を使用します。各数字は所有者、グループ、ユーザーの権限を表します。各数字は 0 から 7 までにする必要があります。8 進数表記は、rwx 記号表記と同じ数値を表します。つまり、4 は r、2 は w、1 は x、0 は-をそれぞれ表します。たとえば permission=755;のように指定すると、所有者には読み取り、書き込み、および実行権限があり、グループメンバーには読み取りおよび実行権限があり、ユーザーには読み取りおよび実行権限があることになります。

RECURSE
指定されたパス名にあるすべてのファイルとディレクトリ 、およびサブディレクトリにあるすべてのファイルとディレクトリに 対するアクセス権限を変

更するように指定します。RECURSE は、指定された HDFS ファイルがディレクトリの場合のみ有効です。たとえば、hdfs chmod='/tmp' permission=755 recurse;のように指定すると、指定されたディレクトリと、そのディレクトリ内にあるすべてのファイルとサブディレクトリに 対する権限が変更されます。

COPYFROMLOCAL='*local-file*' OUT='*output-location*' < **DELETESOURCE** > < **OVERWRITE** > < **RECURSE** >
指定されたローカルファイルを HDFS パスの出力先にコピーします。

'***local-file***'
完全パス名とファイル名を指定します。ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

OUT='*output-location*'
コピーされたファイルの出力先を指定します。これは完全な HDFS パス名およびファイル名です。

DELETESOURCE
コピーコマンドの後に入力ソースファイルを削除します。

OVERWRITE
既存の出力先を上書きするように指定します。

RECURSE
指定されたパス名にあるすべてのファイル、およびサブディレクトリにあるすべてのファイルをコピーするように指定します。RECURSE は、指定されたファイルがディレクトリの場合のみ有効です。

COPYTOLOCAL='*HDFS-file*' OUT='*output-location*' < **DELETESOURCE** > < **KEEPCRC** > < **OVERWRITE** > < **RECURSE** >
指定された HDFS ファイルをローカルファイルの出力先にコピーします。

'***HDFS-file***'
完全パス名とファイル名を指定します。ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

OUT='*output-location*'
コピーされたファイルの出力先を指定します。これはマシンの外部ファイルです。

DELETESOURCE
コピーコマンドの後に入力ソースファイルを削除します。

KEEPCRC
コピーコマンドの後に巡回冗長チェック(CRC)ファイルをローカル出力先に保存します。CRC ファイルは、OUT=オプションで指定された場所と同じ場所に保存されます。CRC ファイルは、コピーされているファイルの正確さを確認するために使用されます。デフォルトで、CRC ファイルは削除されます。

OVERWRITE
既存の出力先を上書きするように指定します。

RECURSE
指定されたパス名にあるすべてのファイル、およびサブディレクトリにあるすべてのファイルをコピーするように指定します。RECURSE は、指定された HDFS ファイルがディレクトリの場合のみ有効です。

DELETE='*HDFS-file*' < **NOWARN** >
指定された HDFS ファイルを削除します。

HDFS-file
パス名、またはパス名とファイル名を指定します。ファイル名を含めると、そのファイルのみが削除されます。ファイル名を含めない場合は、指定されたパス名にあるすべてのファイルと、サブディレクトリにあるすべてのファイルが削除されます。ワイルドカード文字を使用して、パス名またはファイル名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

NOWARN
存在しないファイルを削除しようとすると表示される警告メッセージを非表示にします。

LS='*HDFS-pathname*' < **OUT=*output-location*** >< **RECURSE** >
指定された HDFS パス名にあるファイルをリストします。 各ファイルの出力はその権限、ユーザー ID、ユーザー ID グループ、ファイルサイズ、作成日、作成時刻、ファイル名で構成されます。

HDFS-pathname
パス名を指定します。ワイルドカード文字を使用して、パス名に含まれる任意の数の文字を置き換えることができます。*を使用して任意の数の文字にマッチさせたり、?を使用して単一の文字にマッチさせたりすることができます。

OUT=*output-location*
ファイルの一覧の出力先を指定します。マシンの外部にあるファイルにすることも、FILENAME ステートメントで割り当てられたファイル参照名にすることもできます。デフォルトで、出力先は SAS log になります。

RECURSE
指定されたパス名にあるファイル、およびサブディレクトリにあるすべてのファイルをリストするように指定します。RECURSE は、指定されたファイルがディレクトリの場合のみ有効です。

デフォルト　デフォルトの出力先は SAS log になります。

MKDIR='*HDFS-pathname*'
指定された HDFS パス名を作成します。完全な HDFS パス名を指定します。

RENAME='*HDFS-file*' OUT='*output-location*'
指定された HDFS ファイルの名前を変更します。

'***HDFS-file***'
名前を変更するパス名とファイル名を指定します。

OUT='*output-location*'
新しい HDFS のパス名とファイル名を指定します。

MAPREDUCE ステートメント

MapReduce プログラムを Hadoop クラスターにサブミットします。

要件 Hadoop サーバーに MapReduce プログラムをサブミットするには、Hadoop 構成ファイルに MapReduce (MR1)または MapReduce 2 (MR2)と YARN を実行するためのプロパティが含まれている必要があります。

操作: MapReduce プログラムを Java API を使用してサブミットするには、SAS クライアントからアクセスできる物理的な場所に Hadoop ディストリビューション JAR ファイルをコピーする必要があります。SAS 環境変数 SAS_HADOOP_JAR_PATH に Hadoop JAR ファイルの場所を設定する必要があります。詳細については、“Hadoop への Java アクセスまたは WebHDFS アクセスの構成” (1031 ページ)を参照してください。

注: 2025.06 以降、MAPREDUCE ステートメントは非推奨になりました。このステートメントはまだ機能していますが、サポートは今後の更新で削除される可能性があります。代わりに、Spark または Hive 機能を使用することをお勧めします。

例: “例 3: MapReduce プログラムのサブミット” (1035 ページ)

構文

MAPREDUCE <*hadoop-server-options*> <*mapreduce-options*>;

オプション引数の要約

COMBINE='*class-name*'
: combiner クラスの名前をドット表記で指定します。

DELETERESULTS
: MapReduce ジョブを開始する前に、出力ディレクトリが存在する場合にはそれを削除するよう指定します。

GROUPCOMPARE='*class-name*'
: grouping comparator (GroupComparator)クラスの名前をドット表記で指定します。

INPUT='*HDFS-pathname*'
: MapReduce 入力ファイルへの HDFS パス名を指定します。

INPUTFORMAT='*class-name*'
: input format クラスの名前をドット表記で指定します。

JAR='*external-file(s)*'
: MapReduce プログラムと名前の付いたクラスを含む JAR ファイルの場所を指定します。

MAP='*class-name*'
: map クラスの名前をドット表記で指定します。

OUTPUT='*HDFS-pathname*'
: Hadoop サーバーに接続している場合、MapReduce 出力用に新しい HDFS パス名を指定します。

OUTPUTFORMAT='*class-name*'
: output format クラスの名前をドット表記で指定します。

OUTPUTKEY='*class-name*'
: output key クラスの名前をドット表記で指定します。

OUTPUTVALUE='*class-name*'
: ドット表記で表された output value クラスの名前です。

PARTITIONER='*class-name*'
: partitioner クラスの名前をドット表記で指定します。

REDUCE='*class-name*'
: reducer クラスの名前をドット表記で指定します。

REDUCETASKS=*integer*
: reduce タスクの数を指定します。

SORTCOMPARE='*class-name*'
: sort comparator クラスの名前をドット表記で指定します。

MapReduce オプション

COMBINE='*class-name*'
: combiner クラスの名前をドット表記で指定します。

DELETERESULTS
: MapReduce ジョブを開始する前に、出力ディレクトリが存在する場合にはそれを削除するよう指定します。

GROUPCOMPARE='*class-name*'
: grouping comparator (GroupComparator)クラスの名前をドット表記で指定します。

INPUT='*HDFS-pathname*'
: MapReduce 入力ファイルへの HDFS パス名を指定します。

INPUTFORMAT='*class-name*'
: input format クラスの名前をドット表記で指定します。

JAR='*external-file(s)*'
: MapReduce プログラムと名前の付いたクラスを含む JAR ファイルの場所を指定します。 完全なパス名とファイル名を含める必要があります。

  要件　各場所を単一引用符または二重引用符で囲みます。

MAP='*class-name*'
: map クラスの名前をドット表記で指定します。 map クラスには、キー値とマップされた値の組み合わせで構成される要素が含まれます。

OUTPUT='*HDFS-pathname*'
: Hadoop サーバーに接続している場合、MapReduce 出力用に新しい HDFS パス名を指定します。

  要件　MapReduce 出力場所を指定する必要があります。

  物理名は単一引用符または二重引用符で囲みます。

OUTPUTFORMAT='*class-name*'
output format クラスの名前をドット表記で指定します。

OUTPUTKEY='*class-name*'
output key クラスの名前をドット表記で指定します。

OUTPUTVALUE='*class-name*'
ドット表記で表された output value クラスの名前です。

PARTITIONER='*class-name*'
partitioner クラスの名前をドット表記で指定します。 partitioner クラスは、中間マップ出力のキーの区分を制御します。

REDUCE='*class-name*'
reducer クラスの名前をドット表記で指定します。 reduce クラスは、キーを共有する中間値のセットを小さい値のセットに縮小します。

REDUCETASKS=*integer*
reduce タスクの数を指定します。

SORTCOMPARE='*class-name*'
sort comparator クラスの名前をドット表記で指定します。

PIG ステートメント

Pig 言語コードを Hadoop クラスターにサブミットします。

操作: Pig 言語コードを Java API を使用してサブミットするには、SAS クライアントからアクセスできる物理的な場所に Hadoop ディストリビューション JAR ファイルをコピーする必要があります。SAS 環境変数 SAS_HADOOP_JAR_PATH に Hadoop JAR ファイルの場所を設定する必要があります。詳細については、"Hadoop への Java アクセスまたは WebHDFS アクセスの構成" (1031 ページ)を参照してください。

注: 2025.06 以降、PIG ステートメントは非推奨になりました。このステートメントはまだ機能していますが、サポートは今後の更新で削除される可能性があります。代わりに、Spark または Hive 機能を使用することをお勧めします。

例: "例 4: Pig 言語コードのサブミット" (1037 ページ)

構文

PIG<*hadoop-server-options*> <*pig-code-options*>;

オプション引数の要約

CODE=*fileref* | '*external-file*'
実行する Pig 言語コードを含むソースを指定します。

DELETERESULTS

Oozie RESTful API を介して Hadoop サーバーに接続している場合、既存の出力先を削除してから Oozie ジョブを開始するように指定します。

OUTPUT='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop サーバーに接続している場合、既存の出力先を削除してから Oozie ジョブを開始するように指定します。

PARAMETERS=*fileref* | '*external-file*'
Pig コードが実行されるときに、引数として渡されるパラメーターを含むソースを指定します。

REGISTERJAR='*external-file(s)*'
実行する Pig スクリプトを含む JAR ファイルの場所を指定します。

REPLACE
Oozie RESTful API を介して Hadoop に接続している場合、Oozie アプリケーションで既存のワークフローと JAR ファイルを削除してから、作業ディレクトリに新しいファイルをコピーするように指定します。

WORKINGDIR='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop に接続している場合、Oozie ワークフローアプリケーションディレクトリ用に HDFS パス名を指定します。

Pig コードオプション

CODE=*fileref* | '*external-file*'
実行する Pig 言語コードを含むソースを指定します。

fileref
ソースファイルに割り当てられる SAS ファイル参照名です。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'***external-file***'
ソースファイルの物理的な場所です。完全パス名とファイル名を指定します。

要件 物理名は単一引用符または二重引用符で囲みます。

DELETERESULTS
Oozie RESTful API を介して Hadoop サーバーに接続している場合、既存の出力先を削除してから Oozie ジョブを開始するように指定します。

操作 DELETERESULTS オプションは OUTPUT=オプションとともに使用します。

OUTPUT='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop サーバーに接続している場合、既存の出力先を削除してから Oozie ジョブを開始するように指定します。

要件 物理名は単一引用符または二重引用符で囲みます。

操作 OUTPUT=オプションは DELETERESULTS オプションとともに使用します。

PARAMETERS=*fileref* | '*external-file*'
Pig コードが実行されるときに、引数として渡されるパラメーターを含むソースを指定します。

fileref
ソースファイルに割り当てられる SAS ファイル参照名です。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'***external-file***'
ソースファイルの物理的な場所です。完全パス名とファイル名を指定します。

要件 物理名は単一引用符または二重引用符で囲みます。

REGISTERJAR='*external-file(s)*'
実行する Pig スクリプトを含む JAR ファイルの場所を指定します。 完全パス名とファイル名を指定します。

要件 各場所を単一引用符または二重引用符で囲みます。

REPLACE
Oozie RESTful API を介して Hadoop に接続している場合、Oozie アプリケーションで既存のワークフローと JAR ファイルを削除してから、作業ディレクトリに新しいファイルをコピーするように指定します。

WORKINGDIR='*HDFS-pathname*'
Oozie RESTful API を介して Hadoop に接続している場合、Oozie ワークフローアプリケーションディレクトリ用に HDFS パス名を指定します。

要件 Oozie RESTFUL API を介して Hadoop に接続している場合、この引数は必須です。

HDFS-pathname 名は単一引用符または二重引用符で囲みます。

PROPERTIES ステートメント

構成プロパティを Hadoop サーバーにサブミットします。

別名: PROP

構文

PROPERTIES '*configuration-property-1*' < '*configuration-property-2*'>...;

必須引数

configuration-property
Hadoop 構成ファイルで指定できるプロパティを指定します。

要件 各プロパティを単一引用符または二重引用符で囲み、各値も単一引用符または二重引用符で囲みます。次に例を示します: prop

```
'mapred.job.tracker'='xxx.us.company.com:8021' 'fs.default.name'='hdfs://
xxx.us.company.com:8020';
```

使用法: HADOOP プロシジャ

Hadoop Distributed File System コマンドのサブミット

Hadoop Distributed File System (HDFS)は、Hadoop フレームワーク用の分散型でスケーラブルなポータブルファイルシステムです。HDFS は、大量のデータをネットワーク上に分散して格納し、多くのクライアントによってデータへのアクセスを 提供する設計になっています。

PROC HADOOP HDFS ステートメントは、Hadoop サーバーに HDFS コマンドをサブミットします。HDFS コマンドは、Hadoop シェルコマンドと同様に、HDFS とのやりとりを行い、ファイルを操作します。HDFS コマンドのリストについては、“HDFS ステートメント” (1020 ページ)を参照してください。

MapReduce プログラムのサブミット

MapReduce は、開発者が大量のデータを処理するプログラムを作成するための並列処理フレームワークです。MapReduce フレームワークには、次の 2 つの主要な関数があります。

- map 関数は処理対象のデータを独立したチャンクに分離します
- reduce 関数はデータに関する分析を実行します

PROC HADOOP MAPREDUCE ステートメントでは、MapReduce プログラムを Hadoop クラスターにサブミットします。詳細については、“MAPREDUCE ステートメント” (1025 ページ)を参照してください。

注: 2025.06 以降、MAPREDUCE ステートメントは非推奨になりました。このステートメントはまだ機能していますが、サポートは今後の更新で削除される可能性があります。代わりに、Spark または Hive 機能を使用することをお勧めします。

Pig 言語コードのサブミット

Apache Pig 言語は、Hadoop で使用される MapReduce プログラムをサブミットする高レベルのプログラミング言語です。

PROC HADOOP PIG ステートメントは、Pig 言語コードを Hadoop クラスターにサブミットします。詳細については、“PIG ステートメント” (1027 ページ)を参照してください。

注: 2025.06 以降、PIG ステートメントは非推奨になりました。このステートメントはまだ機能していますが、サポートは今後の更新で削除される可能性があります。代わりに、Spark または Hive 機能を使用することをお勧めします。

構成プロパティのサブミット

Hadoop 構成ファイルを指定するのではなく、PROC HADOOP PROPERTIES ステートメントで構成プロパティをサブミットできます。詳細については、“PROPERTIES ステートメント” (1029 ページ)を参照してください。

Hadoop への Java アクセスまたは WebHDFS アクセスの構成

PROC HADOOP の構成は、SAS/ACCESS Interface to Hadoop の構成と同じです。詳細については、“Configuring Java Access to Hadoop” (*SAS/ACCESS for Relational Databases: Reference*)を参照してください。

2025.08 以降では、WebHDFS を使用している場合、WEBHDFSURL=オプションで REST URL を指定できます。WEBHDFSURL=オプションを使用すると、Hadoop トレーサスクリプトを実行して Hadoop JAR および構成ファイルを SAS クライアントマシンにコピーする必要はありません。2025.08 より前は、これらの手順が必要でした。さらに、環境変数 SAS_HADOOP_CONFIG_PATH または SAS_HADOOP_RESTFUL=1 を指定する必要がなくなりました。詳細は SAS/ACCESS Interface to Hadoop と同様です。詳細については、“Using WebHDFS Access” (*SAS/ACCESS for Relational Databases: Reference*)を参照してください。

Apache Knox Gateway のセキュリティが有効になっている場合は、WEBHDFSURL=オプションで Knox Gateway エンドポイントを指定します。環境変数 KNOX_GATEWAY_URL を指定する必要がなくなりました。詳細については、“Using WebHDFS Access with Apache Knox Gateway Security” (*SAS/ACCESS for Relational Databases: Reference*)を参照してください。

Hadoop 使用時のセマフォ要件

Hadoop をクエリするたびに、SAS はオペレーティングシステムに一連のセマフォ配列を要求します。(これらの配列は、セマフォセットと呼ばれることもあります)接続が閉じられるかクエリが完了すると、SAS はセマフォ配列を解放してオペレーティングシステムに戻します。

SAS では、Hadoop サーバーのセマフォ配列の制限を少なくとも 256 に設定することをお勧めします。より良い数は 512 です。

注: 配列の最大数に関するセマフォ制限は、システム全体のセマフォの最大数に関するセマフォ制限とは異なります。Linux **ipcs -sl** コマンドは、サーバーに設定されている一般的なデフォルトのセマフォ関連の制限を表示します。

```
------ Semaphore Limits --------
max number of arrays = 512
max semaphores per array = 250
max semaphores system wide = 64000
max ops per semop call = 32
semaphore max value = 32767
```

例: HADOOP プロシジャ

例 1: HDFS コマンドのサブミット

要素:

SAS_HADOOP_CONFIG_PATH 環境変数
SAS_HADOOP_JAR_PATH 環境変数
PROC HADOOP ステートメント
HDFS ステートメント
OPTIONS ステートメント
SET システムオプション

詳細

この PROC HADOOP の例では、HDFS コマンドを Hadoop サーバーにサブミットします。ステートメントで、ディレクトリの作成、ディレクトリの削除、HDFS からローカルの出力先へのファイルのコピーを実行します。

プログラム

```
options set=SAS_HADOOP_CONFIG_PATH="pathname";
options set=SAS_HADOOP_JAR_PATH="pathname";

proc hadoop username='sasabc' password='sasabc' verbose;

  hdfs mkdir='/user/sasabc/new_directory';

  hdfs delete='/user/sasabc/temp2_directory';

  hdfs copytolocal='/user/sasabc/testdata.txt'
      out='C:\Users\sasabc\Hadoop\testdata.txt' overwrite;
run;
```

プログラムの説明

SAS_HADOOP_CONFIG_PATH 環境変数と SAS_HADOOP_JAR_PATH 環境変数を定義します。 OPTIONS ステートメントには、環境変数を定義するための SET システムオプションが含まれています。環境変数によって Hadoop クラスター構成ファイルと Hadoop JAR ファイルの場所が設定されるため、SAS セッションで必要なファイルを使用できるようになります。

```
options set=SAS_HADOOP_CONFIG_PATH="pathname";
options set=SAS_HADOOP_JAR_PATH="pathname";
```

PROC HADOOP ステートメントを抽出します。 PROC HADOOP ステートメントは、Hadoop サーバー上のユーザー ID とパスワードを識別することにより、Hadoop サーバーへのアクセスを制御します。このステートメントは、SAS ログへの追加メッセージの書き込みを可能にする VERBOSE オプションを指定します。

```
proc hadoop username='sasabc' password='sasabc' verbose;
```

HDFS パス名を作成します。 最初の HDFS ステートメントでは、HDFS パス名を作成する MKDIR=オプションを指定しています。

```
  hdfs mkdir='/user/sasabc/new_directory';
```

HDFS ファイルを削除します。 2 つ目の HDFS ステートメントでは、HDFS ファイルを削除する DELETE=オプションを指定しています。

```
  hdfs delete='/user/sasabc/temp2_directory';
```

HDFS ファイルをコピーします。 3 つ目の HDFS ステートメントでは、コピーする HDFS ファイルを指定する COPYTOLOCAL=オプション、ローカルマシン上の出力先

を指定する OUT=オプション、および出力先が存在する場合はその場所を指定して上書きする OVERWRITE オプションを指定しています。

```
    hdfs copytolocal='/user/sasabc/testdata.txt'
        out='C:\Users\sasabc\Hadoop\testdata.txt' overwrite;
run;
```

例 2: ワイルドカード文字を使用した HDFS コマンドのサブミット

要素:

SAS_HADOOP_CONFIG_PATH 環境変数
SAS_HADOOP_JAR_PATH 環境変数
SAS_HADOOP_RESTFUL 環境変数
PROC HADOOP ステートメント
HDFS ステートメント
ワイルドカード文字
OPTIONS ステートメント
SET システムオプション

詳細

この PROC HADOOP の例では、HDFS コマンドを Hadoop サーバーにサブミットします。ステートメントによって、指定されたファイルのコンテンツが表示され、1 つの HDFS ファイルに対する権限が変更され、指定された HDFS パス名にあるファイルがリストされます。

プログラム

```
options set=SAS_HADOOP_CONFIG_PATH="pathname";
options set=SAS_HADOOP_JAR_PATH="pathname";
options set=SAS_HADOOP_RESTFUL 1;

proc hadoop username='sasabc' password='sasabc' verbose;

  hdfs cat='/user/sasabc/*';

  hdfs chmod='/user/sasabc/' permission=rwxr-xr-x;

  hdfs ls='/user/sasabc/*';
run;
```

プログラムの説明

SAS_HADOOP_CONFIG_PATH 環境変数、SAS_HADOOP_JAR_PATH 環境変数、および SAS_HADOOP_RESTFUL 環境変数を定義します。 OPTIONS ステートメントには、環境変数を定義するための SET システムオプションが含まれています。最初の 2 つの環境変数によって Hadoop クラスター構成ファイルと Hadoop JAR ファイルの場所が設定されるため、SAS セッションで必要なファイルを使用できるようになります。SAS_HADOOP_RESTFUL 環境変数は、WebHDFS REST API を使用して Hadoop サーバーに接続することを指定します。

```
options set=SAS_HADOOP_CONFIG_PATH="pathname";
options set=SAS_HADOOP_JAR_PATH="pathname";
options set=SAS_HADOOP_RESTFUL 1;
```

PROC HADOOP ステートメントを抽出します。 PROC HADOOP ステートメントは、Hadoop サーバー上のユーザー ID とパスワードを識別することにより、Hadoop サーバーへのアクセスを制御します。このステートメントは、SAS ログへの追加メッセージの書き込みを可能にする VERBOSE オプションを指定します。

```
proc hadoop username='sasabc' password='sasabc' verbose;
```

HDFS ファイルのコンテンツを表示します。 最初の HDFS ステートメントで、HDFS ファイルのコンテンツを表示する CAT=オプションを指定します。ワイルドカード文字*は、1 つ以上の文字がマッチされるように指定します。ディレクトリ/user/sasabc/に含まれているすべてのファイルが SAS ログに表示されます。

```
hdfs cat='/user/sasabc/*';
```

ファイルアクセス権限を変更します。 2 番目の HDFS ステートメントで、指定された HDFS パス名に対するファイルアクセス権限を変更する CHMOD=オプションを指定します。ファイルアクセス権限により、所有者に読み取り、書き込み、および実行権限が付与され、グループメンバーに読み取りおよび実行権限が付与され、ユーザーに読み取りおよび実行権限が付与されます。

```
hdfs chmod='/user/sasabc/' permission=rwxr-xr-x;
```

HDFS パス名にあるファイルをリストします。 3 番目の HDFS ステートメントで、指定された HDFS パス名にあるファイルを SAS ログにリストする LS=オプションを指定します。ワイルドカード文字*は、1 つ以上の文字がマッチされるように指定します。ディレクトリ/user/sasabc/に含まれているすべてのファイルが SAS ログに表示されます。各ファイルの出力はその権限、ユーザー ID、ユーザー ID グループ、ファイルサイズ、作成日、作成時刻、ファイル名で構成されます。

```
hdfs ls='/user/sasabc/*';
run;
```

例 3: MapReduce プログラムのサブミット

要素:

SAS_HADOOP_CONFIG_PATH 環境変数
SAS_HADOOP_JAR_PATH 環境変数
PROC HADOOP ステートメント
MAPREDUCE ステートメント
FILENAME ステートメント

詳細

この PROC HADOOP の例では、MapReduce プログラムを Hadoop サーバーにサブミットします。このコードは、JavaAPI を使用して MapReduce ジョブを実行します。

注: 2025.06 以降、MAPREDUCE ステートメントは非推奨になりました。このステートメントはまだ機能していますが、サポートは今後の更新で削除される可能性があります。代わりに、Spark または Hive 機能を使用することをお勧めします。

この例では、Hadoop MapReduce のアプリケーションである WordCount を使用します。WordCount はテキスト入力ファイルを読み込み、各行を単語に分割し、単語数をカウントしてから、単語数を出力テキストファイルに書き込みます。

プログラム

```
options set=SAS_HADOOP_CONFIG_PATH="pathname";
options set=SAS_HADOOP_JAR_PATH="pathname";

proc hadoop username='sasabc' password='sasabc' verbose;

  mapreduce input='/user/sasabc/architectdoc.txt'

    output='/user/sasabc/outputtest'

    jar='pathname/WordCount.jar'

    outputkey='org.apache.hadoop.io.Text'

    outputvalue='org.apache.hadoop.io.IntWritable'

    reduce='org.apache.hadoop.examples.WordCount$IntSumReducer'

    combine='org.apache.hadoop.examples.WordCount$IntSumReducer'

    map='org.apache.hadoop.examples.WordCount$TokenizerMapper';
run;
```

プログラムの説明

SAS_HADOOP_CONFIG_PATH 環境変数と SAS_HADOOP_JAR_PATH 環境変数を定義します。 OPTIONS ステートメントには、環境変数を定義するための SET システムオプションが含まれています。環境変数によって Hadoop クラスター構成ファイルと Hadoop JAR ファイルの場所が設定されるため、SAS セッションで必要なファイルを使用できるようになります。

```
options set=SAS_HADOOP_CONFIG_PATH="pathname";
options set=SAS_HADOOP_JAR_PATH="pathname";
```

PROC HADOOP ステートメントを抽出します。 PROC HADOOP ステートメントは、Hadoop サーバー上のユーザー ID とパスワードを識別することにより、Hadoop サーバーへのアクセスを制御します。このステートメントは、SAS ログへの追加メッセージの書き込みを可能にする VERBOSE オプションを指定します。

```
proc hadoop username='sasabc' password='sasabc' verbose;
```

MapReduce プログラムをサブミットします。 MAPREDUCE ステートメントには、いくつかのオプションが含まれています。INPUT=では、ArchitectDoc.txt という名前の入力 Hadoop ファイルの HDFS パス名とファイル名を指定しています。

```
mapreduce input='/user/sasabc/architectdoc.txt'
```

HDFS パス名を作成します。 OUTPUT=では、OutputTest という名前のプログラム出力先の HDFS パス名を指定しています。

```
output='/user/sasabc/outputtest'
```

JAR ファイルを指定します。 JAR=では、WordCount.jar という名前の MapReduce プログラムを含む JAR ファイルを指定しています。

```
jar='pathname/WordCount.jar'
```

output key クラスを指定します。 OUTPUTKEY=は、出力キークラス org.apache.hadoop.io.Text の名前を指定します。

```
outputkey='org.apache.hadoop.io.Text'
```

output value クラスを指定します。 OUTPUTVALUE=では、output value クラスの名前 org.apache.hadoop.io.IntWritable を指定しています。

```
outputvalue='org.apache.hadoop.io.IntWritable'
```

reducer クラスを指定します。 REDUCE=では、reducer クラスの名前 org.apache.hadoop.examples.WordCount$IntSumReducer を指定しています。

```
reduce='org.apache.hadoop.examples.WordCount$IntSumReducer'
```

combiner クラスを指定します。 COMBINE=では、combiner クラスの名前 org.apache.hadoop.examples.WordCount$IntSumReducer を指定しています。

```
combine='org.apache.hadoop.examples.WordCount$IntSumReducer'
```

map クラスを指定します。 MAP=では、map クラスの名前 org.apache.hadoop.examples.WordCount$TokenizerMapper を指定しています。

```
map='org.apache.hadoop.examples.WordCount$TokenizerMapper';
run;
```

例 4: Pig 言語コードのサブミット

要素:

SAS_HADOOP_CONFIG_PATH 環境変数
SAS_HADOOP_JAR_PATH 環境変数
PROC HADOOP ステートメント
PIG ステートメント
FILENAME ステートメント

詳細

この PROC HADOOP の例では、Pig 言語コードを Hadoop クラスターにサブミットします。このコードは、JavaAPI を使用して Pig ジョブを実行します。

注: 2025.06 以降、PIG ステートメントは非推奨になりました。このステートメントはまだ機能していますが、サポートは今後の更新で削除される可能性があります。代わりに、Spark または Hive 機能を使用することをお勧めします。

これが Pig 言語コードで、sample_pig.txt という名前のテキストファイルに保存されています。

```
A = LOAD '/user/sasabc/class.txt' USING PigStorage(' ')
AS (name,grade,age,height,weight);
B = FOREACH A GENERATE name, grade, some.custom.class.promotionDecision(grade) as promote;
store B into '/user/sasabc/pigout' USING PigStorage(',');
```

Pig コードは、class.txt という名前のデータファイルを参照します。Pig コードは、some.custom.class.promotionDecision()という名前のユーザー定義の Java 関数も呼び出して、データを操作します。ユーザーがその関数を作成し、JAR ファイルで使用できるようにしました。

プログラム

```
options set=SAS_HADOOP_CONFIG_PATH="pathname";
options set=SAS_HADOOP_JAR_PATH="pathname";

filename mycode 'pathname/sample_pig.txt';

proc hadoop username='sasabc' password='sasabc' verbose;

  pig code=mycode registerjar='pathname/gradeAnalysis.jar';
run;
```

プログラムの説明

SAS_HADOOP_CONFIG_PATH 環境変数と SAS_HADOOP_JAR_PATH 環境変数を定義します。 OPTIONS ステートメントには、環境変数を定義するための SET システムオプションが含まれています。環境変数によって Hadoop クラスター構成ファイルと Hadoop JAR ファイルの場所が設定されるため、SAS セッションで必要なファイルを使用できるようになります。

```
options set=SAS_HADOOP_CONFIG_PATH="pathname";
options set=SAS_HADOOP_JAR_PATH="pathname";
```

ファイル参照を Pig 言語コードに割り当てます。 この FILENAME ステートメントでは、Sample_Pig.txt という名前の Pig 言語コードを含むファイル(上記)の物理的な場所にファイル参照 MYCODE を割り当てます。

```
filename mycode 'pathname/sample_pig.txt';
```

PROC HADOOP ステートメントを実行します。 PROC HADOOP ステートメントは、Hadoop サーバー上の USERNAME=および PASSWORD=オプションによるユーザー ID とパスワードを識別することにより、Hadoop サーバーへのアクセスを制御します。このステートメントは、SAS ログへの追加メッセージの書き込みを可能にする VERBOSE オプションを指定します。

```
proc hadoop username='sasabc' password='sasabc' verbose;
```

PIG ステートメントを実行します。 CODE=オプションは、Pig 言語コードを含むファイルの物理的な場所に割り当てられる SAS ファイル参照名 MYCODE を指定します。REGISTERJAR=オプションは、ユーザーが作成した JAR ファイル gradeAnalysis.jar を指定して、ユーザー定義関数を提供します。

```
  pig code=mycode registerjar='pathname/gradeAnalysis.jar';
run;
```

30

HTTP プロシジャ

概要: HTTP プロシジャ

HTTP プロシジャの機能

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

PROC HTTP では、制約のない一連のメソッドが許可されています。PROC HTTP は、標準 HTTP メソッドの他に、HTTP/1.1 標準に準拠していてターゲットの Web サーバーによって認識される任意のメソッドを受け入れます。また、永続的な接続、Cookie のキャッシング、EXPECT_100_CONTINUE サポートなどの HTTP/1.1 機能も実装しており、認証の種類を指定できます。

これをサポートする Web サーバーについては、PROC HTTP はデフォルトで接続キャッシングと Cookie キャッシングを使用します。プロシジャ内で、この 2 つの種類のキャッシングの動作を切り替えて、キャッシュをクリアするには、プロシジャの引数を指定します。または、マクロ変数を使用して Cookie キャッシュをオフにすることもできます。

認証の指定機能を使用すると、要求に対して 1 つ以上の認証の種類を指定できます。

フォームデータを簡単に投稿したり、HTTP マルチパート要求を実行したりできます。QUERY=プロシジャオプションは、URL=引数にクエリパラメーターをサブミットするプロセスを簡素化します。詳細については、“PROC HTTP での URL エンコードデータの使用” (1059 ページ)および“MULTI オプションを使用したマルチパートデータのアップロード” (1059 ページ)を参照してください。

このプロシジャには、DEBUG ステートメント、応答ステータスマクロ変数、要求のタイムアウト期間を指定する機能が含まれています。SSLPARMS ステートメントを使用すると、PROC HTTP 要求で SSL オプションを指定できます。

構文: HTTP プロシジャ

制限事項:

このプロシジャは、CAS サーバーではサポートされません。

SAS がロック状態のときは、HTTP プロシジャは有効です。プロシジャは、LOCKDOWN ステートメントの ENABLE_AMS=オプションを使用して再有効化することができます。DS2 HTTP パッケージまたは FILENAME、URL アクセス方式が再度有効になると、HTTP プロシジャは自動的に再度有効になります。詳細については、“LOCKDOWN” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

注: 2026.03 以降では、FILENAME URL または PROC HTTP がロックダウンモードになっている場合、URL パスは SAS 管理者が指定した URL 許可リストによって制御できます。

ヒント: PROC HTTP は、各ステートメントの実行後、特定の値を使用してマクロ変数を設定します。これらのマクロ変数をマクロ内で使用して、HTTP エラーをテストできます。詳細については、“PROC HTTP 応答ステータスレポート” (1061 ページ)を参照してください。

PROC HTTPURL="*URL-to-target*" <*options*>;

DEBUG *options*;

HEADERS "*HeaderName*"="*HeaderValue*"
< "*HeaderName-n*"="*HeaderValue-n*">;

SSLPARMS *host-specific-SSL-options*;

ステートメント	タスク	例
“PROC HTTP ステートメント”	HTTP 要求の発行	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 9, Ex. 10, Ex. 11
“DEBUG ステートメント”	デバッグメッセージを表示します	Ex. 12
“HEADERS ステートメント”	HTTP 要求の要求ヘッダーの指定	Ex. 8

PROC HTTP ステートメント

要求を発行する Web サービスを起動します。

例:

構文

PROC HTTP URL="*URL-to-target*</redirect/*n*>"
<METHOD=<">*http-method*<">>
<*authentication-type-options*>
<*caching-options*>
<*header-options*>
<*proxy-server-connection-options*>
<*web-server-authentication-options*>
<EXPECT_100_CONTINUE>
<FOLLOWLOC | NOFOLLOWLOC>
<IN=<*fileref* | "*string*" | FORM (*arguments*) | MULTI <FORM | "type">
(*inputs*)>>
<MAXREDIRECTS=*n*>
<OUT=*fileref*>
<QUERY=("*parm1*"="*value1*" "*parm2*="*value2*" ...)>
;

オプション引数の要約

EXPECT_100_CONTINUE
ターゲットサーバー側に要求を受け入れる用意があるかどうかを、クライアントが判断できるようにします。

FOLLOWLOC
書き込みメソッドが URL リダイレクトに自動的に従えるようにします。

IN=*fileref* | "*string*" | FORM (*arguments*) | MULTI <FORM | "type" > (*inputs*)
入力データを指定します。

MAXREDIRECTS=*n*
許可されるリダイレクトの最大数を指定します。

METHOD=<">*http-method*<">
HTTP メソッドを指定します。

NOFOLLOWLOC
GET メソッドが URL リダイレクトを追従できないようにします。

OUT=*fileref-to-response-data*
出力が書き込まれるファイル参照名を指定します。

QUERY=(<NOENCODE>"*parm1*"="*value1*" "*parm2*"="*value2*")
URL=引数のクエリパラメーターをサブミットするための代替メソッドを提供します。

TIMEOUT=*integer*
HTTP 要求をキャンセルする前の待ち時間としての非アクティブな秒数を指定します。

Authenticate to Web Server

WEBPASSWORD="*basic-authentication-password*"

基本認証のためのパスワードを指定します。

WEBUSERNAME="*basic-authentication-name*"
基本認証のためのユーザー名を指定します。

Connect to Proxy Server

PROXYHOST="*proxy-host-name*"
HTTP プロキシサーバーのインターネットホスト名を指定します。

PROXYPASSWORD="*proxy-passwd*"
HTTP プロキシサーバーのパスワードを指定します。

PROXYPORT=*proxy-port-number*
HTTP プロキシサーバーのポートを指定します。

PROXYUSERNAME="*proxy-user-name*"

Disable Shared Connection and Cookie Caching

CLEAR_CACHE
HTTP 要求の実行前に、共有接続キャッシュと Cookie キャッシュの両方がクリアされるように指定します。

CLEAR_CONN_CACHE
HTTP 要求の実行前に、共有接続キャッシュがクリアされるように指定します。

CLEAR_COOKIES
HTTP 要求の実行前に、共有 Cookie キャッシュがクリアされるように指定します。

NO_CONN_CACHE
このプロシジャ実行の接続キャッシングを無効にします。

NO_COOKIES
キャッシュされた Cookie がこのプロシジャ実行に使用されないように指定します。

Specify Authentication Type

AUTH_ANY
接続サーバーに対する認証に、任意の種類の認証を使用できることを指定します。

AUTH_BASIC
接続サーバーに対する認証に、ユーザー ID 認証が使用されるように指定します。

AUTH_NEGOTIATE
接続サーバーに対する認証に Kerberos またはその他の種類の HTTP 認証が使用されるように指定します

AUTH_NONE
基本認証を使用しないこと、または認証をネゴシエートすることを指定します。

OAUTH_BEARER=*token*
HTTP 呼び出しとともに OAuth アクセストークンを送信します。

PROXY_AUTH_BASIC
プロキシサーバー経由のユーザー ID 認証が実行されるように指定します。

PROXY_AUTH_NEGOTIATE

プロキシサーバー経由の Kerberos またはその他の種類の HTTP 認証が実行されるように指定します。

Specify HTTP Headers

CT="*content-type*"
: 要求ヘッダーの送信する HTTP コンテンツの種類を指定します。

HEADERIN=*fileref-to-request-header-file*
: 出力形式 *key:value* の要求ヘッダーごとに 1 行を含むテキストファイルへのファイル参照名を指定します。

HEADEROUT_OVERWRITE
: リダイレクトの発生時に Web サーバーによって送信された最後のヘッダーブロックのみが、応答ヘッダーで記録されるようにします。

HEADEROUT=*fileref-to-response-header-file*
: 応答ヘッダーが出力形式 *key:value* で書き込まれるテキストファイルへのファイル参照名を指定します。

必須引数

URL="*URL-to-target*"
: HTTP 要求のエンドポイントを識別する完全修飾 URL パスを指定します。

注 PROC HTTP に渡される URL は、URL エンコードされていると見なされます。確実かつ適切にエンコードするには、ターゲット Web サーバーに適した接続クラスを使用します。たとえば、Amazon Web Services には AWSV4Signer クラスを使用します。または、RFC3986 の説明に従って、予約された文字をエンコードします。

ヒント URL で HTTP プロトコルを指定する必要はありません。パス(たとえば、"httpbin.org")を設定した場合、使用される実際の URL は http://httpbin.org です。

オプション引数

AUTH_ANY
: ユーザー名とパスワードを指定した場合は、そのユーザー名とパスワードが、接続サーバーの認証に使用されます。それ以外の場合は、使用可能な他の認証フォームが使用されます。AUTH_ANY を指定した場合の動作は、プロシジャステートメントで AUTH_NEGOTIATE、および AUTH_BASIC を指定する場合と同じです。

デフォルト 認証の種類が指定されていない場合は、これがデフォルトの認証の種類として使用されます。

ヒント HTTP サーバーには複数回アクセスする可能性があるため、データが複数回アップロードされないように、EXPECT_100_CONTINUE を指定します。

AUTH_BASIC
: 接続サーバーの認証に、ユーザー ID 認証が使用されるように指定します。ユーザー名とパスワードは、WEBUSERNAME 引数と WEBPASSWORD 引数で指定されます。

AUTH_NEGOTIATE
接続サーバーに対する認証に Kerberos またはその他の種類の HTTP 認証が使用されるように指定します現在のユーザー ID に権限がある場合は、認証が確立されます。

AUTH_NONE
基本認証を使用しないこと、または認証をネゴシエートすることを指定します。

OAUTH_BEARER=プロシジャオプションは、NO_AUTH とともに使用できます。

CLEAR_CACHE
HTTP 要求の実行前に、共有接続キャッシュと Cookie キャッシュの両方がクリアされるように指定します。

CLEAR_CONN_CACHE
HTTP 要求の実行前に、共有接続キャッシュがクリアされるように指定します。

CLEAR_COOKIES
HTTP 要求の実行前に、共有 Cookie キャッシュがクリアされるように指定します。

CT="*content-type*"
HEADERIN=引数とともに使用され、要求ヘッダーで設定される HTTP コンテンツの種類を指定します。コンテンツの種類には、受信側のユーザーエージェントがユーザーにデータを表示するために十分であるように、本体に含まれるデータを記述します。

コンテンツの種類の指定例は次のとおりです。

```
CT="Text/HTML; charset=ISO-8859-4"

CT="Text/plain; charset=us-ascii"

CT="Application/x-www-form-urlencoded"
```

注 このオプションは、SAS 9.4 PROC HTTP プログラムとの互換性のためにサポートされています。“HEADERS ステートメント” (1054 ページ)を CT=の代わりに使用します。

EXPECT_100_CONTINUE
要求本文が含まれる要求メッセージを送信するクライアントが、ターゲットサーバー側にその要求を受け入れる用意があるかどうかを、要求ヘッダーに基づいて判断できるようにします。大きなデータを送信する場合、不要なデータ転送を行いたくないときに、EXPECT_100_CONTINUE を使用します。詳細については、http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.2.3 を参照してください。

該当要素 PUT で最もよく使用される、IN=引数を指定する HTTP 要求。

操作 このオプションは、HEADEROUT=引数とともに使用されます。

例 “例 10: EXPECT_100_CONTINUE を指定する PUT” (1070 ページ)

FOLLOWLOC
書き込みメソッドが URL リダイレクトに自動的に従えるようにします。 デフォルトでは、POST や PUT などのデータを書き込む PROC HTTP メソッドは、別の場所にリダイレクトされると処理を終了します。FOLLOWLOC が指定されてい

る場合、PROC HTTP は最初に 300 レベルの応答を返し、次に POST または PUT をリダイレクトされた場所に再度サブミットします。

FOLLOWLOC は、GET メソッドのデフォルトの動作です。

参照項目: “NOFOLLOWLOC” (1051 ページ)

HEADERIN=*fileref-to-request-header-file*
出力形式 *key:value* の要求ヘッダーごとに 1 行を含むテキストファイルへのファイル参照名を指定します。

注 このオプションは、SAS 9.4 PROC HTTP プログラムとの互換性のためにサポートされています。“HEADERS ステートメント” (1054 ページ)を HEADERIN=の代わりに使用します。

注意 **HEADERS ステートメントと HEADERIN=引数の両方を指定しないでください。** 両方のオプションを一緒に指定した場合の動作は定義されていません。

HEADEROUT=*fileref-to-response-header-file*
応答ヘッダーが出力形式 *key:value* で書き込まれるテキストファイルへのファイル参照名を指定します。

例 “例 6: 応答ヘッダーを取得する POST” (1067 ページ)

“例 10: EXPECT_100_CONTINUE を指定する PUT” (1070 ページ)

HEADEROUT_OVERWRITE
HEADEROUT=引数とともに使用され、リダイレクトの発生時に Web サーバーによって送信された最後のヘッダーブロックのみが、応答ヘッダーで記録されるようにします。

例 “例 7: HEADEROUT_OVERWRITE を指定する GET” (1067 ページ)

IN=*fileref* | "*string*" | FORM (*arguments*) | MULTI <FORM | "type" > (*inputs*)
入力データを指定します。入力データをサブミットする方法は複数あります。

fileref
ファイル参照名を指定します。ファイル参照名は、別の場所に存在するデータへのポインターです。ファイル参照名は、FILENAME ステートメントで割り当てられます。

例 “例 2: 単純な POST 要求” (1063 ページ)

“string”
入力データを引用符で囲まれた文字列で指定します。

例 “例 4: 入力データを文字列として指定” (1064 ページ)

FORM (“*name1*”=“*value1*” “*name2*”=“*value2*” ...)
入力データを標準の HTML フォームとして送信します。データは名前=値ペアとして入力されます。FORM オプションは、フォームサブミットの標準として、各ペアを URL エンコードして&(アンパサンド)で区切ります。

name=value のペアを URL エンコードしない場合は、name=value のペアの前に NOENCODE フラグを指定できます。

参照項目: “PROC HTTP での URL エンコードデータの使用” (1059 ページ)

MULTI <FORM | "*type*"> (*inputs*)
MULTI はマルチパートアップロードを指定します。

FORM
multipart/form-data と呼ばれる特殊なマルチパートタイプを使用してアップロードが実行されることを指定します。FORM を使用すると、"content-disposition"という名前の特別なヘッダーが自動的に生成され、各入力とともに送信されます。

この生成された content-disposition ヘッダーの形式は次のとおりです。

```
Content-Disposition: form-data;
 name="field-name";
 <filename=filename.ext">
```

"*type*"
要求が一般的なマルチパート要求であることを示します。一般的なマルチパート要求では、ユーザーは生成されたコンテンツの種類ヘッダーに含める値を指定します。たとえば、引用された値が“mixed”の場合、ヘッダー Content-Type: multipart/mixed 送信されます。引用された値が“related”の場合、ヘッダー Content-Type: multipart/related 送信されます。

(*inputs*)
かっこ内にカンマ区切りの入力のリストを指定します。

フォームデータ要求の入力の構文は次のとおりです。

***"Name"=value* <FILENAME="*filename*" | NOFILENAME>**

"Name"=value
(必須)フォームの基本的なコンテンツを説明します。"*Name*"は、対応するコントロールのコントロール名です。

Value は、ファイル参照名または引用符で囲まれた文字列にすることができます。ファイル参照 TEMP は許可されます。

生成された content-disposition ヘッダーの **Filename=**フィールドは、ファイル名がファイル参照名から判別できる場合に自動的に生成されます。

FILENAME="*filename*"
(オプション)自動的に決定できない場合に、生成された content-disposition ヘッダーの **Filename=**フィールドのファイル名を指定します。

NOFILENAME
(オプション)生成された content-disposition ヘッダーから **Filename=**フィールドを省略することを指定します。このオプションを使用すると、フォームに関する限り、ファイルではない入力としてファイル参照名を使用できます。

一般的なマルチパート要求の入力の構文は次のとおりです。

value <header>

value
ファイル参照名または引用符で囲まれた文字列です。

Header
(オプション) 0 個以上のヘッダーを指定します。各ヘッダーは次の形式の適切なヘッダーにします。

header="header-value"

たとえば、次のように指定します。

header="Content-Type: application/json"

注 ヘッダーは、パート内のキーワードの後に指定する必要があります。

要件 POST メソッドおよび PUT メソッドが使用されている場合、IN=オプションは必須です。

注 MULTI オプションには、ネストされたマルチパートアップロードのサポートは含まれていません。マルチパートコンテンツの種類の詳細については、 https: //www.w3.org/Protocols/rfc1341/7_2_Multipart.html を参照してください。

例 “MULTI オプションを使用したマルチパートデータのアップロード” (1059 ページ)

MAXREDIRECTS=*n*
許可されるリダイレクトの最大数を指定します。 PROC HTTP は、NOFOLLOWLOC オプションが指定されていない限り、300 レベルの応答コードからのリダイレクトに自動的に従います。このオプションを使用すると、実行されるリダイレクトの数を制限できます。

デフォルト 5

METHOD=<">*http-method*<">
HTTP メソッドを指定します。標準メソッドには、HEAD、TRACE、GET、POST、PUT、DELETE が含まれます。ただし、このメソッドには制約がありません。HTTP/1.1 標準に準拠し、ターゲットの Web サーバーによって認識される任意のメソッドが受け入れ可能です。詳細については、HTTP/1.1 の仕様(www.w3.org)を参照してください。

デフォルト METHOD 引数を省略し、IN 引数を指定しない場合、デフォルトメソッドは GET です。METHOD 引数を省略し、IN 引数を指定する場合、デフォルトメソッドは POST です。

例 “例 1: 単純な GET 要求” (1062 ページ)

“例 2: 単純な POST 要求” (1063 ページ)

“例 9: 非標準のメソッド” (1070 ページ)

NO_CONN_CACHE
この HTTP 要求の接続キャッシングを無効にします。接続は、指定した接続パラメーターによって確立されます。

注 NO_CONN_CACHE を有効にすると、キャッシュされた認証を含む、キャッシュされた接続のすべての利点が失われます。認証を使用する各呼び出しは再認証が必要であり、これには時間がかかる場合があります。

NO_COOKIES
キャッシュされた Cookie がこの HTTP 要求に使用されないように指定します。このオプションを使用しても、"Cookie"ヘッダーによって Cookie を手動で送信することは可能です。

NOFOLLOWLOC
GET メソッドが URL リダイレクトを追従できないようにします。

NOFOLLOWLOC は、データを書き込む HTTP メソッドのデフォルトの動作です。

参照項目: “FOLLOWLOC” (1047 ページ)

OAUTH_BEARER=*token*
HTTP 呼び出しとともに OAuth アクセストークンを送信します。 有効なトークン値は、文字列、ファイル参照名、または定数 SAS_SERVICES です。文字列値は引用符で囲む必要があります。すべてのトークンタイプについて、引数は Authorization: Bearer Value の形式で承認ヘッダーを送信します。

ヘッダーを追加するだけでなく、OAUTH_BEARER=は他の種類の HTTP 認証を無効にします。戻る HTTP 認証ヘッダーは無視されます。

OUT=*fileref-to-response-data*
出力が書き込まれる場所を示すファイル参照名を指定します。

例 “例 2: 単純な POST 要求” (1063 ページ)

PROXY_AUTH_BASIC
プロキシサーバー経由のユーザー ID 認証が実行されるように指定します。ユーザー名とパスワードは、PROXYUSERNAME 引数と PROXYPASSWORD 引数で指定されます。

PROXY_AUTH_NEGOTIATE
プロキシサーバー経由の Kerberos またはその他の種類の HTTP 認証が実行されるように指定します。現在のユーザー ID に権限がある場合は、認証が確立されます。

PROXYHOST="*proxy-host-name*"
HTTP プロキシサーバーのインターネットホスト名を指定します。PROXYHOST 引数にホスト名とポート番号の両方を次の形式で指定できます。

host-name:port-number

この構文を使用すると、PROXYPORT 引数を指定する必要はありません。

例 “例 3: HTTP 要求でプロキシを指定する” (1064 ページ)

PROXYPASSWORD="*proxy-passwd*"
HTTP プロキシサーバーのパスワードを指定します。

ヒント パスワードは、プロキシサーバーで認証情報が必要な場合のみ要求されます。

PROC PWENCODE によって生成されるエンコーディングがサポートされます。

PROXYPORT=*proxy-port-number*
オプションとして HTTP プロキシサーバーのポートを指定します。PROXYHOST 引数に HTTP プロキシーサーバーのホスト名とポート番号の両方を指定した場合は、PROXYPORT を指定する必要はありません。

PROXYUSERNAME="*proxy-user-name*"
HTTP プロキシサーバーのユーザー名を指定します。

ヒント ユーザー名は、プロキシサーバーで認証情報が必要な場合のみ要求されます。

QUERY=(<NOENCODE>"*parm1*"="*value1*" "*parm2*"="*value2*")
URL=引数のクエリパラメーターをサブミットするための代替メソッドを提供します。

name=value ペアの前にキーワード NOENCODE を指定して、そのペアを URL エンコードしないことを示すことができます。NOENCODE は、それが先行するペアにのみ適用されます。スペースのように、通常はエンコードされている他の文字は引き続きエンコードされます。

注 QUERY=が指定されていて、入力 URL に既存のクエリ文字列がすでに含まれていると、生成された置換文字列全体で?の代わりに&が使用されます。

参照項目: “PROC HTTP での URL エンコードデータの使用” (1059 ページ)

TIMEOUT=*integer*
HTTP 要求をキャンセルする前の待ち時間としての非アクティブな秒数を指定します。 サーバーが応答しない可能性がある場合、ハングを防止するには、このオプションを使用します。既定値の 0 (ゼロ)は、タイムアウト期間がないことを意味します。

WEBPASSWORD="*basic-authentication-password*"
基本認証のためのパスワードを指定します。

別名 PASSWORD

ヒント PROC PWENCODE によって生成されるエンコーディングがサポートされます。

WEBUSERNAME="*basic-authentication-name*"
基本認証のためのユーザー名を指定します。

別名 USERNAME

DEBUG ステートメント

デバッグ情報を SAS ログに書き出します。

サポート: すべての HTTP メソッド

例: “例 12: LEVEL=オプションを指定して DEBUG ステートメントを使用” (1075 ページ)

構文

DEBUG *options*;

オプション引数

Level= 0 | 1 | 2 | 3

0
デバッグはありません。これは、DEBUG ステートメントなしで PROC HTTP を指定するのと同じです。

1
ログに要求ヘッダーと応答ヘッダーを表示します。

デバッグレベルを 1 に設定することは、DEBUG ステートメントで REQUEST_HEADERS および RESPONSE_HEADERS オプションを設定することと同じです。

2
要求データとレベル 1 のメッセージをログに表示します。

デバッグレベルを 2 に設定することは、DEBUG ステートメントで REQUEST_HEADERS、RESPONSE_HEADERS、および REQUEST_BODY オプションを設定することと同じです。

3
応答データとレベル 2 のメッセージをログに表示します。

デバッグレベルを 3 に設定することは、DEBUG ステートメントで REQUEST_HEADERS、RESPONSE_HEADERS、REQUEST_BODY、および RESPONSE_BODY オプションを設定することと同じです。

注意
レベル 3 は注意して使用してください。 応答がバイナリデータの場合、システムが不安定になることがあります。

NO_REQUEST_BODY
デバッグレベル 2 以上を指定した場合に表示される情報から要求本文を抑制します。

NO_REQUEST_HEADERS
デバッグレベル 1 以上を指定した場合に表示される情報から要求ヘッダーを抑制します。

NO_RESPONSE_BODY
デバッグレベル 3 を指定した場合に表示される情報から応答の本文を抑制します。

NO_RESPONSE_HEADERS
デバッグレベル 1 以上を指定した場合に表示される情報から応答ヘッダーを抑制します。

OUTPUT_TEXT
要求本文と応答本文をテキストのように表示します。

REQUEST_BODY
要求本文をログに表示します。

REQUEST_HEADERS
要求ヘッダーをログに表示します。

RESPONSE_BODY
応答の本文をログに表示します。

RESPONSE_HEADERS
応答ヘッダーをログに表示します。

詳細

ログに書き込まれるデバッグ情報には、DEBUG ステートメントで少なくとも 1 つのオプションを指定する必要があります。

ユーザーは、LEVEL=オプションを使用して印刷される情報の量を制御できます。または、情報が必要な特定のコンポーネントを指定できます。

個々のコンポーネントを表すオプションは、単独で指定することも、LEVEL=オプションと一緒に指定して、特定の LEVEL=値について表示される情報を制限または拡張することもできます。たとえば、RESPONSE_HEADERS を指定して応答ヘッダーのみのデバッグ情報を表示したり、RESPONSE_HEADERS と RESPONSE_BODY の両方を指定して応答情報のみを表示したりできます。または、LEVEL=3 を指定してすべてのデバッグ情報を表示したり、NO_RESPONSE_BODY を指定して応答本文を省略することもできます。LEVEL=を使用する場合、デバッグ情報を表示するには 1 以上の値が必要です。

HTTP 要求の本文と応答の本文はデフォルトでバイナリとして書き込まれます。情報をテキストとして印刷するには、OUTPUT_TEXT オプションを設定します。

HEADERS ステートメント

HTTP 要求の要求ヘッダーを指定します。

サポート: すべての HTTP メソッド

注: PROC HTTP CT=および HEADERIN=引数の代わりに HEADERS ステートメントを使用します。

例: “例 8: HEADERS ステートメントを使用する GET” (1069 ページ)

構文

HEADERS "*HeaderName*"="*HeaderValue*" < "*HeaderName-n*"="*HeaderValue-n*">;

必須引数

"*HeaderName*"="*HeaderValue*"
ヘッダー名とその値を表す名前と値のペアです。HeaderName は、標準ヘッダー名またはカスタムヘッダー名です。ヘッダーフィールド定義については、HTTP/1.1 仕様(www.w3.org)を参照してください。

注: ヘッダー名ではコロン(:)を指定しないでください。名前と値のペアは、次のフォームに自動的に変換されます。

HeaderName : HeaderValue

詳細

HEADERS ステートメントを使用すると、プロシジャ要求でヘッダー値を簡単に指定できます。完全にフォーマットされた入力ファイルをファイル参照名で指定する必要はありません。HEADERS ステートメントを使用して、値がメソッドのデフォルト値と異なる場合にアップロードするドキュメントのコンテンツの種類と文字のセットを指定します。

表 30.1 POST および PUT メソッドのデフォルトのコンテンツの種類

HTTP メソッド	デフォルトのコンテンツの種類
POST	application/x-www-form-urlencoded
PUT	application/octet-stream

SSLPARMS ステートメント

PROC HTTP 要求の SSL パラメーターを設定します。

要件: この機能を使用するには、Kubernetes 管理者は、SSLPARMS で設定されたパス、値、またはパラメーターを処理するために、SAS が実行されているポッドにマウントされている場所を提供する必要があります。

SAS Viya プラットフォームでは、証明書と秘密キーを提供するために、Kubernetes 用の X509 証明書管理ツールである cert-manager を使用する必要があります。詳細については、*SAS Viya プラットフォームの暗号化: 移動中のデータ*の「暗号化の概要」を参照してください。

SAS Viya プラットフォームセキュリティチームと 連携して、SAS Security Framework 内で作業していることを確認してください。

サポート: すべての HTTP メソッド

構文

SSLPARMS *SSL-options*;

必須引数

"SSLSystemOption"="OptionValue" <*"SSLSystemOption"="OptionValue"*>

ターゲットサーバーとの安全なクライアントサーバー接続を可能にする暗号化のために 1 つまたは複数の SAS システムオプションを指定します。 使用可能なシステムオプションのリストについては、*SAS Viya プラットフォームの暗号化: 移動中のデータ*の「暗号化のための SAS システムオプション」を参照してください。

注: “SSL”で始まるシステムオプションのみがサポートされています。

SSL システムオプションは、次のいずれかの方法で指定できます。

```
"SSLSystemOption"="OptionValue"
SSLSystemOption="OptionValue"
```

注 SSL オプションはホスト固有です。

詳細

SSL オプションをグローバルにではなくローカルに適用するには、SSLPARMS ステートメントを使用します。

使用法: HTTP プロシジャ

ハイパーテキスト転送プロトコルセキュア(HTTPS)の使用

HTTP セキュリティ: TLS とデータ暗号化

トランスポートレイヤーセキュリティ(TLS)とそれに先行するセキュアソケットレイヤー(SSL)は、データの暗号化により、保護された接続を経由して Web ブラウザーと Web サーバーが通信できるようにします。ブラウザーとサーバーでは、データが送信前に暗号化されます。受信するブラウザーまたはサーバーでは、データが処理前に解読されます。

注: TLS に関する説明はすべて、先行プロトコルである SSL に対しても適用されます。

PROC HTTP で HTTPS を使用する

HTTP (HTTPS)を介した安全な通信は、SAS Viya プラットフォームのフルデプロイメントではデフォルトでグローバルに提供されます。HTTPS は、信頼されたルート CA バンドルによって制御されます。デフォルトの信頼されたルート CA バンドルは、インストール時に配置されます。

PROC HTTP では、SSLPARMS ステートメントを使用して、グローバル通信セキュリティ設定をローカル通信セキュリティ設定でオーバーライドできます。SSLPARMS ステートメントで、暗号化のための SAS システムオプションを指定します。SSLPARMS ステートメントに設定されたオプションは、それらが指定された PROC HTTP 要求にのみ適用されます。詳細については、“SSLPARMS ステートメント” (1055 ページ)を参照してください。

SAS Viya プラットフォームの SSL システムオプションの設定については、“SAS System Options for Encryption” (*SAS Viya Platform Encryption: Data in Motion*)を参照してください。

基本認証以外の認証の使用

PROC HTTP を使用すると、認証の種類を指定することができます。認証の種類を指定する機能は、要求を成功させるのに必要な認証の種類があらかじめわかっている場合に便利です。適切な種類を指定すると、認証についてプロシジャがネゴシエートする必要がなくなり、プロシジャの実行が最適化されます。たとえば、サーバーがサポートする認証が Kerberos だけであることがわかっている場合は、AUTH_NEGOTIATE 引数を指定することをお勧めします。

認証の種類を指定しない場合、デフォルトの種類は AUTH_ANY です。AUTH_ANY は、AUTH_NEGOTIATE および AUTH_BASIC を要求で一緒に指定するのと同じです。接続先のサーバーがまたは Kerberos 認証プロトコルをサポートしている場合、通常は、ユーザー名とパスワードを指定する必要はありません。現在のユーザー ID に権限がある場合は、認証が確立されます。

EXPECT_100_CONTINUE は、複数回サーバーにアクセスする必要がある要求を最適化するためにサポートされます。このオプションを指定すると、データが複数回アップロードされるのを防ぐことができます。

注: 認証を使用する場合は、NO_CONN_CACHE を有効にしないでください。NO_CONN_CACHE が有効になっている場合、認証を使用する各呼び出しは再認証する必要があり、これには時間がかかる場合があります。

回線ロギング

回線ロギングは、ネットワーク上に流れるパケットの情報をそのままロギングします。この情報は、通常はダンプと呼ばれます。回線のダンプを使用すると、サーバーにどのような情報が送信され、サーバーによってどのような情報が送り返されたのかを表示できます。表示されるのは生のデータであるため、回線ダンプはプログラムのデバッグにも有用です。

HTTP 固有のメッセージのログへの書き込みには、ロガー APP.TK.HTTPC が使用されます。ロガーによって生成される回線ダンプは、ロガー APP.TK.HTTPC を DEBUG レベル以上に設定することで有効にできます。DEBUG レベルでは、受信および送信データの最初の 64 バイトがロギングされます。TRACE レベルでは、すべてのデータがログに書き込まれます。TRACE レベルでは、パフォーマンスが大幅に低下することに注意してください。

詳細については、*SAS Logging: Configuration and Programming Reference* の SAS Logging を参照してください。

応答エンコーディング

応答はセッションエンコーディングにエンコードされません。要求に使用するエンコーディングを提供し、コンテンツの種類を設定する必要があります。

PROC HTTP での URL エンコードデータの使用

次のように、IN=引数の FORM パラメーターを使用して、URL エンコードされたデータを送信できます。FORM オプションを使用すると、PROC HTTP を使用してシングルパートフォームをサブミットできます。

```
%let firstname=Mickey;
%let lastname=Mouse;
proc http url="http://httpbin.org/post"
  in = form ("firstname"="&firstname"
        "lastname"="&lastname");
run;
```

QUERY=プロシジャオプションを使用すると、URL エンコードされたクエリパラメーターをサブミットできます。以下に示すように、URL にクエリパラメーターを手動で追加することができます。ただし、URL を手動で変更する場合は、特殊文字をエンコードし、&(アンパサンド)が正しく表示され、マクロ変数と間違えられないようにする必要があります。

```
url="http://httpbin.org/GET?firstname=&firstname%nrstr(&lastname)=&lastname";
```

新しい QUERY=プロシジャオプションを使用すると、上記の要求を次のようにサブミットできるようになります。

```
%let firstname=Mickey;
%let lastname=Mouse;
proc http url="http://httpbin.org/get"
  query = ("firstname"="&firstname"
        "lastname"="&lastname");
run;
```

FORM および QUERY オプションは、name=value ペアの前に NOENCODE オプションを指定しない限り、指定された入力文字列のすべてのデータを URL エンコードします。

MULTI オプションを使用したマルチパートデータのアップロード

HTTP マルチパート要求は、通常、ブラウザーおよび HTTP クライアントで HTTP サーバーにファイルをアップロードするために使用されます。PROC HTTP は、一般的なマルチパートサブミットとマルチパートフォームサブミットをサポートします。

一般的なマルチパートサブミットの例としては、メタデータを含むファイルのアップロードが挙げられます。ファイルとは、人間が読める名前を割り当てた単なるバイナリデータです。ファイル名とファイルに関する情報(メタデータ)は、通常、次のような JSON 本文で記述されます。

```
{
   "name": "Picture"
}
```

PROC HTTP を使用して画像ファイルとメタデータをアップロードするために必要なコードの例を次に示します。

```
filename image "profile.png";
filename meta TEMP;

data _null_;
 file meta recfm=f lrecl=1;
 put "{";
 put """name"": ""picture"",";
 put "}";
run;

proc http
url="httpbin.org/post"
method="post"
in = multi "related" ( meta header="Content-Type: application/json",
                  image header="Content-Type: image/png");
run;
```

この例では、メタデータとアップロードするファイルのファイル参照名を定義します。DATA ステップは、JSON 形式でメタデータコンテンツを作成します。PROC HTTP では、IN=引数は、MULTI キーワード、"related"の種類、およびメッセージ入力(かっこ内)を表すファイル参照名を指定します。各ファイル参照のコンテンツの種類は、メッセージ本文内のヘッダーとして指定されます。

マルチパートフォームをサブミットする例を次に示します。

```
filename in 'your-input-file';
filename putHdr temp;
filename getHdf temp;

data _null_;
  file putHdr recfm=f lrecl=1;
  put "Content-Type: text/plain";
run;

data _null_;
  file getHdr recfm=f lrecl=1;
  put "Content-Type: text/plain";
run;

proc http url="your-web-location"
  in = multi form ( "file"=in)
HeaderIn=putHdr
HeaderOut=getHdr;
run;
```

マルチパートフォーム要求を使用すると、フォームサブミットでファイル全体を送信できます。この例では、入力ファイルのファイル参照名、ファイル入力ヘッダーおよび出力ヘッダーを定義します。PROC HTTP では、IN=引数は、FORM キーワード付きの MULTI キーワードと、メッセージ入力(かっこ内)としての名前と値のペアを指定します。"File"は、対応するコントロールのコントロール名です。入力ファイルはペアの値です。アップロードされたファイルのコンテンツの種類は、PROC HTTP の HeaderIn=および HeaderOut=引数で指定されます。

FORM キーワードと MULTI キーワードを使用すると、コンテンツの種類"multipart/form-data"が生成されます。 w3.org では、大きなファイル、非 ASCII データ、およびバイナリデータを含むフォームを送信する場合は、コンテンツの種類に"multipart/form-data"を使用することをお勧めします。

HEADERS=を使用したチャンクデータの送信

チャンクデータを強制的に送信するには、次のようにヘッダーを指定します。

```
proc http url="httpbin.org/post"
 in="hello";
 headers "Transfer-Encoding"="chunked";
 debug level=3;
run;
```

PROC HTTP 応答ステータスレポート

PROC HTTP は、応答ステータス値を SAS ログに書き込みます。

また、PROC HTTP は、各ステートメントの実行後、特定の値を使用してマクロ変数を設定します。これらのマクロ変数をマクロ内で使用して、HTTP エラーをテストできます。HTTP エラーは、ホスト接続が成功し、HTTP 要求が HTTP プロシジャによって正常に解析された後に発生するエラーです。マクロ変数には、ホスト接続エラーまたは PROC HTTP 構文エラーの値は格納されません。マクロ変数は、PROC HTTP の呼び出しの度にリセットされます。

SYS_PROCHTTP_STATUS_CODE
: HTTP 要求のステータスコードを保存します。

SYS_PROCHTTP_STATUS_PHRASE
: ステータスコードに関連付けられている説明フレーズを格納します。

次の例を考えます。

```
HTTP/1.1 200 OK
```

SYS_PROCHTTP_STATUS_CODE マクロ変数には値 200 が格納されます。SYS_PROCHTTP_STATUS_PHRASE マクロ変数は OK を格納します。これは、この HTTP 要求が成功したことを示します。

詳細については、“例 11: ステータス応答値をキャプチャするプログラムを作成する” (1072 ページ)を参照してください。

グローバル値を設定するためのマクロ変数

PROC HTTP によって 2 つの自動マクロ変数が生成され、デフォルトの PROC HTTP を設定したり、その設定を変更したりできます。

PROCHTTP_PROXY=*proxy-server-name-and-port-number*;
PROC HTTP 要求のデフォルトのプロキシサーバーを設定します。一度設定されたら、PROC HTTP 要求で PROXYHOST=引数を指定しない限り、指定されたプロキシサーバーによって、SAS セッションのすべての PROC HTTP 要求に対するプロキシが確立されます。プロシジャの引数で指定された値は、マクロ変数で指定された値を無効にします。マクロ変数とは異なる値で PROXYHOST=引数を指定して、異なるプロキシサーバーを要求に対して使用します。要求に対してプロキシの使用を無効にするには、値を指定せずに PROXYHOST=を指定します。詳細については、“例 5: マクロ変数でプロキシを指定する” (1066 ページ)を参照してください。

PROCHTTP_NOCOOKIES= blank | *integer*;
PROC HTTP 要求で Cookie キャッシングをグローバルに制御できるようにします。マクロ変数を省略するか、値を指定せずにマクロ変数を指定すると、Cookie キャッシングが有効になります(Cookie キャッシングはデフォルトでオンです)。Cookie キャッシングをグローバルに無効にするには、ゼロ以外の値をマクロ変数で指定します。特定の PROC HTTP 要求に対して Cookie キャッシングを無効にするには、PROC HTTP 要求で NO_COOKIES プロシジャオプションだけでなく、CLEAR_COOKIES 引数または CLEAR_CACHE 引数も指定します。

マクロ変数は、%LET ステートメントで設定されます。Cookie キャッシングを無効にした場合は、%SYMDEL ステートメントでマクロ変数をシンボルテーブルから削除できます。

例: HTTP プロシジャ

例 1: 単純な GET 要求

要素: METHOD=引数
URL=引数
OUT=引数

詳細

この例では、ローカルネットワークで、サーバーへのな GET 要求を行います。GET は、PROC HTTP で作成できる最も一般的で単純な要求です。PROC HTTP 要求で IN 引数が省略され、引数がオプションになる場合、GET がデフォルトの METHOD 値になります。

プログラム

```
filename resp TEMP;

proc http
  url="http://httpbin.org/get"
  out=resp;
run;
```

例 2: 単純な POST 要求

要素: METHOD=引数
IN=引数
OUT=引数

詳細

この例では、ローカルネットワークで、サーバーへの単純な POST 要求を行います。アップロードするファイルは、IN=引数でファイル参照名によって特定されます。IN 引数を指定すると、すべての SAS リリースのデフォルトの METHOD=値は POST になります。応答ヘッダーおよび出力ヘッダーは、ファイル参照名に書き込まれます。

プログラム

```
filename resp TEMP;
filename headout TEMP;
filename input TEMP;

data _null_;
  file input;
  put "this is some sample text";
```

```
run;

proc http
  url="http://httpbin.org/post"
  in=input
  out=resp
  headerout=headout;
run;
```

例 3: HTTP 要求でプロキシを指定する

要素: PROXYHOST=引数
PROXYPORT= Argument

詳細

この例では、“例 2: 単純な POST 要求” (1063 ページ)と同じ要求を行いますが、呼び出しは外部サーバーに送信されるため、プロキシサーバーを使用する必要があります。この例では、PROXYHOST 引数で外部サーバーの名前とプロキシポート番号の両方を指定します。

プログラム

```
filename out "u:\prochttp\Testware\ProxyTest_out.txt";
filename input TEMP;

data _null_;
  file input;
  put "this is some sample text";
run;

proc http
  url="http://httpbin.org/post"
  method="post"
  in=input
  out=out
  proxyhost="proxyhost.company.com: 889";
run;
```

例 4: 入力データを文字列として指定

要素: IN= "*string*"

詳細

PROC HTTP の IN=引数は、引用符付き入力文字列またはファイル参照名を受け入れて、入力データをサブミットします。入力を文字列で指定すると、テキストポストおよびフォームベースのポストがさらに送信しやすくなります。この例では、http://httpbin.org/forms/post にあるフォームをサブミットします。応答は、応答ファイルに書き込まれます。

プログラム

```
filename resp TEMP;

proc http
  method="post"
  url="http://httpbin.org/post"
  in='custname=Sas+User&custtel=919-555-5555&custemail=sas.user%40
  sas.com&size=medium&topping=cheese&delivery=12%3A00&comments=Dont+Drop+It'
  out=resp;
run;

data _null_;
  infile resp;
  input;
  put _infile_;
run;
```

これは、Resp ファイルのコンテンツです。

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "comments": "Dont Drop It",
    "custemail": "sas.user@sas.com",
    "custname": "Sas User",
    "custtel": "919-555-5555",
    "delivery": "12:00",
    "size": "medium",
    "topping": "cheese"
  },
  "headers": {
    "Accept": "*/*",
    "Content-Length": "133",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "SAS/9",
  },
  "json": null,
  "origin": "149.173.1.80, 104.129.194.85",
  "url": "http://httpbin.org/post"
}
```


例 5: マクロ変数でプロキシを指定する

要素: PROCHTTP_PROXY=マクロ変数
IN="*string*"

詳細

この例では、“例 3: HTTP 要求でプロキシを指定する” (1064 ページ)と同じ要求を行います。ただし、プロキシサーバーがマクロ変数で指定され、入力テキストが IN 引数で文字列として指定されている場合を除きます。PROCHTTP_PROXY マクロ変数は、プロキシサーバーのインターネットホスト名とポート番号を 1 つの値として指定します。プロキシはマクロ変数で設定されるため、以降 SAS セッションで行われるすべての HTTP 要求で使用できます。この要求では、POST へのパラメーターは、IN=引数で指定された文字列から読み込まれます。

プログラム

```
%let PROCHTTP_PROXY=proxyhost.company.com:889;

filename out "u:\prochttp\Testware\ProxyTest_out.txt";
filename input TEMP;
```

```
proc http
  url="http://httpbin.org/post"
  method="post"
  in="text to write out"
  out=out;
run;
```

例 6: 応答ヘッダーを取得する POST

要素: IN="*string*"
HEADEROUT=引数

詳細

この例では、“例 5: マクロ変数でプロキシを指定する” (1066 ページ)と同じ POST 要求を行いますが、headerOut.txt というファイルの応答ヘッダーを取得します。

プログラム

```
%let PROCHTTP_PROXY=proxyhost.company.com:889;

filename out "u:\prochttp\Testware\ProxyTest_out.txt";
filename hdrout "u:\prochttp\Testware\headerOut.txt";

 proc http
  url="http://httpbin.org/post"
  method="post"
  in="text to write out"
  out=out
  headerout=hdrout;
run;
```

例 7: HEADEROUT_OVERWRITE を指定する GET

要素: HEADEROUT 引数
HEADEROUT_OVERWRITE 引数

詳細

この例は、HEADEROUT_OVERWRITE 引数の効果を示しています。GET 要求は、出力先に到達する前に 2 回リダイレクトされます。HEADEROUT_OVERWRITE では、最後の出力ヘッダーのみが記録されます。

リダイレクト後の標準の HEADEROUT 出力の例

```
filename hdrs "u:\prochttp\Testware\GetHdr_out.txt";
filename out "u:\prochttp\Testware\GetTest_out.txt";

proc http
  url="http://httpbin.org/redirect/2"
  method=GET
  headerout=hdrs
  out=out;
run;
```

これは GetHdr_out.txt のコンテンツです。

```
HTTP/1.1 302 FOUND
Server: nginx
Date: Mon, 20 Apr 2015 14:19:52 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 247
Connection: keep-alive
Location: /relative-redirect/1
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

HTTP/1.1 302 FOUND
Server: nginx
Date: Mon, 20 Apr 2015 14:19:53 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: keep-alive
Location: /get
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

HTTP/1.1 200 OK
Server: nginx
Date: Mon, 20 Apr 2015 14:19:53 GMT
Content-Type: application/json
Content-Length: 195
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
```

HEADEROUT_OVERWRITE での HEADEROUT 要求の例

```
filename hdrs "u:\prochttp\Testware\GetHdr2_out.txt";
filename out "u:\prochttp\Testware\GetTest2_out.txt";

proc http
 url="http://httpbin.org/redirect/2"
method=GET
headerout=hdrs
out=out
HEADEROUT_OVERWRITE;
run;
```

これは GetHdr2_out.txt のコンテンツです。

```
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 20 Apr 2015 14:22:48 GMT
Content-Type: application/json
Content-Length: 195
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
```

例 8: HEADERS ステートメントを使用する GET

要素: HEADERS ステートメント
GET メソッド

詳細

HEADERS ステートメントを指定する GET メソッド要求の例を示します。

プログラム

```
filename resp TEMP;

proc http
  url="http://httpbin.org/headers"
  out=resp;
 headers
  "Accept"="application/json";
```

```
run;

data _null_;
  infile resp;
  input;
  put _infile_;
run;
```

出力は次のようになります。

```
  "headers": {
    "Accept": "*/*,application/json",
    "Host": "httpbin.org",
    "User-Agent": "SAS/9",
  }
}
```

例 9: 非標準のメソッド

要素: METHOD 引数

詳細

この例では、MKCOL WEBDAV http メソッドをサブミットします。出力は、Resp という名前の一時ファイルに書き込まれます。非標準のメソッドに入力および出力の要件はありません。ターゲットサーバーからデータが返されたときに、有効な OUT を指定していれば、データは OUT ファイル参照名に書き込まれます。Resp に書き込まれる出力を次に示します。

プログラム

```
filename resp TEMP;

proc http
  url="http://hostname/directory/"
  method=MKCOL
  out=resp;
run;
```

例 10: EXPECT_100_CONTINUE を指定する PUT

要素: EXPECT_100_CONTINUE 引数

HEADEROUT=引数

詳細

この例では、EXPECT_100_CONTINUE ヘッダーを指定します。

プログラム

```
filename resp TEMP;
filename hdrs TEMP;

proc http
  url="http://httpbin.org/put"
  method=PUT
  in='Some Put Data'
  out=resp
  headerout=hdrs
  EXPECT_100_CONTINUE;
run;

data _null_;
  infile hdrs;
  input;
  put _infile_;
run;

data _null_;
  infile resp;
  input;
  put _infile_;
run;
```

HDRS の出力は次のようになります。

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Server: myserver/18.0
Date: Mon, 24 Nov 2014 20:18:29 GMT
Content-Type: application/json
Content-Length: 652
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
X-Cache: MISS from transproxy
Via: 1.1 vegur, 1.1 transproxy (squid)
Connection: keep-alive
```

Resp ファイルの出力は次のようになります。

```
{
  "args": {},
  "data": "Some Put Data",
  "files": {},
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Content-Length": "13",
    "Content-Type": "application/octet-stream",
    "Host": "httpbin.org",
    "User-Agent": "SAS/9",
    "Xxpect": "100-continue",
  },
  "json": null,
  "origin": "149.173.1.80, 104.129.194.85",
  "url": "http://httpbin.org/put"
}
```


例 11: ステータス応答値をキャプチャするプログラムを作成する

要素: SYS_PROCHTTP_STATUS_CODE マクロ変数
SYS_PROCHTTP_STATUS_PHRASE マクロ変数

詳細

この例では、SYS_PROCHTTP_STATUS_CODE マクロ変数に設定された値をテストする簡単なマクロプログラムを作成します。結果コードが SYS_PROCHTTP_STATUS_CODE マクロ変数の指定された値と一致しない場合、プログラムはエラーメッセージを出力するように指定します。プログラムはまた、エラーメッセージ内の SYS_PROCHTTP_STATUS_CODE および SYS_PROCHTTP_STATUS_PHRASE マクロ変数によって戻された実際の値を出力するように指定します。

さまざまな状況下で期待できる結果を示すために、さまざまな PROC HTTP 要求の後にマクロプログラムが呼び出されます。マクロプログラムは、実行ごとに 200 の値で呼び出されます。値 200 は、HTTP 要求の正常終了を示します。

プログラム

次のコードは、マクロプログラムを作成します。

```
%macro prochttp_check_return(code);
%if %symexist(SYS_PROCHTTP_STATUS_CODE) ne 1 %then %do;
  %put ERROR: Expected &code., but a response was not received from
 the HTTP Procedure;
```

```
  %abort;
  %end;
%else %do;
  %if &SYS_PROCHTTP_STATUS_CODE. ne &code. %then %do;
   %put ERROR: Expected &code., but received &SYS_PROCHTTP_STATUS_CODE.
 &SYS_PROCHTTP_STATUS_PHRASE.;
   %abort;%end;
%end;
%mend;
```

成功した HTTP 要求の結果

次に、PROC HTTP 要求の成功例を示します。マクロプログラムは、エラーメッセージと、コード値 200 を返さない HTTP 要求の戻り値と状態値を出力するよう指定します。

```
proc http url="httpbin.org/get";
run;
%prochttp_check_return(200);
```

要求のログを次に示します。要求は 200 を返しました。戻りコードがマクロプログラムで指定されたコード値と一致する PROC HTTP 要求の場合、マクロ変数の値は無視されます。エラーメッセージはありません。したがって、ステータス報告マクロ変数の値を出力する必要はありません。

```
173  proc http url="httpbin.org/get";
174  run;

NOTE: PROCEDURE HTTP used (Total process time):
      real time           3.40 seconds
      cpu time            0.03 seconds

NOTE: 200 OK

175
176  %prochttp_check_return(200);
177
178
```

失敗したホスト接続の結果

The following is an example of a PROC HTTP request that cannot establish a host connection:

```
proc http url="foo.bar";
run;
%prochttp_check_return(200);
```

サーバー接続が確立できない場合は、エラーメッセージが返されます。ただし、エラーメッセージには戻りコードまたは状態フレーズは含まれません。

```
179  proc http url="foo.bar";
180  run;

ERROR: Host name resolution failed
NOTE: PROCEDURE HTTP used (Total process time):
      real time          2.28 seconds
      cpu time           0.00 seconds

NOTE: The SAS System stopped processing this step because of errors.
181
182  %prochttp_check_return(200);
ERROR: Expected 200, but a response was not received from the HTTP Procedure
ERROR: Execution terminated by an %ABORT statement.
183
```

有効な HTTP エラーの結果

次は接続するが無効なセカンダリパスを指定する PROC HTTP 要求の例です。

```
proc http url="httpbin.org/get2";
run;
%prochttp_check_return(200);
```

サーバー接続が確立されますが、HTTP 要求が失敗する場合、エラーメッセージが実際の戻りコードとそれに対応する状態フレーズを出力します。

```
184  proc http url="httpbin.org/get2";
185  run;

NOTE: PROCEDURE HTTP used (Total process time):
      real time          0.04 seconds
      cpu time           0.00 seconds

NOTE: 404 Not Found

186
187  %prochttp_check_return(200);
ERROR: Expected 200, but received 404 Not Found
ERROR: Execution terminated by an %ABORT statement.
188
```

解析エラーの結果

無効な引数を指定した PROC HTTP 要求の例を示します。

```
proc http url="httpbin.org/get" foo;
run;
%prochttp_check_return(200);
```

サーバー接続が確立されても、PROC HTTP 要求を解析できない場合、マクロはエラーメッセージを返します。このメッセージには、戻りコードまたは状態フレーズは含まれません。

```
189  proc http url="httpbin.org/get" foo;
                              ---
                              22
                              202
ERROR 22-322: Syntax error, expecting one of the following: ;, AUTH_ANY, AUTH_BASIC,
          AUTH_NEGOTIATE, AUTH_NONE, CLEAR_CACHE, CLEAR_CONN_CACHE,
          CLEAR_COOKIES, CT, EXPECT_100_CONTINUE, FOLLOWLOC, HEADERIN, HEADEROUT,
          HEADEROUT_OVERWRITE, IN, METHOD, NOFOLLOW, NOFOLLOWLOC,
          NO_CONN_CACHE, NO_COOKIES, OAUTH_BEARER, OUT, PASSWORD, PROXYHOST,
          PROXYPASSWORD, PROXYPORT, PROXYUSERNAME, PROXY_AUTH_BASIC, PROXY_AUTH_NEGOTIATE,
          PROXY_AUTH_NONE, TIMEOUT, URL, USERNAME, VERBOSE,WEBPASSWORD, WEBUSERNAME.
ERROR 202-322: The option or parameter is not recognized and will be ignored.
190  run;
191
192  %prochttp_check_return(200);
ERROR: Expected 200, but a response was not received from the HTTP Procedure
NOTE: PROCEDURE HTTP used (Total process time):
    real time          0.07 seconds
    cpu time           0.07 seconds

ERROR: Execution terminated by an %ABORT statement.
NOTE: The SAS System stopped processing this step because of errors.
```

例 12: LEVEL=オプションを指定して DEBUG ステートメントを使用

要素:

DEBUG ステートメント
LEVEL=ステートメントオプション:
ステートメント出力

詳細

この例は、DEBUG ステートメントの Level=オプションで返される情報を示しています。

DEBUG レベル 1

デバッグレベル 1 は、HTTP 要求ヘッダーと応答ヘッダーを表示します。

```
proc http
  in = "**************testing prochttp****************"
```

```
    method="post"
    url="http://httpbin.org/post";
    debug level = 1;
run;
```

次の情報がログに表示されます。

```
> POST /post HTTP/1.1
> User-Agent: SAS/9
> Host: httpbin.org
> Accept: */*
> Connection: Keep-Alive
> Content-Length: 45
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 OK
< Connection: keep-alive
< Server: myserver/19.9.0
< Date: Tue, 07 Aug 2018 19:12:34 GMT
< Content-Type: application/json
< Content-Length: 418
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: true
< Via: 1.1 vegur
<
```

DEBUG レベル 2

デバッグレベル 2 は、HTTP 要求ヘッダーと応答ヘッダーと HTTP 要求の本文を表示します。

```
proc http
  in = "**************testing prochttp****************"
  method="post"
  url="http://httpbin.org/post";
  debug level = 2;
run;
```

次の情報がログに表示されます。要求の本文はデフォルトでバイナリとして書き込まれます。情報をテキストとして書き込む場合は、OUTPUT_TEXTDEBUG ステートメントオプションを使用します。

```
> POST /post HTTP/1.1
> User-Agent: SAS/9
> Host: httpbin.org
> Accept: */*
> Connection: Keep-Alive
> Content-Length: 45
> Content-Type: application/x-www-form-urlencoded
>
> 000000000A9FC4C0: 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 74 65 73
**************tes
> 000000000A9FC4D0: 74 69 6E 67 20 70 72 6F 63 68 74 74 70 2A 2A 2A ting
prochttp***
> 000000000A9FC4E0: 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A          *************
< HTTP/1.1 200 OK
< Connection: keep-alive
< Server: myserver/19.9.0
< Date: Thu, 06 Sep 2018 14:21:26 GMT
< Content-Type: application/json
< Content-Length: 418
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: true
< Via: 1.1 vegur
<
```

DEBUG レベル 3

デバッグレベル 3 は、要求ヘッダー、応答ヘッダー、要求の本文、および応答の本文を表示します。

```
proc http
  in = "**************testing prochttp*****************"
  method="post"
  url="http://httpbin.org/post";
  debug level = 3;
run;
```

次の情報がログに表示されます。要求と応答の本文はデフォルトでバイナリとして書き込まれます。情報をテキストとして書き込む場合は、OUTPUT_TEXTDEBUG ステートメントオプションを使用します。

```
> POST /post HTTP/1.1
> User-Agent: SAS/9
> Host: httpbin.org
> Accept: */*
> Connection: Keep-Alive
> Content-Length: 45
> Content-Type: application/x-www-form-urlencoded
>
> 000000000A9FC4C0: 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 74 65 73 **********tes
> 000000000A9FC4D0: 74 69 6E 67 20 70 72 6F 63 68 74 74 70 2A 2A 2A ting prochttp***
> 000000000A9FC4E0: 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A          **********
< HTTP/1.1 200 OK
< Connection: keep-alive
< Server: myserver/19.9.0
< Date: Thu, 06 Sep 2018 14:25:41 GMT
< Content-Type: application/json
< Content-Length: 418
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: true
< Via: 1.1 vegur
<
< 000000000A9FC5B6: 7B 0A 20 20 22 61 72 67 73 22 3A 20 7B 7D 2C 20 {.  "args": {},
< 000000000A9FC5C6: 0A 20 20 22 64 61 74 61 22 3A 20 22 22 2C 20 0A .  "data": "", .
< 000000000A9FC5D6: 20 20 22 66 69 6C 65 73 22 3A 20 7B 7D 2C 20 0A   "files": {}, .
< 000000000A9FC5E6: 20 20 22 66 6F 72 6D 22 3A 20 7B 0A 20 20 20 20   "form": {.
< 000000000A9FC5F6: 22 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 74 65 "**********te
< 000000000A9FC606: 73 74 69 6E 67 20 70 72 6F 63 68 74 74 70 2A 2A sting prochttp**
< 000000000A9FC616: 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 22 3A **********":
< 000000000A9FC626: 20 22 22 0A 20 20 7D 2C 20 0A 20 20 22 68 65 61  "".  }, .  "hea
< 000000000A9FC636: 64 65 72 73 22 3A 20 7B 0A 20 20 20 20 22 41 63 ders": {.    "Ac
< 000000000A9FC646: 63 65 70 74 22 3A 20 22 2A 2F 2A 22 2C 20 0A 20 cept": "*/*", .
< 000000000A9FC656: 20 20 20 22 43 6F 6E 6E 65 63 74 69 6F 6E 22 3A    "Connection":
< 000000000A9FC666: 20 22 63 6C 6F 73 65 22 2C 20 0A 20 20 20 20 22  "close", .    "
< 000000000A9FC676: 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 22 3A Content-Length":
< 000000000A9FC686: 20 22 34 35 22 2C 20 0A 20 20 20 20 22 43 6F 6E  "45", .    "Con
< 000000000A9FC696: 74 65 6E 74 2D 54 79 70 65 22 3A 20 22 61 70 70 tent-Type": "app
< 000000000A9FC6A6: 6C 69 63 61 74 69 6F 6E 2F 78 2D 77 77 77 2D 66 lication/x-www-f
< 000000000A9FC6B6: 6F 72 6D 2D 75 72 6C 65 6E 63 6F 64 65 64 22 2C orm-urlencoded",
< 000000000A9FC6C6: 20 0A 20 20 20 20 22 48 6F 73 74 22 3A 20 22 68  .    "Host": "h
< 000000000A9FC6D6: 74 74 70 62 69 6E 2E 6F 72 67 22 2C 20 0A 20 20 ttpbin.org", .
< 000000000A9FC6E6: 20 20 22 55 73 65 72 2D 41 67 65 6E 74 22 3A 20   "User-Agent":
< 000000000A9FC6F6: 22 53 41 53 2F 39 22 0A 20 20 7D 2C 20 0A 20 20 "SAS/9".  }, .
< 000000000A9FC706: 22 6A 73 6F 6E 22 3A 20 6E 75 6C 6C 2C 20 0A 20 "json": null, .
< 000000000A9FC716: 20 22 6F 72 69 67 69 6E 22 3A 20 22 31 32 2E 37  "origin": "12.7
< 000000000A9FC726: 30 2E 31 30 31 2E 31 31 34 22 2C 20 0A 20 20 22 0.101.114", .  "
< 000000000A9FC736: 75 72 6C 22 3A 20 22 68 74 74 70 3A 2F 2F 68 74 url": "http://ht
< 000000000A9FC746: 74 70 62 69 6E 2E 6F 72 67 2F 70 6F 73 74 22 0A tpbin.org/post".
< 000000000A9FC756: 7D 0A                                         }.
```

例 13: UNIX での暗号化のローカルオプションの指定

要素: SSLPARMS ステートメント
SSLCERTLOC=システムオプション
SSLPVTKEYLOC= システムオプション
SSLPVTKEYPASS=システムオプション

SSLPKCS12LOC=システムオプション
SSLPKCS12PASS=システムオプション

詳細

次は、UNIX でローカル証明書を指定する暗号化オプションを指定する PROC HTTP 要求の例です。UNIX では、指定するシステムオプションは、証明書の出力形式が PEM であるか DER であるによって異なります。この例は、ユーザー John Doe の PEM および DER 証明書を送信するコードを示しています。

SAS PROC HTTP は、相互 TLS(mTLS)とも呼ばれる TLS クライアント証明書認証を使用した安全な API 通信をサポートしています。これは、Web サービスでサーバーとクライアントの両方が X.509 証明書を使用して認証する必要がある場合に役立ちます。適切な SSL システムオプションと SSLPARMS ステートメントを指定することにより、PEM ファイルまたは PKCS#12 アーカイブを使用してクライアント証明書を送信するように PROC HTTP を構成できます。

注: 証明書のパスは、Web サーバーが実行中の場所への相対パスです。そのため、証明書はディスク上でアクセスできる必要があります。

SAS Viya の SSL システムオプションの詳細については、*SAS Viya プラットフォームの暗号化: 移動中のデータ*の SAS System Options for Encryption を参照してください。

PEM 証明書の SSLPARMS 引数

SSLCERTLOC=システムオプションおよび SSLPVTKEYLOC=システムオプションは、PEM 証明書の場所を指定します。SSLPVTKEYPASS=は、必要な場合にのみ指定されます。

```
proc http
  method=GET
  url="https://internal.site.com/";
  sslparms
    sslcertloc="/users/johndoe/certificates/server.pem"
    sslpvtkeyloc="/users/johndoe/certificates/serverkey.pem"
    sslpvtkeypass="password"
  ;
run;
```

DER 証明書の SSLPARMS 引数

SSLPKCS12LOC=システムオプションおよび SSLPKCS12PASS=システムオプションは、DER 証明書の場所を指定します。

例のコード 30.1　DER でエンコーディングされた証明書のシステムオプション

```
proc http
 method=GET
 url=https://internal.site.com/;
 sslparms
  sslpkcs12loc="/users/server/certificates/server.p12"
  sslpkcs12pass="password"
 ;
run;
```

31

IMPORT プロシジャ

概要: IMPORT プロシジャ

IMPORT プロシジャの機能

IMPORT プロシジャは外部データソースからデータを読み取り、それを SAS データセットに書き出します。次の外部データソースが含まれます。

- 区切りファイル
- JMP ファイル
- Microsoft Excel ワークシート
- Paradox ファイル
- SPSS ファイル
- Stata ファイル

区切りファイルでは、区切り文字(ブランク、カンマ、タブなど)でデータ値の列が区切られます。SAS/ACCESS Interface to PC Files のライセンスがある場合、追加の外部データソースに Microsoft Access データベースファイル、Microsoft Excel ファイルや Lotus スプレッドシートを含めることができます。

JMP 7 以降のファイルからデータをインポートできます。また、JMP 変数には最大 255 文字を使用できます。値ラベルを SAS フォーマットカタログにインポートすることもできます。拡張属性が自動的に使用されるようになり、META=ステートメントがサポートされなくなりました。

CAS エンジンは、CAS サーバーのメモリ内の CAS テーブルにインポートしたり、CAS テーブルを PC ファイル形式にエクスポートしたりするための JMP、Excel (.xlsx および.xls)、SPSS、および Stata ファイルもサポートしています。CAS エンジンを使用して CAS データをインポートまたはエクスポートするには、CAS サーバーへの接続を確立する必要があります。CAS テーブルにインポートする場合、PC ファイル形式のすべての CHAR データ型または文字列は、CAS テーブルの VARCHAR データ型に変換されます。

IMPORT プロシジャは実行時に入力ファイルを読み込み、データを指定された SAS データセットに書き出します。デフォルトでは、変数名が最初の行に現れることを IMPORT プロシジャは仮定しています。プロシジャでは変数を数えるために最初の 20 行をスキャンし、各変数について適切な入力形式と出力形式の決定を試みます。IMPORT プロシジャのステートメントを使用すると次のことが行えます。

- 変数の種類と長さを決定するために SAS がスキャンする行の数 (GUESSINGROWS=)
- SAS がデータの読み込みを開始する行(DATAROW=)
- SAS が変数名を抽出するかどうかの変更(GETNAMES=)

これらのステートメントは、デフォルト値を変更するためにも使用できます。

IMPORT プロシジャは、区切り文字で区切られたファイルの読み込み時に、次のいずれかの方法を使用して、データをインポートします:

- 生成された DATA ステップコード
- 生成された SAS/ACCESS コード

入力データソース固有のオプションやステートメントを指定して、結果をカスタマイズすることができます。IMPORT プロシジャは指定の出力 SAS データセットまたはテーブルを生成し、インポートに関する情報を SAS ログに書き込みます。ログには、IMPORT プロシジャによって生成された DATA ステップコードまたは SAS/ACCESS コードが表示されます。

プロシジャの実行後にコードを修正する必要がある場合は、RECALL コマンドを発行(または F4 キーを押下)して、生成された DATA ステップをリコールします。この段階で、INFILE ステートメントからオプションを追加または削除でき、またデータの INFORMAT、FORMAT、INPUT ステートメントをカスタマイズできます。

このメソッドを使用して入力形式を修正した場合は、出力形式の同じ変数についても修正を行ってください。入力形式と出力形式のすべての変数は、種類(文字または数値)も同じである必要があります。さらに種類が文字である場合は、データの表示時に切り捨てが起こらないような長さの変数を出力形式に割り当てる必要があります。たとえば、文字変数の長さが 400 文字であるのに、出力形式に$char50 が割り当てられている場合、データの表示時には最初の 50 文字しか表示されません。

PROC IMPORT コードをリコールするには、RECALL コマンドを再度発行(または F4 キーをもう一度押下)します。

注: デフォルトでは、IMPORT プロシジャは、区切り文字で区切られたファイルを可変レコード長のファイルとして読み込みます。外部ファイルが固定長のファイル形式を使用している場合は、INFILE ステートメントに RECFM=F と LRECL=オプションを含んだ SAS DATA ステップを使用します。INFILE ステートメントの RECFM=オプションの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。

ヒント **区切りファイルのホスト間での共有:**IMPORT プロシジャを使用して区切りファイルを SAS に読み込む場合、各行はホストに固有の行末区切り文字で終了している必要があります。あるホストで作成された区切りファイルを他のホストで共有する場合、デフォルトの行末の区切り文字が一致しないことがあります。この場合、新しいホストでの行末区切り文字をファイルで指定する必要があります。

- Linux: デフォルトの行末の区切り文字は改行(LF)です。Windows で作成されたファイルを読み込むには、FILENAME ステートメントに TERMSTR=CRLF オプションを使用します。FILENAME ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。

PROC IMPORT では、COMMA 入力形式の代わりに NLNUM 入力形式が使用されます。14,000.01 のようにカンマがある値を含むファイルをインポートする場合、COMMA 入力形式は、カンマおよびその他の数字以外の文字を数値から削除します。

これらの文字を削除すると、値の解釈エラーが引き起こされる可能性があります。NLNUM は、LOCALE システムオプションの指定値を使用してカンマがある数値を解釈することにより、これらのエラーを防ぎます。

たとえば、一万四千・〇一に相当する数字を入力するには、LOCALE=English_UnitedStates を指定したユーザーは 14,000.01 と入力します。LOCALE=French_France を指定したユーザーは 14.000,01 と入力します。NLNUM は、指定されたロケールに基づいて、どちらの入力値も正しく解釈し、正しい値を書き込みます。NLNUM および LOCALE=French_France を指定して 14.000,01 を読み込み、データセットに保存し、その後、NLNUM および LOCALE=English_UnitedStates を指定して書き込むと、14,000.01 と表示されます。

AAFFGG_ 06JAN2016:10:04:26 などの文字値を SAS データセットにインポートする場合は、 SAS はそれを日付として読み取ろうとする場合があります。

詳細については、次を参照してください。

- “COMMAw.d 入力形式” (*SAS 出力形式と入力形式: リファレンス*)
- “NLNUMw.d 入力形式” (*SAS 各国語サポート(NLS): リファレンスガイド*)
- “各国語サポート関連のロケール概念の概要” (*SAS 各国語サポート(NLS): リファレンスガイド*)

SAS Viya プラットフォームでのフォーマットカタログのエンコーディング

SAS Viya プラットフォームでは、UTF-8 エンコーディングのみがサポートされています。

フォーマットカタログのエンコーディングの詳細については、*SAS Viya プラットフォームでの UTF-8 へのデータの移行* および *UTF-8 データの処理における SAS Viya FAQ* を参照してください。

ヒント

- XLSX エンジンを使用して、UTF-8 データを読み取ります。
- UTF-8 データを含むスプレッドシートをインポートするのに、DBMS=xls オプションを使用しないでください。
- Microsoft Excel を使用して、.xls スプレッドシートを.xlsx スプレッドシートに変換してから、DBMS=xlsx オプションでインポートします。

VARCHAR データ型のサポート

CAS では、PROC IMPORT で VARCHAR データ型がサポートされています。VARCHAR では、可変長の文字変数が保存されます。変数に指定する長さは、保存する文字の最大数を表します。

VARCHAR データ型は、CHAR データ型に似ています。CHAR 変数の長さは、バイト単位で測定されます。VARCHAR 変数の長さは、バイト単位ではなく文字単位で測定されます。VARCHAR の使用については、*SAS Cloud Analytic Services: ユーザーガイド*の「CAS DATA ステップでサポートされているデータ型」を参照してください。

次の例では、CAS Engine を LENGTH ステートメントと一緒に使用して、VARCHAR 変数と CHAR 変数を作成します。VARCHAR 変数 X の長さは 30 で、CHAR 変数 Y の長さも 30 です。

```
libname mycas cas;
data mycas.string;
  length x varchar(30);
  length y $30;
  x = 'abc'; y = 'def';
run;
proc contents data=mycas.string; run;
```

このコードで生成される出力を次に示します。

The CONTENTS Procedure

Data Set Name	MYCAS.STRING	Observations	1
Member Type	DATA	Variables	2
Engine	CAS	Indexes	0
Created	09/11/2016 13:48:30	Observation Length	48
Last Modified	09/11/2016 13:48:30	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Limit	100MB

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	x	Varchar	30
2	y	Char	30

外部ファイルインターフェイス(EFI)マクロ変数

外部ファイルインターフェイス(EFI)マクロ変数を使用して、PROC IMPORT が文字値を読み取って解釈する方法を制御できます。

EFI_ALLCHARS=YES | NO

デフォルト NO

注 EFI_ALLCHARS の値は、後続のすべての PROC IMPORT ステップに適用されます。

AllChars SAS レジストリオプションは、EFI_ALLCHARS マクロ変数と同じ効果があります。

例 %let EFI_ALLCHARS=YES;

SAS 日時の誤った解釈を避けるために、すべての列がデフォルトで CHAR データ型になるかどうかを指定します。SAS 9.4TS1M4 以降では、EFI マクロ変数 EFI_ALLCHARS を指定できます。これにより、PROC IMPORT はすべての 変数を文字として定義します。PROC IMPORT で EFI_ALLCHARS を使用する方法の例は、次のようになります:

```
%let EFI_ALLCHARS=YES;
proc import datafile='c:\temp\class.csv' out=class dbms=csv replace;
run;
%let EFI_ALLCHARS=NO;
```

PROC IMPORT ステップの後で、EFI_ALLCHARS の値をリセットします。それ以外の場合、EFI_ALLCHARS=YES に続くすべての PROC IMPORT ステップは、すべての変数を文字として定義します。

EFI_MISSING_NUMERICS=YES | NO

デフォルト NO

注 MissingNumerics SAS レジストリオプションは、EFI_MISSING_NUMERICS マクロ変数と同じ効果があります。

例 %let EFI_MISSING_NUMERICS=YES;

単一のピリオドが数字か文字かを指定します。MissingNumerics が No に設定されている場合、単一のピリオドは文字として解釈されます。

EFI_NOQUOTED_DELIMITER=YES | NO

デフォルト NO

注 QuotedDelimiter SAS レジストリオプションは、EFI_NOQUOTED_DELIMITER マクロ変数と同じ効果があります。

例 %let EFI_NOQUOTED_DELIMITER=YES;

変数値内の単一引用符を、開いた引用符内の区切り文字として読み取るか、変数値の一部として読み取るかを指定します。変数に区切り文字が含まれている場合、または変数値の一部として単一引用符で読み取る場合は、EFI_NOQUOTED_DELIMITER を Yes に設定します。

EFI_QUOTED_NUMERICS=YES | NO

デフォルト NO

注 QuotedNumerics SAS レジストリオプションは、EFI_QUOTED_NUMERICS マクロ変数と同じ効果があります。

例 %let EFI_QUOTED_NUMERICS=YES;

引用符で囲まれた数値を文字として読み取るか、または数値として読み取るかを指定します。PROC IMPORT が外部ファイルを読み取ると、引用符で囲まれた値が文字値として自動的に作成されます。EFI_QUOTED_NUMERICS を Yes に設定すると、引用符で囲まれた数値を数値として読み取ることができます。たとえば、10MAY14 は文字値ですが、EFI_QUOTED_NUMERICS を Yes に設定すると、日付値である 10MAY14 になります。

構文: IMPORT プロシジャ

制限事項: IMPORT プロシジャは次の動作環境で使用できます。

- Linux

ファイルのパス名には、最大で 201 文字を使用できます。

操作: パーセント記号(%)の付いたすべてのデータは、誤って解釈されないように文字データとみなされます。パーセントデータは、誤って解釈される可能性があるため、文字データとみなされます。

サポート: CAS で PROC IMPORT がサポートするファイルの種類は CSV、TAB、DLM、JMP です。

注: PROC IMPORT を使用すると、SAS データセットまたはテーブルに外部ファイルをインポートできます。

ヒント: CAS テーブルでは、PROC IMPORT で VARCHAR データ型がサポートされています。詳細については、“VARCHAR データ型のサポート” (1085 ページ)を参照してください。

XLSX エンジンを使用して、UTF-8 データを読み取ります。詳細については、“LIBNAME Statement: XLSX Engine” (*SAS/ACCESS Interface to PC Files: Reference for the SAS Viya Platform*)を参照してください。

UTF-8 データを含むスプレッドシートをインポートするのに、DBMS=xls オプションを使用しないでください。

Microsoft Excel を使用して、.xls スプレッドシートを.xlsx スプレッドシートに変換してから、DBMS=xlsx でインポートします。

参照項目: “ANYDTDTMw. 入力形式” (*SAS 出力形式と入力形式: リファレンス*)

PROC IMPORT

DATAFILE="*filename*" | TABLE="*tablename*"

OUT=<*libref*.>*SAS data set*<(*SAS data set options*)>

<DBMS=*identifier*> <REPLACE>;

statements for importing from delimited files

DATAROW=*n*;

DELIMITER=*char*'' | '*nn*x;

GETNAMES=YES | NO;

GETSHEETNAMES=YES | NO;

GUESSINGROWS=*n* | MAX;

VARNAMEROW=*n*;

statements for importing from JMP files

DBENCODING=*12-char SAS encoding-value*;

FMTLIB=<*libref*.>*format-catalog*;

META=*libref.member-data-set*;

ステートメント	タスク	例
“PROC IMPORT ステートメント”	外部データファイルを SAS データセットにインポートします	Ex. 1, Ex. 2, Ex. 3, Ex. 4
“DATAROW ステートメント”	区切りテキストファイルの特定行からのデータ読み込みを開始します	Ex. 3
“DBENCODING ステートメント”	JMP ファイルに使用するエンコーディング文字セットを示します	
“DELIMITER ステートメント”	入力ファイルのデータ列を区切る区切り文字を指定します	Ex. 1, Ex. 3, Ex. 4
“FMTLIB ステートメント”	値ラベルを指定の SAS フォーマットカタログに保存します	
“GETNAMES ステートメント”	入力ファイルの最初の行のデータ値から SAS 変数名を生成します	Ex. 1, Ex. 2
“GETSHEETNAMES ステートメント”	Excel ファイル内のワークシート名を SAS データセットとして返します。	
“GUESSINGROWS ステートメント”	スキャンする入力ファイルの行数を指定して、変数に適切なデータの種類と長さを決定します	
“META ステートメント”	JMP メタデータ情報を指定の SAS データセットに保存します	
“VARNAMEROW ステートメント”	PROC IMPORT に区切りファイルの変数名を含むファイルの行を指定します。	

PROC IMPORT ステートメント

外部データファイルを SAS データセットまたはテーブルにインポートします。

制限事項: ファイルのパス名には、最大で 201 文字を使用できます。

CAS LIBNAME エンジンを使用して CAS テーブルをインポートする場合は、CAS サーバーに接続する必要があります。

注: SAS Viya プラットフォームは UTF-8 エンコードのみをサポートし、Linux でのみ実行されます。

Excel がファイルを開くことができるように、無効な UTF-8 文字は疑問符に置き換えられます。

ヒント: PROC IMPORT は、FILENAME ステートメントで使用可能なアクセスの種類をすべてサポートしています。

CAS テーブルでは、PROC IMPORT で VARCHAR データ型がサポートされています。詳細については、“VARCHAR データ型のサポート” (1085 ページ)を参照してください。

XLSX エンジンを使用して、UTF-8 データを読み取ります。詳細については、“LIBNAME Statement: XLSX Engine” (*SAS/ACCESS Interface to PC Files: Reference for the SAS Viya Platform*)を参照してください。

UTF-8 データを含むスプレッドシートをインポートするのに、DBMS=xls オプションを使用しないでください。

Microsoft Excel を使用して、.xls スプレッドシートを.xlsx スプレッドシートに変換してから、DBMS=xlsx でインポートします。

参照項目: 31 章, “IMPORT プロシジャ,” (1081 ページ)

構文

PROC IMPORT

DATAFILE="*filename* " | TABLE="*tablename* "

OUT=<*libref* .>*SAS data set* <(*SAS data set options*)>

<DBMS=*identifier*> <REPLACE>;

オプション引数の要約

DBMS=*data-source-identifier*
: インポートするデータの種類を指定します。

<file-format-specific-statements>

REPLACE
: 既存する SAS データセットを上書きします。

SAS data set options

SAS データセットオプションを指定します。

SERVERAUTHDOMAIN=*server authentication-domain*

必須引数

DATAFILE="*filename*" | "*fileref*"

入力 PC ファイル、スプレッドシート、または区切り外部ファイルの完全パスとファイル名またはファイル参照名を指定します。ファイル参照名は、出力ファイルの物理的な場所に関連付けられた SAS 名です。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。FILENAME ステートメントの詳細については、*SAS グローバルステートメント: リファレンス*を参照してください。PC ファイル形式の詳細については、*SAS/ACCESS Interface to PC Files: SAS Viya プラットフォームリファレンス*

ファイル参照名を指定した場合、または完全パスとファイル名にパスのバックスラッシュなどの特殊文字、小文字やスペースが含まれていない場合、引用符を省略することができます。

別名　FILE=

制限事項　IMPORT プロシジャは、DISK 以外の FILENAME ステートメントのデバイスの種類またはアクセスメソッドをサポートしていません。たとえば、IMPORT プロシジャは一時外部ファイルを作成する TEMP デバイスの種類をサポートしていません。

IMPORT プロシジャがデータをインポートできるのは、SAS でそのデータの種類がサポートされている場合のみです。SAS では数値および文字の種類のデータはサポートされていますが、たとえばバイナリオブジェクトはサポートされていません。

CAS エンジンは、NUMERIC、CHAR、および VARCHAR データ型をサポートしています。PC ファイル内の CHAR または文字列値が CAS テーブルにインポートされると、CAS エンジンは CHAR を CAS テーブル内の VARCHAR に変換します。インポートするデータの種類が SAS でサポートされていない場合、IMPORT プロシジャはデータを正しくインポートできないことがあります。多くの場合、プロシジャは可能な限り最善の方法でデータを変換します。ただし、種類によっては変換できないものもあります。

操作　デフォルトでは、IMPORT プロシジャは、区切り文字で区切られたファイルを可変レコード長のファイルとして読み込みます。外部ファイルが固定長のファイル形式を使用している場合は、INFILE ステートメントに RECFM=F と LRECL=オプションを含んだ SAS DATA ステップを使用します。詳細については、INFILE ステートメントを参照してください。

fileref を使用してインポートする区切り文字で区切られたファイルを指定する場合、FILENAME ステートメントで LRECL=オプションを指定した場合を除き、論理レコード長(LRECL)のデフォルトは 256 になります。IMPORT プロシジャでサポートされている最大 LRECL は、32767 です。

区切り文字で区切られたファイルの場合、最初の 20 行がスキャンされ、変数の属性が決定されます。GUESSINGROWS=ステートメントを使用して、スキャンされる行数を増やすことができます。すべての値は、文字列とし

て読み込まれます。日時出力形式または数値入力形式がデータ値に適用可能な場合、その種類は数値として宣言されます。その他の場合は、その種類は文字のままです。

Microsoft Excel ワークブックなどの一部の入力データソースでは、データの最初の 8 行がスキャンされます。最も一般的なデータ型(数値または文字)が列に使用されます。これがデフォルト設定です。最初の 8 行のほとんどのデータが欠損している場合、SAS はデフォルトで CHAR データ型に設定され、その列の後続の数値データはすべて欠損に設定されます。

CAS エンジンは、CAS サーバーのメモリ内の CAS テーブルに対して CHAR を VARCHAR にデフォルト設定します。他のファイル形式には、他のデフォルト値がある場合があります。

注 SAS によるデータ型の変換方法については、インポートするデータソースファイル形式に固有の情報を参照してください。

参照項目: *SAS DATA ステップステートメント: リファレンス*の FILENAME ステートメント

例 “例 1: 区切り文字で区切られたファイルのインポート” (1102 ページ)

“例 2: ファイル参照名を使用した、特定の区切り文字で区切られたファイルのインポート” (1106 ページ)

“例 3: タブ区切り文字で区切られたファイルのインポート” (1109 ページ)

“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1112 ページ)

OUT=<*libref.*>*SAS data set*

出力 SAS データセットを 1 レベルまたは 2 レベルの SAS 名(ライブラリとメンバー名)で識別します。指定した SAS データセットが存在しない場合は、IMPORT プロシジャが作成します。1 レベルの名前を指定すると、デフォルトで IMPORT プロシジャは USER ライブラリ(割り当てられている場合)、または WORK ライブラリ(USER が割り当てられていない場合)のいずれかを使用します。物理データセット名は、大文字を使用して作成されます。大文字と小文字を区別するオペレーティングシステムで実行されている SAS プログラムでは、このデータセット名を大文字で参照する必要がある場合があります。

VALIDMEMNAME=EXTEND システムオプションも指定されている場合、SAS データセット名には単一引用符を含めることができます。VALIDMEMNAME=を使用すると、SAS データセット名など、特定の SAS メンバーの名前に関するルールが拡張されます。詳細については、“SAS データセットと変数の命名規則” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

例 “例 1: 区切り文字で区切られたファイルのインポート” (1102 ページ)

“例 2: ファイル参照名を使用した、特定の区切り文字で区切られたファイルのインポート” (1106 ページ)

“例 3: タブ区切り文字で区切られたファイルのインポート” (1109 ページ)

“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1112 ページ)

TABLE="*tablename*"
入力 DBMS テーブルの名前を指定します。名前に特殊文字(疑問符など)、小文字、またはスペースが 含まれていない場合は、引用符を省略できます。DBMS テーブル名では大文字と小文字が区別される場合もあります。

要件 DBMS テーブルをインポートするには SAS/ACCESS Interface to PC Files のライセンスが必要です。

DBMS テーブルをインポートするときは、DBMS=オプションを指定する必要があります。

Microsoft Access データベーステーブルをインポートする 場合、SAS/ACCESS はテーブル名を SAS メンバー名に変換します。SAS は、32 バイトを超えるメンバー名をサポートしていないので注意してください。DBMS テーブルをインポートするときは、DBMS=識別子オプションを指定する必要があります。

オプション引数

DBMS=*data-source-identifier*
インポートするデータの種類を指定します。DBMS テーブルをインポートするには、SAS/ACCESS Interface to PC Files: Reference for SAS Viya Platform の Summary of DBMS Specifications にリストされているサポート対象データベース識別子を使用して DBMS=を指定します。たとえば、DBMS=CSV は、カンマ区切りの値を持つ区切りファイルをインポートするように指定します。

JMP ファイルをインポートするには、識別子として JMP を指定します。JMP ファイルはバージョン 7 以降である必要があり、JMP 変数名の長さは最大で 255 文字までです。SAS では、32,767 以上の変数がある JMP ファイルのインポートがサポートされています。

タブ区切り文字で区切られたファイルをインポートするには、識別子として TAB を指定します。.CSV で終わらないその他の区切り文字で区切られたファイルをインポートするには、識別子として DLM を指定します。拡張子が.CSV のカンマ区切りファイルの場合、DBMS=は任意です。IMPORT プロシジャは、カンマ区切り文字で区切られたファイルの拡張子を.CSV と認識します。

拡張子のないファイルをインポートする場合は、DBMS 引数が必要であり、データはタブで区切られます。カンマで区切られたデータを含む.TXT ファイルをインポートする場合にも必要です。

この表は、使用可能な DBMS=識別子を示しています。表に記載されている場合を除き、すべての DBMS=指定はローカルアクセスを参照しています。

CSV
出力データソースは、カンマ区切りのファイルです。ファイル拡張子は.CSV です。

DLM
出力データソースは、区切り文字で区切られたファイルです。DELIMITER='*char*'を使用して、エクスポートするデータのタイプを指定できます。

DTA
出力データソースは、Stata ファイルです。ファイル拡張子は.dta です。

JMP
出力データソースは、バージョン 7 以降の形式の JMP ファイルです。ファイル拡張子は.JMP です。

PARADOX
出力データソースは Paradox DB ファイルです。ファイル拡張子は.db です。

SAV
出力データソースは、SPSS ファイルです。ファイル拡張子は.sav です。

TAB
出力データソースはタブ区切りファイルです。ファイル拡張子は.txt です。

XLS
ファイル形式を使用する Microsoft Excel 97、2000、2002、または 2003 スプレッドシート。ファイル拡張子は.xls です。

XLSX
Excel 2007、2010、およびそれ以降のスプレッドシート(.xlsx ファイル形式を使用)。ファイル拡張子は.xlsx です。

トランスコーディングは、DBMS=XLS ではサポートされていません。.xls ファイルを.xlsx ファイルに保存し、DBMS=XLSX を使用してトランスコーディングをサポートします。

Linux で Excel ワークブックを読み書きするために、DBMS=XLSX と DBMS=XLS を指定できます。SAS PC Files Server を使用する必要はありません。

SAS は、サポートと機能強化のために.xlsx を使用することをお勧めします。

注: DBMS=PCFS を除いて、すべての DBMS=指定はローカルアクセスを参照します。

制限事項　データソースを利用できるかどうかは、動作環境、場合によってはプラットフォーム、およびサイトに SAS/ACCESS Interface for PC Files ライセンスがあるかどうかによって決まります。

参照項目:　このオプションの識別子の詳細については、SAS でサポートされている DBMS 識別子 (647 ページ)を参照してください。

“DELIMITER ステートメント” (1097 ページ)

例　“例 1: 区切り文字で区切られたファイルのインポート” (1102 ページ)

“例 2: ファイル参照名を使用した、特定の区切り文字で区切られたファイルのインポート” (1106 ページ)

“例 3: タブ区切り文字で区切られたファイルのインポート” (1109 ページ)

“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1112 ページ)

Microsoft Excel

Excel ファイルに DBMS=XLS または DBMS=XLSX を指定すると、SAS PC Files Server にアクセスしなくても、Linux で Excel ワークブックを直接読み書きできます。この例は、セルの範囲を指定する DBMS=XLSX の使用を示しています。

```
proc import datafile="fieldtypes.xlsx"
   out=small dbms=xlsx;
   range=colsb_d;
run;
```

注: PROC IMPORT で DBMS=XLS を指定すると、2 つ以上の文字が通常の命名規則で有効でない場合、コードは変数名を **Var#**に切り替えます。

DBMS=XLSX を指定すると、IMPORT プロシジャは、Microsoft Excel ワークブックに保存されている EXCEL2007 以降のバージョンのファイルのみを読み取ることができます。

この表は、各 DBMS=識別子がサポートする Microsoft Excel ファイルのバージョンを示しています。

表 31.1 Microsoft Excel ワークブック指定

DBMS=識別子	Excel 2007 以降	Excel97、2000、2002、2003	Excel 5.0、95	Excel 4.0
XLS	いいえ	はい	はい	いいえ
XLSX	はい	いいえ	いいえ	いいえ

この表は、Excel 固有の DBMS=識別子、使用するソフトウェア、必要なソフトウェア、および DBMS=指定を使用できるオペレーティングプラットフォームを示しています。

表 31.2 Linux 上の Excel の DBMS 指定

DBMS=識別子	使用法	必須
XLS	ファイル形式技術	
XLSX	ファイル形式技術	

この表は、データの場所に基づいて Microsoft Excel ファイルに使用する DBMS=データソース識別子のクイックリファレンスです。

表 31.3 Microsoft Excel および Access ファイルの DBMS データソース識別子

DBMS=識別子	Linux
XLS	はい
XLSX	はい

注 トランスコーディングとマルチバイト文字は、DBMS=XLS ではサポートされていません。出力は予測できない結果をもたらします。Linux を使用する場合は、トランスコードに DBMS=XLSX を使用してください。

REPLACE
既存する SAS データセットを上書きします。REPLACE を省略すると、IMPORT プロシジャは既存するデータセットを上書きしません。

注意
既存の SAS 世代データセットに書き込むために REPLACE オプションを付けて IMPORT プロシジャを使用すると、最新(base)の世代データセットまたは世代データセットのグループが削除されます。

IMPORT プロシジャに REPLACE オプションを付けて使用して、既存の世代データセットに書き込みを行う場合、次のいずれかを実行します。

- 世代数を増加または減少するために GENMAX=データセットオプションを使用すると、既存のすべての世代が削除されて新規に単一のベース世代データセットに置き換えられます。
- GENMAX=データセットオプションを省略すると、既存のすべての世代は削除され同じ名前の新規に単一のデータセットに置き換えられますが、これは世代データセットではありません。

上記のかわりに SAS DATA ステップで REPLACE=データセットオプションを使用して、永久 SAS データセットを置き換え、SAS データセットの世代グループを保持します。詳細については、“Definitions for Generation Data Sets” (*SAS V9 LIBNAME Engine: Reference*)を参照してください。

例 “例 1: 区切り文字で区切られたファイルのインポート” (1102 ページ)

“例 2: ファイル参照名を使用した、特定の区切り文字で区切られたファイルのインポート” (1106 ページ)

“例 3: タブ区切り文字で区切られたファイルのインポート” (1109 ページ)

“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1112 ページ)

SAS data set options
SAS データセットオプションを指定します。たとえば、結果の SAS データセットにパスワードを割り当てるには、ALTER=、PW=、READ=または WRITE=データセットオプションを使用できます。指定の条件を満たすデータのみインポートするには、WHERE データセットオプションを使用できます。

制限事項　区切り外部ファイル、カンマ区切り外部ファイル、タブ区切り外部ファイルのインポート時は、データセットオプションを指定できません。

参照項目:　*SAS データセットオプション: リファレンス*

<file-format-specific-statements>
データソースでサポートされている構文については、"File Format-Specific Reference for the IMPORT and EXPORT Procedures" (*SAS/ACCESS Interface to PC Files: Reference for the SAS Viya Platform*)を参照してください。

SERVERAUTHDOMAIN=*server authentication-domain*
認証ドメインメタデータオブジェクトの名前を指定して、サーバーへの接続を許可します。

注: このオプションは、2025.02 以降で使用できます。

注: SERVERAUTHDOMAIN はより高いセキュリティを提供しますので、SERVERPASS や SERVERUSER の代わりに使用してください。

DATAROW ステートメント

区切りテキストファイルの指定行番号からのデータの読み込みを開始します。

デフォルト:　GETNAMES=NO の場合: 1、GETNAMES=YES の場合: 2

制限事項:　GETNAMES=NO の場合、DATAROW は 1 以上である必要があります。
GETNAMES=YES の場合、DATAROW は 2 以上である必要があります。

操作:　DATAROW ステートメントは、区切り文字で区切られたファイルと XLSX ファイルに対して有効です。

参照項目:　"GETNAMES ステートメント" (1098 ページ)

例:　"例 3: タブ区切り文字で区切られたファイルのインポート" (1109 ページ)

構文

DATAROW=*n*;

必須引数

n
IMPORT プロシジャがデータの読み込みを開始する入力ファイルの行番号を指定します。

DBENCODING ステートメント

JMP ファイルに使用するエンコーディング文字セットを示します。

操作: DBENCODING ステートメントは、DBMS=JMP の場合のみ有効です。

構文

DBENCODING=*12-char SAS encoding-value*;

必須引数

12-char SAS encoding-value
JMP ファイルと使用するエンコーディングを示します。エンコーディングによって、文字セットの各文字が一意の数値表現にマッピングされ、コードポイントのテーブルが構成されます。各文字は、エンコーディングごとに異なる数値表現がされます。この値には最大で 12 文字を使用できます。

DELIMITER ステートメント

入力ファイルのデータ列を区切る区切り文字を指定します。

デフォルト: .CSV ファイルのカンマ
他のすべてのファイルの種類の空白スペース

操作: DBMS=DLM を指定した場合は、DELIMITER=ステートメントも指定する必要があります。

参照項目: “DBMS=data-source-identifier”には、Viya でサポートされている DBMS 識別子のリストがあります。

例: “例 1: 区切り文字で区切られたファイルのインポート” (1102 ページ)

構文

DELIMITER=*char*'' | '*nn*'x;

必須引数

char | '*nn*'x
入力ファイルのデータ列を区切る区切り文字を指定します。区切り文字は、単一文字または 16 進値として指定できます。たとえば、データ列をアンパサンドで区切る場合、DELIMITER='&'を指定します。

注: DELIMITER=を省略する場合、IMPORT プロシジャは区切り文字をスペースとみなします。

次のいずれかの条件を満たすファイルをインポートする場合は、DELIMITER ステートメントが必要です。

- ファイル拡張子のないファイル
- 拡張子が.TXT で、タブ以外で区切られたデータを含むファイル
- 拡張子が.TXT、.CSV、または.JMP で、空白で区切られたデータを含むファイル

この例は、DBMS 引数と DELIMITER ステートメントを使用して、拡張子が.TXT のファイルにカンマ区切り文字を指定する方法を示しています。

```
proc import datafile="C:\temp\test.txt"
  out=test
  dbms=dlm
  replace;
  dlm=',';
run;
```

FMTLIB ステートメント

値ラベルを指定の SAS フォーマットカタログに保存します。

操作: FMTLIB ステートメントは、DBMS=JMP の場合のみ有効です。

構文

FMTLIB=*<libref.>format-catalog*;

必須引数

<libref.>format-catalog
値ラベルが保存されるフォーマットカタログを指定します。

GETNAMES ステートメント

IMPORT プロシジャが入力ファイルの最初の行のデータ値から SAS 変数名を生成するかどうかを指定します。

デフォルト: YES

制限事項: IMPORT プロシジャと同時に使用する場合にのみ有効です。

VALIDVARNAME=ANY が使用された場合、GETNAMES=はデータ値の接頭辞にアンダースコアを使用しません。

操作: GETNAMES ステートメントは、区切り文字で区切られたファイルにのみ有効です。

例: “例 1: 区切り文字で区切られたファイルのインポート” (1102 ページ)
“例 2: ファイル参照名を使用した、特定の区切り文字で区切られたファイルのインポート” (1106 ページ)
“例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート” (1112 ページ)

構文

GETNAMES=YES | NO | COMPAT ;

必須引数

YES | NO

YES インポートした区切りファイルの最初の行にあるデータ値から、IMPORT プロシジャによって SAS 変数名が生成されることを指定します。

NO IMPORT プロシジャによって SAS 変数名が VAR1、VAR2 などのように生成されることを指定します。

注: 入力ファイルの最初の行のデータ値が読み込まれ、これにブランクなど SAS 名で無効な特殊文字が含まれている場合、SAS はその文字をアンダースコアに変換します。たとえば、変数名 Occupancy Code は、SAS 変数名 Occupancy_Code になります。SAS 変数名は数字からは始められないため、GETNAMES=により、その値の最初の文字を置き換えるのではなく、変数名の接頭辞にアンダースコアを追加します。たとえば 2014.CHANGES は_2014_CHANGES になります。

COMPAT
データファイル内の使用可能な名前に基づいて SAS 名を生成する前者の方法が実装されるように指定します。

> 重要 COMPAT は、XLSX の場合にのみ実装されます。

注 “IMPORT_COMPAT_VARNAME 環境変数” (*SAS グローバルステートメント: リファレンス*)および“LIBNAME Statement: XLSX Engine” (*SAS/ACCESS Interface to PC Files: Reference for the SAS Viya Platform*)で、以前の SAS 名生成方法を指定することもできます。

例 getnames=COMPAT;

GETSHEETNAMES ステートメント

Excel ファイル内のワークシート名を SAS データセットとして返します。

デフォルト: NO

制限事項: IMPORT プロシジャで DBMS=XLS または DMBS=XLSX 引数を指定した場合にのみ有効です。

構文

GETSHEETNAMES=YES | NO;

必須引数

YES | NO

YES IMPORT プロシジャがワークシート名を Excel ファイルから SAS データセットとして返すことを指定します。

NO IMPORT プロシジャがワークシート名を Excel ファイルから SAS データセットとして返さないことを指定します。

GUESSINGROWS ステートメント

スキャンするファイルの行数を指定して、変数に適切なデータの種類と長さを決定します。

デフォルト: 20

制限事項: この値は、DATAROW に対して指定された値より大きい必要があります。

操作: GUESSINGROWS ステートメントは、区切り文字で区切られたファイルにのみ有効です。

構文

GUESSINGROWS=*n* | MAX;

必須引数

n
IMPORT プロシジャがスキャンする入力ファイルの行数を示し、変数の適切なデータの種類と長さを決定します。範囲は 1 から 2147483647 (または MAX)です。

スキャンデータプロセスは、行 1 から GUESSINGROWS オプションによって指定される番号までをスキャンします。

注: SAS レジストリでデフォルトの行の値を変更できます。SAS コマンドラインで、regedit と入力します。**レジストリエディター**が開いたら、**Products** ⇨ **BASE** ⇨ **EFI** ⇨ **GuessingRows** を選択します。

MAX
2147483647 のかわりに指定されます。最大値を指定すると、パフォーマンスに悪影響を及ぼすことがあります。

META ステートメント

JMP メタデータ情報を指定の SAS データセットに保存します。(非推奨)

操作: META ステートメントは、DBMS=JMP の場合のみ有効です。

構文

META=*libref.member-data-set*;

必須引数

libref.member-data-set
メタデータ情報が書き込まれる SAS データセットを指定します。

META ステートメントは、JMP ファイルのインポートではサポートされなくなり、無視されます。代わりに、拡張属性が自動的に使用されます。拡張属性による JMP ファイルのインポート時には、拡張属性は新規 SAS データセットに自動的に割り当てられます。

プログラム内に META ステートメントが残っていることもありますが、その場合は、拡張属性に置換されたことを示す NOTE がログに生成され、無視されます。

VARNAMEROW ステートメント

変数名を含む、区切りテキストファイルの行を指定します。

デフォルト: GETNAMES=YES の場合: 1; GETNAMES=NO の場合、VARNAMEROW は使用されません。

制限事項: IMPORT プロシジャと同時に使用する場合にのみ有効です。

VARNAMEROW ステートメントを使用する場合、DATAROW 値は VARNAMEROW 値よりより大きい必要があります。

操作: VARNAMEROW ステートメントは、区切りファイルに対してのみ有効です。

GETNAMES=NO の場合、VARNAMEROW ステートメントは無視されます。

注: VARNAMEROW は 2024.05 から有効になります。

参照項目: “GETNAMES ステートメント”
“DATAROW ステートメント”

構文

VARNAMEROW=*n*;

必須引数

n
変数名を含む、入力区切りファイル内の行番号を指定します。

詳細

VARNAMEROW ステートメントは、変数名が配置されている区切りファイルの行番号を指定します。この行は DATAROW 指定の前に置く必要があります。DATAROW が指定されていない場合は、VARNAMEROW+1 が使用されます。GETNAMES=NO は、VARNAMEROW の指定をオーバーライドします。

例: IMPORT プロシジャ

例 1: 区切り文字で区切られたファイルのインポート

要素: PROC OPTIONS ステートメントオプション
DATAFILE=
DBMS=
GETNAMES=
OUT=
REPLACE
DELIMITER=ステートメント
OPTIONS ステートメント
PRINT プロシジャ

詳細

この例では、次の区切り外部ファイルをインポートし、WORK.MYDATA という名前の一時 SAS データセットを作成します。

```
Region&State&Month&Expenses&Revenue
Southern&GA&JAN2001&2000&8000
Southern&GA&FEB2001&1200&6000
Southern&FL&FEB2001&8500&11000
Northern&NY&FEB2001&3000&4000
Northern&NY&MAR2001&6000&5000
Southern&FL&MAR2001&9800&13500
Northern&MA&MAR2001&1500&1000
;
```

プログラム

```
options nodate ps=60 ls=80;

proc import datafile="C:\My Documents\myfiles\delimiter.txt"
        dbms=dlm
        out=mydata
        replace;

   delimiter='&';

   getnames=yes;

run;


proc print data=mydata;
run;
```

プログラムの説明

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

```
options nodate ps=60 ls=80;
```

入力ファイルを指定します。 入力ファイルには区切り文字で区切られたファイルを指定します。既存のデータセットが存在する場合、置き換えられます。出力 SAS データセットを識別します。

```
proc import datafile="C:\My Documents\myfiles\delimiter.txt"
        dbms=dlm
        out=mydata
```

```
      replace;
```

区切り文字として& (アンパサンド)を指定します。

```
    delimiter='&';
```

データの最初の行から変数名を生成します。

```
    getnames=yes;

run;
```

出力データセットを印刷します。

```
proc print data=mydata;
run;
```

ログ

SAS ログには、正常なインポートに関する情報が表示されます。この例の場合、次の一部のログにあるように、IMPORT プロシジャが SAS DATA ステップを生成します。

例のコード 31.1 SAS データセットを作成するためにインポートされた外部ファイル

```
1    options nodate ps=60 ls=80; proc import datafile="C:\My
1 ! Documents\myfiles\delimiter.txt" dbms=dlm out=mydata replace; delimiter='&'
1 !  delimiter='&'; getnames=yes;run;

2    /*************************************************************************
3    *   PRODUCT:   SAS
4    *   VERSION:   9.3
5    *   CREATOR:   External File Interface
6    *   DATE:      31JAN11
7    *   DESC:      Generated SAS Datastep Code
8    *   TEMPLATE SOURCE:  (None Specified.)
9    *************************************************************************/
10      data WORK.MYDATA    ;
11      %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
12      infile 'C:\My Documents\myfiles\delimiter.txt' delimiter = '&' MISSOVER
12 !  DSD lrecl=32767 firstobs=2 ;
13        informat Region $8.;
14        informat State $2.;
15        informat Month MONYY7.;
16        informat Expenses best32.;
17        informat Revenue best32.;
18        format Region $8.;
19        format State $2.;
20        format Month MONYY7.;
21        format Expenses best12.;
22        format Revenue best12.;
23      input
24                  Region $
25                  State $
26                  Month
27                  Expenses
28                  Revenue
29      ;
30      if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection
30 ! macro variable */
31      run;

NOTE: The infile 'C:\My Documents\myfiles\delimiter.txt' is:
      Filename=C:\My Documents\myfiles\delimiter.txt,
      RECFM=V,LRECL=32767,File Size (bytes)=254,
      Last Modified=31Jan2011:11:44:29,
      Create Time=31Jan2011:11:44:29

NOTE: 7 records were read from the infile 'C:\My Documents\myfiles\delimiter.txt'.
      The minimum record length was 29.
      The maximum record length was 30.
NOTE: The data set WORK.MYDATA has 7 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.06 seconds
      cpu time            0.03 seconds

7 rows created in WORK.MYDATA from C:\My Documents\myfiles\delimiter.txt.

NOTE: WORK.MYDATA data set was successfully created.
```

出力

アウトプット 31.1　データセット Work.MyData

The SAS System

Obs	Region	State	Month	Expenses	Revenue
1	Southern	GA	JAN2001	2000	8000
2	Southern	GA	FEB2001	1200	6000
3	Southern	FL	FEB2001	8500	11000
4	Northern	NY	FEB2001	3000	4000
5	Northern	NY	MAR2001	6000	5000
6	Southern	FL	MAR2001	9800	13500
7	Northern	MA	MAR2001	1500	1000

例 2: ファイル参照名を使用した、特定の区切り文字で区切られたファイルのインポート

要素:

PROC OPTIONS ステートメントオプション
- DATAFILE=
- DBMS=
- GETNAMES=
- OUT=
- REPLACE

FILENAME ステートメント

PRINT プロシジャ

詳細

この例では、次のスペース区切り文字で区切られたファイルをインポートし、Work.States という名前の一時 SAS データセットを作成します。

```
Region State Capital Bird
South Georgia Atlanta 'Brown Thrasher'
South 'North Carolina' Raleigh Cardinal
North Connecticut Hartford Robin
West Washington Olympia 'American Goldfinch'
Midwest Illinois Springfield Cardinal
```

プログラム

```
filename stdata 'c:\temp\state_data.txt' lrecl=100;


proc import datafile=stdata
      dbms=dlm
      out=states
      replace;
   delimiter=' ';
   getnames=yes;
run;

proc print data=states;
run;
```

プログラムの説明

ファイル名を指定します。

```
filename stdata 'c:\temp\state_data.txt' lrecl=100;
```

入力ファイルを指定します。 入力ファイルには区切り文字で区切られたファイルを指定します。既存のデータセットが存在する場合、置き換えられます。出力 SAS データセットを識別します。

```
proc import datafile=stdata
      dbms=dlm
      out=states
      replace;
```

DELIMITER ステートメントに空白の値を指定します。GETNAMES ステートメントを使用して、データの最初の行から変数名を生成します。

```
   delimiter=' ';
   getnames=yes;
run;
```

データセットを出力します。

```
proc print data=states;
run;
```

ログ

SAS ログには、正常なインポートに関する情報が表示されます。この例の場合、次の一部のログにあるように、IMPORT プロシジャが SAS DATA ステップを生成します。

例のコード 31.2　ファイル参照名を使用した、特定の区切り文字で区切られたファイルのインポート

```
filename stdata 'c:\myfiles\state_data.txt' lrecl=100;

proc import datafile=stdata
            dbms=dlm
            out=states
            replace;
     delimiter=' ';
     getnames=yes;

run;

/*************************************************************************
*   PRODUCT:   SAS
*   VERSION:   9.4
*   CREATOR:   External File Interface
*   DATE:      18APR14
*   DESC:      Generated SAS Datastep Code
*   TEMPLATE SOURCE:  (None Specified.)
*************************************************************************/
    data WORK.STATES    ;
    %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
    infile STDATA delimiter = ' ' MISSOVER DSD lrecl=32767 firstobs=2 ;
       informat Region $7.;
       informat State $16.;
       informat Capital $11.;
       informat Bird $20.;
       format Region $7.;
       format State $16.;
       format Capital $11.;
       format Bird $20.;
    input
                Region $
                State $
                Capital $
                Bird $
    ;
    if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection
! macro variable */
    run;

NOTE: The infile STDATA is:
      Filename=c:\myfiles\state_data.txt,
      RECFM=V,LRECL=32767,File Size (bytes)=225,
      Last Modified=15Apr2014:11:27:14,
      Create Time=15Apr2014:11:25:38

NOTE: 5 records were read from the infile STDATA.
      The minimum record length was 32.
      The maximum record length was 44.
NOTE: The data set WORK.STATES has 5 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.03 seconds
      cpu time            0.03 seconds


5 rows created in WORK.STATES from STDATA.



NOTE: WORK.STATES data set was successfully created.
```

出力

アウトプット 31.2　Work.States データセット

The SAS System

Obs	Region	State	Capital	Bird
1	South	Georgia	Atlanta	Brown Thrasher
2	South	North Carolina	Raleigh	Cardinal
3	North	Connecticut	Hartford	Robin
4	West	Washington	Olympia	American Goldfinch
5	Midwest	Illinois	Springfield	Cardinal

例 3: タブ区切り文字で区切られたファイルのインポート

要素:

PROC OPTIONS ステートメントオプション
- DATAFILE=
- DATAROW=
- DBMS=
- OUT=
- REPLACE

DELIMITER=ステートメント

PRINT プロシジャ

詳細

この例では、次のタブ区切り文字で区切られたファイルをインポートし、Work.Class という名前の一時 SAS データセットを作成します。

入力データ 31.1　入力

```
Name    Gender   Age
Joyce   F        11
Thomas  M        11
Jane    F        12
Louise  F        12
James   M        12
John    M        12
```

```
Robert   M      12
Alice    F      13
Barbara  F      13
Jeffery  M      13
Carol    F      14
Judy     F      14
Alfred   M      14
Henry    M      14
Jenet    F      15
Mary     F      15
Ronald   M      15
William  M      15
Philip   M      16
```

プログラム

```
proc import datafile='C:\userid\pathname\Class.txt'
        out=class
        dbms=dlm
        replace;

     datarow=5;

   delimiter='09'x;
run;

proc print data=class;
run;
```

プログラムの説明

入力ファイルを指定します。 GETNAMES=オプションはデフォルトの'yes'になります。入力ファイルには区切り文字で区切られたファイルを指定します。既存のデータセットが存在する場合、置き換えられます。出力データセットを指定します。

```
proc import datafile='C:\userid\pathname\Class.txt'
        out=class
        dbms=dlm
        replace;
```

最初の行の読み込みは、DATAROW=オプションの指定により行 5 になります。

```
     datarow=5;
```

区切り文字を指定します。 ASCII プラットフォームではタブの 16 進表現は'09'x になります。EBCDIC プラットフォームではタブの 16 進表現は'05'x になります。

```
   delimiter='09'x;
run;
```

出力データセットを印刷します。

```
proc print data=class;
run;
```

ログ

SAS ログには、正常なインポートに関する情報が表示されます。この例の場合、次の一部のログにあるように、IMPORT プロシジャが SAS DATA ステップを生成します。

例のコード 31.3 タブ区切り文字で区切られたファイルのインポート

```
1   proc import datafile='C:\userid\pathname\Class.txt'
2               out=class
3               dbms=dlm
4               replace;
5           datarow=5;
6          delimiter='09'x;
7   run;
8    /*******************************************************************************
9    *  PRODUCT:  SAS
10   *  VERSION:  9.4
11   *  CREATOR:  External File Interface
12   *  DATE:     04DEC18
13   *  DESC:     Generated SAS Datastep Code
14   *  TEMPLATE SOURCE:  (None Specified.)
15   *******************************************************************************/
16      data WORK.CLASS    ;
17      %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
18      infile 'C:\userid\pathname\Class.txt' delimiter='09'x MISSOVER DSD lrecl=32767
18 ! firstobs=5 ;
19          informat Name $7.;
20          informat Gender $1.;
21          informat Age $3.;
22          format Name $7.;
23          format Gender $1.;
24          format Age $3.;
25      input
26                  Name  $
27                  Gender  $
28                  Age  $
29      ;
30      if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro variable */
31      run;

NOTE: The infile 'C:\userid\pathname\Class.txt' is:
      Filename=C:\userid\pathname\Class.txt,
      RECFM=V,LRECL=32767,File Size (bytes)=253,
      Last Modified=26Nov2018:14:06:26,
      Create Time=26Nov2018:13:50:27

NOTE: 16 records were read from the infile 'C:\userid\pathname\Class.txt'.
      The minimum record length was 9.
      The maximum record length was 12.
NOTE: The data set WORK.CLASS has 16 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.01 seconds


16 rows created in WORK.CLASS from C:\userid\pathname\Class.txt.



NOTE: WORK.CLASS data set was successfully created.
NOTE: The data set WORK.CLASS has 16 observations and 3 variables.
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           0.45 seconds
      cpu time            0.20 seconds
```

Output

アウトプット 31.3　Work.Class Data Set

The SAS System

Obs	Name	Gender	Age
1	Louise	F	12
2	James	M	12
3	John	M	12
4	Robert	M	12
5	Alice	F	13
6	Barbara	F	13
7	Jeffery	M	13
8	Carol	F	14
9	Judy	F	14
10	Alfred	M	14
11	Henry	M	14
12	Jenet	F	15
13	Mary	F	15
14	Ronald	M	15
15	William	M	15
16	Philip	M	16

例 4: CSV 拡張子を使用したカンマ区切り文字で区切られたファイルのインポート

要素:

PROC OPTIONS ステートメントオプション
- DATAFILE=
- DBMS=
- GETNAMES=
- OUT=
- REPLACE

PRINT プロシジャ

詳細

この例では、次のカンマ区切り文字で区切られたファイルをインポートし、Work.Shoes という名前の一時 SAS データセットを作成します。

注: SAS では、CSV ファイルのデフォルトの区切り文字はカンマです。

```
"Africa","Boot","Addis Ababa","12","$29,761","$191,821","$769"
"Asia","Boot","Bangkok","1","$1,996","$9,576","$80"
"Canada","Boot","Calgary","8","$17,720","$63,280","$472"
"Central America/Caribbean","Boot","Kingston","33","$102,372","$393,376","$4,454"
"Eastern Europe","Boot","Budapest","22","$74,102","$317,515","$3,341"
"Middle East","Boot","Al-Khobar","10","$15,062","$44,658","$765"
"Pacific","Boot","Auckland","12","$20,141","$97,919","$962"
"South America","Boot","Bogota","19","$15,312","$35,805","$1,229"
"United States","Boot","Chicago","16","$82,483","$305,061","$3,735"
"Western Europe","Boot","Copenhagen","2","$1,663","$4,657","$129"
```

プログラム

```
proc import datafile="C:\temp\test.csv"
      out=shoes
      dbms=csv
      replace;


   getnames=no;
run;

proc print data=work.shoes;
run;
```

プログラムの説明

入力データファイルを指定します。 既存のデータセットが存在する場合、置き換えられます。出力データセットを指定します。

```
proc import datafile="C:\temp\test.csv"
      out=shoes
      dbms=csv
      replace;
```

GETNAMES=オプションを'no'に設定すると、レコード 1 にある変数名は使用されなくなります。

```
   getnames=no;
run;
```

データセットを出力します。

```
proc print data=work.shoes;
run;
```

ログ

SAS ログには、正常なインポートに関する情報が表示されます。この例の場合、次の一部のログにあるように、IMPORT プロシジャが SAS DATA ステップを生成します。

例のコード 31.4　カンマ区切り文字で区切られたファイルのインポート

```
457  proc import datafile="C:\myfiles\test.csv"
458      dbms=csv
459      out=shoes
460      replace;
461      getnames=no;
462  run;

463  /*************************************************************************
464  *   PRODUCT:   SAS
465  *   VERSION:   9.4
466  *   CREATOR:   External File Interface
467  *   DATE:      18APR14
468  *   DESC:      Generated SAS Datastep Code
469  *   TEMPLATE SOURCE:  (None Specified.)
470  *************************************************************************/
471     data WORK.SHOES    ;
472     %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
473     infile 'C:\myfiles\test.csv' delimiter = ',' MISSOVER DSD lrecl=32767 ;
474        informat VAR1 $27.;
475        informat VAR2 $6.;
476        informat VAR3 $13.;
477        informat VAR4 $4.;
478        informat VAR5 $10.;
479        informat VAR6 $10.;
480        informat VAR7 $8.;
481        format VAR1 $27.;
482        format VAR2 $6.;
483        format VAR3 $13.;
484        format VAR4 $4.;
485        format VAR5 $10.;
486        format VAR6 $10.;
487        format VAR7 $8.;
488     input
489                VAR1 $
490                VAR2 $
491                VAR3 $
492                VAR4 $
493                VAR5 $
494                VAR6 $
495                VAR7 $
496     ;
497     if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection
497! macro variable */
498     run;

NOTE: The infile 'C:\myfiles\test.csv' is:
      Filename=C:\myfiles\test.csv,
      RECFM=V,LRECL=32767,File Size (bytes)=657,
      Last Modified=18Apr2014:10:34:47,
      Create Time=18Apr2014:10:33:23

NOTE: 10 records were read from the infile 'C:\myfiles\test.csv'.
      The minimum record length was 51.
      The maximum record length was 81.
NOTE: The data set WORK.SHOES has 10 observations and 7 variables.
```

出力

アウトプット 31.4　Work.Shoes データセット

The SAS System

Obs	VAR1	VAR2	VAR3	VAR4	VAR5	VAR6	VAR7
1	Africa	Boot	Addis Ababa	12	$29,761	$191,821	$769
2	Asia	Boot	Bangkok	1	$1,996	$9,576	$80
3	Canada	Boot	Calgary	8	$17,720	$63,280	$472
4	Central America/Caribbean	Boot	Kingston	33	$102,372	$393,376	$4,454
5	Eastern Europe	Boot	Budapest	22	$74,102	$317,515	$3,341
6	Middle East	Boot	Al-Khobar	10	$15,062	$44,658	$765
7	Pacific	Boot	Auckland	12	$20,141	$97,919	$962
8	South America	Boot	Bogota	19	$15,312	$35,805	$1,229
9	United States	Boot	Chicago	16	$82,483	$305,061	$3,735
10	Western Europe	Boot	Copenhagen	2	$1,663	$4,657	$129

32

JAVAINFO プロシジャ

概要: JAVAINFO プロシジャ

JAVAINFO プロシジャの機能

JAVAINFO プロシジャは、SAS が使用している Java 環境に関する診断情報をユーザーに通知します。診断情報は、SAS Java 環境が正しく設定されていることを確認するために使用でき、SAS テクニカルサポートに問題を報告するときに役立ちます。また、PROC JAVAINFO は Java を使用してその診断を報告するため、SAS Java 環境が正しく動作していることを検証するために頻繁に使用されます。

構文: JAVAINFO プロシジャ

PROC JAVAINFO <*options*>;

ステートメント	タスク
“PROC JAVAINFO ステートメント”	SAS Java 環境に関する診断情報を表示します

PROC JAVAINFO ステートメント

SAS Java 環境に関する診断情報を表示します。

操作: デフォルトでは、SAS Server がロックダウン状態の場合に PROC JAVAINFO が有効になります。SAS 管理者は VIYA_LOCKDOWN_USER_DISABLED_METHODS=JAVA を指定して、PROC JAVAINFO を無効にできます。詳細については、“Enable or Disable Access Methods” (*SAS Viya Platform: Programming Run-Time Servers*)を参照してください。

構文

PROC JAVAINFO <*options*>;

オプション引数

ALL
SAS Java 環境に関する最新情報を表示します。

CLASSPATHS
Java が使用しているクラスパスに関する情報を表示します。

HELP
JAVAINFO プロシジャを使用する際の使用上の支援を提供します。

JREOPTIONS
JREOPTIONS 構成オプションの指定時に設定される Java プロパティを表示します。

- JREOPTIONS は PROC JAVAINFO での使用時に、Java の起動時に設定される JREOPTIONS Java プロパティを指定します。
- JREOPTIONS は PROC OPTIONS での使用時に、SAS の起動時に構成ファイルにある Java オプションを指定します。

注: SAS.cfg はインストール時に指定される構成ファイルですが、その他の構成ファイルが指定可能です。

OS
SAS が実行中のオペレーティングシステムに関する情報を表示します。

version
SAS が使用している Java Runtime Environment (JRE)を表示します。

33

JSON プロシジャ

概要: JSON プロシジャ

JSON プロシジャの機能

JSON プロシジャは SAS データセットからデータを読み込み、そのデータを JSON で外部ファイルに書き込みます。[1] 表現。複数のオプションを使用してエクスポートしたデータを制御できます。各オプションによってコンテンツは削除され、形式に影響が及ぼされます。データを SAS データセットからエクスポートするほか、PROC JSON はステートメントを提供し、このステートメントを使用して追加データを外部ファイルに書き込んだり、JSON コンテナーを制御したりできます。

SAS は、JSON ファイルを読み取るための JSON エンジンを提供します。詳細については、*"LIBNAME ステートメント: JSON エンジン" (SAS グローバルステートメント: リファレンス)*を参照してください。

概念: JSON プロシジャ

JSON 出力ファイルコンテナー

JSON 出力ファイルの構造

JSON 出力ファイルは、最低でも 1 つのコンテナーから構成され、これが最上位コンテナーと呼ばれます。すべてのデータ、メタデータ、および追加コンテナーは最上位コンテナー内に格納されます。最上位コンテナーの種類は、PROC JSON ステートメントの直前のステートメントおよびそのステートメントで有効化されるオプションによって決定されます。

1. Java Script Object Notation (JSON)は、テキストベースのオープンスタンダードデータ形式で、人間が読み取り可能なデータを交換できるように設計されています。JSON は JavaScript プログラミング言語のサブセットを基本として、データオブジェクトの記述には JavaScript 構文を使用します。

JSON コンテナー

JSON 出力は、データ構造コンテナーの 2 つの種類から構成されます。

JSON オブジェクトコンテナー({ })
: 左中かっこ({)で始まり、右中かっこ(})で終わります。オブジェクトコンテナーは、名前と値のペアとして書き込まれるキーと値のペアを収集します。値には、サポートされる JSON データ型、オブジェクト、または配列のいずれかを指定できます。各名前の後にはコロン、値の順に続きます。キーと値のペアはカンマで区切ります。

JSON 配列コンテナー([ ])
: 左角かっこ([)で始まり、右角かっこ(])で終わります。配列コンテナーは、名前を指定せずに値のリストとして書き込まれる値のリストを収集します。値には、サポートされる JSON データ型、オブジェクト、または配列のいずれかを指定できます。値はカンマで区切ります。

オブジェクトコンテナーの作成

次のステートメントによって、オブジェクトコンテナーが作成されます。

- デフォルトオプションを有効にした状態の EXPORT ステートメントは、オブジェクトコンテナーを最上位コンテナーとして開きます。このオブジェクトコンテナーは暗黙的に開かれ、自動的に閉じられます。

```
proc json out="example.json";
  export sashelp.class (where=(age=11));
...
```

- WRITE VALUES ステートメントは、PROC JSON ステートメントの後の最初のステートメントである場合、オブジェクトコンテナーを最上位コンテナーとして開きます。このオブジェクトコンテナーは暗黙的に開かれ、自動的に閉じられます。

```
proc json out="example.json";
  write values "container" "object";
  write values "created" "implicitly";
...
```

- WRITE OPEN OBJECT ステートメントは、オブジェクトコンテナーを明示的に開きます。このコンテナーは、WRITE CLOSE ステートメントを使用して明示的に閉じる必要があります。

```
proc json out="example.json"
  write open object;
  write values "container" "object";
  write values "created" "explicitly";
  write close;
...
```

配列コンテナーの作成

次のステートメントによって、配列コンテナーが作成されます。

- NOSASTAGS オプションを有効にした状態の EXPORT ステートメントは、JSON 配列コンテナーを最上位コンテナーとして開きます。この最上位の配列コンテナーは暗黙的に開かれ、自動的に閉じられます。

```
proc json out="example.json" pretty;
  export sashelp.class (where=(age=11)) / nosastags;
...
```

- WRITE OPEN ARRAY ステートメントは、配列コンテナーを明示的に開きます。このコンテナーは、WRITE CLOSE ステートメントを使用して明示的に閉じる必要があります。

```
proc json out="example.out";
  write open array;
  write values "container" "array";
  write values "created" "explictly";
  write close;
...
```

コンテナーの入れ子構造

最上位コンテナーには、任意の数のコンテナーを含めることができます。同様に、コンテナーは任意の深さまでコンテナーを入れ子構造にできます。コンテナーを入れ子構造にするときには、現在のコンテナーのデータ構造要件を順守するよう注意してください。

- オブジェクトにはキーと値のペアのリストが必要になり、その場合、値自体をオブジェクトまたは配列に指定できます。
- 配列にはキーと値のペアのそうした構造要件が含まれないため、値、オブジェクト、または配列のリストのみです。
- WRITE VALUES ステートメントまたはオブジェクトコンテナーのステートメントは、偶数の値になる必要があり、キーと値のペアのキー部分は文字列にする必要があります。

この例では、黙示的に開いたオブジェクト内で配列が入れ子構造になります。

```
proc json out="example.json";
  write values "level" 1;            /* implicit open of object       */
  write values "container" "object";   /* write data to object          */
  write values "created" "implicitly"; /* write data to object          */
  write values "nest";               /* required string to start object */
  write open array;                 /* explicit open of array        */
  write values "level" 2;            /* write data to array           */
  write values "container" "array";    /* write data to array          */
  write values "created" "explicitly"; /* write data to array          */
```

```
  write close;                  /* close explicit open          */
run;
```

```
{"level":1,"container":"object","created":"implicitly","nest":["level",2,
"container","array","created","explicitly"]}
```

この例では、エクスポートしたデータは 3 つの配列コンテナー内で入れ子構造になります。

```
proc json out="example.json";
  write open array;                  /* explicit open of array */
  write values "level" 1;             /* write data to array    */
  write values "container" "array";          /* write data to array    */
  write values "created" "explicitly";       /* write data to array    */
  write open array;                 /* explicit open of array */
  write values "level" 2;             /* write data to array    */
  write values "container" "array";          /* write data to array    */
  write values "created" "explicitly";       /* write data to array    */
  export sashelp.class (where=(age=11)) / nokeys;  /* export data without keys*/
  write close;                      /* close explicit open    */
  write close;                      /* close explicit open    */
run;
```

```
["level",1,"container","array","created","explicitly",["level",2,"container",
"array","created","explicitly","SASJSONExport","1.0 NO",
"SASTableData+CLASS",[["Joyce","F",11,51.3,50.5],["Thomas","M",11,57.5,85]]]]
```

欠損値

SAS 内の欠損値は、特定のオブザベーションまたは変数のデータを含まない変数の値タイプです。デフォルトでは、SAS は 1 つの期間として欠損数値を表し、空白スペースとして欠損文字値を表します。

また、JSON には null 値という欠損値の概念も含まれ、これは情報が存在しないことを示す特別な値です。

PROC JSON は、次が TRUE のときに JSON null 値を JSON 出力ファイルに書き込みます。

- SAS データセットの数値変数には欠損値が含まれます
- WRITE VALUES ステートメントは NULL キーワードを値として書き込みます

デフォルトでは、SAS データセット文字変数に欠損値が含まれるとき、SAS は空の文字列("")を JSON 出力ファイルに書き込みます。ただし、NOTRIMBLANKS オプションを指定する場合、ブランクの文字列全体が JSON 出力ファイルに書き込まれます。

入力文字列のスキャン

JSON 文字列は、二重引用符で囲まれたゼロ以上の文字シーケンスです。JSON 文字列には任意の Unicode 文字を指定できますが、二重引用符(")、バックスラッシュ(\)、または制御文字は指定できません。JSON 文字列にはエスケープ文字としてバックスラッシュを含めることができ、このエスケープ文字によって文字シーケンスの次の文字に関する代替翻訳が呼び出されます。たとえば、JSON 文字列に二重引用符を含めることができるようにするには、"の前に\文字を置くと"はエスケープされます。

デフォルトでは、JSON 文字列として受け入れ可能な文字のみが含まれていることを保証するため、PROC JSON は入力文字列をスキャンします。入力文字列内の受け入れられない文字は、適切なエスケープシーケンスに置換されます。

NOSCAN オプションを使用すると、入力文字列に受け入れ可能な文字が含まれるか、またはすでにスキャン済みと認識されるように指定できます。NOSCAN がサポートされるのは、PROC JSON ステートメント、EXPORT ステートメント、および WRITE VALUES ステートメントです。

JSON 出力ファイルのエンコーディング

デフォルトでは、結果の JSON 出力ファイルでは Unicode エンコーディングの UTF-8 が使用されます。JSON 出力ファイルのエンコードを上書きするには、FILENAME ステートメント内で ENCODING=オプションを使用します。ただし、これを実行できるのは次の状況に限られます。

- 現在の SAS セッションエンコードが UTF-8 の場合、次の Unicode エンコーディング形式のいずれかを指定できます。UTF-8 および次のエンコーディング形式のみが JSON スタンダードに準拠しています。他のエンコーディング形式は、スタンダードに準拠する JSON パーサーによって認識されない可能性があります。

  表 33.1 UTF-8 SAS セッションで有効なエンコーディング

名前	説明
utf-16be	Unicode (UTF-16BE)
utf-32be	Unicode (UTF-32BE)
utf-32le	Unicode (UTF-32LE)

- 現在の SAS セッションのエンコーディングが UTF-8 ではない場合、JSON 出力ファイルのエンコーディングを上書きできるのは、現在の SAS セッションエンコーディングに US-ASCII との互換性があり、JSON 出力ファイルに書き込まれるすべての文字列に含まれるのが小範囲の Latin1 文字(コードポイント 0-127)のみである場合に限られます。

たとえば、次のコードによって、Unicode UTF-16BE 文字セットエンコーディングを使用する JSON 出力ファイルがエクスポートされます。現在の SAS セッションエンコーディングは Wlatin1 です。FILENAME ステートメント内の ENCODING=オプションは、外部ファイルへの書き込み時に PROC JSON がデータを Wlatin1 から指定された Unicode UTF-16BE へトランスコードするよう命令します。

```
filename jsonout "path-to-JsonOutput.json" encoding="utf-16be";

proc json out=jsonout;
  export sashelp.class;
quit;
```

構文: JSON プロシジャ

PROC JSONOUT=*fileref* | "*external-file*" <*options*>;
EXPORT<*libref*.>*SAS-data-set*< (*SAS-data-set-options*)> < /*options* > ;
WRITE VALUES*value(s)*< /*options* > ;
WRITE OPEN *type* ;
WRITE CLOSE ;

ステートメント	タスク	例
“PROC JSON ステートメント”	SAS データセットの JSON ファイルへのエクスポート。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7
“EXPORT ステートメント”	エクスポートする SAS データセットを識別し、結果出力を制御します	Ex. 1, Ex. 2, Ex. 3, Ex. 5, Ex. 6, Ex. 7
“WRITE VALUES ステートメント”	1 つ以上の値を出力ファイルに書き込みます	Ex. 4, Ex. 5, Ex. 6
“WRITE OPEN ステートメント”	出力ファイル内で JSON コンテナーを開いて入れ子構造にします	Ex. 4, Ex. 5, Ex. 6
“WRITE CLOSE ステートメント”	出力ファイル内で開いている JSON コンテナーを閉じます	Ex. 4, Ex. 5, Ex. 6

PROC JSON ステートメント

JSON 出力ファイルを指定し、結果出力を制御します。

例:

構文

PROC JSONOUT=*fileref* | "*external-file*" <*options*>;

オプション引数の要約

FMTCHARACTER
NOFMTCHARACTER
: 文字の SAS 出力形式が SAS データセット変数に関連付けられている場合、文字の SAS 出力形式を結果出力に適用するかどうかを決定します。

FMTDATETIME
NOFMTDATETIME
: 日付、時刻、または日時の SAS 出力形式が SAS データセット変数に関連付けられている場合、日付、時刻、または日時の SAS 出力形式を結果出力に適用するかどうかを決定します。

FMTNUMERIC
NOFMTNUMERIC
: 数値の SAS 出力形式が SAS データセット変数に関連付けられている場合、数値の SAS 出力形式を結果出力に適用するかどうかを決定します。

KEYS
NOKEYS
: エクスポートされたオブザベーションが JSON オブジェクトとして書き込むのか、または JSON 配列として書き込むのかを決定します。

PRETTY
NOPRETTY
: JSON 出力のフォーマット方法を決定します。

SASTAGS
NOSASTAGS
: EXPORT ステートメントを使用するときに SAS メタデータを含めるか抑制するかを決定します。

SCAN

NOSCAN
: 受け入れ可能な文字のみが JSON 出力ファイルにエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

TRIMBLANKS

NOTRIMBLANKS
: JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。

必須引数

OUT=*fileref* | "*external-file*"
: JSON 出力ファイルを識別します。

fileref
: JSON 出力ファイルに割り当てられる SAS ファイル参照名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

"*external-file*"
: JSON 出力ファイルの物理的な場所です。完全なパス名とファイル名を含める必要があります。物理名は単一引用符または二重引用符で囲みます。最大長は 200 文字です。

オプション引数

FMTCHARACTER | NOFMTCHARACTER
: 文字の SAS 出力形式が SAS データセット変数に関連付けられている場合、文字の SAS 出力形式を結果出力に適用するかどうかを決定します。

別名
: FMTCHAR

: NOFMTCHAR

デフォルト
: NOFMTCHARACTER

操作
: PROC JSON ステートメント、EXPORT ステートメントまたはその両方で FMTCHARACTER | NOFMTCHARACTER を指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

注
: ユーザー定義形式もサポートされます。

FMTDATETIME | NOFMTDATETIME
: 日付、時刻、または日時の SAS 出力形式が SAS データセット変数に関連付けられている場合、日付、時刻、または日時の SAS 出力形式を結果出力に適用するかどうかを決定します。 SAS 出力形式を適用すると、結果の JSON 出力内の日付と時刻の値がより読みやすくなります。

別名
: FMTDT

: NOFMDTDT

デフォルト　FMTDATETIME

操作　FMTDATETIME | NOFMTDATETIME は、PROC JSON ステートメント、EXPORT ステートメント、またはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

ヒント　ユーザー定義形式もサポートされます。

FMTNUMERIC | NOFMTNUMERIC
数値の SAS 出力形式が SAS データセット変数に関連付けられている場合、数値の SAS 出力形式を結果出力に適用するかどうかを決定します。

別名　FMTNUM

NOFMTNUM

デフォルト　NOFMTNUMERIC

制限事項　SAS 出力形式 BEST*w*、E*w*、および *w* のみ。*d* は出力ファイルに JSON 番号を書き込みます。その他すべての SAS 出力形式は、JSON 文字列となります。

要件　FMTNUMERIC は数値の SAS 出力形式を適用します。日付、時刻、および日時の SAS 出力形式の場合、FMTDATETIME オプションを使用してください。

操作　NOFMTNUMERIC を使用するか、または数値変数に関連 SAS 出力形式が含まれない場合、最大 12 桁の数値が出力ファイルに書き込まれます。

FMTNUMERIC | NOFMTNUMERIC は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

注　SAS 出力形式(BEST *w*.、E *w*.、および *wd*)による出力ファイルへの JSON 番号の書き込みは、FMTNUMERIC オプションに関係なく、整数値には適用されません。

ヒント　ユーザー定義形式もサポートされます。

KEYS | NOKEYS
エクスポートされたオブザベーションが JSON オブジェクトとして書き込むのか、または JSON 配列として書き込むのかを決定します。 JSON オブジェクトは、オブザベーションの SAS 変数値をキーと値のペアとして格納します。SAS 変数名がキーです。JSON 配列は変数値のみを格納します

デフォルト　KEYS

操作 KEYS | NOKEYS は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

ヒント EXPORT ステートメントの使用時は、NOKEYS を使用して、オブザベーションデータ内の SAS 変数名を抑制します。

例 “例 3: EXPORT ステートメントの使用時にオブザベーションデータ内の SAS 変数名を抑制する” (1146 ページ)

PRETTY | NOPRETTY
JSON 出力のフォーマット方法を決定します。 PRETTY は、インデントを使用して JSON コンテナー構造を表すより人間が読みやすい形式を作成します。 NOPRETTY は、出力を 1 行で書き込みます。

デフォルト NOPRETTY

制限事項 PRETTY | NOPRETTY を指定できるのは、PROC JSON ステートメントのみです。

SASTAGS | NOSASTAGS
EXPORT ステートメントを使用するときに SAS メタデータを含めるか抑制するかを決定します。 メタデータは、SAS エクスポートバージョン、エクスポートした SAS データセット名、および任意のデフォルトオプション指定(PRETTY など) から構成されます。

プロシジャ要求の最初のステートメントが EXPORT である場合、エクスポートしたデータの最上位コンテナーは JSON オブジェクトコンテナーです。SAS メタデータは、エクスポートしたデータに先行します。NOSASTAGS が指定されている場合、最上位コンテナーは JSON 配列コンテナーであり、SAS メタデータは抑制されます。

PROC JSON 要求の中で WRITE OPEN ステートメントが EXPORT ステートメントに先行している場合に発生する内容については、次のトピックを参照してください。

- WRITE OPEN ARRAY — “例 5: 同じプログラム内での値の書き込みとデータのエクスポート” (1150 ページ)。

デフォルト SASTAGS

操作 NOSASTAGS は、エクスポートしたデータの最も外側のコンテナーに影響を及ぼします。最上位 JSON コンテナーは、EXPORT ステートメントがプロシジャ要求の最初のステートメントである場合にのみ影響を受けます。

SASTAGS | NOSASTAGS は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

注 エクスポートされたオブザベーションの出力形式を変更するには、KEYS | NOKEYS オプションを使用します。

例 “例 1: デフォルトオプションを使用した JSON ファイルのエクスポート” (1143 ページ)

“例 2: 最上位 JSON 配列コンテナーへのエクスポート” (1144 ページ)

“例 6: エクスポートされたデータへの値の書き込みとコンテナーの制御” (1154 ページ)

SCAN | NOSCAN
受け入れ可能な文字のみが JSON 出力ファイルにエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

デフォルト SCAN

操作 NOSCAN は、入力文字列に受け入れ可能な JSON テキストが含まれるか、またはすでにスキャン済みと認識されるように指定します。NOSCAN が有効な場合、入力文字列または文字値はそのまま取得され、出力 JSON 文字列は 引用符で囲まれます。

SCAN | NOSCAN は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つ全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

参照項目: “入力文字列のスキャン” (1124 ページ)

TRIMBLANKS | NOTRIMBLANKS
JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。 スペース文字のみ削除されます。

デフォルト TRIMBLANKS

操作 TRIMBLANKS | NOTRIMBLANKS は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つ全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

EXPORT ステートメント

エクスポートする SAS データセットを識別し、結果出力を制御します。

別名: EX

注: 複数の EXPORT ステートメントをサブミットすると、複数の SAS データセットを JSON 出力ファイルにエクスポートできます。

結果の JSON 出力では、Unicode エンコーディング形式 UTF-8 が使用されます。EXPORT ステートメントで ENCODING=データオプションを指定しても、出力ファイル内のエンコーディングを上書きできません。エンコーディングの上書きについては、“JSON 出力ファイルのエンコーディング” (1124 ページ)を参照してください。

参照項目: “データのエクスポート” (1139 ページ)

例: “例 1: デフォルトオプションを使用した JSON ファイルのエクスポート” (1143 ページ)

“例 2: 最上位 JSON 配列コンテナーへのエクスポート” (1144 ページ)

“例 3: EXPORT ステートメントの使用時にオブザベーションデータ内の SAS 変数名を抑制する” (1146 ページ)

“例 5: 同じプログラム内での値の書き込みとデータのエクスポート” (1150 ページ)

“例 6: エクスポートされたデータへの値の書き込みとコンテナーの制御” (1154 ページ)

“例 7: エクスポートされた出力への SAS 出力形式の適用” (1159 ページ)

構文

EXPORT<*libref.*> *SAS-data-set* < (*SAS-data-set-options*)> < /*options*>;

オプション引数の要約

(*SAS-data-set-options*)
入力 SAS データセットに適用される SAS データセットオプションを指定します。

必須引数

<*libref .*>*SAS-data-set*
エクスポートする SAS データセットを 1 または 2 レベルの SAS 名(ライブラリとメンバー名)で識別します。1 レベルの名前を指定した場合、JSON プロシジャはデフォルトで USER ライブラリ(割り当てられている場合)または WORK ライブラリのどちらかを使用します。

オプション引数

(*SAS-data-set-options*)
入力 SAS データセットに適用される SAS データセットオプションを指定します。 たとえば、エクスポートするデータセットにパスワードが割り当てられている場合、ALTER=、PW=、READ=、または WRITE=データセットオプションを使用できます。指定の条件を満たすサブセットのデータをエクスポートするには、WHERE オプション=を使用します。SAS データセットオプションの詳細については、*SAS データセットオプション: リファレンス*を参照してください。

スラッシュ区切り文字の後に使用するオプション

FMTCHARACTER | NOFMTCHARACTER
文字の SAS 出力形式が SAS データセット変数に関連付けられている場合、文字の SAS 出力形式を結果出力に適用するかどうかを決定します。

別名 FMTCHAR

NOFMTCHAR

デフォルト NOFMTCHARACTER

操作 PROC JSON ステートメント、EXPORT ステートメントまたはその両方で FMTCHARACTER | NOFMTCHARACTER を指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

FMTDATETIME | NOFMTDATETIME
日付、時刻、または日時の SAS 出力形式が SAS データセット変数に関連付けられている場合、日付、時刻、または日時の SAS 出力形式を結果出力に適用するかどうかを決定します。 SAS 出力形式を適用すると、結果の JSON 出力内の日付と時刻の値がより読みやすくなります。

別名 FMTDT

NOFMDTDT

デフォルト FMTDATETIME

操作 FMTDATETIME | NOFMTDATETIME は、PROC JSON ステートメント、EXPORT ステートメント、またはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

FMTNUMERIC | NOFMTNUMERIC
数値の SAS 出力形式が SAS データセット変数に関連付けられている場合、数値の SAS 出力形式を結果出力に適用するかどうかを決定します。

別名 FMTNUM

NOFMTNUM

デフォルト NOFMTNUMERIC

制限事項 SAS 出力形式 BEST*w*、E*w*、および *w* のみ。*d* は出力ファイルに JSON 番号を書き込みます。その他すべての SAS 出力形式は、JSON 文字列となります。

要件 FMTNUMERIC は数値の SAS 出力形式を適用します。日付、時刻、および日時の SAS 出力形式の場合、FMTDATETIME オプションを使用してください。

操作 NOFMTNUMERIC を使用するか、または数値変数に関連 SAS 出力形式が含まれない場合、最大 12 桁の数値が出力ファイルに書き込まれます。

FMTNUMERIC | NOFMTNUMERIC は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

KEYS | NOKEYS
エクスポートされたオブザベーションが JSON オブジェクトとして書き込むのか、または JSON 配列として書き込むのかを決定します。 JSON オブジェクトは、オブザベーションの SAS 変数値をキーと値のペアとして格納します。変数名がキーです。JSON 配列は変数値のみを格納します

デフォルト KEYS

操作 KEYS | NOKEYS は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

ヒント EXPORT ステートメントの使用時は、NOKEYS を使用して、オブザベーションデータ内の SAS 変数名を抑制します。

例 “例 3: EXPORT ステートメントの使用時にオブザベーションデータ内の SAS 変数名を抑制する” (1146 ページ)

SASTAGS | NOSASTAGS
EXPORT ステートメントを使用するときに SAS メタデータを含めるか抑制するかを決定します。 メタデータは、SAS エクスポートバージョン、エクスポートした SAS データセット名、および任意のデフォルトオプション指定(PRETTY など)から構成されます。

プロシジャ要求の最初のステートメントが EXPORT である場合、エクスポートしたデータの最上位コンテナーは JSON オブジェクトコンテナーです。SAS メタデータは、エクスポートしたデータに先行します。NOSASTAGS が指定されている場合、最上位コンテナーは JSON 配列コンテナーであり、SAS メタデータは抑制されます。

PROC JSON 要求の中で WRITE OPEN ステートメントが EXPORT ステートメントに先行している場合に発生する内容については、次のトピックを参照してください。

- WRITE OPEN ARRAY — “例 5: 同じプログラム内での値の書き込みとデータのエクスポート” (1150 ページ)。

デフォルト SASTAGS

操作 NOSASTAGS は、エクスポートしたデータの最も外側のコンテナーに影響を及ぼします。最上位 JSON コンテナーは、EXPORT ステートメントがプロシジャ要求の最初のステートメントである場合にのみ影響を受けます。

SASTAGS | NOSASTAGS は、PROC JSON ステートメント、EXPORT ステートメント、あるいはその両方で指定できます。両方のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

注 エクスポートされたオブザベーションの出力形式を変更するには、KEYS | NOKEYS オプションを使用します。

例 “例 1: デフォルトオプションを使用した JSON ファイルのエクスポート” (1143 ページ)

“例 2: 最上位 JSON 配列コンテナーへのエクスポート” (1144 ページ)

“例 6: エクスポートされたデータへの値の書き込みとコンテナーの制御” (1154 ページ)

SCAN | NOSCAN
受け入れ可能な文字のみが JSON 出力にエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

デフォルト SCAN

操作 NOSCAN は、入力文字列に受け入れ可能な JSON テキストが含まれるか、またはすでにスキャン済みと認識されるように指定します。NOSCAN が有効な場合、入力文字列または文字値はそのまま取得され、出力 JSON 文字列は 引用符で囲まれます。

SCAN | NOSCAN は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つのステートメント全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

参照項目: “入力文字列のスキャン” (1124 ページ)

TABLENAME="*name*"
エクスポートした SAS データセットの名前を指定します。 名前は JSON 出力ファイル内の SAS メタデータとしてエクスポートされます。名前を単一引用符または二重引用符で囲みます。

デフォルト デフォルトは、SAS データセットメンバー名です。

要件 TABLENAME=オプションでは、SASTAGS オプションに JSON 出力内の SAS メタデータを含める必要があります。

TRIMBLANKS | NOTRIMBLANKS
JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。 スペース文字のみ削除されます。

デフォルト TRIMBLANKS

操作 TRIMBLANKS | NOTRIMBLANKS は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つのステートメント全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

WRITE VALUES ステートメント

1 つ以上の値を JSON 出力ファイルに書き込みます。

別名: W V

操作: WRITE VALUES ステートメントが PROC JSON ステートメントの後の最初のステートメントである場合、PROC JSON によって JSON オブジェクトが最上位コンテナーとして開かれます。PROC JSON は、黙示的に開かれた最上位コンテナーを自動的に閉じます。

1 つの WRITE VALUES ステートメント内に複数の値を指定することは、複数の WRITE VALUES ステートメントをそれぞれ 1 つの値のみを指定してサブミットすることに相当します。値の順番のみが重要です。

参照項目: “JSON 出力ファイルへの値の書き込み” (1139 ページ)

例: “例 5: 同じプログラム内での値の書き込みとデータのエクスポート” (1150 ページ)
“例 6: エクスポートされたデータへの値の書き込みとコンテナーの制御” (1154 ページ)

構文

WRITE VALUES*value(s)* < /*options*>;

オプション引数の要約

SCAN
NOSCAN
受け入れ可能な文字のみが JSON 出力にエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。
TRIMBLANKS

NOTRIMBLANKS
JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。

必須引数

value(s)
1 つ以上の値を JSON 出力ファイルに書き込むよう指定します。値は空白スペースで区切ります。 値は次のうちいずれかになります。

- *文字列*。文字列が引用符で囲まれる場合、コンテンツまたは長さに関して制約はありません。ただし、文字列が引用符で囲まれない場合、次の規則が適用されます。
  - 文字列の最大長は 256 バイトです。
  - 最初の文字はラテンアルファベット(A–Z、a–z)またはアンダースコアで始める必要があります。開始文字以外には、英字、数字、アンダースコアを使用できます。
  - 文字列にはアンダースコア以外のブランクや特殊文字を含めることができません。文字列には大文字と小文字を両方使用できます。

  注: 文字列は、単一引用符または二重引用符で囲むことができます。
- *number* は整数、浮動小数、または指数関数形式で表されます。
- ブール値 TRUE | T または FALSE | F.
- NULL | N.

たとえば、ステートメント write values "success" true;によって、次の結果が JSON 出力ファイルに書き込まれます。

- **"success":true** 現在のコンテナーが JSON オブジェクトの場合
- **“success”, true** 現在のコンテナーが JSON 配列の場合

要件 JSON オブジェクトコンテナーに値を書き込む際、キーと値のペアのキー部分は文字列にする必要があります。たとえば、write values "abcd" "1";というステートメントが適切です。ただし、ステートメント write values 1 "abcd"; ではエラーが発生します。

スラッシュ区切り文字の後に使用するオプション

SCAN | NOSCAN
受け入れ可能な文字のみが JSON 出力にエクスポートされることを保証するため、PROC JSON が入力文字列をスキャンしてエンコードするかどうかを指定します。

デフォルト SCAN

操作 NOSCAN は、入力文字列に受け入れ可能な JSON テキストが含まれるか、またはすでにスキャン済みと認識されるように指定します。NOSCAN

が有効な場合、入力文字列または文字値はそのまま取得され、出力 JSON 文字列は 引用符で囲まれます。

SCAN | NOSCAN は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つのステートメント全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

参照項目: “入力文字列のスキャン” (1124 ページ)

TRIMBLANKS | NOTRIMBLANKS
JSON 出力内の文字データの最後で末尾のブランクを削除または保持するかどうかを決定します。 スペース文字のみ削除されます。

デフォルト TRIMBLANKS

操作 TRIMBLANKS | NOTRIMBLANKS は、PROC JSON ステートメント、EXPORT ステートメント、WRITE VALUES ステートメント、あるいは 3 つのステートメント全部で指定できます。複数のステートメントでこのオプションが指定されている場合、EXPORT ステートメントの指定が優先されます。

WRITE OPEN ステートメント

出力ファイル内で JSON コンテナーを開いて入れ子構造にします。

別名: W O

操作: WRITE OPEN ステートメントが PROC JSON ステートメントの後の最初のステートメントである場合、WRITE OPEN ステートメントは最上位コンテナーを確立します。
WRITE OPEN ステートメントを使用して明示的に開いたコンテナーの WRITE CLOSE ステートメントをサブミットします。

参照項目: “JSON 出力ファイルへの値の書き込み” (1139 ページ)

例: “例 5: 同じプログラム内での値の書き込みとデータのエクスポート” (1150 ページ)
“例 6: エクスポートされたデータへの値の書き込みとコンテナーの制御” (1154 ページ)

構文

WRITE OPEN *type*;

必須引数

type
JSON コンテナーの種類を指定します。

ARRAY
JSON コンテナーが配列であるよう指定します。この配列によって値のリストが収集されます。

別名 A

例 write open array;

OBJECT
JSON コンテナーがオブジェクトであるよう指定します。このオブジェクトによってキーと値のペアが収集されます。

別名 O

例 write open object;

WRITE CLOSE ステートメント

JSON 出力ファイル内で開いている JSON コンテナーを閉じます。

別名: W C

制限事項: WRITE CLOSE ステートメントは、PROC JSON ステートメントの後の最初のステートメントにすることはできません。

操作: The WRITE CLOSE ステートメントは、WRITE OPEN ステートメントを使用して明示的に開かれたいずれかの種類の最近開いたコンテナーを閉じます。WRITE OPEN ステートメントを使用してコンテナーを明示的に開いた場合に限り、コンテナーの WRITE CLOSE ステートメントをサブミットする必要があります。

参照項目: “JSON 出力ファイルへの値の書き込み” (1139 ページ)

例: “例 5: 同じプログラム内での値の書き込みとデータのエクスポート” (1150 ページ)
“例 6: エクスポートされたデータへの値の書き込みとコンテナーの制御” (1154 ページ)

構文

WRITE CLOSE ;

使用法: JSON プロシジャ

データのエクスポート

JSON プロシジャを使用すると、1 つ以上の SAS データセットを JSON 出力ファイルにエクスポートできます。プロシジャステートメントでは、エクスポートする JSON 出力ファイル名と SAS データセット名を指定します。たとえば、次の PROC JSON コードは Sashelp.Class という名前の SAS データセット内のデータを Output.json という名前の JSON 出力ファイルに書き込みます。

```
proc json out="path-to-Output.json";
  export sashelp.class;
run;
```

エクスポートされたデータはデフォルトにより JSON オブジェクトコンテナー({ }) に書き込まれます。データセットの各オブザベーションも、JSON オブジェクトコンテナーに配置されます。外側のコンテナーとオブザベーションコンテナーの両方を配列として作成できるオプションが利用可能です。SASTAGS (または NOSASTAGS) 引数は、外側のコンテナーを制御します。KEYS (または NOKEYS)引数は、オブザベーションのコンテナーを制御します。詳細については、“例 1: デフォルトオプションを使用した JSON ファイルのエクスポート” (1143 ページ)、“例 2: 最上位 JSON 配列コンテナーへのエクスポート” (1144 ページ)および“例 3: EXPORT ステートメントの使用時にオブザベーションデータ内の SAS 変数名を抑制する” (1146 ページ)を参照してください。

JSON 出力ファイルへの値の書き込み

PROC JSON は、データを明示的に JSON 出力ファイルに書き込むための WRITE VALUES ステートメントを提供します。WRITE VALUES ステートメントを使用すると、1 つ以上の値を JSON オブジェクトコンテナーまたは JSON 配列コンテナーに書き込むことができます。値には、文字列、数字、ブール値 TRUE/FALSE、または NULL キーワードを指定できます。

WRITE VALUES ステートメントは、PROC JSON ステートメントで指定された最初のステートメントである場合、指定された値を収集するための最上位 JSON オブジェクトコンテナーを暗黙的に作成します。このようにして、PROC JSON で WRITE VALUES ステートメントを使用して、SAS データセットをエクスポートせずに独自の JSON 出力ファイルを作成できます。SAS 変数値が、オブジェクトコンテナーに、キーと値のペアとして収集されます。たとえば、次のように指定します。

```
proc json out="example.json";
```

```
  write values "container" "object";
  write values "created" "implicitly";
run;
```

```
{"container":"object","created":"implicitly"}
```

WRITE VALUES ステートメントは、エクスポートされた SAS データセットのデータを含む JSON コンテナーに値を直接追加することはできません。EXPORT ステートメントは、JSON オブジェクトを暗黙的に開き、自動的に閉じます。WRITE VALUES ステートメントを使用して、閉じた JSON コンテナーに値を書き込むことはできません。エクスポートされたデータを含む JSON オブジェクトに値を追加するには、WRITE VALUES ステートメントの前に WRITE OPEN ステートメントを付ける必要があります。

WRITE OPEN ステートメントは、JSON コンテナーを明示的に開きます。また、これを使用することで、JSON コンテナーの種類(オブジェクトまたは配列)を指定することができます。

WRITE VALUES ステートメントが WRITE OPEN OBJECT ステートメントの後に続く場合、ペアの最初の値はキー名と見なされます。2 番目の値はキーの値です。たとえば、次のように指定します。

```
proc json out="example.json"
  write open object;
  write values "container" "object";
  write values "created" "explicitly";
  write close;
run;
```

```
{"container":"object","created":"explicitly"}
```

WRITE VALUES ステートメントは、WRITE OPEN ARRAY ステートメントの後に続く場合、JSON 配列コンテナーに値のカンマ区切りリストを作成します。

```
proc json out="example.out";
  write open array;
  write values "container" "array";
  write values "created" "explictly";
  write close;
run;
```

```
["container","array","created","explictly"]
```

エクスポートされた SAS データを含む JSON ファイルに情報を追加する場合や JSON コンテナーをネストする場合に、WRITE OPEN ステートメントを使用します。

オプションを使用した JSON 出力の制御

PROC JSON は、結果の JSON 出力を制御する複数のオプションをサポートします。PROC JSON、EXPORT、および WRITE VALUES ステートメントでオプションを指定できます。すべてのステートメントですべてのオプションがサポートされているわけではありません。

次のテーブルでは、オプションと、それがステートメントでサポートされるかどうかをリストで示します。

注: デフォルトのオプションキーワードは太字で表示されます。

表 33.2 PROC JSON、EXPORT、および WRITE VALUES ステートメントにおけるステートメントオプションの可用性

オプション	関数	PROC JSON ステートメント	EXPORT ステートメント	WRITE VALUES ステートメント
FMTCHARACTER | **NOFMTCHARACTER**	関連文字の SAS 出力形式を適用するかどうかを決定します。	はい	はい	いいえ
FMTDATETIME | NOFMTDATETIME	関連する日付、時刻、または日時の SAS 出力形式を適用するかどうかを決定します。	はい	はい	いいえ
FMTNUMERIC | **NOFMTNUMERIC**	関連する数値の SAS 出力形式(日付、時刻、および日時の SAS 出力形式を除く)を適用するかどうかを決定します。	はい	はい	いいえ
KEYS | NOKEYS	エクスポートされたオブザベーションを JSON オブジェクトとして書き込むのか、または JSON 配列として書き込むのかを決定します。	はい	はい	いいえ
PRETTY | **NOPRETTY**	JSON 出力のフォーマット方法を決定します。	はい	いいえ	いいえ
SASTAGS | NOSASTAGS	EXPORT ステートメントを使用するときに SAS メタデータを含めるのか、あるいは抑制するのかを決定します。EXPORT	はい	はい	いいえ

オプション	関数	PROC JSON ステートメント	EXPORT ステートメント	WRITE VALUES ステートメント
	は、プロシジャ要求の中で最初のステートメントである場合、最上位 JSON コンテナーがオブジェクトコンテナーであるか、または配列コンテナーであるかを判別します。			
SAS データセットオプション	SAS データセットに適用するアクションを指定します。	いいえ	はい	いいえ
SCAN | NOSCAN	PROC JSON が入力文字列をスキャンしてエンコードするかどうかを決定します。	はい	はい	はい
TABLENAME="*name*"	エクスポートした SAS データセットの SAS メタデータで名前を指定します。	いいえ	はい	いいえ
TRIMBLANKS | NOTRIMBLANKS	末尾のブランクを削除または保持するかどうかを決定します。	Yes	はい	はい

注: PROC JSON ステートメントで指定されるオプションは、プロシジャの持続時間に適用されます。EXPORT および WRITE VALUES ステートメントで指定されるオプションが適用されるのは、そのステートメントのみです。オプションが複数のステートメントで指定される場合、EXPORT ステートメントまたは WRITE VALUES ステートメントで指定されるオプションの方が優先されます。

PROC JSON ビデオ

PROC JSON の使い方のデモを示すビデオは、SAS から JSON 出力を書き込む方法を参照してください。このビデオでは、SAS データセットを JSON 出力ファイルにエ

クスポートする方法、データのサブセットをエクスポートして JSON 出力を制御する方法、自由な形式の JSON 出力を書き込む方法、JSON コンテナーを制御および入れ子にする方法、および、複数の SAS データセットのエクスポート方法を示します。

例: JSON プロシジャ

例 1: デフォルトオプションを使用した JSON ファイルのエクスポート

要素:

PROC JSON ステートメントオプション
PRETTY
EXPORT ステートメント

詳細

この例の PROC JSON は、Sashelp.Class データセットのサブセットを JSON 出力ファイルにエクスポートします。返されるオブザベーションの数を制限するために、WHERE データセットオプションが指定されています。PROC JSON PRETTY プロシジャは、デフォルトの JSON コンテナー構造を示すためにインデントを使用する形式で出力を返すよう指定されています。それ以外の場合は、デフォルトのエクスポートオプションが有効になります。

JSON 出力ファイルおよび EXPORT ステートメントを指定します。 PROC JSON ステートメントは、完全パス名と JSON ファイル名を使用して JSON 出力ファイルの物理的な場所を指定します。.json 拡張子が省略されると、出力はテキストファイルに書き込まれます。

```
proc json out="path-to-DefaultOutput.json" pretty;
  export sashelp.class (where=(age=11));
run;
```

出力: デフォルトオプションを使用した JSON ファイルのエクスポート

アウトプット 33.1　PROC JSON 出力ファイル DefaultOutput.json のコンテンツ

```
{ 1

  "SASJSONExport": "1.0 PRETTY", 2

  "SASTableData+CLASS": [ 3

    { 4

      "Name": "Joyce",
      "Sex": "F",
      "Age": 11,
      "Height": 51.3,
      "Weight": 50.5
    },
    {
      "Name": "Thomas",
      "Sex": "M",
      "Age": 11,
      "Height": 57.5,
      "Weight": 85
    }
  ]
}
```

1 出力ファイル内の最上位コンテナーは、JSON オブジェクトコンテナー({ })です。

2 SAS メタデータは、最上位コンテナーの上部に含まれています。文字列 "SASJSONExport"は、JSON ファイルの作成者を指定します。文字列"1.0 PRETTY"は、SAS JSON 形式のバージョン番号を指定し、その後に JSON 出力ファイルの作成に使用されるデフォルト以外のオプションが続きます。文字列 "SASTableData+CLASS"は、SAS データセットがエクスポートされたことを指定し、SAS データセットの名前を指定します。

3 エクスポートされたオブザベーションセットは、JSON 配列コンテナー([ ])内にあります。

4 デフォルトの動作(KEYS)に従って、データセットの各オブザベーションが JSON オブジェクトコンテナーにエクスポートされます。各オブジェクトコンテナー内の変数値は、キーと値のペアです。

例 2: 最上位 JSON 配列コンテナーへのエクスポート

要素:

PROC JSON ステートメントオプション
　PRETTY

EXPORT ステートメントオプション
　NOSASTAGS

詳細

この PROCJSON 例では、Sashelp.Class データセットのサブセットが JSON 出力ファイルにエクスポートされ、エクスポートされたデータを JSON 配列コンテナーに書き込むよう指定されます。

プログラム

```
proc json out="path-to-NosastagsOutput.json" pretty;

  export sashelp.class (where=(age=11)) / nosastags;
run;
```

プログラムの説明

JSON 出力ファイルを指定します。 PROC JSON ステートメントは、完全パス名と JSON ファイル名を使用して JSON 出力ファイルの物理的な場所を指定します。PRETTY オプションは、より読みやすい形式を作成します。

```
proc json out="path-to-NosastagsOutput.json" pretty;
```

エクスポートする SAS データセットを識別し、最上位の JSON コンテナーを制御します。 EXPORT ステートメント内の NOSASTAGS は、SAS メタデータを抑制し、最上位コンテナーが配列コンテナーであるとを指定します。NOSASTAGS オプションは、PROC JSON ステートメントまたは EXPORT ステートメントに指定することができます。

```
  export sashelp.class (where=(age=11)) / nosastags;
run;
```

出力: 最上位 JSON 配列コンテナーへのエクスポート

出力ファイル内の最上位コンテナーは、配列コンテナー([ ])です。SAS メタデータはありません。SAS データセット内の各オブザベーションは、JSON オブジェクト({ }) コンテナーです。オブジェクトコンテナー内の変数値は、キーと値のペアです。

アウトプット 33.2　PROC JSON 出力ファイル NosastagsOutput.json のコンテンツ

```
[
  {
    "Name": "Joyce",
    "Sex": "F",
    "Age": 11,
    "Height": 51.3,
    "Weight": 50.5
  },
  {
    "Name": "Thomas",
    "Sex": "M",
    "Age": 11,
    "Height": 57.5,
    "Weight": 85
  }
]
```

例 3: EXPORT ステートメントの使用時にオブザベーションデータ内の SAS 変数名を抑制する

要素:

PROC JSON ステートメントオプション
　PRETTY

EXPORT ステートメントオプション
　NOKEYS

詳細

この例では、Sashelp.Class データセットのサブセットが JSON 出力ファイルにエクスポートされ、エクスポートされたデータセットオブザベーションを配列として書き込むよう指定されます。配列コンテナーは、変数値をキーと値のペアとしてではなく、値リストに格納します。

JSON 出力ファイルを指定します。 PROC JSON ステートメントは、完全パス名と JSON ファイル名を使用して JSON 出力ファイルの物理的な場所を指定します。PRETTY オプションは、より読みやすい形式を作成します。

```
proc json out="path-to-NokeysOutput.json" pretty;
```

エクスポートする SAS データセットを識別し、オブザベーションコンテナーを制御します。 EXPORT ステートメント内の NOKEYS は、データセットオブザベーションを、オブジェクトコンテナーにではなく、配列コンテナー内の値として書き込みます。NOKEYS オプションは、PROC JSON ステートメントまたは EXPORT ステートメントに指定することができます。

```
export sashelp.class (where=(age=11)) / nokeys;
```

```
run;
```

出力: EXPORT ステートメントの使用時にオブザベーションデータ内の SAS 変数名を抑制する

アウトプット 33.3　PROC JSON 出力ファイル NokeysOutput.json のコンテンツ

```
{ 1
  "SASJSONExport": "1.0 NOKEYS PRETTY", 2
  "SASTableData+CLASS": [ 3
    [
      "Joyce",
      "F",
      11,
      51.3,
      50.5
    ],
    [
      "Thomas",
      "M",
      11,
      57.5,
      85
    ]
  ]
}
```

1 出力ファイル内の最上位コンテナーは、JSON オブジェクトコンテナー({ })です。

2 SAS メタデータは、最上位コンテナーに含まれています。メタデータ文字列 "SASTableData+CLASS"は、キーと値のペアのキーです。エクスポートされたデータを含む配列は、キーと値のペアの値です。

3 NOKEYS オプションは、デフォルトの KEYS 動作をオーバーライドします。したがって、SAS データセットから選択された各オブザベーションは、配列内のネストされた配列コンテナー([ ])にエクスポートされます。変数値は、変数名なしでリストされます。

例 4: SAS データセットをエクスポートせずに JSON 出力を書き込む

要素:

PROC JSON ステートメントオプション
　PRETTY
WRITE OPEN ステートメント
WRITE VALUES ステートメント
WRITE CLOSE ステートメント

詳細

この PROC JSON の例では、データを SAS データセットからエクスポートせずに JSON 出力を書き込む方法を説明します。これによって、JSON 出力のコンテンツ全体を完全に制御し、任意の JSON 出力を生成できます。

プログラム

```
proc json out="path-to-WriteOpenOutput.json" pretty;

  write open object;
  write values "Nested object sample";

  write open object;
  write values "Comment" "In a nested object";
  write close;

  write values "Nested array sample";
  write open array;
    write open array;
      write values "In a nested array";
      write values 1 true null;
    write close;
  write close;

  write values "Finished" "End of samples";

  write close;
run;
```

プログラムの説明

JSON 出力ファイルを指定します。 PRETTY オプションは、より読み取りやすい形式を作成することで、結果出力を制御します。

```
proc json out="path-to-WriteOpenOutput.json" pretty;
```

JSON オブジェクトコンテナーを開き、値を書き込みます。 WRITE OPEN OBJECT ステートメントは、オブジェクトコンテナー({ })を最上位コンテナーとして明示的に開きます。WRITE VALUES ステートメントは、文字列をオブジェクトコンテナーに書き込みます。

```
  write open object;
  write values "Nested object sample";
```

値を使用してオブジェクトコンテナーを入れ子構造にして、入れ子構造のオブジェクトコンテナーを閉じます。 WRITE OPEN OBJECT ステートメントは、最上位コンテナーに 2 番目のオブジェクトコンテナーを明示的に開きます。WRITE VALUES ス

テートメントは、第 2 レベルオブジェクトコンテナーに 2 つの文字列を書き込みます。WRITE CLOSE ステートメントは、最近開いたコンテナーを閉じます。

```
write open object;
write values "Comment" "In a nested object";
write close;
```

値を JSON 出力ファイルに書き込み、配列コンテナーを入れ子構造にし、値を使用して配列コンテナーを入れ子構造にし、2 つの配列コンテナーを閉じます。 WRITE VALUES ステートメントは、最上位コンテナーに文字列を書き込みます。WRITE OPEN ARRAY ステートメントは、最上位コンテナー内に配列コンテナーを作成してから、前に開いた配列コンテナーに第 2 レベルの配列コンテナーを入れ子構造にします。WRITE VALUES ステートメントは、文字列、数値、ブール値 TRUE、および NULL キーワードを第 2 レベルの配列コンテナーに書き込みます。2 つの WRITE CLOSE ステートメントにより、配列コンテナーが閉じられます。

```
write values "Nested array sample";
write open array;
  write open array;
    write values "In a nested array";
    write values 1 true null;
  write close;
write close;
```

最終値を書き込みます。 WRITE VALUES ステートメントは、最上位コンテナーに文字列を書き込みます。

```
write values "Finished" "End of samples";
```

最上位コンテナーを閉じます。 WRITE CLOSE ステートメントは、残りの開いているコンテナー(最上位コンテナー)を閉じます。

```
  write close;
run;
```

出力: SAS データセットをエクスポートせずに JSON 出力を書き込む

アウトプット 33.4　PROC JSON 出力ファイル WriteOpenOutput.json のコンテンツ

```
{
  "Nested object sample": { 1

    "Comment": "In a nested object"
  },
  "Nested array sample": [ 2

    [
      "In a nested array",
      1,
      true,
      null
    ]
  ], 3

  "Finished": "End of samples"
}
```

1 最上位コンテナーがオブジェクトコンテナーとして指定されたため、SAS は、WRITE VALUES ステートメントがキーと値のペアを指定すると予測します。最初の WRITE VALUES ステートメントは 1 つの値だけを指定します。したがって、文字列“Nested object sample”が、キーと値のペアのキーとして出力されます。最初の入れ子構造になったコンテナーは、このキーと値のペアの値です。入れ子構造になったコンテナーはオブジェクトコンテナーでもあるため、入れ子構造になった WRITE VALUES ステートメントはキーと値のペアとして出力されます。文字列内のキーは“Comment”であり、値は文字列“In a nested object”です。

2 前のオブジェクトコンテナーは明示的に閉じられているため、最上位コンテナーの規則が次の WRITE VALUES ステートメントに適用されます。SAS は、別のキーと値のペアを想定しています。文字列 Nested array sample”は、このキーと値のペアのキーであり、最初の入れ子構造になった配列コンテナーが値です。2 番目の入れ子構造になった配列コンテナーが実際の配列です。入れ子構造になった WRITE VALUES ステートメントは、第 2 レベルの配列コンテナーに 4 つの値のリストを定義します。

3 前の配列コンテナーが明示的に閉じられたため、最後の WRITE VALUES ステートメントは、最上位コンテナーにまた書き込まれ、キーと値のペアを含むことを期待されます。文字列“Finished”は、このキーと値のペアのキーであり、文字列“End of samples”が値です。

例 5: 同じプログラム内での値の書き込みとデータのエクスポート

要素:　PROC JSON ステートメントオプション

PRETTY
WRITE OPEN ARRAY ステートメント
WRITE VALUES ステートメント
EXPORT ステートメント
WRITE CLOSE ステートメント

詳細

この例では、デフォルトのエクスポート設定、NOSASTAGS オプション、およびの NOKEYS オプションの動作を示すために、値を書き込み、同じデータのエクスポートを 3 回実行します。エクスポートされたデータは、最上位配列コンテナーに挿入され、エクスポートされたデータに対する NOSASTAGS の影響を示します。

PROC JSON ステートメントに JSON 出力ファイルを指定します。 PRETTY オプションは、より読みやすい出力形式で出力を作成します。

```
proc json out="path-to-WriteAndExport.json" pretty;
```

最上位の JSON 配列コンテナーを開いて、値を指定します。 WRITE OPEN ARRAY ステートメントは、JSON 配列コンテナー([])を最上位コンテナーとして明示的に開きます。WRITE VALUES ステートメントは、文字列、数値、おうび追加文字列を、その配列の値として指定します。

```
write open array;
  write values "level" 1;
  write values "container" "array";
  write values "created" "explicitly";
```

EXPORT ステートメントを指定します。 EXPORT ステートメントは、SAS データセット Sashelp.Class からオブザベーションのサブセットをエクスポートすることを指定します。WHERE データセットオプションは、出力ファイルに含まれるオブザベーションを制限します。指定されているエクスポートオプションはありません。EXPORT ステートメントは、エクスポートされたデータを保持するコンテナーを暗黙的に開閉します。EXPORT ステートメントの前の WRITE VALUES ステートメントは、記述的なラベルを指定します。

```
  write values "Default Export";
  export sashelp.class (where=(age=11));
```

NOSASTAGS オプションを指定する EXPORT ステートメントを指定します。 EXPORT ステートメントは、先行する EXPORT ステートメントと同じデータをエクスポートするよう指定します。NOSASTAGS は、SAS メタデータを抑制します。EXPORT ステートメントの前の WRITE VALUES ステートメントは、記述的なラベルを指定します。

```
write values "Export with NOSASTAGS Options";
 export sashelp.class (where=(age=11)) / nosastags;
```

NOSASTAGS と NOKEYS オプションを指定する EXPORT ステートメントを指定します。 EXPORT ステートメントは、先行する EXPORT ステートメントと同じデータをエクスポートするよう指定します。NOSASTAGS は、SAS メタデータを抑制します。NOKEYS オプションは、オブジェクトコンテナーではなく、配列コンテナーに各データセットオブザベーションをエクスポートするよう指定します。EXPORT ス

テートメントの前の WRITE VALUES ステートメントは、記述的なラベルを指定します。

```
write values "Export with NOSASTAGS and NOKEYS Option";
export sashelp.class (where=(age=11)) / nosastags nokeys;
```

外側の JSON 配列コンテナーを閉じます。

```
write close;
run;
```

出力: 同じプログラム内での値の書き込みとデータのエクスポート

アウトプット 33.5　PROC JSON 出力ファイル WriteAndExport.json のコンテンツ


```
[ 1

  "level",
  1,
  "container",
  "array",
  "created",
  "explicitly",
  "Default Export",
  "SASJSONExport", 2

  "1.0 PRETTY",
  "SASTableData+CLASS",
  [ 3

    {
      "Name": "Joyce",
      "Sex": "F",
      "Age": 11,
      "Height": 51.3,
      "Weight": 50.5
    },
    {
      "Name": "Thomas",
      "Sex": "M",
      "Age": 11,
      "Height": 57.5,
      "Weight": 85
    }
  ],
  "Export with NOSASTAGS Option", 4

  {
    "Name": "Joyce",
    "Sex": "F",
    "Age": 11,
    "Height": 51.3,
    "Weight": 50.5
  },
  {
    "Name": "Thomas",
    "Sex": "M",
    "Age": 11,
    "Height": 57.5,
    "Weight": 85
  },
  "Export with NOSASTAGS and NOKEYS Option", 5

  [
    "Joyce",
    "F",
    11,
    51.3,
    50.5
  ],
  [
    "Thomas",
    "M",
    11,
    57.5,
    85
  ]
]
```

1. 最初の WRITE VALUES ステートメントによって定義される値は、明示的に開かれた最上位の配列コンテナー内の変数値です。
2. 最初の EXPORT ステートメントからの SAS メタデータも、最上位の配列コンテナーに配列値として追加されます。
3. デフォルトのエクスポート動作は、エクスポートされたデータを、開かれた配列コンテナーに配列値として書き込みます。エクスポートされたオブザベーションは、新規配列内のカンマ区切りの JSON オブジェクトコンテナーです。
4. 2 番目の EXPORT ステートメントからエクスポートされたデータの前に、文字列"Export with NOSASTAGS Option"がラベルとして出力されます。2 番目の EXPORT ステートメントの NOSASTAGS は、エクスポートされたオブザベーションから SAS メタデータと外部配列コンテナーを抑制します。エクスポートされたオブザベーションは、既存配列内のカンマ区切りのオブジェクトコンテナーです。
5. 文字列"Export with NOSASTAGS and NOKEYS Option"は、3 番目の EXPORT ステートメントからエクスポートされたデータの前にラベルとして出力されます。NOKEYS オプションと NOSASTAGS オプションが共に指定されている場合、エクスポートされたオブザベーションは、既存の配列にカンマ区切り配列値として追加されます。

例 6: エクスポートされたデータへの値の書き込みとコンテナーの制御

要素:

PROC JSON ステートメントオプション
- NOSASTAGS
- PRETTY

WRITE OPEN ステートメント

WRITE VALUES ステートメント

EXPORT ステートメント

WRITE CLOSE ステートメント

詳細

この PROC JSON の例では、追加値を JSON 出力ファイルに書き込み、JSON コンテナーを制御して入れ子構造にする方法について説明します。この例では、Sashelp.Cars データセットのサブセットが JSON 出力ファイルにエクスポートされます。

プログラム

```
%let vehicleType=Truck;
%let minCost=26000;

proc json out="path-to-WriteOpenArrayOutput.json" nosastags pretty;

  write open array;
  write values "Vehicles";
  write open array;
  write values "&vehicleType";
  write open array;
  write values "Greater than $&minCost";
  /*********** Asian ***********************/
  %let originator=Asia;
  write open object;
  write values "&originator";
  write open array;
  export sashelp.cars(where=((origin = "&originator")  and
                      (type   = "&vehicleType") and
                      (MSRP   > &minCost)         )
                  keep=make model type origin MSRP);
  write close; /* data values */
  write close; /* Asia */
  /*********** European ***********************/
  %let originator=Europe;
  write open object;
  write values "&originator";
  write open array;
  export sashelp.cars(where=((origin = "&originator")  and
                      (type   = "&vehicleType") and
                      (MSRP   > &minCost)         )
                  keep=make model type origin MSRP);
  write close; /* data values */
  write close; /* Europe */
  /*********** American ***********************/
  %let originator=USA;
  write open object;
  write values "&originator";
  write open array;
  export sashelp.cars(where=((origin = "&originator")  and
                      (type   = "&vehicleType") and
                      (MSRP   > &minCost)         )
                  keep=make model type origin MSRP);
  write close; /* data values */
  write close; /* USA */
  write close; /* expensive */
  write close; /* vehicleType */
  write close; /* cars */
run;
```

プログラムの説明

マクロ変数を割り当てます。 %LET ステートメントは、マクロ変数を作成し、値がコード全体で使用されるように割り当てます。

```
%let vehicleType=Truck;
%let minCost=26000;
```

JSON 出力ファイルを指定し、結果出力を制御します。 PROC JSON ステートメントは、完全パス名と JSON ファイル名を使用して JSON 出力ファイルの物理的な場所を指定します。NOSASTAGS オプションは SAS メタデータの作成を抑制し、PRETTY オプションはより読み取りやすい形式を作成します。

```
proc json out="path-to-WriteOpenArrayOutput.json" nosastags pretty;
```

追加の PROC JSON ステートメントを含めて、要求をサブミットします。 これらのステートメントは、車の種類ごとに入れ子構造になった一連のコンテナーを開き、値をコンテナーのラベルとして書き込みます。EXPORT ステートメントは、エクスポートする SAS データセットを指定し、特定のオブザベーションを選択して、所定の SAS 変数のみを要求します。

```
write open array;
write values "Vehicles";
write open array;
write values "&vehicleType";
write open array;
write values "Greater than $&minCost";
/*********** Asian ***********************/
%let originator=Asia;
write open object;
write values "&originator";
write open array;
export sashelp.cars(where=((origin = "&originator")  and
                    (type   = "&vehicleType") and
                    (MSRP   > &minCost)         )
                keep=make model type origin MSRP);
write close; /* data values */
write close; /* Asia */
/*********** European ***********************/
%let originator=Europe;
write open object;
write values "&originator";
write open array;
export sashelp.cars(where=((origin = "&originator")  and
                    (type   = "&vehicleType") and
                    (MSRP   > &minCost)         )
                keep=make model type origin MSRP);
write close; /* data values */
write close; /* Europe */
/*********** American ***********************/
%let originator=USA;
write open object;
write values "&originator";
write open array;
export sashelp.cars(where=((origin = "&originator")  and
                    (type   = "&vehicleType") and
                    (MSRP   > &minCost)         )
```

```
                keep=make model type origin MSRP);
  write close; /* data values */
  write close; /* USA */
  write close; /* expensive */
  write close; /* vehicleType */
  write close; /* cars */
run;
```

出力: JSON コンテナーの制御と値の書き込み

アウトプット 33.6　PROC JSON 出力ファイル WriteOpenArrayOutput.json のコンテンツ


```
[
 "Vehicles",
 [
  "Truck",
  [
   "Greater than $26000",
   {
    "Asia": [
     {
      "Make": "Nissan",
      "Model": " Titan King Cab XE",
      "Type": "Truck",
      "Origin": "Asia",
      "MSRP": 26650
     }
    ]
   },
   {
    "Europe": [
    ]
   },
   {
    "USA": [
     {
      "Make": "Cadillac",
      "Model": " Escalade EXT",
      "Type": "Truck",
      "Origin": "USA",
      "MSRP": 52975
     },
     {
      "Make": "Chevrolet",
      "Model": " Avalanche 1500",
      "Type": "Truck",
      "Origin": "USA",
      "MSRP": 36100
     },
     {
      "Make": "Chevrolet",
      "Model": " Silverado SS",
      "Type": "Truck",
      "Origin": "USA",
      "MSRP": 40340
     },
     {
      "Make": "Chevrolet",
      "Model": " SSR",
      "Type": "Truck",
      "Origin": "USA",
      "MSRP": 41995
     },
```

```
      {
       "Make": "Ford",
       "Model": " F-150 Supercab Lariat",
       "Type": "Truck",
       "Origin": "USA",
       "MSRP": 33540
      },
      {
       "Make": "GMC",
       "Model": " Sierra HD 2500",
       "Type": "Truck",
       "Origin": "USA",
       "MSRP": 29322
      }
     ]
    }
   ]
  ]
]
```


例 7: エクスポートされた出力への SAS 出力形式の適用

要素:

PROC JSON ステートメントオプション
- PRETTY

EXPORT ステートメントオプション
- FMTDATETIME
- FMTNUMERIC

DATA ステップ

詳細

この PROC JSON の例では、SAS 出力形式に関連付けられた変数を含む Work.Formats という名前の SAS データセットがエクスポートされます。結果の JSON 出力ファイルは SAS 出力形式に適用され、出力値がより読み取りやすくなります。

プログラム

```
data formats;
  input name $ idnumber $ salary hiredate
mmddyy10.;
  format salary dollar7. hiredate date9.;
  datalines;
```

```
Brad 0755 21163 9/24/2012
Lindzey 0767 34321 9/04/2012
;

proc json out="path-to-FormatsOutput.json" pretty;

  export work.formats / fmtnumeric;
run;
```

プログラムの説明

SAS 出力形式に関連付けられた変数を含む SAS データセットを作成します。 DATA ステップによって、4 つの変数を含む Work.Formats という名前の SAS データセットが作成されます。FORMAT ステートメントによって、DOLLAR7 が関連付けられます。変数 Salary を含む SAS 数値形式および変数 HireDate を含む SAS 日付形式 DATE9。

```
data formats;
  input name $ idnumber $ salary hiredate
mmddyy10.;
  format salary dollar7. hiredate date9.;
  datalines;
Brad 0755 21163 9/24/2012
Lindzey 0767 34321 9/04/2012
;
```

JSON 出力ファイルを指定します。 PROC JSON ステートメントは、完全なパス名とファイル名を持つ JSON 出力ファイルの物理的場所を指定し、より読み取りやすい形式を作成するための PRETTY オプションを含めます。

```
proc json out="path-to-FormatsOutput.json" pretty;
```

エクスポートする SAS データセットを識別し、FMTNUMERIC オプションを指定します。 EXPORT ステートメントは、SAS データセット名を指定します。FMTDATETIME オプションは、変数 HireDate に関連付けられた日付の SAS 出力形式を適用するためにデフォルトで使用できます。FMTNUMERIC オプションは、変数 Salary に関連付けられた数値の SAS 出力形式 DOLLAR7.を適用するために指定されます。

```
  export work.formats / fmtnumeric;
run;
```

出力: エクスポートされた出力への SAS 出力形式の適用

アウトプット 33.7　FMTDATETIME および FMTNUMERIC を含む PROC JSON 出力ファイル FormatsOutput.json


```
{
  "SASJSONExport": "1.0 PRETTY FMTNUMERIC",
  "SASTableData+FORMATS": [
    {
      "name": "Brad",
      "idnumber": "0755",
      "salary": "$21,163",
      "hiredate": "24SEP2012"
    },
    {
      "name": "Lindzey",
      "idnumber": "0767",
      "salary": "$34,321",
      "hiredate": "04SEP2012"
    }
  ]
}
```


EXPORT ステートメントが NOFMTDATETIME オプションを指定し、FMTNUMERIC オプションは指定しなかった場合、結果の JSON 出力ファイルには、読み取りやすさが低い Salary および HireDate の値が含まれます。

アウトプット 33.8　NOFMTDATETIME および NOFMTNUMERIC を含む PROC JSON 出力ファイル FormatsOutput.json


```
{
  "SASJSONExport": "1.0 PRETTY NOFMTDATETIME",
  "SASTableData+FORMATS": [
    {
      "name": "Brad",
      "idnumber": "0755",
      "salary": 21163,
      "hiredate": 19260
    },
    {
      "name": "Lindzey",
      "idnumber": "0767",
      "salary": 34321,
      "hiredate": 19240
    }
  ]
}
```

34

LUA プロシジャ

概要: LUA プロシジャ

LUA プロシジャの機能

Lua プログラミング言語は、埋め込み可能なスクリプト言語で、標準の C コンパイラを持つすべてのプラットフォームで実行します。Lua ではシンプルな構文が使用され、計算処理が高速です。また、メモリ割り当てが自動的に管理されます。

LUA プロシジャを使用すると、SAS コード内の Lua プログラミング言語からステートメントを実行できます。Lua ステートメントを外部 Lua スクリプトからサブミットしたり、SAS コードに直接入力したりできます。

PROC LUA を使用すると、次のタスクを実行できます。

- SAS セッションで Lua コードを実行する
- Lua ステートメントでほとんどの SAS 関数を呼び出す
- Lua ステートメントで PROC FCMP 関数を呼び出す
- Lua から SAS コードをサブミットする
- CAS アクションを呼び出す
- VARCHAR データを読み取る

注: マクロ定義内では、PROC LUA 内の SUBMIT/ENDSUBMIT ステートメントは許可されません。かわりに、%INCLUDE ステートメントまたは INFILE=引数を使用して、マクロ定義から LUA コードを実行できます。

概念: LUA プロシジャ

Lua スクリプトのセキュリティの確認

Lua ランタイムにはセキュリティの脆弱性があるため、信頼された Lua スクリプトのみが実行できます。信頼はデジタル証明書を作成することによって確立されます。SECURELUA システムオプションが TRUE に設定されている場合、PROC LUA は、信頼されていなければ、Lua スクリプトを実行したり、Lua モジュールをロードしたりしません。SECURELUA システムオプションのデフォルト値は FALSE です。

Lua スクリプトは、次の方法で実行できます。

- コンパイルされていない単純な Lua スクリプトは、Lua ステートメントの INFILE オプション、または Lua スクリプトの load または loadfile 関数を使用して呼び出すことができます。
- C コードで実装されていない Lua モジュールは、Lua スクリプトの required 関数によって呼び出すことができます。

このような状況では、Lua スクリプトまたはモジュールが存在するディレクトリを、LOCKDOWN 許可リストと SECURELUALOC システムオプションで指定する必要があります。

注: Lua スクリプトがバイナリとしてコンパイルされている場合、または Lua モジュールが C で記述されている場合、SECURELUA システムオプションがオンの場合は Lua コードを実行できません。

Lua スクリプトが PROC LUA Submit ステートメントから実行される場合は、PROC LUA ステートメントの SECURELUATOKEN 引数にデジタル署名を指定する必要があります。

管理者は、ユーザーが作成した Lua スクリプトまたはモジュールの安全性を確認する責任があります。

- SECURELUALOC で指定されるディレクトリパスは、カスタマーサイトの信頼できるセキュリティ管理者のみが書き込み可能である必要があります。
- ユーザーが作成したスクリプトとモジュールは、SECURELUALOC パス内に保存する前にセキュリティレビューを受ける必要があります。
- 管理者は、ディレクトリパスを LOCKDOWN 許可リストと SECURELUALOC オプションに追加するためのサイト固有の構成オプションを提供する責任があります。

Lua コンテンツの検証に使用される証明書の場所は、LUACERTLOC 環境変数で指定する必要があります。デジタル証明書の作成については、“ Generate Digital Certificates Using OpenSSL ” (*Encryption in SAS*)を参照してください。

外部 Lua ファイルの実行

SAS セッションで%INCLUDE ステートメントを使用すると、外部 Lua スクリプト(*.lua ファイルまたは*.luc ファイル)を実行できます。たとえば、Lua スクリプト abc.luc を実行するには、SAS プログラムで次の行を入力します。

```
%include "./tmp/abc.luc";
```

PROC LUA での CAS アクションの呼び出し

CAS サーバーに接続するための要件

PROC LUA から CAS アクションを呼び出すことができます。CAS サーバーに接続するには、次の情報が必要です。

- CAS サーバーを実行しているサーバー(ホスト)の名前
- CAS サーバーポート
- ユーザー ID とパスワード(CAS サーバーにアクセスするためのアクセス許可が必要)。または、authinfo ファイルを作成しておく必要があります。

次のリソースも必要です。これらのファイルは、通常、インストールの一部として自動的にインストールされます。

- middleclass.lua
- swat.lua
- tkluaswat.so

詳細については、“要件” (*SAS Viya プラットフォームプログラミング: CAS 用 Lua 入門ガイド*)を参照してください。

CAS サーバーに接続して Lua コードを実行するには、SAS プログラムに適切な設定とユーティリティをロードする必要があります。SAS Scripting Wrapper for Analytics Transfer (SWAT)ライブラリもロードする必要があります。詳細については、“例 13: CAS サーバーへの接続” (1212 ページ)を参照してください。

CAS サーバーと対話する関数

CAS サーバーと対話する関数は次のとおりです。CAS サーバーに接続する方法を示す例については、“例 13: CAS サーバーへの接続” (1212 ページ)を参照してください。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

Lua-var:HELP{}
Lua 変数に関連付けられている CAS セッションの使用可能な CAS アクションのリストを要求します。たとえば、CAS セッションを Lua 変数に割り当てた場合は、s:help{}を指定します。

CAS.OPEN("*host-name*", *port*<, "*user-ID*"><, "*password*">)
CAS サーバー上でセッションを開きます。CAS サーバーのホスト名とポートを指定し、オプションとしてユーザー ID とパスワードの値を指定します。ユーザー ID には、CAS サーバーにアクセスするためのアクセス許可が必要です。ユーザー ID とパスワードを含めない場合は、authinfo ファイルを定義している必要があります。詳細については、*“要件” (SAS Viya プラットフォームプログラミング: CAS 用 Lua 入門ガイド)*を参照してください。

以下の代替バージョンに示しているように、CAS セッションを表す Lua 変数に CAS.OPEN の結果を割り当てます。

```
s = cas.open("host.mycompany.com", 5570, "myuserid", "password")
```

```
s = cas.open("host.mycompany.com", 5570)
```

接続を確立したら、s の属性(s.port や s.hostname など)を参照できます。

Lua-var:SHUTDOWN{}
Lua 変数に関連付けられている CAS セッションを終了し、CAS サーバーへの接続を閉じます。たとえば、Lua 変数 s で表される CAS セッションを閉じるには、次のコマンドを使用します。

```
s:shutdown{}
```

PROC LUA での VARCHAR データのサポート

VARCHAR データは CAS サーバーでのみサポートされています。これは、CAS サーバー上で完全に実行される DATA ステップと、VARCHAR データをサポートするすべてのアクションで、VARCHAR 値を使用できることを意味します。他のタスクの場合、VARCHAR データは最初に固定長文字データに変換されます。

詳細については、*“DATA ステップの実行場所の決定” (SAS Cloud Analytic Services: DATA ステッププログラミング)*および ARCHAR データが確実にサポートされるようにするためのアクション、プロシジャ、または関数のドキュメントを参照してください。

Lua ステートメントでの標準 SAS 関数の呼び出し

Lua ステートメントで標準 SAS 関数を実行するには、関数の前に接頭辞"SAS."を付けます。これにより、標準 SAS 関数のほとんどを実行できます。たとえば、 SAS 関数の MAX を呼び出すには、次に示すのように関数の前に"SAS."を付けます。

```
local max = sas.max(a,b,c)
```

注: ADDRLONG など、DATA ステップでのみ実行される関数は、Lua コードで実行しません。

SAS 関数で必要な引数に対して値を指定しないと、その引数は、SAS 関数によって欠損値に変換されます。

Lua ステートメントでの SAS コードのサブミット

SAS コードをサブミットする関数

Lua は SAS 内のスクリプト言語であるため、SAS コードをサブミットし、Lua ステートメント内で SAS 値をかわりに使うことができます。SAS をサブミットするには、次の関数を使用します。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

SAS.SUBMIT(*[[SAS code]]*, {*substitution(s)*})

Lua ステートメント内で関数が発行されるときに、指定された SAS コードがサブミットされ、実行されます。SAS コードは、角かっこ([[および]])で囲みます。

代入値が指定されている場合、その値は最初に割り当てられます。残りの代入値はすべて、ローカル変数によって指定されます。

代入値がサブミットされていない場合は、呼び出し側の関数環境からローカル変数が使用されます。代入用の変数は@記号で囲みます。たとえば、userID という変数の値を、可変ユーザーに割り当てるには、次の割り当てを発行します。

```
user = @userID@
```

SAS.SUBMIT への呼び出しによって代入を使用する例を次に示します。

```
sas.submit([[
  data @name@;
  x = @x@;
run;
```

]],{name="foo"})

SAS コードをサブミットする際には、SAS コメントに次のキーワードを含めないでください。

run; %macro

quit; %mend

SAS コードブロックのコメント内にこれらのキーワードが存在すると、SAS ログに警告またはエラーがトリガーされ、予期しない結果が生じることがあります。このような状況を回避するには、キーワード内に空白スペースまたはエスケープ文字を挿入して変更します。たとえば、コメント内の run ;または\%\macro によって予期しない動作がトリガーされることはありません。

SAS.SUBMIT からの戻り値は、SYSERR 自動マクロ変数の値に割り当てられます。

SAS.SUBMIT_(*SAS code*, *substitution(s)*)
SAS コードと代入値をサブミットしますが、コードは実行しません。サブミットされたすべての SAS コマンドを実行し、SAS.SUBMIT_への前の呼び出しを終了するには、SAS.SUBMIT 関数(アンダースコアなし)を発行する必要があります。

代入値がサブミットされていない場合は、呼び出し側の関数環境からローカル変数が使用されます。代入用の変数は@記号で囲みます。

詳細については、“Lua コード内で SAS データセットを開く” (1185 ページ)を参照してください。

SAS ステートメントでの Lua 変数代入について

変数代入を実行するには、キーと値のペアが含まれるハッシュテーブルを使用します。キーと値のペアは、サブミットされた Lua ステートメント内で行われた Lua 変数割り当てによって定義されます。値を代入するには、キーを@記号で囲みます(例:@key@)。@key@の値は、それに対応する値で置き換えられます。代入では大文字と小文字が区別されます。つまり、@key@と@Key@のキー値は異なります。

PROC LUA の範囲

PROC LUA の呼び出し内で定義されたファイル参照名、ライブラリ参照名、およびローカル変数は、そのプロシジャ呼び出しの間のみ定義されます。同様に、PROC LUA 呼び出し内で作成されたマクロ変数とマクロ定義は、そのプロシジャ呼び出し内でのみ使用できます。LUA プロシジャの外部でこれらのオブジェクトを参照することはできません。

PROC LUA での Math.Huge の解釈

Lua では定数 math.huge は無限大のコンピューター表現です。SAS は、この値を定数値 1.7976931348623E308 で表します。

Lua によるブール値の解釈

Lua には、ブールコンテキストで使用されるデータ型が 2 つあります。ニル型は、1 つの値 **nil** をとることができます。ブール型は、値として **true** または **false** をとることができます。Lua では、**nil** 値または **false** 値が false と見なされます。0 を含め、その他の値はすべて、true と見なされます。

SAS には、条件が true の場合に値 1 を返し、条件が false の場合に値 0 を返す数値関数が多数あります。たとえば、SYMEXIST、SYMLOCAL、SYMGLOBAL、MISSING が、こうした関数に該当します。ANY*文字関数など、その他の SAS 関数は、指定された文字を文字列内で検索し、最初に見つかった文字の位置を返します。その文字が見つからない場合、これらの関数は 0 を返します。

Lua コードで任意の SAS 関数を呼び出す場合は、意図したとおりに Lua コードが結果を解釈することを確認します。

たとえば、マクロ変数 FOOBAR が定義されていないとします。その場合、ローカルシンボルテーブルには存在しません。次のコードは、変数が存在すること、およびローカルであることを誤って記述しています。

```
/* INCORRECT use of SAS Boolean functions */
proc lua;
  submit;
    if sas.symexist("foobar") then
      if sas.symlocal("foobar") then
        print("In Proc LUA, foobar exists and is LOCAL.")
      else
        print("In Proc LUA, foobar exists but is not LOCAL.")
      end
    else
      print(In Proc LUA, foobar does not exist.")
    end
  endsubmit;
run;
```

前のコードは、FOOBAR が存在すること、およびローカルであることを誤って記述しています。SAS 関数 SYMEXIST および SYMLOCAL によって返された 0 値が、Lua で true として解釈されているからです。

かわりに、SAS ブール関数を呼び出すときに、必要な戻り値を Lua コードで明示的にテストします。次のコードは、マクロ変数 FOOBAR が存在するかどうか、また、ローカルかどうかを確認する適切なテストを示しています。

```
/* CORRECT use of SAS Boolean functions */
proc lua;
  submit;
    if sas.symexist("foobar") == 1 then
      if sas.symlocal("foobar") == 1 then
        print("In Proc LUA, foobar exists and is LOCAL.")
      else
        print("In Proc LUA, foobar exists but is not LOCAL.")
      end
    else
      print("In Proc LUA, foobar does not exist.")
```

```
    end
  endsubmit;
run;
```

上記のコードは、マクロ変数 FOOBAR が、定義されていないため、存在せず、またローカルでもないことを正しく記述して示います。SAS 関数 SYMEXIST からの戻り値 0 は、明示的に値 1 と比較されています。

LUA 関数の呼び出し

PROC LUA を実行すると、Lua ステートメントでほとんどの Lua 関数を呼び出すことができます。詳細については、LUA のドキュメントを参照してください。SAS ログをチェックして、呼び出したすべての関数がエラーなしで実行されることを確認します。

LUA プロシジャのデータセット関数

次の関数は、LUA プロシジャで実行されるように定義されています。これらの関数は、SAS またはデータセットと対話します。

小さなデータセットについては、SAS データセット全体を Lua テーブルに読み込むことができます。その後、Lua テーブルを変更するか、テーブルに対してクエリを実行できます。また、新しい SAS データセットに変更を書き込むことができます。ただし、大きなデータセットの場合は、DATA ステップでの処理と同じように、一度に 1 つのオブザベーションでデータを処理する方が効率的です。この理由から、大きなデータセットについては、Lua コードで、DATA ステップをサブミットすることをお勧めします。

LUA プロシジャ内で実行されるデータセット関数を次に示します。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

SAS.ADD_VARS(*data-set-ID*, *variable-definitions*)
指定された Lua 変数を新しいデータセットに追加します。つまり、SAS.ADD_VARS は、モード'o'の SAS.OPEN を使用してデータセットを開くときにのみ使用します。

変数を定義するための出力形式は次のとおりです。

{ {name="*varname*", type="N|C",

format="*format.*", length="*value*",

label="*varlabel*", informat="*informat.*"},

{*additional variable definition*}, ...

}

変数名のみが必須です。デフォルトの変数の種類は数値(N)です(同じ名前の Lua 変数も数値の場合)。デフォルトの文字変数の長さは 200 文字です。

完全な SAS 出力形式(format="best12.3"など)を指定できます。ピリオド(.)が含まれる出力形式を指定すると、完全な出力形式と見なされ、追加の出力形式の属性は無視されます。完全な SAS 出力形式を指定しない場合は、これらの変数属性の組み合わせを使用して、変数の出力形式を指定できます。

- FORMAT では、長さも小数点も指定しません(例:format="best")
- FORMAT_WIDTH では、文字数または桁数を指定します(例:format_width=12)
- FORMAT_DEC では、小数点以下の桁数を指定します(例:format_dec=3)

詳細については、“例 12: 変数の定義と、データセットへの変数の追加” (1210 ページ)を参照してください。

SAS.ATTR(*data-set-ID*, *attribute-name*)
データセットの *attribute-name* の値を返します。たとえば、sas.attr(*data-set-ID*, 'label')は、指定されたデータセットのラベルを返します。

SAS.CLOSE(*data-set-ID*)
開いているデータセットを閉じます。この関数は、データセット ID が無効の場合は終了します。

SAS.EXISTS(*SAS-data-set-name*)
データセットが存在する場合はブール値 **true** を、存在しない場合は **false** を返します。

注: この関数は、0 または 1 の値を返す標準 SAS 関数 EXIST(末尾に's'がありません)とは異なります。Lua コードでは、0 および 1 は true として解釈されます。別の方法として、条件(sas.exist("work.test") > 0)が true かどうかをテストします。SAS.EXISTS 関数は、Lua コードブロック内でのみ使用できます。

SAS.NOBS(*data-set-ID*)
データセット内のオブザベーションの数を返します。

SAS.NVARS(*data-set-ID*)
データセット内の変数の数を返します。

SAS.OPEN(*SAS-data-set-name<, mode>*)
SAS データセットを開き、データセットが適切に開いた場合は、データセット ID を返します。データセットが開かない場合、関数は **nil** を返します。したがって、SAS.EXISTS 関数は、SAS.OPEN 関数を呼び出す前に使用します。

SAS.OPEN 関数では、KEEP=、DROP=、または WHERE=などのデータセットオプションを使用できます。たとえば、次のコードでは、データセット Sashelp.Class を開き、変数 Age のみを保持できます。

```
local dsid = sas.open('sashelp.class(keep=age)')
```

有効なデータセットモードは、*I*(読み込み)、*O*(作成)および *U*(更新)です。データセットモードを指定しないと、デフォルト値 I が使用されます。

SAS.READ_DS(*SAS-data-set-name*)
指定された SAS データセットのデータを含む Lua テーブルを返します。データセットが存在しない場合、関数は **nil** を返します。したがって、SAS.EXISTS 関数は、SAS.READ_DS 関数を呼び出す前に使用します。

最良の方法として、小さなデータセットの場合にのみ、SAS.READ_DS 関数を使用してください。大きなデータセットの場合は、個別のオブザベーションを処理する関数を使用してオブザベーションを反復します。詳細については、"オブザベーションを処理する関数" (1173 ページ)を参照してください。

Alias: SAS.LOAD_DS(*SAS-data-set-name*)

SAS.SET_ATTR(*data-set-ID*, *attribute-name*, *value*)
値をデータセットの属性に割り当てます。

SAS.WHERE(*data-set-ID*, *where-clause*)
WHERE 句をデータセットに適用します。データセットにすでに WHERE 句が存在する場合、指定された WHERE 句は、前の WHERE 句に追加されます。ただし、オプションの *replace-where-clause* 引数のブール値が true の場合は、指定された WHERE 句によって既存の WHERE 句が置き換えられます。

この関数のリターンコードは整数値です。Lua はすべての整数値を true として解釈するため、戻り値を明示的にテストして、0 に等しいかどうかを確認します (SAS では false)。

sas.where(dsid, "also <new condition>")、sas.where(dsid, "undo")、sas.where(dsid, "clear")などの操作がサポートされています。詳細については、*SAS コンポーネント言語: リファレンス*を参照してください。

次のような Lua ステートメントをサブミットして、WHERE 句をデータセットに適用できます。

```
sas.submit("data work.air; set sashelp.air; run;")
dsid = sas.open("work.air")
rc, msg = sas.where(dsid,"air=222 or air=999")
print(rc, msg)
rc=sas.close(dsid)
```

SAS.WRITE_DS(*Lua-table*, *SAS-data-set-name*)
Lua テーブルから SAS データセットを作成します。SAS データセットには、Work.Random などの 2 レベルの名前を指定できます。Lua テーブルは、SAS.READ_DS 関数で返される構造に従っている必要があります。ソーステーブルには少なくとも 1 つのオブザベーションが必要です。

データセットオブザベーションの処理

オブザベーションを処理する関数

次の Lua 関数を使用して、データセット内の個別のオブザベーションを処理できます。まず、更新モード('u')でデータセットを開く必要があります。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

SAS.APPEND(*data-set-ID*)
新しく作成されたオブザベーションをデータセットに追加します。

SAS.DELOBS(*data-set-ID*)
データセット内の現在のオブザベーションを削除します。

SAS.GET_VALUE(*data-set-ID*, *variable-number* | *variable-name*)
現在のオブザベーションで指定された変数の値を返します。データセット内の位置(数値)または名前で変数を特定します。

SAS.NEXT(*data-set-ID*)
処理のためにデータセット内の次のオブザベーションに移動します。データセットの処理を開始していない場合、SAS.NEXT 関数は、データセット内の最初のオブザベーションに移動します。SAS.NEXT 関数を使用すると、SAS データセットで直接操作できます。最初にデータを Lua テーブルに読み込む必要がありません。この関数は、大きなデータセットで有用です。

SAS.PUT_VALUE(*data-set-ID*, *variable-name*, *value*)
データセットの指定された変数に値をロードします。

SAS.ROWS(*data-set-ID*)
データセット内のオブザベーションを反復し、各行を処理のために Lua テーブルにロードして、処理の終了時に Lua nil を追加します。この関数は、変数が比較的少ないデータセットで有用です。

SAS.UPDATE(*data-set-ID*)
SAS.PUT_VALUE 関数を呼び出して、追加された値でオブザベーションを更新します。

SAS.VARS(*data-set-ID*)
データセットの変数を反復します。

オブザベーションを追加する順序

新しいオブザベーションを追加するには、オブザベーション処理関数を次の順序で呼び出します。

1 SAS.APPEND。

2 SAS.PUT_VALUE。オブザベーションの必要な変数すべてに値が設定されるまで、次の関数呼び出しを繰り返します。

3 SAS.UPDATE。

範囲外の警告

データセット ID(ハンドル)が範囲外になると、関連する SAS データセットは自動的に閉じます(まだ閉じていない場合)。SAS ログには、次のような警告が表示されます。

```
WARNING: Closing SASHELP.CLASS - handle has gone out of scope.
```

プログラムがアクセスできなくなると、そのデータセット ID は範囲外になります。たとえば、データセット ID が、ユーザー定義関数でローカル変数として定義されている場合、データセット ID は、その関数定義の最後で範囲外になります。最良の方法として、データセット ID が範囲外になる前にデータセットを閉じてください。

LUA プロシジャのシステム関数

次の関数は、サブミットされた SAS コードの動作を LUA プロシジャ内から制御します。詳細については、“Lua ステートメントでの SAS コードのサブミット” (1168 ページ)を参照してください。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

SAS.GET_MAX_SYSERR()
: SAS ログでエラーをトリガーせずに、SAS.SUBMIT への呼び出しから返すことができる現在の SYSERR の最大値を返します。

SAS.IS_QUIET()
: SAS.SET_QUIET 関数を使用して設定された値に応じて、**true** または **false** を返します。このブール値は、SAS.SUBMIT 関数にサブミットされた SAS 言語ステートメントが、SAS ログに書き込まれたかどうかを示します。

SAS.SET_MAX_SYSERR(*value*)
: SAS.SUBMIT 呼び出しによって返すことができる SYSERR の最大許容値を指定します。SAS が、指定値よりも大きい SYSERR 値を返した場合は、エラーが SAS ログに印刷されます。デフォルト値はゼロで、可能な値には正の整数が含まれます。

SAS.SET_QUIET(*value*)
: SAS.SUBMIT 関数にサブミットされた SAS 言語ステートメントが、SAS ログに書き込まれるかどうかを指定します。

  可能な引数値は **true** または **false** です。デフォルト値は **false** です。

Lua ライブラリについて

Lua ライブラリと Lua の SAS 拡張機能

Lua プログラミング言語には、テーブルや文字列を操作したり、一般的な数学関数を実行したりするために使用できるいくつかのライブラリが含まれています。これらの関数のほとんどは、LUA プロシジャで使用できます。さらに、Lua 言語の拡張として PROC LUA 用に作成された関数があります。これらの関数を使用すると、テーブルや文字列を操作したり、データに対して計算を実行したりできます。これら

の拡張機能は PROCLUA 内で使用できますが、Lua プログラミング言語の一部ではなく、PROC LUA の外部で使用することはできません。

Lua IO ライブラリの制限

SAS がロックダウンモード(ロックダウン状態)で実行されている場合、LuaIO ライブラリの関数へのアクセスは制限されます。SAS 管理者はこの制限を無効にすることができます。詳細については、*"Security Features" (SAS Viya Platform: Programming Run-Time Servers)*を参照してください。

PROC LUA のテーブル関数

ここでは、PROC LUA で使用できるテーブル関数の一部を示します。このテーブル関数を呼び出すには、table.size(t)のように、'TABLE'を前に付けます。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

以下は、PROC LUA で使用できる、Lua テーブルライブラリの一般的に使用されるテーブル関数の一部です。

TABLE.CONCAT(*Lua-table*<, "*delimiter*"><, *start-position*<, *end-position*>>)

TABLE.INSERT(*Lua-table*, <*position*, > *value*)

TABLE.REMOVE(*Lua-table*<, *position*>)

TABLE.SORT(*Lua-table*<, *comparison-function*>)

これは、PROC LUA 用に特別に作成されたテーブル関数です。これらの関数は、PROC LUA 内でのみ呼び出すことができます。

TABLE.CONTAINS(*Lua-table-name*,*v*)
指定されたテーブルに値 *v* が含まれている場合、ブール値 **true** を返します。文字値は、単一引用符または二重引用符で囲みます。

たとえば、名前のリストを使用してテーブル T を作成したとします。そのテーブルにテキスト文字列が含まれているかどうかを確認し、適切なメッセージをログに印刷できます。

```
local t = {"John", "Paul", "George", "Ringo"}

if (table.contains(t,"Ringo")) then
  print ("The table contains 'Ringo'.")
else
  print ("The table does not contain 'Ringo'.")
end
```

TABLE.SIZE(*Lua-table-name*)
指定されたテーブルの要素の数を返します。

TABLE.TOSTRING(*Lua-table-name*)
指定されたテーブルの出力形式が定義された文字列表現を返します。

詳細については、“例 3: SAS データセットのロードと、結果として生成される Lua テーブルの表示” (1188 ページ)、および“例 5: Table 関数の使用” (1194 ページ)を参照してください。

文字列を操作する関数

PROC LUA で使用できる文字列関数は次のとおりです。このテーブル関数を呼び出すには、string.trim(*text-variable*)のように、'STRING'を前に付けます。

注: 関数名はすべて大文字で記述するのが SAS の規則です。ただし、これらの関数を LUA プロシジャ内で呼び出す場合は、すべて小文字を使用する必要があります。

これらの文字列関数は、PROC LUA 専用に作成されています。これらの関数は、PROC LUA 内でのみ呼び出すことができます。

STRING.ENDS_WITH(*string1*, *string2*)
String1 の終わりが String2 の値と一致するかどうかを示すブール値を返します。比較文字では大文字と小文字が区別されます。リテラル文字値は引用符で囲みます。

STRING.RESOLVE(*string*, {token1="*text-value1*"<, token2="*text-value2*"<, ...>>});
トークンで置換された文字列を返します。次の形式の置換トークンを含む値を持つ文字列変数を、出力形式@ *トークン名*@で指定します 。トークンごとに、返される文字列値に置き換えるテキスト値を指定します。

たとえば、変数コードを次のように宣言したとします。

```
local code="data @results@; set @in@; where @where@; run;"
```

トークンの結果、in、および where の代入値を指定できます。

```
string.resolve(code, {results="work.foo", in="bar", where="x > 2"})
```

この STRING.RESOLVE の呼び出しは、次の文字列を返します。

```
data work.foo; set bar; where x > 2; run;
```

STRING.SPLIT(*string*, *delimiter*, *remove-empty-strings*)
指定された区切り文字値を使用して、文字列値をサブ文字列のテーブルに分割します。区切り文字の値を単一引用符または二重引用符で囲むか、二重角かっこ([[ ]])で囲みます。最後の引数であるブール値を使用して、結果のテーブルから空の文字列を削除するかどうかを指定します。デフォルトでは、最後の引数は TRUE です。

STRING.STARTS_WITH(*string1*, *string2*)
String1 の先頭が String2 の値と一致するかどうかを示すブール値を返します。比較文字では大文字と小文字が区別されます。リテラルテキスト値を単一引用符または二重引用符で囲むか、二重角かっこ([[ ]])で囲みます。

STRING.TRIM(*文字列*)
元の文字列値の最初と最後から空白文字が削除された文字列を返します。リテラルテキストを単一引用符または二重引用符で囲むか、二重角かっこ([[ ]])で囲みます。

詳細については、“例 6: String 関数の使用” (1196 ページ)を参照してください。

Lua ステートメント内のオブジェクト構文

データセット ID など、一部のオブジェクトについては、Lua コードによってオブジェクト構文がサポートされており、関数の前で、その関数の処理対象となるオブジェクトを指定します。たとえば、次の Lua ステートメントは同等です。

```
local luavar = sas.get_value(dsid, 'some_var')

local luavar = dsid:get_value('some_var')
```

関数の処理対象のオブジェクトの名前を、その関数の前に配置する場合は、必ずコロン(:)を使用してください。

同様に、“例 11: 大きなテーブルへの反復子関数の使用” (1208 ページ)の次のコードブロックは、2 つの方法で書き込まれる可能性があります。

```
-- Iterate over the rows of the data set
local i=0
while sas.next(dsid) do
  i=i+1
  print("OBS=" .. i)
  for vname,var in pairs(vars) do
    print(vname, '=', sas.get_value(dsid, vname) )
  end
end
```

SAS.NEXT 関数と SAS.GET_VALUE 関数は両方とも、オブジェクト構文で表すことができます。

```
-- Iterate over the rows of the data set
local i=0
while dsid:next() do
  i=i+1
  print("OBS=" .. i)
  for vname,var in pairs(vars) do
    print(vname, '=', dsid:get_value(vname) )
  end
end
```

Lua ステートメントでの PROC FCMP 関数の呼び出し

Lua コード内で FCMP プロシジャを使用して作成された関数をサブミットできます。PROC FCMP 関数を呼び出す場合は、関数が保存されているパッケージを、SAS OPTIONS ステートメントで指定する必要があります。

PROC FCMP 関数によって引数のいずれかが変更された場合、その引数は、OUTARGS ステートメントで指定されています。OUTARGS ステートメントの引数

に対する変更を Lua コードで取得するには、その引数は、配列として定義されている必要があります。

詳細については、次の情報を参照してください。

- 21 章, “FCMP プロシジャ,” (667 ページ)
- “例 14: PROC FCMP 関数の実行” (1216 ページ)

PROC LUA で FULLSTIMER オプションを使用する

PROC LUA で FULLSTIMER オプションを使用する場合、FULLSTIMER がログにレポートする実行時間は、LUA プロシジャ内で呼び出される SAS コード集計を反映します。実行時間には、LUA プロシジャ自体の実行時間が含まれます。

たとえば、SORT プロシジャを 3 回呼び出す PROC LUA コードについて考えます。SAS ログには次の出力が表示されます。

アウトプット 34.1　FULLSTIMER を使用した PROC LUA 出力

```
proc sort data=one; by i; run;
NOTE: There were 10000 observations read from the data set WORK.ONE.
NOTE: The data set WORK.ONE has 10000 observations and 2 variables.
NOTE: PROCEDURE SORT used (Total process time):
real time           0.03 seconds
user cpu time       0.03 seconds
system cpu time     0.00 seconds
memory              2252.60k
OS Memory           14492.00k
Timestamp           10/20/2015 11:43:40 AM
Step Count                        10  Switch Count  0

proc sort data=two; by i; run;
NOTE: There were 10000000 observations read from the data set WORK.TWO.
NOTE: The data set WORK.TWO has 10000000 observations and 2 variables.
NOTE: PROCEDURE SORT used (Total process time):
real time           8.03 seconds
user cpu time       12.34 seconds
system cpu time     0.70 seconds
memory              71231.17k
OS Memory           83136.00k
Timestamp           10/20/2015 11:43:48 AM
Step Count                        10  Switch Count  0

proc sort data=three; by i; run;
NOTE: There were 100 observations read from the data set WORK.THREE.
NOTE: The data set WORK.THREE has 100 observations and 2 variables.
NOTE: PROCEDURE SORT used (Total process time):
real time           0.01 seconds
user cpu time       0.00 seconds
system cpu time     0.01 seconds
memory              71231.17k
OS Memory           83136.00k
Timestamp           10/20/2015 11:43:48 AM
Step Count                        10  Switch Count  0

NOTE: PROCEDURE LUA used (Total process time):
real time           8.15 seconds
user cpu time       12.37 seconds
system cpu time     0.78 seconds
memory              71231.17k
OS Memory           83136.00k
Timestamp           10/20/2015 11:43:48 AM
Step Count                        10  Switch Count  0
```

出力では、0.78 秒のシステム CPU 時間は、PROC SORT への呼び出しの集計と、PROC LUA で使用される追加の CPU 時間を反映しています。

構文: LUA プロシジャ

制限事項:

SAS がロックダウン状態にある場合、Lua IO および OS ライブラリへのアクセスは制限されます。サーバー管理者は、次のリンクの手順に従って、これらの制限を削除できます。

- “Example 4: Enabling the Lua OS Library While in Lockdown Mode” (*SAS Viya Platform: Programming Run-Time Servers*)

- “Example 5: Enabling the Lua IO Library While in Lockdown Mode” (*SAS Viya Platform: Programming Run-Time Servers*)

PROC LUA <INFILE='*filename*'> <SECURELUATOKEN= "*digital_signature*"><RESTART> <TERMINATE>;
    <SUBMIT <"*assignment(s);*">;>
        ... *Lua statements* ...
    <ENDSUBMIT;>
run;

ステートメント	タスク
“PROC LUA ステートメント”	SAS コード内で Lua ステートメントを実行するか、実行する Lua ステートメントが含まれるファイルを指定します
“SUBMIT ステートメント”	Lua ステートメントのブロックの先頭を特定します
“ENDSUBMIT ステートメント”	Lua ステートメントのブロックの末尾を特定します

PROC LUA ステートメント

SAS セッションで Lua ステートメントを実行します。

構文

PROC LUA <INFILE='*filename*'><SECURELUATOKEN= "*digital_signature*"> <RESTART> <TERMINATE>;

オプション引数

INFILE= '*filename*'
SAS セッションで実行する Lua ステートメントが含まれるソースファイルを特定します。 SAS では、このファイル名の末尾には.lua ファイル拡張子が付いているものとされますが、この拡張子は、指定するファイル名には含まれていません。

要件 ファイル名を単一引用符または二重引用符で囲みます。

INFILE=オプションを使用する場合は、Lua スクリプトへのパスを指定する必要があります。PROC LUA ステートメントの前で LUAPATH ファイル名の値を指定して、Lua スクリプトへのパスを定義します。詳細については、“例 2: 外部 Lua スクリプトからの入力の指定” (1187 ページ)を参照してください。

例 open_data.lua Lua スクリプトファイルのコードを使用するには、
INFILE='*open_data*'
を指定します。

SECURELUATOKEN= "*digital_signature*"
BASE64URL エンコーディングでエンコードされた *digital_signature*

制限事項 PROC LUA では、Lua ロード関数がコンパイル済みバイナリコードをロードすることを許可していません。load 関数は、コンパイルされていない通常の Lua スクリプトのみをサポートします。

要件 Lua コンテンツを検証するために使用される証明書は、LUACERTLOC 環境変数で指定されたパスにあります。この場所は、LOCKDOWN パスリストに含まれ、SECURELUALOC システムオプションで指定されている必要があります。

- キーのペアは、RSA 公開キーアルゴリズムと 4096 ビットのキー長で生成する必要があります。
- Lua スクリプトのダイジェストは、SHA 256 アルゴリズムで計算する必要があります。Lua スクリプトの行末は、LF 制御文字(0x0A)を使用して表す必要があります。CRLF ではありません。
- Lua スクリプトのダイジェストは、秘密キーと RFC 3447 で定義された PKCS1 v1.5 パディングスキームを使用して署名する必要があります。
- 証明書ファイルは、LUACERTLOC 環境変数を介して検出されます。
- 証明書ファイルの場所は、LOCKDOWN PATH ステートメントを介して LOCKDOWN パスリストに追加され、SECURELUALOC オプションを使用して指定する必要があります。
- デジタル署名の作成の詳細については、OpenSSL の使用方法: ハッシュ、デジタル署名などを参照してください。

例 open_data.lua Lua スクリプトファイルのコードを使用するには、
SECURELUATOKEN="*digital_signature*"
Lua スクリプトのセキュリティを検証します。

RESTART
SAS セッションの Lua コードサブミットの状態をリセットします。 LUA プロシジャは、LUA プロシジャへの呼び出し間で Lua コードの状態が維持されるリエントラントプロシジャです。つまり、RESTART オプションまたは TERMINATE オプションを発行するか、SAS セッションを終了するまで、Lua グローバル変数割り当てまたは関数定義がメモリに残ります。

RESTART は、Lua コードの新しいブロックの先頭で指定できます。

例
```
proc lua restart;
submit;
 <lua statements...>
```

```
endsubmit;
run;
```

TERMINATE
LUA プロシジャの完了時に、メモリへの Lua コード状態の保持を停止し、Lua 状態を終了します。以降、LUA プロシジャを呼び出すと、Lua コード状態の新しいインスタンスが開始されます。

SUBMIT ステートメント

Lua コードのブロックの先頭を特定します。SUBMIT ステートメントと ENDSUBMIT ステートメントの間に、Lua ステートメントを入力します。

要件 各 SUBMIT ステートメントには、対応する ENDSUBMIT ステートメントが必要です。

構文

SUBMIT <'*assignments;*'>;

オプション引数

assignments;
Lua ステートメントのブロックに渡された 1 つ以上のマクロ変数割り当てを特定します。割り当てが 1 つだけの場合は、引用符内のセミコロン(;)は不要です。SAS では、Lua ステートメントのブロック内ではマクロ変数が拡張されません。したがって、SUBMIT ステートメントの割り当てリスト内でマクロ値を渡す必要があります。

例 マクロ変数の値 N を Lua 変数 Name に割り当てるには、次の SUBMIT ステートメントを入力します。

```
SUBMIT "name=&n";
```

詳細

マクロ定義内では、PROC LUA 内の SUBMIT/ENDSUBMIT ステートメントは許可されません。かわりに、%INCLUDE ステートメントまたは INFILE=引数を使用して、マクロ定義から LUA コードを実行できます。

ENDSUBMIT ステートメント

Lua ステートメントのブロックの末尾を特定します。ENDSUBMIT ステートメントと同じ行に他のステートメントを入力しないでください。

構文

ENDSUBMIT;

詳細

マクロ定義内では、PROC LUA 内の SUBMIT/ENDSUBMIT ステートメントは許可されません。かわりに、%INCLUDE ステートメントまたは INFILE=引数を使用して、マクロ定義から LUA コードを実行できます。

使用法: LUA プロシジャ

SAS プログラム内での Lua ステートメントのサブミット

PROC LUA 起動内の、SUBMIT ステートメントと ENDSUBMIT ステートメントの間で Lua ステートメントをサブミットできます。次のコードは、1 つの Lua print ステートメントを実行します。

```
proc lua;
submit;
  print("Hello from Lua")
endsubmit;
run;
```

SUBMIT および ENDSUBMIT ブロック内で定義するファイル参照名、ライブラリ参照名、マクロ変数などは、そのコードブロック内でのみ使用できます。

たとえば、マクロ変数 Mymacrovar は、次の PROC LUA の呼び出しで SUBMIT および ENDSUBMIT ブロック内で定義されます。ただし、この変数は、例の最後にある%PUT ステートメントで定義されていません。

```
proc lua restart;
submit;
  sas.submit([[%let mymacrovar=Hi there;]])
  txt = sas.symget("mymacrovar")
  print(txt)
endsubmit;
run;

%put &mymacrovar;
```

アウトプット 34.2　PROC LUA のマクロ変数の範囲

```
%let mymacrovar=Hi there;
Hi there
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.14 seconds
      cpu time            0.07 seconds


WARNING: Apparent symbolic reference MYMACROVAR not resolved.
17
18   %put &mymacrovar;
&mymacrovar
```

Lua コード内で SAS データセットを開く

SAS.SUBMIT 関数を呼び出すと、SAS コードのブロックをサブミットできます。SAS コードは、角かっこ([[および]])で囲みます。このサンプルでサブミットされたコードは、最初に、SAS データセットが存在するかどうかを確認します。サブミットされた DATA ステップを介して、Base SAS コードで行うようにデータを変更できます。

```
/* Test whether a data set exists */
proc lua;
submit;
  if sas.exists("sashelp.air") then
    print("The data set SASHELP.AIR exists.")
  else
    print("The data set SASHELP.AIR does not exist.")
  end
endsubmit;
run;
```

データセットが存在する場合は、データセットを開いて、それを新しいデータセット WORK.AIR に読み込むことができます。サブミットされた DATA ステップを介して、標準の DATA ステップで行うようにデータを変更できます。

注: 通常、より長い SAS コードブロックが Lua 変数に割り当てられます。

```
proc lua;
submit;
  sas.submit( [[ data work.air; set sashelp.air; run; ]] )
```

```
endsubmit;
run;
```

また、SAS コード内で Lua 変数値を代入することもできます。次のコードは、シンプルな代入を示しています。詳細については、“例 9: Lua 変数代入による SAS コードのサブミット” (1204 ページ)を参照してください。

```
proc lua;
submit;
  local dest = 'work.class'
  local source = 'sashelp.class'

  sas.submit( [[ data @dest@; set @source@; run; ]] )
endsubmit;
run;
```

例: LUA プロシジャ

例 1: サンプルデータセットの作成

この例では、複数の変数と 20 のオブザベーションを含むサンプル SAS データセット Homes を作成します。このデータセットは、次に示すいくつかの例の入力です。

PROC LUA を使用してアクセスできるサンプルデータセットを 作成するコードは次のとおりです。

```
data homes;
  input bad loan mortdue value reason $ job $ yoj derog delinq
      clage ninq clno debtinc;
datalines;
1 1100  25860  39025  HomeImp Other  10.5 0 0 94.37  1 9  .
0 4700  71855  88566  HomeImp Other  2.0  2 0 283.96 0 5  36.475
0 5500  72147  69918  HomeImp Sales  4.0  2 0 158.53 0 23 43.404
1 6400  25144  45200  HomeImp Other  25.0 0 2 128.00 4 17 .
0 7000  58114  93391  HomeImp ProfEx 6.0  0 0 200.08 1 24 31.737
1 7900  67222  75189  HomeImp Other  4.0  0 0 95.68  0 23 36.980
0 8300  54039  89301  HomeImp Other  18.0 0 0 173.19 0 28 26.267
0 8800  .     32221  DebtCon Other  0.0  0 0 276.84 0 14 24.199
0 9400  71600  99682  DebtCon Other  16.0 . . 159.04 4 16 25.607
0 10000 68807  76581  HomeImp Office 14.0 0 0 237.65 0 32 42.336
0 10200 16322  86505  DebtCon Other  8.0  0 0 259.16 0 14 21.253
0 10700 115118 124198 DebtCon Self   6.0  1 0 174.34 0 19 31.758
0 11100 148235 182053 HomeImp Office 4.0  0 0 198.81 0 55 33.539
0 11700 56441  86987  HomeImp Other  17.0 0 0 198.03 0 16 33.931
0 12100 58556  76724  HomeImp Other  3.0  0 1 234.66 1 16 37.639
0 12500 81865  101048 DebtCon Other  1.0  0 0 147.40 0 23 38.855
0 12900 18106  37881  DebtCon Mgr    8.0  0 0 134.38 0 10 20.516
0 13400 98701  129679 HomeImp ProfEx 10.0 0 0 179.13 2 32 29.549
```

```
1 13900 .    .    HomeImp Other  4.0  0 2 209.48 0 15 35.775
0 14400 45516  63924  DebtCon Other  .   0 0 109.33 1 19 40.872
;
```

例 2: 外部 Lua スクリプトからの入力の指定

要素: FILENAME ステートメント
PROC LUA ステートメント、INFILE=オプション

詳細

この例では、FILENAME ステートメントと、PROC LUA ステートメントの INFILE= オプションを使用して、外部 Lua スクリプトと、そのスクリプトへの考えられるパスを 1 つ以上指定します。

プログラム

```
filename LUAPATH ('/usr/local/scripts/lua','/home/user/myname/my_scripts');

proc lua infile='my_script';
run;
```

プログラムの説明

入力 Lua スクリプトを検索するディレクトリを指定します。 FILENAME ステートメントは、Lua スクリプトが保存されているディレクトリを定義し、それを LUAPATH ファイル参照名に割り当てます。ディレクトリパスは単一引用符または二重引用符で囲みます。

```
filename LUAPATH ('/usr/local/scripts/lua','/home/user/myname/my_scripts');
```

PROC LUA ステートメントを実行し、Lua スクリプト名を指定します。 INFILE=オプションは、Lua ステートメントが含まれる Lua スクリプトの名前を指定します。この例では、SAS が my_script.lua ファイルの Lua ステートメントを実行します。'.lua'または'.luc'のファイル拡張子は使用しないでください。システムの package.path 変数を確認して、最初に LUA ファイルまたは LUC ファイルが開いているかどうかを確かめます。

```
proc lua infile='my_script';
run;
```

例 3: SAS データセットのロードと、結果として生成される Lua テーブルの表示

要素: SAS.EXISTS 関数
SAS.READ_DS 関数
オブザベーション処理: WHILE および FOR ループ

詳細

この例では、例 1 で作成した Homes データセットを使用します。

この例では、SAS データセット Homes を読み取り、データを SAS ログに印刷します。個々のオブザベーションを配列のエントリとして扱います。各配列のエントリには、元の SAS データセットの変数から派生する、関連付けられた属性が含まれます。たとえば、t[2].loan の値は **4700** です。

プログラム

```
proc lua;
submit;
  if (sas.exists("work.homes")) then
    local t = sas.read_ds("work.homes")
    i=1

    while(t[i] ~= nil) do
      print("Obs #" .. i)
      for k,v in pairs(t[i]) do
          print(k,v)
      end
      print("\n")
      i = i+1
    end

  end
endsubmit;
run;
```

プログラムの説明

SAS データセット Homes が存在することを確認し、それを Lua テーブル T に読み込みます。 データセット内の各オブザベーションは、t[i]などの配列にあるかのようにアクセスできます。

```
proc lua;
submit;
  if (sas.exists("work.homes")) then
    local t = sas.read_ds("work.homes")
    i=1
```

データセットの各オブザベーションを処理します。 反復子を初期化し、WHILE ループを使用して各オブザベーションを処理します。次のオブザベーションが存在するかどうかをチェックし、変数と値の各ペアを出力します。FOR ループを使用して各変数の名前と値を処理し、両方を SAS ログに出力します。各オブザベーションの終わりに改行文字を出力し、i を増分します。

```
    while(t[i] ~= nil) do
      print("Obs #" .. i)
      for k,v in pairs(t[i]) do
          print(k,v)
      end
      print("\n")
      i = i+1
    end
```

SAS に PROC LUA 呼び出しをサブミットします。

```
  end
endsubmit;
run;
```

出力: Lua テーブル

アウトプット 34.3　サンプル Lua データ

```
Obs #1
reason    HomeImp
derog    0
job    Other
yoj    10.5
clno    9
loan    1100
bad    1
debtinc    .
mortdue    25860
value    39025
clage    94.37
ninq    1
delinq    0

...
Obs #20
reason    DebtCon
derog    0
job    Other
yoj    .
clno    19
loan    14400
bad    0
debtinc    40.872
mortdue    45516
value    63924
clage    109.33
ninq    1
delinq    0


NOTE: PROCEDURE LUA used (Total process time):
      real time           0.17 seconds
      cpu time            0.18 seconds
```

例 4: Lua テーブルからの SAS データセットの書き込み

要素:

PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント
SAS.WRITE_DS 関数
TABLE.TOSTRING 関数

詳細

この例では、Lua テーブルを作成し、そのテーブルを SAS データセットに書き込みます。Lua テーブルの値は、SAS RANNOR および RANUNI 乱数ジェネレーター関数を呼び出すことで生成されます。このコードは、配列を処理するための Lua 規則も使用しています。

プログラム

```
proc lua;
submit;
  local tbl = {}


  for i=1,10 do
     vars = {}
     vars.seed = 1234 * i;
     vars.randnor = sas.rannor( vars.seed )
     vars.randuni = sas.ranuni( vars.seed )
     vars.color = "purple"
     tbl[#tbl+1] = vars
  end


  print("Lua table:", table.tostring(tbl))


  sas.write_ds(tbl, "work.random")
endsubmit;
run;


proc print data=random;run;
```

プログラムの説明

PROC LUA ステートメントを実行して、Lua コードのブロックを開始します。 TBL と呼ばれるローカル Lua 配列を宣言します。

```
proc lua;
submit;
  local tbl = {}
```

Lua テーブルのコンテンツを生成します。 この例では、配列 TBL のコンテンツを生成します。変数 Seed、Randnor、Randuni、Color が作成され、FOR ループの 10 の反復にわたって値が割り当てられます。

```
  for i=1,10 do
     vars = {}
```

```
    vars.seed = 1234 * i;
    vars.randnor = sas.rannor( vars.seed )
    vars.randuni = sas.ranuni( vars.seed )
    vars.color = "purple"
    tbl[#tbl+1] = vars
end
```

生成された Lua テーブルを出力します。 結果として生成される Lua テーブルは、SAS ログに出力されます。

```
print("Lua table:", table.tostring(tbl))
```

Lua テーブルを SAS データセットに書き込みます。 この例では、Lua テーブルを、Work ライブラリの Random と呼ばれる SAS データセットに書き込みます。Work ライブラリに保存したデータセットには、同じ SAS セッションでのみアクセスできます。データセットを永久に保存するには、そのデータセットを、work などのライブラリに保存します。

```
  sas.write_ds(tbl, "work.random")
endsubmit;
run;
```

SAS データセットを出力します。 SAS データセットを書き込んだ後、その SAS データセットには、LUA プロシジャ外でアクセスできます。データセットを Sasuser などのライブラリに保存すると、そのデータセットには、後の SAS セッションでもアクセスできます。

```
proc print data=random;run;
```

出力: Lua テーブルの SAS テーブル

アウトプット 34.4　SAS ログの一部の Lua テーブル

```
Lua table:    table: 000000002892EAE0=
{
  [1]=table: 00000000289359A0=
  {
    ["color"]="purple"
    ["randuni"]=0.3831937143
    ["randnor"]=1.4215132075
    ["seed"]=1234
  }
  [2]=table: 0000000028937640=
  {
    ["color"]="purple"
    ["randuni"]=0.088249594
    ["randnor"]=-0.102644377
    ["seed"]=2468
  }
  [3]=table: 000000000C195400=
  {
    ["color"]="purple"
    ["randuni"]=0.4458283407
    ["randnor"]=2.0089305781
    ["seed"]=3702
  }
  [4]=table: 0000000028928FE0=
  {
    ["color"]="purple"
    ["randuni"]=0.4562234927
    ["randnor"]=1.9125666228
...
```

アウトプット 34.5　生成された SAS テーブル

The SAS System

Obs	seed	randuni	randnor	color
1	1234	0.38319	1.42151	purple
2	2468	0.08825	-0.10264	purple
3	3702	0.44583	2.00893	purple
4	4936	0.45622	1.91257	purple
5	6170	0.96744	1.82697	purple
6	7404	0.37316	-0.63993	purple
7	8638	0.28390	-1.49036	purple
8	9872	0.43640	-0.05252	purple
9	11106	0.00276	-0.45912	purple
10	12340	0.23450	0.96227	purple

例 5: Table 関数の使用

要素:

PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント
TABLE.CONCAT 関数
TABLE.INSERT 関数
TABLE.REMOVE 関数
TABLE.SIZE 関数
TABLE.SORT 関数
TABLE.TOSTRING 関数

詳細

この例では、単純なテーブルを作成し、要素の数を出力して、テーブルを SAS ログに出力します。

プログラム

```
proc lua;
submit;

 local t={"a", "b", "c", "foo", "bar"}

 print(table.size(t))
 print("Output with function TABLE.TOSTRING")
 print(table.tostring(t))
 print("Output with function TABLE.CONCAT")
 print(table.concat(t, ", "))

 print("Inserting and Removing Values")
 table.insert(t,5,"new")
 print("Table with new value: ", table.concat(t,", "))
 table.remove(t,4)
 print("Table after removing value at position 4: ", table.concat(t,", "))

 print("Sorting a Table")
 table.sort(t)
 print("Table with sorted values: ",table.concat(t,", "))
 table.sort(t, function(a,b) return a>b end)
 print("Table sorted in descending order: ",table.concat(t,", "))
```

```
endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントを抽出します。 PROC LUA ステートメントを使用すると、SAS セッションで Lua コードを呼び出すことができます。

```
proc lua;
```

Lua ステートメントのブロックの先頭を特定します。 SUBMIT ステートメントは、Lua ステートメントのブロックの先頭を特定します。

```
submit;
```

単純なテーブルを作成します。 テーブル T の要素を割り当てます。

```
local t={"a", "b", "c", "foo", "bar"}
```

テーブルに関する情報を出力します。 TABLE.TOSTRING 関数と TABLE.CONCAT 関数を使用して、テーブル内の要素の数を出力し、テーブルを出力します。

```
print(table.size(t))
print("Output with function TABLE.TOSTRING")
print(table.tostring(t))
print("Output with function TABLE.CONCAT")
print(table.concat(t, ", "))
```

テーブルの項目を挿入して削除します。 ラベルを出力し、TABLE.INSERT 関数を使用してテーブルの 5 番目の位置に値"new"という値を挿入します。テーブルを出力して、追加された値を確認します。TABLE.REMOVE 関数を使用して、テーブルの 4 番目の位置の値"foo"を削除します。結果のテーブルを出力します。

```
print("Inserting and Removing Values")
table.insert(t,5,"new")
print("Table with new value: ", table.concat(t,", "))
table.remove(t,4)
print("Table after removing value at position 4: ", table.concat(t,", "))
```

テーブル内の値を並べ替えます。 ラベルを出力し、TABLE.SORT 関数を使用してテーブルを昇順に並べ替えます。結果のテーブルを出力して、並べ替えられた値を確認します。次に、テーブルを再度並べ替えて、降順でテーブルを並べ替える関数を指定します。結果のテーブルを出力します。

```
print("Sorting a Table")
table.sort(t)
print("Table with sorted values: ",table.concat(t,", "))
table.sort(t, function(a,b) return a>b end)
print("Table sorted in descending order: ",table.concat(t,", "))
```

ENDSUBMIT ステートメントを使用して、Lua ステートメントのブロックの末尾を特定します

```
endsubmit;
```

```
run;
```

出力: テーブルの詳細を表示する SAS ログ

アウトプット 34.6　TABLE 関数の結果

```
Output with function TABLE.TOSTRING
table: 00000000289200E0=
{
  [1]="a"
  [2]="b"
  [3]="c"
  [4]="foo"
  [5]="bar"
}
Output with function TABLE.CONCAT
a, b, c, foo, bar
Inserting and Removing Values
Table with new value:     a, b, c, foo, new, bar
Table after removing value at position 4:     a, b, c, new, bar
Sorting a Table
Table with sorted values:     a, b, bar, c, new
Table sorted in descending order:     new, c, bar, b, a
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.03 seconds
      cpu time            0.01 seconds
```

例 6: String 関数の使用

要素:

PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント
STRING.ENDS_WITH 関数
STRING.RESOLVE 関数
STRING.SPLIT 関数
STRING.STARTS_WITH 関数
STRING.TRIM 関数

詳細

この例では、STRING 関数を使用してテキスト値を操作します。

プログラム

```
proc lua;
  submit;
  mystring = "@noun@ @verb@ the @object@"

  newstring1 = string.resolve(mystring, {noun="Pigs",verb="eat",object="pie"})
  newstring2 = string.resolve(mystring, {noun="Cars",verb="guzzle",
                        object="gasoline"})

  print ("Original string: " .. mystring)
  print ("New string 1: " .. newstring1)
  print ("New string 2: " .. newstring2)


  if string.starts_with(newstring2, "Cars") then
    print("Could be about cars: " .. newstring2)
  else
    print("Not about cars: " .. newstring2)
  end

  if string.ends_with(newstring2, "pie") then
    print ("I love pie!" .. newstring2)
  else
    print("Too bad. No pie: " .. newstring2)
  end


  mystring = "     The fox ate the chicken.  "
  trimmedstring = string.trim(mystring)

  print ("Original string between ***s: ***" .. mystring .. "***")
  print ("Trimmed string between ***s:  ***" .. trimmedstring .. "***")


  t = string.split(mystring," ")

  for element in pairs(t) do
    print("Table index " .. element .. ", Value is: " .. t[element] .. "\n")
  end


  endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントを実行して、Lua ステートメント処理を開始します。
PROC LUA ステートメントを使用すると、SAS セッションで Lua コードを呼び出すことができます。SUBMIT ステートメントは、Lua ステートメントのブロックの先頭を特定します。

```
proc lua;
  submit;
```

文字列変数を作成し、その中にさまざまな値を代入します。 置換トークンの名詞、動詞、目的語を含む文字列 Mystring を作成します。STRING.RESOLVE 関数を使用して、新しい文字列、Newstring1 および Newstring2 を生成します。元の文字列と新しい文字列を SAS ログに出力します。

```
mystring = "@noun@ @verb@ the @object@"

newstring1 = string.resolve(mystring, {noun="Pigs",verb="eat",object="pie"})
newstring2 = string.resolve(mystring, {noun="Cars",verb="guzzle",
                    object="gasoline"})

print ("Original string: " .. mystring)
print ("New string 1: " .. newstring1)
print ("New string 2: " .. newstring2)
```

文字列変数の最初と最後で一致するものを探します。 STRING.STARTS_WITH 関数を使用して、Newstring2 が Cars で始まるかどうかを確認してから、適切なメッセージを出力します。次に、STRING.ENDS_WITH 関数を使用して、Newstring2 が pie で終わっているかどうかを確認し、適切なメッセージを出力します。

```
if string.starts_with(newstring2, "Cars") then
  print("Could be about cars: " .. newstring2)
else
  print("Not about cars: " .. newstring2)
end

if string.ends_with(newstring2, "pie") then
  print ("I love pie!" .. newstring2)
else
  print("Too bad. No pie: " .. newstring2)
end
```

文字列値から先頭と末尾の空白文字を削除します。 Mystring に新しい値を割り当てます。STRING.TRIM 関数を使用して、先頭と末尾の空白文字を削除します。比較のために、元の文字列とトリミングされた文字列を出力します。

```
mystring = "    The fox ate the chicken.  "
trimmedstring = string.trim(mystring)

print ("Original string between ***s: ***" .. mystring .. "***")
print ("Trimmed string between ***s:  ***" .. trimmedstring .. "***")
```

文の文字列に単語のリストを作成します。 STRING.SPLIT 関数を使用して、文字列 Mystring から個々の単語のリストを生成します。デフォルトでは、空白の値は結果のリストに含まれません。Mystring の単語のリストを出力します。

```
t = string.split(mystring," ")

for element in pairs(t) do
  print("Table index " .. element .. ", Value is: " .. t[element] .. "\n")
end
```

Lua ステートメントのブロックを終了し、LUA プロシジャを実行します。
ENDSUBMIT ステートメントで、Lua ステートメントのブロックを終了します。

```
  endsubmit;
run;
```

出力: 文字列関数結果を表示する SAS ログ

アウトプット 34.7　STRING 関数の結果

```
NOTE: Lua initialized.
Original string: @noun@ @verb@ the @object@
New string 1: Pigs eat the pie
New string 2: Cars guzzle the gasoline
Could be about cars: Cars guzzle the gasoline
Too bad. No pie: Cars guzzle the gasoline
Original string between ***s: ***    The fox ate the chicken.    ***
Trimmed string between ***s: ***The fox ate the chicken.***
Table index 1, Value is: The

Table index 2, Value is: fox

Table index 3, Value is: ate

Table index 4, Value is: the

Table index 5, Value is: chicken.

NOTE: PROCEDURE LUA used (Total process time):
      real time           0.20 seconds
      cpu time            0.06 seconds
```

例 7: Lua テーブルへのオブザベーションの追加

要素:

SAS.APPEND 関数
SAS.CLOSE 関数
SAS.GET_VALUE 関数
SAS.PUT_VALUE 関数
SAS.UPDATE 関数
連結演算子(..)

詳細

この例は、PROC LUA の SAS 関数を使用してオブザベーションを追加する方法を示しています。まず、単純なデータセットを作成し、SAS.APPEND、SAS.PUT_VALUE、および SAS.UPDATE 関数を使用してレコードを追加します。次に、更新されたデータセットから値を取得して出力します。

プログラム

```
data foo;
 input x y;
 id+1;
 datalines;
1 10
2 20
3 30
;

proc lua;
submit;
  dsid = sas.open("work.foo",'u')

  while(sas.next(dsid) ~= nil) do
    myid = sas.get_value(dsid,"id")
    myx = sas.get_value(dsid,"x")
    myy = sas.get_value(dsid,"y")
    print("ID: " .. myid .. " x: " .. myx .. " y: " .. myy)
  end

  while(sas.next(dsid) ~= nil) do
    myid = sas.get_value(dsid,"id")
    myx = sas.get_value(dsid,"x")
    myy = sas.get_value(dsid,"y")
    print("ID: " .. myid .. " x: " .. myx .. " y: " .. myy)
  end


  sas.append(dsid)
  sas.put_value(dsid,"id",4)
  sas.put_value(dsid,"x",4)
  sas.put_value(dsid,"y",40)
  sas.update(dsid)
  rc=sas.close(dsid)
endsubmit;
run;


proc lua restart;
submit;
  dsid=sas.open("work.foo")


  while(sas.next(dsid) ~= nil) do
    myid = sas.get_value(dsid,"id")
    myx = sas.get_value(dsid,"x")
    myy = sas.get_value(dsid,"y")
    print("ID: " .. myid .. " x: " .. myx .. " y: " .. myy)
  end


  rc=sas.close(dsid)
endsubmit;
run;
```

プログラムの説明

データセットを作成します。 DATA ステップを使用して、Foo という小さなデータセットを作成します。

```
data foo;
 input x y;
 id+1;
 datalines;
1 10
2 20
3 30
;
```

PROC LUA を呼び出し、更新のために Foo データセットを開きます。 SAS.OPEN 関数を使用して、Foo データセットのコンテンツを Dsid Lua 変数に割り当てます。更新モードで('u'引数を使用して)データセットを開くように指定することによって、データセットに書き込むことができます。更新モードを指定しないと、データセットはデフォルトで読み取り専用モードで開きます。

```
proc lua;
submit;
  dsid = sas.open("work.foo",'u')

  while(sas.next(dsid) ~= nil) do
    myid = sas.get_value(dsid,"id")
    myx = sas.get_value(dsid,"x")
    myy = sas.get_value(dsid,"y")
    print("ID: " .. myid .. " x: " .. myx .. " y: " .. myy)
  end
```

Foo の値を読み取り、SAS ログに出力します。 SAS.NEXT 関数で WHILE ループを使用して、データセット内のオブザベーションを反復処理します。各オブザベーションに対して、ID、X、および Y それぞれの変数から Myid、Myx、および Myy の値を取得します。これらの値を SAS ログに出力します。

```
  while(sas.next(dsid) ~= nil) do
    myid = sas.get_value(dsid,"id")
    myx = sas.get_value(dsid,"x")
    myy = sas.get_value(dsid,"y")
    print("ID: " .. myid .. " x: " .. myx .. " y: " .. myy)
  end
```

新しいオブザベーションを追加し、各変数に値を割り当てます。 SAS.APPEND 関数を呼び出して、新しいオブザベーション用のデータセットを準備します。SAS.PUT_VALUE 関数を使用して値を割り当て、最後に SAS.UPDATE 関数でデータセットを更新します。データセットを閉じます。

```
  sas.append(dsid)
  sas.put_value(dsid,"id",4)
  sas.put_value(dsid,"x",4)
  sas.put_value(dsid,"y",40)
  sas.update(dsid)
  rc=sas.close(dsid)
endsubmit;
run;
```

PROC LUA を再開し、Work.Foo データセットを開きます。 Work.Foo のコンテンツを Dsid Lua 変数に割り当てます。

```
proc lua restart;
submit;
  dsid=sas.open("work.foo")
```

Work.Foo データセットからデータを取得します。 WHILE ループを使用して、Work.Foo データセットのオブザベーションを処理します。SAS.GET_VALUE 関数を使用して値を取得し、値を SAS ログに出力します。

```
  while(sas.next(dsid) ~= nil) do
    myid = sas.get_value(dsid,"id")
    myx = sas.get_value(dsid,"x")
    myy = sas.get_value(dsid,"y")
    print("ID: " .. myid .. " x: " .. myx .. " y: " .. myy)
  end
```

Work.Foo データセットを閉じます。

```
  rc=sas.close(dsid)
endsubmit;
run;
```

アウトプット 34.8　元の Work.Foo データセットのリスト

```
ID: 1 x: 1 y: 10
ID: 2 x: 2 y: 20
ID: 3 x: 3 y: 30
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.02 seconds
      cpu time            0.01 seconds
```

アウトプット 34.9　新しいオブザベーションが追加された Work.Foo データセットのリスト

```
ID: 1 x: 1 y: 10
ID: 2 x: 2 y: 20
ID: 3 x: 3 y: 30
ID: 4 x: 4 y: 40
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.16 seconds
      cpu time            0.06 seconds
```

例 8: Lua ステートメントでの SAS マクロ変数値の使用

要素:

PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント
マクロ変数割り当て

連結演算子(..)

詳細

SAS.SUBMIT コードブロック内のものを除き、SUBMIT ステートメントの末尾にあるセミコロン(;)と ENDSUBMIT ステートメントの先頭の間では、マクロ置換が発生しません。Lua コマンドのマクロ値の割り当てを SUBMIT ステートメント内で指定する必要があります。この例では、Lua コード内で使用するためにマクロ変数値を割り当てる方法を示します。

複数の割り当てがある場合はそれぞれをセミコロン(;)で区切り、すべての割り当てを 1 つの引用符で囲みます。この例では、次のテキストを SAS ログに出力します。
Hello, George. Have a nice day.

プログラム

```
%let g='George';
%let h='Have a nice day.';

proc lua;

submit "name=&g; msg=&h";

  print('Hello, ' .. name .. '. ' .. msg)

endsubmit;
run;
```

プログラムの説明

マクロ変数値を定義します。 マクロ変数 G および H は、文字値に割り当てられます。Lua では、マクロ変数割り当てに単一引用符を使用する必要があります。SAS のマクロ変数割り当ての場合、通常、単一引用符は不要です。

```
%let g='George';
%let h='Have a nice day.';
```

PROC LUA ステートメントを実行します。 PROC LUA ステートメントを使用すると、SAS セッションで Lua コードを呼び出すことができます。

```
proc lua;
```

Lua ステートメントのブロックの先頭を特定し、任意のマクロ変数を割り当てます。 SUBMIT ステートメントは、Lua ステートメントのブロックの先頭を特定します。この例では、マクロ変数 g および h の値を、ローカル Lua 変数 name および msg にそれぞれ割り当てます。SUBMIT ステートメントの末尾と ENDSUBMIT ステートメントの先頭の間では、マクロの展開が発生しません。Lua コードは、ローカル Lua 変数を参照します。

```
submit "name=&g; msg=&h";
```

Lua ステートメントを実行します。 SUBMIT ステートメントと ENDSUBMIT ステートメントの間に、Lua ステートメントを入力します。Lua ステートメントはセミコロン(;)で終了する必要はありません。この例では、テキスト値を SAS ログに出力します。'..'演算子を使用して文字列を連結します。ログでは、使用する print ステートメントはそれぞれ改行されて、表示されます。

```
print('Hello, ' .. name .. '. ' .. msg)
```

ENDSUBMIT ステートメントを使用して、Lua ステートメントのブロックの末尾を特定します

```
endsubmit;
run;
```

例 9: Lua 変数代入による SAS コードのサブミット

要素:

PROC LUA ステートメント
SUBMIT ステートメントと ENDSUBMIT ステートメント
SAS.SUBMIT 関数
変数代入

詳細

この例では、Lua ステートメントのブロックをサブミットします。Lua ステートメント内で、SAS コードのブロックをローカル変数に割り当てます。その後、SAS コードを SAS.SUBMIT 関数で呼び出して、変数値を代入します。

この例では、Work.Answer データセットを、ローカル Lua 変数コードへの入力として使用します。

図 34.1 Work.Answer データセット

The SAS System

Obs	CCUID	Value
1	60	12
2	61	27
3	62	5
4	62	8
5	67	9
6	67	4

プログラム

```
proc lua;
submit;

  local rc
  local code = [[
      data sample; set answer;
      where CCUID = @ccuid@;
      y = @subValue@;
      run;
  ]]
  rc = sas.submit(code, {ccuid="67", subValue=72})
endsubmit;
run;


proc print data=sample;
run;
```

プログラムの説明

LUA プロシジャを実行します。

```
proc lua;
```

実行する Lua ステートメントのブロックの先頭を特定します。 SUBMIT ステートメントを使用して、Lua ステートメントを特定します。ローカル変数 Rc を宣言します。Lua ステートメントの末尾にセミコロン(;)は不要です。

```
submit;

  local rc
```

SAS コードのブロックをローカル Lua 変数に割り当てます。 ローカル Lua 変数コードに、角かっこ([[および]])で示されている SAS コードのブロックが割り当てられています。SAS コードは、WORK.ANSWER データセットを開き、CCUID の値が、@ccuid@によって指定されている値と一致するレコードのみを保持します。SAS コードが実行されると、値がキー@ccuid@に割り当てられます。新しい変数 Y は、@subValue@によって指定されている値に割り当てられます。結果データセットは、WORK.SAMPLE データセットに保存されます。

```
  local code = [[
      data sample; set answer;
      where CCUID = @ccuid@;
      y = @subValue@;
      run;
  ]]
```

SAS コードを実行し、結果を Lua 変数 Rc に割り当てます。 SAS.SUBMIT 関数は、SAS コードを実行するようシステムに指示します。これにより、Code 変数に割り当てられたコードブロックが実行されます。SAS.SUBMIT 関数への呼び出しには 2 つ

の変数代入が含まれます。SAS コードが実行されたら、文字値*"67"*は@ccuid@に代入され、値 *72* は@subValue@に代入されます。

```
rc = sas.submit(code, {ccuid="67", subValue=72})
```

ENDSUBMIT ステートメントは、Lua ステートメントのブロックの末尾を特定します。 RUN ステートメントによって、PROC LUA への呼び出しが完了します。

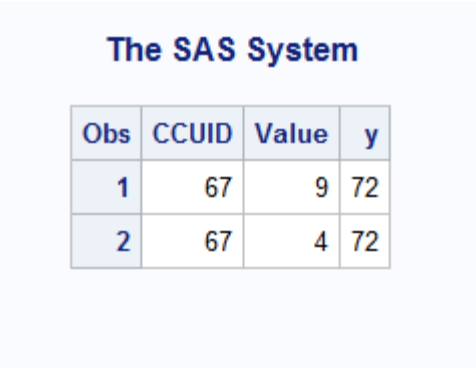
```
endsubmit;
run;
```

結果データセット Work.Sample を出力します。

```
proc print data=sample;
run;
```

出力: 代入された Lua 値の結果 SAS テーブル

アウトプット 34.10　生成された Work.Sample テーブル



The SAS System

Obs	CCUID	Value	y
1	67	9	72
2	67	4	72

例 10: 小さなテーブルに反復子関数を使用

要素:

SAS.OPEN 関数
SAS.ROWS 関数
SAS.CLOSE 関数

詳細

この例では、データセットを走査し、変数名と対応する値をオブザベーションごとに出力します。SAS.ROWS 関数は、変数が比較的少ない場合に使用します。

プログラム

```
proc lua;
submit;

  local dsid = sas.open("sashelp.class")

  for row in sas.rows(dsid) do
    for n,v in pairs(row) do
      if type(n)=="string" then
        print(n,'=', v)
      end
    end
  end

  sas.close(dsid)
endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントと SUBMIT ステートメントを実行して、Lua コードのブロックを開始します。

```
proc lua;
submit;
```

Sashelp.Class SAS データセットを開いて、コンテンツを Lua 変数 Dsid にロードします。 デフォルトでは、データセットは読み込みのために開かれます。

```
  local dsid = sas.open("sashelp.class")
```

データセットの行を処理します。 外部 FOR ループは、データセット内の各行を処理します。PAIRS 関数は、変数名と値のペアを Sashelp.Class データセットの各行から取り出します。その後、変数名と値のペアは SAS ログに出力されます。Lua では、IF 条件で 2 つの値が等しいかどうかを評価するときに、2 つの等号(==)が使用されることに注意してください。

```
  for row in sas.rows(dsid) do
    for n,v in pairs(row) do
      if type(n)=="string" then
        print(n,'=', v)
      end
    end
  end
```

データセットを閉じて、Lua コードのブロックを終了し、LUA プロシジャへの呼び出しを完了します。

```
  sas.close(dsid)
endsubmit;
run;
```

出力: 小さなデータセット用の ROWS 反復子

アウトプット 34.11　ROWS 関数で小さなデータセットを反復(部分出力)

```
sex    =    M
height    =    69
name    =    Alfred
weight    =    112.5
age    =    14
sex    =    F
height    =    56.5
name    =    Alice
weight    =    84
age    =    13
sex    =    F
height    =    65.3
name    =    Barbara
weight    =    98
age    =    13
sex    =    F
height    =    62.8
name    =    Carol
weight    =    102.5
age    =    14
...
sex    =    M
height    =    66.5
name    =    William
weight    =    112
age    =    15
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.08 seconds
      cpu time            0.06 seconds
```

例 11: 大きなテーブルへの反復子関数の使用

要素:

SAS.OPEN 関数
SAS.VARS 関数
SAS.NEXT 関数
SAS.GET_VALUE 関数
SAS.CLOSE 関数

詳細

この例では、SAS.VARS 関数と SAS.NEXT 関数を使用して、SAS データセットを走査し、そのコンテンツを出力します。

プログラム

```
proc lua;
submit;

  local dsid = sas.open("sashelp.company") -- open for input

  local vars = {}
  -- Iterate over the variables in the data set
  for var in sas.vars(dsid) do
    vars[var.name:lower()] = var
  end

  -- Iterate over the rows of the data set
  local i=0
  while (sas.next(dsid) ~= nil) do
    i=i+1
    print("OBS=" .. i)
    for vname,var in pairs(vars) do
      print(vname, '=', sas.get_value(dsid, vname) )
    end
  end

  sas.close(dsid)
endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントと SUBMIT ステートメントを実行して、Lua コードのブロックを開始します。

```
proc lua;
submit;
```

Sashelp.Company データセットを開きます。

```
local dsid = sas.open("sashelp.company") -- open for input
```

ローカル Lua 配列 VARS を宣言し、その配列に、データセットの変数の値を入力します。 波かっこ({ })は、Lua の配列を特定します。SAS.VARS 関数を使用して、データセット内のすべての変数を反復します。この例では、各変数の値を Vars 配列に割り当てます。ここで、配列キーは小文字の変数名です。

```
local vars = {}
-- Iterate over the variables in the data set
for var in sas.vars(dsid) do
  vars[var.name:lower()] = var
end
```

各データセットオブザベーションを反復し、変数ごとに値を出力します。 この例では、反復子変数 I を 0 に初期化します。その後、SAS.NEXT 反復子関数は、データセットのオブザベーションを繰り返します。この例は、オブザベーションごとに、変数名と各変数に対応する値を出力しています。

```
-- Iterate over the rows of the data set
local i=0
```

```
  while (sas.next(dsid) ~= nil) do
    i=i+1
    print("OBS=" .. i)
    for vname,var in pairs(vars) do
      print(vname, '=', sas.get_value(dsid, vname) )
    end
  end
```

データセットを閉じて、LUA プロシジャへの呼び出しを終了します。

```
  sas.close(dsid)
endsubmit;
run;
```

出力: 大きなデータセットでの反復子

アウトプット 34.12　大きなデータセットでの反復子関数の使用

```
OBS=1
level4    =    CONTRACTS
job1    =    MANAGER
level3    =    ADMIN
level2    =    TOKYO
n    =    1
depthead    =    1
level1    =    International Ai
level5    =    So Suumi
OBS=2
level4    =    CONTRACTS
job1    =    ASSISTANT
level3    =    ADMIN
level2    =    TOKYO
n    =    1
depthead    =    2
level1    =    International Ai
level5    =    Steffen Graff
...
OBS=48
level4    =    MIS
job1    =    TECH. CONS.
level3    =    TECHN. SERVICES
level2    =    NEW YORK
n    =    1
depthead    =    2
level1    =    International Ai
level5    =    Roy Hobbs
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.23 seconds
      cpu time            0.21 seconds
```

例 12: 変数の定義と、データセットへの変数の追加

要素:　PROC LUA ステートメント
SAS.OPEN 関数と'o'オプション

SAS.ADD_VARS 関数
SAS.CLOSE 関数

詳細

この例では、Lua コード内で新しい変数を定義し、その変数をデータセットに追加します。この例は、SAS.ADD_VARS 関数を使用して変数を定義し、出力データセットに追加しています。

プログラム

```
proc lua;
submit;
  local dsid = sas.open("work.sample","o")
  sas.add_vars(dsid, { {name="var1", type="N", format="BEST12.2"},
                       {name="var2", type="C", length=500, format="$char80."},
                       {name="var3", type="C", label="Character Data"}
                     })
  sas.close(dsid)
endsubmit;
run;
```

プログラムの説明

PROC LUA ステートメントを実行して、Lua ステートメントのブロックを開始します。 PROC LUA ステートメントを使用すると、SAS セッションで Lua コードを呼び出すことができます。SUBMIT ステートメントは、Lua ステートメントのブロックの先頭を特定します。

```
proc lua;
submit;
```

データセットを開いて、そのコンテンツをローカル Lua 変数に割り当てます。 SAS.OPEN 関数を呼び出して、WORK.SAMPLE データセットを開きます。データセットが存在しない場合は、'o'モードを使用して作成します。

```
local dsid = sas.open("work.sample","o")
```

変数を定義して、データセットに追加します。 この例では、SAS.ADD_VARS 関数を呼び出して、3 つの変数(Var1、Var2、Var3)をデータセットに追加します。種類(数値(N)または文字(C))は、変数ごとに指定されています。変数には、さまざまな追加属性が割り当てられます。名前属性のみが必須です。詳細については、“LUA プロシジャのデータセット関数” (1171 ページ)を参照してください。

```
sas.add_vars(dsid, { {name="var1", type="N", format="BEST12.2"},
                     {name="var2", type="C", length=500, format="$char80."},
```

```
    {name="var3", type="C", label="Character Data"}
  })
```

データセットを閉じて、LUA プロシジャへの呼び出しを終了します。 この例では、SAS.CLOSE 関数を実行して、Lua 変数 Dsid に関連付けられているデータセットを閉じます。

```
  sas.close(dsid)
endsubmit;
run;
```

出力: データセットへの変数の追加

アウトプット 34.13　Work.Sample の PROC CONTENTS からの部分出力

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	var1	Num	8	BEST12.2	
2	var2	Char	500	$CHAR80.	
3	var3	Char	200		Character Data

例 13: CAS サーバーへの接続

要素:

CAS.OPEN 関数
CLOSE 関数
SUBMIT ステートメントと ENDSUBMIT ステートメント

詳細

この例では、CAS サーバーに接続する方法を示します。ホスト名は **server.mycompany.com** 、ポートは **5570** です。

プログラム

```
proc lua;
submit;

-- Show content of swat_enabled variable
```

```
print("NOTE: swat_enabled=", swat_enabled)

-- Show path and cpath info at start
print("NOTE: before")
print("NOTE: package.path=\n" .. package.path)
print("NOTE: package.cpath=\n" .. package.cpath)

-- Add required directories to path and cpath
if swat_enabled == false or swat_enabled == nil then
  print("NOTE: changing path and cpath")
package.path = package.path ..
    ';/opt/sas/viya/home/SASFoundation/misc/casluaclnt/lua/?.lua'
package.path = package.path ..
    ';/opt/sas/viya/home/SASFoundation/misc/casluaclnt/lua/deps/?.lua'
package.cpath = package.cpath ..
    ';/opt/sas/viya/home/SASFoundation/misc/casluaclnt/lua/lib/?.so'

swat_enabled = true

-- Show updated path and cpath values
print("NOTE: after")
print("NOTE: package.path=\n" .. package.path)
print("NOTE: package.cpath=\n" .. package.cpath)
end

-- Preload library
cas = require "swat"


-- Connect to the CAS server on your host
s=cas.open("server.mycompany.com",5570)

-- Print the session ID
print("NOTE: s=", s)

-- Run the builtins.serverStatus action
-- and display some information
result = s:builtins_serverStatus
print("NOTE: server=\n",result['server'])
print("NOTE: nodestatus=\n",result['nodestatus'])


-- Run the addFmtLib action to add a format library
s:addFmtLib{fmtlibname="myformats"}

-- Create a format called demo in the myformats library
s:addFormat{fmtLibName="myformats",
          fmtname="demo",
          ranges={"0-17='demo1'","18-34='demo2'",
                "35-49='demo3'","50-64='demo4'"
               }}

-- Format some values with the newly created format
result = s:listFmtValues{fmtname = "demo",
            nvals = {15,23,39,61}}

-- Display the results
print(result)
```

```
-- Shut down the CAS session
s:close{}

endsubmit;
run;
```

プログラムの説明

PROC LUA を開始し、ユーティリティリソースをロードします。 連結演算子を使用して、middleclas.lua の場所を package.path 値に追加します。同様に、tkluaswat.so ファイルの場所を package.cpath 値に追加します。このプログラムを複数回実行した場合に package.path と package.cpath の値が更新されないように、Swat_enabled という変数が作成されます。SWAT の詳細については、see *"要件" (SAS Viya プラットフォームプログラミング: CAS 用 Lua 入門ガイド)*を参照してください。

```
proc lua;
submit;

-- Show content of swat_enabled variable
print("NOTE: swat_enabled=", swat_enabled)

-- Show path and cpath info at start
print("NOTE: before")
print("NOTE: package.path=\n" .. package.path)
print("NOTE: package.cpath=\n" .. package.cpath)

-- Add required directories to path and cpath
if swat_enabled == false or swat_enabled == nil then
  print("NOTE: changing path and cpath")
package.path = package.path ..
    ';/opt/sas/viya/home/SASFoundation/misc/casluaclnt/lua/?.lua'
package.path = package.path ..
    ';/opt/sas/viya/home/SASFoundation/misc/casluaclnt/lua/deps/?.lua'
package.cpath = package.cpath ..
    ';/opt/sas/viya/home/SASFoundation/misc/casluaclnt/lua/lib/?.so'

swat_enabled = true

-- Show updated path and cpath values
print("NOTE: after")
print("NOTE: package.path=\n" .. package.path)
print("NOTE: package.cpath=\n" .. package.cpath)
end
```

SWAT ライブラリをロードします。

```
-- Preload library
cas = require "swat"
```

CAS サーバーに接続します。 CAS.OPEN 関数を使用して、CAS サーバー上で CAS セッションを開始します。CAS セッションを Lua 変数 S に割り当てます。接続が成

功したことを確認し、セッション情報を出力します。このバージョンの CAS.OPEN 関数は、authinfo ファイルが作成されていることを前提としています。詳細については、*"接続とセッションの開始" (SAS Viya プラットフォームプログラミング: CAS 用 Lua 入門ガイド)*を参照してください。

```
-- Connect to the CAS server on your host
s=cas.open("server.mycompany.com",5570)

-- Print the session ID
print("NOTE: s=", s)

-- Run the builtins.serverStatus action
-- and display some information
result = s:builtins_serverStatus
print("NOTE: server=\n",result['server'])
print("NOTE: nodestatus=\n",result['nodestatus'])
```

CAS アクションを実行します。 addFmtLib アクションを実行して、新しいフォーマットライブラリを追加します。addFormat アクションを使用して新しいフォーマット定義を追加します。次に、フォーマットをいくつかの値に適用し、出力をチェックしてフォーマットされた値を確認します。

```
-- Run the addFmtLib action to add a format library
s:addFmtLib{fmtlibname="myformats"}

-- Create a format called demo in the myformats library
s:addFormat{fmtLibName="myformats",
        fmtname="demo",
        ranges={"0-17='demo1'","18-34='demo2'",
             "35-49='demo3'","50-64='demo4'"
            }}

-- Format some values with the newly created format
result = s:listFmtValues{fmtname = "demo",
          nvals = {15,23,39,61}}

-- Display the results
print(result)
```

CAS セッションをシャットダウンし、CLOSE 関数を使用して CAS サーバーから切断します。 Lua コードブロックを終了し、PROC LUA への呼び出しをサブミットします。

```
-- Shut down the CAS session
s:close{}

endsubmit;
run;
```

アウトプット 34.14　CAS サーバー出力への接続

```
NOTE: Lua initialized.

NOTE: swat_enabled=    nil

NOTE: before

NOTE: package.path=
/usr/local/.../init.lua;./?.lua

NOTE: package.cpath=
/usr/local/.../loadall.so;./?.so

NOTE: changing path and cpath

NOTE: after

NOTE: package.path=
/usr/local/.../init.lua;./?.lua;/opt/sas/viya/home/SASFoundation/misc/casluaclnt/
lua/?.lua;
/opt/sas/viya/home/SASFoundation/misc/casluaclnt/lua/deps/?.lua

NOTE: package.cpath=
/usr/local/.../loadall.so;./?.so;/opt/sas/viya/home/SASFoundation/misc/
casluaclnt/lua/lib/?.so

NOTE: s=    CAS{'server.mycompany.com', 5570, 'myuserID',
session='7239a98g3-1l5c-6932-7jjc-9ez39s0d1'}

NOTE: server=
          Server Status
Node Count  Total Actions
         1              1

NOTE: nodestatus=
                              Node Status
Node Name             Role        Uptime (Sec)  Running  Stalled
server.mycompany.com  controller         0.647        0        0
[listFmtValues]

                  ListFmtValues
Format Name  Number Value  Dest   Character value
demo                  15  demo1
                      23  demo2
                      39  demo3
                      61  demo4
```

例 14: PROC FCMP 関数の実行

要素:
- PROC FCMP ステートメント
- PROC LUA ステートメント
- FILENAME ステートメント
- SUBMIT ステートメントと ENDSUBMIT ステートメント

制限事項:
- FCMP プロシジャで作成された関数を呼び出すときは、配列である OUTARG 引数のみを変更できます。

詳細

この例では、FCMP プロシジャを使用して定義された関数を呼び出します。これは標準 SAS 関数を呼び出すのと似ています。関数の定義は、同じ SAS プログラムに含める必要はありません。PROC FCMP 関数を関数ライブラリに保存すると、CMPLIB システムオプションが定義されている任意のプログラムで使用できます。この例では、SUMX 関数と ADD_SCALAR 関数を定義して、ArrayFuncs パッケージの work.myFuncs データセットに保存します。*package* は、ユーザーによって指定された関連ルーチンの集合です。パッケージ名は、PROC FCMP 関数が含まれるデータセットで関連する関数をグループ化します。

プログラム

```
proc fcmp outlib=work.myFuncs.ArrayFuncs;
function sumx(x[*]);
  sum = 0;
  do i = 1 to dim(x);
    sum = sum + x[i];
  end;
  return(sum );
endsub;

function add_scalar( scalar, x[*] );
   outargs x;
   do i = 1 to dim(x);
     x[i] = x[i] + scalar;
   end;
   return( dim(x) );
endsub;
run;


options cmplib=work.myFuncs;


proc lua;
submit;
   array = { 1, 2, 3, 4, 5 }
   sum = sas.sumx(array)
   print(sum)
endsubmit;
run;


proc lua;
submit;
   array = { 1, 2, 3, 4, 5 }
   dim = sas.add_scalar(5, array)
   a1 = array[1]
   a2 = array[2]
```

```
  print(a1)
  print(a2)
endsubmit;
run;
```

プログラムの説明

FCMP プロシジャを使用して関数を定義します。 SUMX 関数は、配列の値を合計し、その値を返します。ADD_SCALAR 関数は、スカラー値を配列の各メンバーに追加します。ADD_SCALAR の OUTARGS 引数は配列を指定するため、PROC LUA を使用してこの関数を呼び出すことで、サブミットされた配列の値を変更できます。ADD_SCALAR 関数は、配列の要素の数を返します。

```
proc fcmp outlib=work.myFuncs.ArrayFuncs;
function sumx(x[*]);
 sum = 0;
 do i = 1 to dim(x);
   sum = sum + x[i];
 end;
 return(sum );
endsub;

function add_scalar( scalar, x[*] );
  outargs x;
  do i = 1 to dim(x);
    x[i] = x[i] + scalar;
  end;
  return( dim(x) );
endsub;
run;
```

コンパイルされた関数の場所を指定します。 この例では、work.myFuncs データセットで、前に定義された関数を検索します。関数が保存されている場合、FCMP プロシジャへの呼び出しは、LUA プロシジャと同じプログラムに含める必要はありません。LUA プロシジャ内で関数を呼び出す前に、OPTIONS ステートメントを使用して関数の場所を指定します。

```
options cmplib=work.myFuncs;
```

LUA プロシジャ内で SUMX 関数を呼び出します。 SUMX 関数を呼び出すには、呼び出しの前に"SAS."を付けます。これは、標準 SAS 関数を呼び出すのと似ています。

```
proc lua;
submit;
  array = { 1, 2, 3, 4, 5 }
  sum = sas.sumx(array)
  print(sum)
endsubmit;
run;
```

LUA プロシジャ内で ADD_SCALAR 関数を呼び出します。 ADD_SCALAR 関数を呼び出すには、呼び出しの前に"SAS."を付けます。これは、標準 SAS 関数を呼び出す

のと似ています。SUMX と ADD_SCALAR への呼び出しは、同じ LUA プロシジャ内で実行できます。

```
proc lua;
submit;
  array = { 1, 2, 3, 4, 5 }
  dim = sas.add_scalar(5, array)
  a1 = array[1]
  a2 = array[2]
  print(a1)
  print(a2)
endsubmit;
run;
```

出力: ユーザー定義の配列関数の操作

アウトプット 34.15　PROC FCMP を使用して作成された関数の呼び出し

```
388  proc fcmp outlib=work.myFuncs.ArrayFuncs;
389  function sumx(x[*]);
390    sum = 0;
391    do i = 1 to dim(x);
392      sum = sum + x[i];
393    end;
394    return(sum );
395  endsub;
396
397  function add_scalar( scalar, x[*] );
398    outargs x;
399    do i = 1 to dim(x);
400      x[i] = x[i] + scalar;
401    end;
402    return( dim(x) );
403  endsub;
404  run;

NOTE: Function add_scalar saved to work.myFuncs.ArrayFuncs.
NOTE: Function sumx saved to work.myFuncs.ArrayFuncs.
NOTE: PROCEDURE FCMP used (Total process time):
      real time           1.06 seconds
      cpu time            0.15 seconds


405
406  options cmplib=work.myFuncs;
407
408  proc lua;
409  submit;
410    array = { 1, 2, 3, 4, 5 }
411    sum = sas.sumx(array)
412    print(sum)
413  endsubmit;
414  run;

NOTE: Resuming Lua state from previous PROC LUA invocation.
15
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.30 seconds
      cpu time            0.07 seconds


415
416  proc lua;
417  submit;
418    array = { 1, 2, 3, 4, 5 }
419    dim = sas.add_scalar(5, array)
420    a1 = array[1]
421    a2 = array[2]
422    print(a1)
423    print(a2)
424  endsubmit;
425  run;

NOTE: Resuming Lua state from previous PROC LUA invocation.
6
7
NOTE: PROCEDURE LUA used (Total process time):
      real time           0.05 seconds
      cpu time            0.04 seconds
```

35

MEANS プロシジャ

概要: MEANS プロシジャ

MEANS プロシジャの機能

MEANS プロシジャは、すべてのオブザベーションにわたる変数とオブザベーショングループ内の変数に対する記述統計量を計算するためのデータ要約ツールを提供します。たとえば、PROC MEANS では次を行います。

- モーメントを基準として記述統計量を計算する
- 中央値を含む分位点を推定する
- 平均値の信頼限界を計算する
- 極値を識別する
- *t* 検定を実行する

デフォルトでは、PROC MEANS は出力を表示します。OUTPUT ステートメントを使用して、統計量を SAS データセットに保存することもできます。

PROC MEANS と PROC SUMMARY は類似しています。相違点の説明については、60 章, “SUMMARY プロシジャ,” (2141 ページ)を参照してください。

PROC MEANS が作成可能な出力の種類について

PROC MEANS デフォルト出力

出力 1.1 に、PROC MEANS が表示するデフォルトの出力を示します。PROC MEANS が分析するデータセットには、1 から 10 までの整数が含まれています。出力には、オブザベーション数、平均値、標準偏差、最小値、最大値がレポートされます。出力を生成するステートメントは次のとおりです。

```
proc means data=OnetoTen;
run;
```

アウトプット 35.1　デフォルト記述統計量

```
                              The SAS System                                1

                           The MEANS Procedure

                        Analysis Variable : Integer

        N            Mean         Std Dev         Minimum         Maximum
-------------------------------------------------------------------------------
       10       5.5000000       3.0276504       1.0000000      10.0000000
-------------------------------------------------------------------------------
```

PROC MEANS カスタマイズ出力

次の出力に、2 つの変数、MoneyRaised と HoursVolunteered の複数の広範分析の結果を示します。分析データセットには、地元の慈善団体に対し高校生たちが集めた金額とボランティア活動に充てた時間数に関する情報が含まれています。PROC MEANS は、2 つのカテゴリ変数の 6 つの組み合わせを使用して、オブザベーション数、平均値、範囲を計算します。最初の変数 School には 2 つの値があり、もう 1 つの変数 Year には 3 つの値があります。出力を生成するプログラムの説明については、“例 11: 出力統計量を使用し、極値を識別する” (1320 ページ)を参照してください。

アウトプット 35.2　分類水準と最大値の ID に対して指定された統計量

```
          Summary of Volunteer Work by School and Year              1

                     The MEANS Procedure

                 N
School       Year  Obs  Variable         N       Mean       Range
------------------------------------------------------------------------
Kennedy       1992  15  MoneyRaised      15   29.0800000   39.7500000
                        HoursVolunteered  15   22.1333333   30.0000000

              1993  20  MoneyRaised      20   28.5660000   23.5600000
                        HoursVolunteered  20   19.2000000   20.0000000

              1994  18  MoneyRaised      18   31.5794444   65.4400000
                        HoursVolunteered  18   24.2777778   15.0000000

Monroe        1992  16  MoneyRaised      16   28.5450000   48.2700000
                        HoursVolunteered  16   18.8125000   38.0000000

              1993  12  MoneyRaised      12   28.0500000   52.4600000
                        HoursVolunteered  12   15.8333333   21.0000000

              1994  28  MoneyRaised      28   29.4100000   73.5300000
                        HoursVolunteered  28   19.1428571   26.0000000

------------------------------------------------------------------------
```

```
        Best Results: Most Money Raised and Most Hours Worked           2

                            Most     Most      Money     Hours
 Obs  School   Year  _TYPE_  _FREQ_  Cash     Time     Raised  Volunteered

  1              .     0     109   Willard  Tonya    78.65      40
  2           1992     1      31   Tonya    Tonya    55.16      40
  3           1993     1      32   Cameron  Amy      65.44      31
  4           1994     1      46   Willard  L.T.     78.65      33
  5  Kennedy     .     2      53   Luther   Jay      72.22      35
  6  Monroe      .     2      56   Willard  Tonya    78.65      40
  7  Kennedy  1992     3      15   Thelma   Jay      52.63      35
  8  Kennedy  1993     3      20   Bill     Amy      42.23      31
  9  Kennedy  1994     3      18   Luther   Che-Min  72.22      33
 10  Monroe   1992     3      16   Tonya    Tonya    55.16      40
 11  Monroe   1993     3      12   Cameron  Myrtle   65.44      26
 12  Monroe   1994     3      28   Willard  L.T.     78.65      33
```

プログラムではレポートだけでなく、最も多くのお金を集めた学生とボランティア活動にほとんどの時間を充てた学生のうちすべてのオブザベーションを超えるものと School と Year の組み合わせ内のものを識別する出力データセット(出力の 2 ページ目)も作成します。

- データセットの最初のオブザベーションには、MoneyRaised と HoursVolunteered の値が全体で最大の生徒が示されます。
- オブザベーション 2 から 4 には、学校に関係なく、年度ごとの値が最大の学生が示されます。
- オブザベーション 5 と 6 には、年度に関係なく、学校ごとの値が最大の学生が示されます。

- オブザベーション 7 から 12 には、学校と年度の組み合わせごとの値が最大の生徒が示されます。

PROC MEANS と ODS OUTPUT ステートメント

MEANS および SUMMARY プロシジャのテンプレートによって、ODS はデフォルトの出力形式および出力形式属性を出力に適用し、入力データセットの変数に関連付けられ、適用されている任意の出力形式を上書きします。これは、オプション USE_FORMAT_DEFAULTS がテンプレート内で指定されているためです。このオプションは、ODS 形式の出力の幅および配置に影響するため、これらの幅および配置は、入力データセットまたは FORMAT ステートメントや ATTRIB ステートメントに見られる出力形式の幅および配置とは異なる場合があります。デフォルトの出力形式は ODS 出力に影響し、内部の数値表現や計算には影響しません。

たとえば、デフォルト出力形式の幅がデータセットまたはステートメントで指定した幅よりも広い場合、数値形式の出力の幅が予想よりも広くなったり、より多くの文字を含んだりすることがあります。

ODS OUTPUT ステートメントではなく OUTPUT ステートメントを使用して出力データセットを作成することにより、デフォルト出力形式の出力データセットへの適用を防止できます。

デフォルトの出力形式を ODS 出力に適用したくない場合は、次のコードを使用して Base.Summary ODS テーブルテンプレートを変更できます。

```
proc template;
  edit base.summary;
  use_format_defaults=off;
end;
run;
```

概念: MEANS プロシジャ

分類変数の使用

TYPES ステートメントと WAYS ステートメントの使用

TYPES ステートメントは、PROC MEANS がデータのサブグループ化に使用する利用可能な分類変数を制御します。入力データセットの 1 つのオブザベーションで発生するこれらのアクティブな分類変数値の一意の組み合わせにより、データサブグループが決定されます。指定した種類に対して PROC MEANS が生成するサブグループはそれぞれ、その種類の水準と呼ばれます。すべての種類に対し、非アクティブな分類変数が欠損値を含むオブザベーションのリジェクトのオブザベーション合計数に影響する可能性があります。

WAYS ステートメントを使用すると、PROC MEANS は、分類変数の完全セットから選択された *n* 分類変数のすべての可能な一意の組み合わせに対応する種類を生成します。たとえば、次のようになります。

```
proc means;
 class a b c d e;
 ways 2 3;
 run;
```

これは、次と同じです。

```
proc means;
 class a b c d e;
 types a*b a*c a*d a*e b*c b*d b*e c*d c*e d*e
      a*b*c a*b*d a*b*e a*c*d a*c*e a*d*e
      b*c*d b*c*e c*d*e;
 run;
```

TYPES ステートメントと WAYS ステートメントを省略すると、PROC MEANS はすべての分類変数を使用して表示されている出力に対しデータ(種類 NWAY)をサブグループ化し、出力データセットに対しすべての種類(2^k)を計算します。

分類値の並べ替え

PROC MEANS は、対応する 1 次元の種類での分類変数の順序を検証して、すべての種類での各分類変数の順序を決定します。オプション ORDER=DATA または ORDER=FREQ でのこの動作の影響を確認します。DBMS テーブルの入力データで ORDER=DATA オプションを指定すると、PROC MEANS の計算によって同じ分析の実行ごとに異なる結果が算出される可能性があります。ORDER=オプションは、クロス集計テーブルで変数水準の順序を決定するのに加え、PROC MEANS が計算する数多くの検定統計量と測定にも影響を与える可能性があります。PROC MEANS が入力データセットをサブセットに分割する場合、分類処理はサブグループごとにオプション ORDER=DATA または ORDER=FREQ にそれぞれ適用されません。代わりに、1 つの度数とデータ順序が、すべての出力に対し全体のデータセットの未分割ビューに基づいて作成されます。たとえば、次のステートメントについて考えます。

```
data pets;
 input Pet $ Gender $;
 datalines;
dog  m
dog  f
dog  f
dog  f
cat  m
cat  m
cat  f
;

proc means data=pets order=freq;
   class pet gender;
run;
```

ステートメントにより、次の出力が生成されます。

アウトプット 35.3　分類値の並べ替え

```
                     The SAS System                                  1

                   The MEANS Procedure

                                        N
              Pet          Gender     Obs
              --------------------------
              dog          f            3

                           m            1

              cat          f            1

                           m            2
              --------------------------
```

例では、PROC MEANS は雄猫を雌猫より前にリストしません。代わりに、データセット全体のすべての 種類に対する性別の順序を決定します。PROC MEANS によっ

て、雌のペットのオブザベーションの方が多いことがわかります(f=4、m=3)。
ORDER のデフォルト値は ORDER=INTERNAL です。

コンピューターリソース

PROC MEANS は、すべての動作環境にわたり同一のメモリ割り当てスキームを使用します。分類変数が含まれると、PROC MEANS はメモリに各分類変数の一意の値ごとのコピーを保持する必要があります。次を計算して、分類変数をグループ化するためのメモリ要件を推定できます。

$$Nc_1(Lc_1 + K) + Nc_2(Lc_2 + K) + ... + Nc_n(Lc_n + K)$$

ここで、

Nc_i
分類変数に対する一意の値の数です。

Lc_i
c_iのフォーマットされていない長さとフォーマットされた長さの組み合わせです。

K
約 32 バイト(64 ビットアーキテクチャの場合は 64 バイト)の一部の定数です。

CLASS ステートメントで GROUPINTERNAL オプションを使用する場合、Lc_iはc_iの未フォーマットの長さです。

指定されている種類の分類変数の一意の各組み合わせ$c_{1_i}\ c_{2_j}$によって、その種類の水準が形成されます。参照: “TYPES ステートメント” (1274 ページ)。次を計算して、すべての組み合わせが実際にデータ(完全な種類)に存在する場合に、指定された種類のすべての水準に対し可能性のある最大必要メモリを推定できます。

$$W * Nc_1 * Nc_2 * ... * Nc_n$$

ここで、

W
分析された変数の数と計算された統計量の数に基づく定数です(分位点の計算をするための QMETHOD=OS を要求しない限り)。

$Nc_1...Nc_n$
指定された種類のアクティブな分類変数に対する一意の水準の数です。

明らかに、水準の必要メモリは分類変数の水準に負担をかけます。このため、PROC MEANS は 1 つ以上のユーティリティファイルを開き、1 つ以上の種類の水準をディスクに書き込むことができます。これらの種類は、PROC MEANS が入力データスキャン時に作成したプライマリの種類、または派生の種類です。

PROC MEANS がすべてのプライマリの種類を入力データの処理時にディスクに一部書き込む必要がある場合、1 つ以上のマージパスがメモリの種類の水準とディスクの水準を組み合わせるために必要になる可能性があります。また、分類変数に DATA 以外の順序を使用する場合、PROC MEANS はディスクで完了した種類をグル

ープ化します。このため、最大のディスク必要量は、指定されている種類の必要メモリの 2 倍になることがあります。

PROC MEANS が一時作業ファイルを使用する場合、次の記載が SAS ログに表示されます。

```
Processing on disk occurred during summarization.
Peak disk usage was approximately nnn
Mbytes.
Adjusting MEMSIZE or REALMEMSIZE may improve performance.
```

ほとんどの場合、処理は正常に終了します。

CLASS ステートメントで分類変数を指定する場合、PROC MEANS がユーティリティファイルに書き込む前に使用するデータ依存メモリ量が、SAS システムオプション REALMEMSIZE=によって制御されます。REALMEMSIZE=の値は、SAS で割り当て可能な仮想メモリとは対照的な実メモリ量を示します。PROC MEANS は、これらの 2 つの値のうち小さい方を計算することによってユーティリティファイルに 書き込む前に使用するデータ依存メモリ量を決定します。

- REALMEMSIZE=の値
- 0.8*(M-U) (M は MEMSIZE=の値で、U はすでに使用中のメモリ量)

REALMEMSIZE は、PROC SORT など、その他のメモリ集中型 PROC の動作にも影響します。

代替として、PROC オプション SUMSIZE=を使用できます。PROC オプション SORTSIZE=と同様に、SUMSIZE=は、ディスク基準操作が開始するメモリしきい値を設定します。最高の結果を得るには、SUMSIZE=をタスクに利用可能になる可能性がある実メモリ量よりも少ない値に指定します。効率的にするため、PROC MEANS は SUMSIZE=の値の端数を内部的に切り上げることができます。SUMSIZE=は、分類変数を指定しない限り影響しません。

動作環境の情報: REALMEMSIZE= SAS システムオプションは、すべての動作環境で使用できるわけではありません。詳細については、使用している動作環境に関する SAS コンパニオンを参照してください。

PROC MEANS によりメモリが不十分であることがレポートされる場合、SUMSIZE=(または REALMEMSIZE=)の値を増やします。MEMSIZE=より大きい SUMSIZE= (または REALMEMSIZE=)値は影響しません。そのため、MEMSIZE=の値も増やす必要があることがあります。PROC MEANS によりディスク容量が不十分であることがレポートされる場合、WORK スペース割り当てを増やします。計算リソースパラメーターの調整方法については、使用している動作環境に関する SAS ドキュメントを参照してください。

パフォーマンスを向上させるもう 1 つの方法として、TYPES ステートメントまたは WAYS ステートメントを注意して適用し、計算を関連する分類変数の組み合わせにのみ制限します。特に、すべての分類変数の組み合わせを要求しないことによって重要なリソースの節約が可能になります。

PROC MEANS の In-Database 処理

大きなデータセットが外部データベースに格納される場合、SAS を実行するコンピューターへのデータセットの転送がパフォーマンス、セキュリティ、リソース管理

問題によって影響を受ける可能性があります。SAS In-Database 処理でデータベースで初期データ集計が実行されるようにすることにより、データ転送を大幅に減らすことができます。

PROC MEANS の In-Database 処理では、次のデータベース管理システムがサポートされています。

- Amazon Redshift
- DB2
- Google BigQuery
- Greenplum
- Hadoop
- Impala
- Microsoft SQL Server
- MySQL (2021.1.5 現在)
- Netezza
- Oracle
- PostgreSQL
- SAP HANA
- Snowflake
- Spark
- Teradata
- Vertica

適切な条件下で、PROC MEANS は使用されるステートメントと、PROC ステップで指定される出力統計量に基づいて SQL クエリを生成します。分類変数が指定されると、プロシジャは N 次元の種類を表す SQL GROUP BY 句を作成します。データベースでの集計クエリの実行時に作成される結果セットが内部 PROC MEANS データ構造に読み込まれ、すべての後続の種類が元の N 次元の種類から派生し、最終分析結果を形成します。SAS 出力形式定義がデータベースで配置される際に、分類変数のフォーマットがデータベースで発生します。SAS 出力形式定義がデータベースで配置されていない場合、未加工値に対し In-Database 集計が実行され、結果のセットが PROC MEANS 内部構造と結合されるときに関連する出力形式が適用されます。マルチラベルフォーマットは常に、データベースによって返される最初に集計された結果セットを使用して実行されます。ステートメント CLASS、TYPES、WAYS、VAR、BY、FORMAT、WHERE は、PROC MEANS がデータセット内で処理される場合にサポートされます。FREQ、ID、IDMIN、IDMAX、IDGROUPS は、サポートされていません。In-Database 処理に対しサポートされている統計量は、N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN、CSS、USS、VAR、STD、STDERR、PRET、UCLM、LCLM、CLM、CV です。

In-Database 処理の重み付けは、N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN に対してのみサポートされています。

In-Database 処理に対し現在サポートされていない統計量は、SKEW、KURT、P1、P5、P10、P20、P25/Q1、P30、P40、P50/MEDIAN、P60、P70、P75/Q3、P80、P90、P95、P99、MODE です。

SQLGENERATION システムオプションまたは LIBNAME ステートメントオプションは、In-Database プロシジャがデータベース内で実行されるかどうか、およびその実行方法を制御します。デフォルトで、In-Database プロシジャは可能な場合はデータベース内で実行されます。In-Database 処理を行わないようにする、OBS=、FIRSTOBS=、RENAME=、DBCONDITION=などの多くのデータセットオプションがあります。完全なリストについては、"Data Set Options" (*SAS In-Database Products: User's Guide*)を参照してください。

In-Database 処理は、分類変数がない(1 行が返される)、または選択されている分類変数に少数の一意の値が含まれている場合に、プロシジャに転送されるデータ量を大幅に減らすことができます。ただし、PROC MEANS は結果セットをその内部構造に読み込むため、SAS プロセスの必要メモリは In-Database 処理が実行されない場合に要求されたものと同じになります。データ要約のほとんどがデータベース内で実行される場合、SAS プロセスの必要 CPU を大幅に削減する必要があります。要約に必要な実時間を大幅に削減する必要があります。多くのデータベースプロセスクエリが並列処理されるためです。

In-Database 処理の詳細については、*SAS/ACCESS for Relational Databases: リファレンス*を参照してください。

入力データセットのスレッド処理

THREADS オプションは、入力データセットの並列処理を有効化または無効化します。スレッド処理により、処理操作における一定の並列処理が達成されます。この並列処理の目的は、所定の操作を完了するための処理時間を削減し、それによって追加 CPU リソースのコストを抑えることです。詳細については、並列処理の使用を参照してください。

SAS システムオプション CPUCOUNT=の値はスレッド化された並べ替えのパフォーマンスに影響します。CPUCOUNT=は、スレッド化されたプロシジャで使用できるシステム CPU の数を示します。

詳細については、*SAS システムオプション: リファレンス*の"THREADS System Option"および"CPUCOUNT= System Option"を参照してください。

計算された統計は、オブザベーションが処理される順序に応じてわずかに異なる場合があります。このような変動は、浮動小数点演算によって導入される数値エラーによるものであり、その結果は概算であり、正確ではないと見なす必要があります。オブザベーション処理の順序は、マルチスレッド処理または並列処理の非決定論的影響を受ける可能性があります。処理の順序は、ACCESS エンジンを介してクエリ結果を提供する DBMS などのデータソースによって 生成されるオブザベーションの一貫性のない、または非決定論的な並べ替えによっても 影響を受ける可能性があります。詳細については、"Base SAS のスレッド処理" (*SAS プログラマガイド: エッセンシャル*)を参照してください。

PROC MEANS の CAS 処理

CAS サーバーにあるデータテーブルを分析する場合、PROC MEANS によって実行される作業の一部は CAS 内で実行されます。CAS で実行される作業は、In-Database

処理によって DBMS 内で実行される作業と似ています。あなたの入力データセットが CAS からのものであれば、PROC MEANS 処理の一部を CAS サーバーで実行できます。PROC MEANS を CAS アクションで実行することは、SAS 内での処理に比べていくつかの利点があります。これらの利点には、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。インメモリテーブルは、比較的低速のネットワーク接続を介して転送されるのではなく、サーバー上でローカルに操作されるため、処理が高速になります。より多くの処理リソースが用意されている可能性があるため、CAS が使用されます。

適切な条件下で、PROC MEANS は、使用されるステートメントと、PROC ステップで指定される出力統計量に基づいて CAS アクションを生成および実行します。分類変数が指定されると、プロシジャは *n* 次元の種類を表す GROUPBY 入力テーブルパラメーターを作成します。アクションが CAS サーバーで実行される際に作成される結果セットが、SAS によって内部 PROC MEANS データ構造に読み込まれます。以降のすべてのタイプは、元の *n* 次元の種類から派生して、最終的な分析結果を形成します。

組み込み出力形式の場合、クラス変数のフォーマットは CAS で行われます。ユーザー定義の出力形式の場合、出力形式がサーバーで定義されているかサーバーにコピーされている場合、クラス変数のフォーマットは CAS で行われます。出力形式が CAS サーバー上で利用できない場合、初期集計が生の値を使用するサーバー上で発生し、結果のセットが PROC MEANS 内部構造に結合される際に関連する出力形式が SAS により適用されます。マルチラベルフォーマットは常に、CAS によって返される最初に集計された結果セットを使用して実行されます。

VARCHAR 変数は PROC MEANS ではサポートされていません。VARCHAR として宣言された変数は CHAR に変換されます。

ステートメント CLASS、TYPES、WAYS、VAR、BY、FORMAT、WHERE は、PROC MEANS が CAS サーバー内で処理される場合にサポートされます。FREQ、ID、IDMIN、IDMAX、IDGROUPS は、サポートされていません。

サポートされている統計量は、N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN、CSS、USS、VAR、STD、STDERR、PRET、UCLM、LCLM、CLM、CV です。

CAS 処理の重み付けは、N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN に対してのみサポートされています。

CAS で現在サポートされていない統計量は、SKEW、KURT、P1、P5、P10、P20、P25/Q1、P30、P40、P50/MEDIAN、P60、P70、P75/Q3、P80、P90、P95、P99、MODE です。

デフォルトでは、DATA=入力データセットが CAS のインメモリテーブルまたはビューを参照している場合、MEANS プロシジャは可能な場合は CAS サーバーで実行されます。CAS での処理を行わないようにする、OBS=、FIRSTOBS=、RENAME=などの多くのデータセットオプションがあります。

CAS で処理することで、分類変数がない(1 行が返される)、または選択されている分類変数に少数の一意の値が含まれている場合に、プロシジャに転送されるデータ量を減らすことができます。ただし、PROC MEANS は結果セットをその内部構造に読み込むため、SAS プロセスの必要メモリは CAS で処理しない場合に必要となる必要メモリと同等になります。データ要約のほとんどが CAS サーバー内で実行される場合、SAS プロセスの CPU 要件は大幅に低下することになります。CAS はデータを並列的に処理できるため、要約に必要な実時間処理は、大幅に低下するはずです。PROC MEANS の結果が OUTPUT ステートメントを使用して CAS サーバーに戻さ

れる場合でも、中間集計の処理は SAS の MEANS プロシジャによって実行される必要があります。すべての処理を CAS で実行する場合は、PROC CAS または他の多くの CAS クライアントと言語のいずれかを使用して、適切なアクションを直接呼び出すことができます。

注: 中間結果は、最終的な出力先(CAS またはクライアント)に関係なく、常にクライアントによって処理されます。

次に、CAS で PROC MEANS を実行する方法の例を示します。CAS LIBNAME エンジンは、CAS セッション名または CAS セッション UUID を介して SAS 9.4 セッションを既存の CAS セッションに接続するために使用されます。結果のライブラリ参照名は、特定の CAS セッションと通信するために SAS によって使用されます。

```
/* Connect to a CAS server */
cas casauto host="cloud.example.com" port=5570;
/* Specify the CAS engine LIBNAME statement and use the CAS engine libref*/
libname mycas cas sessref=casauto;
/* Enable INFO messages so that the SAS log reports the use of CAS actions. */
options msglevel=i;
/*For demonstration purposes, move Grade to CAS and rename it to
GradeBySection.*/
proc casutil;
load data=Grade casout=GradeBySection;
quit;
/*Execute the PROC MEANS procedure*/
/* A SORT step is not needed for BY-processing when running in CAS */
proc means data=mycas.GradeBySection min max mean;
  by Section;
  var Score;
  class Status Year;
  title1 'Final Exam Scores for Student Status and Year of Graduation';
  title2 'Within Each Section';
run;
cas casauto terminate;
```

構文: MEANS プロシジャ

ヒント: ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。詳細については、“言語の概念” (20 ページ)を参照してください。グローバルステートメントを使用することもできます。リストについては、*SAS グローバルステートメント: リファレンス*の“Global Statements”を参照してください。

PROC MEANS<*options*> <*statistic-keyword(s)*>;

ATTRIB *variable-list(s) attribute-list(s)*;

BY<DESCENDING>*variable-1*<<DESCENDING> *variable-2* ...> <NOTSORTED>;

CLASS *variable(s)*</ *options*>;

FORMAT *variable-1*<...*variable-n*> <*format*> <DEFAULT=*default-format*>;

FREQ *variable*;

ID *variable(s)*;

LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;

OUTPUT<OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
<*id-group-specification(s)*> <*maximum-id-specification(s)*>
<*minimum-id-specification(s)*> </ *options*>;

TYPES *request(s)*;

VAR *variable(s)*</ WEIGHT=*weight-variable*>;

WAYS *list*;

WEIGHT *variable*;

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク	例
“PROC MEANS ステートメント”	変数の記述統計量を計算します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10, Ex. 12, Ex. 13
“BY ステートメント”	BY グループごとに別の統計量を計算します。	Ex. 3
“CLASS ステートメント”	値が分析対象のサブグループを定義する変数を識別します。	Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10, Ex. 11, Ex. 12
“FREQ ステートメント”	値が各オブザベーションの度数を表す変数を識別します。	
“ID ステートメント”	追加の ID 変数を出力データセットに含めます。	
“OUTPUT ステートメント”	指定されている統計量と ID 変数を含む出力データを作成します。	Ex. 8, Ex. 9, Ex. 10, Ex. 11, Ex. 12
“TYPES ステートメント”	データの分割に使用する分類変数の特定の組み合わせを識別します。	Ex. 2, Ex. 5, Ex. 12
“VAR ステートメント”	分析変数と結果での順序を識別します。	Ex. 1
“WAYS ステートメント”	分類変数の一意の組み合わせを作成するための方法の数を指定します。	Ex. 6
“WEIGHT ステートメント”	値が統計量計算で各オブザベーションを重み付ける変数を識別します。	Ex. 6

PROC MEANS ステートメント

変数の記述統計量を計算します。

サポート: スレッド処理

参照項目: 60 章, “SUMMARY プロシジャ,” (2141 ページ)

例:
“例 1: 特定の記述統計量の計算” (1292 ページ)
“例 2: 分類変数を使用した記述統計量の計算” (1294 ページ)
“例 3: BY ステートメントを分類変数とともに使用する” (1296 ページ)
“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)
“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)
“例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)
“例 7: 平均の信頼限界の計算” (1310 ページ)
“例 8: 出力統計量を計算する” (1313 ページ)
“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1315 ページ)
“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)
“例 11: 出力統計量を使用し、極値を識別する” (1320 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)
“例 13: STACKODSOUTPUT オプションによるデータの制御” (1328 ページ)

構文

PROC MEANS <*options*> <*statistic-keyword(s)*>;

オプション引数の要約

DATA=*SAS-data-set*
入力データセットを指定します。

INCAS=(YES | NO)
CAS 内での処理を許可するかどうかを指定します。

NOTRAP
浮動小数点例外の復旧を無効化します。

SUMSIZE=*value*
分類変数によるデータ要約に使用するメモリ量を指定します。

THREADS

NOTHREADS
SAS システムオプション THREADS | NOTHREADS を無効にします。

Control the classification levels

CLASSDATA=*SAS-data-set*
: 分析する分類変数の組み合わせを含む 2 次データセットを指定します。

COMPLETETYPES
: 分類変数値の可能な組み合わせをすべて 作成します。

EXCLUSIVE
: CLASSDATA=データセットにない分類変数値の組み合わせをすべて分析から除外します。

MISSING
: 欠損値を有効値として使用し、分類変数の組み合わせを作成します。

Control the output

FW=*field-width*
: 統計量のフィールド幅を指定します。

MAXDEC=*number*
: 統計量に対し小数点以下の桁数を指定します。

NOLABEL
: 変数ラベルを抑制します。

NONOBS
: 分類変数の一意の各組み合わせに対するオブザベーション合計数のレポートを行いません。

NOPRINT
: 表示されているすべての 出力を非表示にします。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
: 分類変数の値を指定されている順序に従って並べ替えます。

PRINT
: 出力を表示します。

PRINTALLTYPES
: 分類変数の要求されたすべての組み合わせに対する分析を表示します。

PRINTIDVARS
: ID 変数の値を表示します。

STACKODSOUTPUT
: ODS 出力オブジェクトを生成します

Control the output data set

CHARTYPE
: _TYPE_変数に文字値が含まれるように指定します。

DESCENDTYPES
: _TYPE_値の降順で出力データセットを並べ替えます。

IDMIN
: ID 変数を最小値に基づいて選択します。

NWAY
: 出力統計量を_TYPE_の値が最高のオブザベーションに制限します。

Control the statistical analysis

ALPHA=*value*
: 信頼限界に対する信頼水準を指定します。

EXCLNPWGT
: 非正の重みが付いたオブザベーションを分析から除外します。

QMARKERS=*number*
P2 分位点推定方法に使用するサンプルサイズを指定します。

QMETHOD=OS | P2
分位点推定方法を指定します。

QNTLDEF=1 | 2 | 3 | 4 | 5
分位点の計算に使用される数学的定義を指定します。

statistic-keyword(s)
統計量を選択します。

VARDEF=DF | N | WDF | WEIGHT
分散の除数を指定します。

オプション引数

ALPHA=*value*
平均値に対する信頼限界を計算する信頼水準を指定します。信頼限界のパーセントは、(1−*value*)×100 です。例は次のとおりです(ALPHA=.05 は、95%の信頼水準になります)。

デフォルト .05

範囲 0 から 1 まで

操作 信頼限界を計算するには、*statistic-keyword* CLM、LCLM または UCLM を指定します。

参照項目: “信頼限界” (1287 ページ)

“例 7: 平均の信頼限界の計算” (1310 ページ)

CHARTYPE
出力データセットの_TYPE_変数が_TYPE_のバイナリ値の文字表現になるように指定します。変数の長さは分類変数の数と同じです。

操作 32 を超える分類変数を指定すると、_TYPE_は自動的に文字変数になります。

参照項目: “出力データセット” (1290 ページ)

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

CLASSDATA=*SAS-data-set*
出力に表示が必要な分類変数の値の組み合わせを含むデータセットを指定します。入力データセットではなく CLASSDATA=データセットで発生する分類変数の値の組み合わせが出力に表示され、度数はゼロとなります。

制限事項 CLASSDATA=データセットにすべての分類変数を含める必要があります。このデータの種類と出力形式は、入力データセットの対応する分類変数と一致する必要があります。

操作 EXCLUSIVE オプションを使用する場合、PROC MEANS は分類変数の組み合わせが CLASSDATA=データセットにない入力データセットのオブザベーションを除外します。

ヒント CLASSDATA=データセットを使用して、入力データセットをフィルタリングまたは補足します。

参照項目: “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

COMPLETETYPES
組み合わせが入力データセットで発生しない場合でも、分類変数の可能な組み合わせをすべて作成します。

操作 CLASS ステートメントの PRELOADFMT オプションにより、度数がゼロの場合でも PROC MEANS が分類変数の組み合わせに対するユーザー定義出力形式範囲または値を出力に書き込むようになります。

ヒント COMPLETETYPES を使用しても、必要メモリは増えません。

参照項目: “例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

DATA=*SAS-data-set*
入力 SAS データセットを識別します。

参照項目: “入力データセット” (24 ページ)

DESCENDTYPES
_TYPE_値の降順で出力データセットのオブザベーションを並べ替えます。

別名 DESCENDING

DESCEND

操作 NWAY を指定すると、降順は無効です。

ヒント DESCENDTYPES を使用して、全体合計(_TYPE_=0)を各 BY グループの最終オブザベーションにします。

参照項目: “出力データセット” (1290 ページ)

“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1315 ページ)

EXCLNPWGT
非正(ゼロまたは負)の重みがついたオブザベーションを分析から除外します。デフォルトでは、PROC MEANS は非正の重みが付いたオブザベーションを重みがゼロのオブザベーションのように扱い、オブザベーションの合計数にカウントします。

別名 EXCLNPWGTS

参照項目: “WEIGHT ステートメント” (1278 ページ)

VAR ステートメントの WEIGHT=オプション (1275 ページ)

EXCLUSIVE
CLASSDATA=データセットにない分類変数の組み合わせをすべて分析から除外します。

要件 CLASSDATA=データセットが指定されていない場合、このオプションは無視されます。

参照項目: “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

FW=*field-width*
統計量を印刷出力または表示出力に表示するフィールド幅を指定します。FW=は、出力データセットに保存される統計量に影響しません。

デフォルト 12

ヒント PROC MEANS が出力の列ラベルを切り捨てる場合、フィールド幅を増やします。

参照項目: “例 1: 特定の記述統計量の計算” (1292 ページ)

“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

INCAS=(YES | NO)
CAS 内での処理を許可するかどうかを指定します。

YES
CAS 内での処理を使用します。YES がデフォルトです。

NO
CAS 内での処理を使用しないでください。

IDMIN
出力データセットに ID 変数の最小値が含まれるように指定します。

操作 出力に ID 変数の値を表示する PRINTIDVARS を指定します。

参照項目: “ID ステートメント” (1261 ページ)

MAXDEC=*number*
印刷出力または表示出力に統計量を表示する小数点以下の桁の最大数を指定します。MAXDEC=は、出力データセットで保存される統計量に影響しません。

デフォルト BEST.(柱状出力形式の幅。通常は約 7)。

範囲 0-8

参照項目: “例 2: 分類変数を使用した記述統計量の計算” (1294 ページ)

“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

MISSING
欠損値を有効値とみなし、分類変数の組み合わせを作成します。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING を省略すると、PROC MEANS は欠損分類変数値を含むオブザベーションを分析から除外します。

参照項目: *SAS プログラマガイド: エッセンシャル*で特別な意味を持つ欠損値の説明を参照してください。

“例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

NOLABEL
変数ラベルを抑制します。

例
```
proc means data=sashelp.shoes maxdec=2 nolabels;
  var sales returns;
run;
```

NONOBS
分類変数の値の一意の各組み合わせに対するオブザベーション合計数を表示する列を非表示にします。この列は、出力データセットの_FREQ_変数に対応しています。

参照項目: “N Obs 統計量” (1290 ページ)

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

“例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

NOPRINT
すべての出力を抑制します。`“PRINT”オプションを参照してください。

ヒント OUT=出力データセットのみ作成する場合は、NOPRINT を使用します。

NOTRAP
データ処理時の浮動小数点例外(Floating Point Exception (FPE))の復旧を無効化します。デフォルトでは、PROC MEANS はこれらのエラーをトラップし、統計量を欠損に設定します。

FPE 回復のオーバーヘッドが重要な動作環境では、NOTRAP はパフォーマンスを向上させることができます。算術例外の場合に PROC MEANS が強制終了するように、通常の SAS FPE 処理は有効です。

NWAY
出力データセットに_TYPE_と_WAY_の値が最高のオブザベーションの統計量のみ含まれるように指定します。分類変数の指定時に、NWAY はすべての分類変数の組み合わせに対応します。

操作 TYPES ステートメントまたは WAYS ステートメントを指定すると、PROC MEANS はこのオプションを無視します。

参照項目: “出力データセット” (1290 ページ)

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
並べ替え順序を指定して、出力の分類変数の値の一意の組み合わせを作成します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 CLASS ステートメントで PRELOADFMT を使用すると、各分類変数の値の順序が、PROC FORMAT が関連するユーザー定義出力形式の値の格納に使用する順序と一致します。CLASSDATA=オプションを使用すると、PROC MEANS は CLASSDATA=データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC MEANS はまずユーザー定義出力形式を使用して、出力を並べ替えます。EXCLUSIVE を省略すると、PROC MEANS はユーザー定義出力形式と CLASSDATA=値の後に入力データセットの分類変数の一意の値を発生順に基づいて追加します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。この順序は、使用している動作環境によって異なります。

別名 FMT

EXTERNAL

FREQ
最も多くのオブザベーションを含む水準が最初に表示されるように、値を度数カウントの降順で並べ替えます。

操作 分類変数の多重組み合わせに対し、PROC MEANS は個別の分類変数度数から分類変数組み合わせの順序を決定します。

CLASS ステートメントで ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

UNFORMATTED
値をフォーマットされていない値で並べ替えます。PROC SORT と同じ順序になります。この順序は、使用している動作環境によって異なります。

別名 UNFMT

INTERNAL

デフォルト UNFORMATTED

参照項目: “分類値の並べ替え” (1227 ページ)

PRINT
PROC MEANS が統計量分析を表示するかどうかを指定します。NOPRINT は、すべての出力を行いません。

デフォルト PRINT

ヒント OUT=出力データセットのみ作成する場合は、NOPRINT を使用します。

参照項目: “例 8: 出力統計量を計算する” (1313 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

PRINTALLTYPES
印刷または表示出力の分類変数の要求された組み合わせをすべて(すべての _TYPE_値)表示します。通常、PROC MEANS は種類 NWAY のみ表示します。

別名 PRINTALL

操作 NWAY オプション、TYPES ステートメントまたは WAY ステートメントを使用すると、PROC MEANS はこのオプションを無視します。

参照項目: “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

PRINTIDVARS
ID 変数の値を印刷または表示出力に表示します。

別名 PRINTIDS

操作 ID 変数の最小値を表示する IDMIN を指定します。

参照項目: “ID ステートメント” (1261 ページ)

QMARKERS=*number*
P2 分位点推定方法に使用するマーカーのデフォルト数を指定します。マーカー数によって固定メモリ空間のサイズを制御します。

デフォルト デフォルト値は、要求する分位点によって異なります。中央値(P50)の場合、*number* は 7 です。分位点(P25 と P50)の場合、*number* は 25 です。分位点 P1、P5、P10、P75 P90、P95 または P99 の場合、*number*

は 105 です。複数の分位点を要求すると、PROC MEANS は *number* の最大値を使用します。

範囲 3 より大きい奇数の整数

ヒント マーカー数をデフォルト設定よりも増やして推定値の正確性を向上させます。マーカー数を減らして、メモリと計算時間を節約します。

参照項目: “分位点” (1288 ページ)

QMETHOD=OS | P2
PROC MEANS が分位点の計算時に入力データの処理に使用する方法を指定します。オブザベーション数が QMARKERS=値および QNTLDEF=5 以下の場合、両方の方法による結果は同じになります。

OS
順序統計量を使用します。この方法は、PROC UNIVARIATE が使用する方法と同じです。

注: この方法は、非常に多くのメモリを必要とする可能性があります。

P2
P2 分位点推定方法を使用して、分位点を概算します。

デフォルト OS

制限事項 QMETHOD=P2 の場合、PROC MEANS では MODE または重み付き分位点は計算されません。

ヒント QMETHOD=P2 の場合、一部の分位点(P、P5、P95、P99)の信頼性のある推定は、一部のデータセットに対し可能でないことがあります。

参照項目: “分位点” (1288 ページ)

QNTLDEF=1 | 2 | 3 | 4 | 5
QMETHOD=OS の場合に PROC MEANS が分位点の計算に使用する数学的定義を指定します。QMETHOD=P2 を使用するには、QNTLDEF=5 を使用する必要があります。

別名 PCTLDEF=

デフォルト 5

参照項目: “分位数と関連統計量” (2415 ページ)

statistic-keyword(s)
計算する統計量と、それを出力に表示する順序を指定します。PROC ステートメントで使用可能なキーワードは、“統計量キーワード” (1246 ページ)にリストされています。

デフォルト N、MEAN、STD、MIN、MAX

要件 平均値の標準誤差、信頼限界、スチューデントの *t* 検定を計算するには、VARDEF=オプションのデフォルト値、DF を使用する必要があります。歪度または尖度を計算するには、VARDEF=N または VARDEF=DF を使用する必要があります。

ヒント CLM または LCLM と UCLM の両方を使用して、平均値の両側信頼限界を計算します。LCLM のみ、または UCLM を使用して、片側信頼限界を計算します。

参照項目: キーワードの定義と関連する統計量の式については、“キーワードと式” (2410 ページ)を参照してください。

“例 1: 特定の記述統計量の計算” (1292 ページ)

“例 3: BY ステートメントを分類変数とともに使用する” (1296 ページ)

STACKODSOUTPUT
データセットが印刷出力に類似している ODS 出力オブジェクトを生成します。

STACKODSOUTPUT オプションは、PROC MEANS OUTPUT ステートメントではなく、ODS OUTPUT ステートメントで作成された出力データセットに影響します。

別名 STACKODS

参照項目: “例 13: STACKODSOUTPUT オプションによるデータの制御” (1328 ページ)

SUMSIZE=*value*
分類変数の使用時にデータ要約に使用できるメモリ量を指定します。*value* は、次のうちのいずれかになる可能性があります。

n
***n*K**
***n*M**
***n*G**
利用可能なメモリ量をバイト、キロバイト、メガバイト、ギガバイトでそれぞれ指定します。*n* が 0 の場合、PROC MEANS は SAS システムオプション SUMSIZE=の値を使用します。

MAXIMUM
MAX
利用可能なメモリの最大量を指定します。

デフォルト SUMSIZE=システムオプションの値。

注 SUMSIZE=0 を指定すると、PROC MEANS は優先するグローバル REALMEMSIZE オプションを使用できるようになります。

ヒント 最高の結果を得るには、SUMSIZE=に PROC ステップで利用可能な物理メモリ量より大きい値を指定しないでください。追加容量が必要な場合、PROC MEANS はユーティリティファイルを使用します。

参照項目: *SAS システムオプション: リファレンス*の SAS システムオプション SUMSIZE=。

THREADS | NOTHREADS
入力データセットの並列処理を有効化または無効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします。(制約を参照)。詳細については、並列処理の使用を参照してください。

デフォルト: SAS システムオプション THREADS | NOTHREADS の値。

制限事項: サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

操作: PROC MEANS は、BY ステートメントが指定されている、または SAS システムオプション CPUCOUNT の値が 2 未満の場合を除いて SAS システムオプション THREADS を有効にします。PROC MEANS ステートメントで THREADS を使用して、PROC MEANS がこれらの状況で並列処理を使用するように強制できます。

注: THREADS が指定され(SAS システムオプションとして、または PROC MEANS ステートメントで)、もう 1 つのプログラムが入力データセットを読み込み、書き込み、更新のために開いておくと、PROC MEANS が入力データセットを開けない場合があります。この場合、PROC MEANS は処理を停止し、メッセージを SAS ログに書き込みます。

VARDEF=DF | N | WDF | WEIGHT
分散と標準偏差の計算に使用する除数を指定します。次のテーブルに、*divisor* に可能な値と、関連する除数を示します。

表 35.1 VARDEF=に可能な値

値	除数	除数の式
DF	自由度	$n - 1$
N	オブザベーション数	n
WDF	重みの合計 - 1	$(\Sigma_i w_i) - 1$
WEIGHT | WGT	重みの合計	$\Sigma_i w_i$

プロシジャは分散を*CSS*/*divisor*として計算します。*CSS*は修正平方和で、$\Sigma\,(x_i - \bar{x})^2$と等しくなります。分析変数に重み付けをする場合、*CSS*は$\Sigma\, w_i(x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト　DF

要件　平均の標準誤差、平均の信頼限界またはスチューデントの *t*-検定を計算するには、VARDEF=のデフォルト値を使用します。

ヒント　WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。*i* 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$、およびw_iは *i* 番目のオブザベーションの重みです。この方法により、単位重みを含むオブザベーションの分散の推定が生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな *n* に対し)$\sigma^2/\bar{w}$の推定になります。$\bar{w}$は平均重みです。この方法により、平均重みを含むオブザベーションの分散の漸近的な推定が生成されます。

参照項目:　“キーワードと式” (2410 ページ)

“WEIGHT ステートメント” (1278 ページ)

統計量キーワード

CLM
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (1243 ページ)を参照してください。

CSS
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (1243 ページ)を参照してください。

KURTOSIS
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (1243 ページ)を参照してください。

別名　KURT

LCLM
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (1243 ページ)を参照してください。

MAX
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (1243 ページ)を参照してください。

MEAN
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (1243 ページ)を参照してください。

MIN
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

MODE
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

N
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

NMISS
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

RANGE
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

SKEWNESS
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

別名 SKEW

STDDEV
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

別名 STD

STDERR
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

SUM
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

SUMWGT
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

UCLM
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

USS
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

VAR
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

MEDIAN
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

別名 P50

P1
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P5
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P10
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P20
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P40
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P70
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

Q1
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

別名 P25

Q3
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

別名 P75

P90
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P95
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P99
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P30
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P60
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

P80
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

QRANGE
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

PROBT
仮説検定キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

別名 PBT

T
仮説検定キーワードです。 詳細については、*“statistic-keyword(s)”* (1243 ページ)を参照してください。

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

BY ステートメント

出力を BY グループ順に並べ替えます。

制限事項: BY ステートメントではグループ変数を指定しないでください。BY 変数をグループ変数と同じにすることはできません。

構文

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

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

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

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

詳細

SAS システムオプション NOBYLINE を指定して BY ステートメントを使用

BY ステートメントを、BY グループ処理によって生成される出力に通常表示される BY 行を非表示にする SAS システムオプション NOBYLINE と使用する場合、PROC MEANS は常に BY グループごとに新しいページを開始します。この動作により、タイトルに BY グループ情報を入力し、NOBYLINE でデフォルトの BY 行を非表示にしてカスタマイズした BY 行を作成した場合、タイトルの情報がページのレポートと一致するようになります。“BY グループの情報を含むタイトルの作成 ” (55 ページ)を参照してください。

BY グループ処理

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

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

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

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

- BY ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。
- PROC SORT の詳細については、55 章, “SORT プロシジャ,” (2027 ページ)を参照してください。
- CAS での BY グループ処理の詳細については、“グループ化と順序付け” (*SAS Cloud Analytic Services: DATA ステッププログラミング*)を参照してください。
- インデックスの作成の詳細については、“INDEX CREATE ステートメント” (435 ページ)を参照してください。

BY 変数値のフォーマティング

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

1 プロシジャで、値を BY 変数の内部(未フォーマット)値順に並べ替えるどうかが決定されます。

2 プロシジャで、BY 変数に出力形式を適用されたかどうかが決定されます。BY 変数が数値で、ユーザー適用の出力形式がない場合は、BY グループ処理のために BEST12.出力形式が適用されます。

3 プロシジャは、BY 変数または変数の内部値と未フォーマット値が両方とも変更されるまで、現在の BY グループにオブザベーションを追加し続けます。

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

2 つのデータセットで長さが異なる BY 変数

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

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

- プロシジャで、BY ステートメントが常にセカンダリデータセットに適用される場合があります。この場合、BY ステートメントの変数が 1 つ以上、セカンダリデータセットに出現している必要があります。
- プロシジャで、BY ステートメントがセカンダリデータセットに一度も適用されない場合もあります。この場合、セカンダリデータセットには、BY ステートメントの変数は不要です。
- プロシジャで、セカンダリデータセットに BY 変数があるかどうかチェックされる場合があります。セカンダリデータセットに BY 変数が 1 つもない場合、セカンダリデータセットに対する BY 処理は発生しません。セカンダリデータセットに BY 変数が 1 つ以上あり、それがプライマリデータセットの BY 変数と一致す

る場合、セカンダリデータセットに対して BY 処理が行われます。セカンダリデータセットに全部ではなく一部の BY 変数がある場合、プロシジャでエラーメッセージが出力され、終了する場合があります。または、その特定プロシジャのドキュメントで説明しているその他のアクションが実行される場合もあります。

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

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

- A
- A、B
- A、B、C
- A、B、C、D

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

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブランクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデータセットの'ABCD'とセカンダリデータセットの'ABCD 'は一致します。セカンダリデータセットに'ABCDEF'が含まれている場合は、一致しません。

BY ステートメントをサポートする Base SAS プロシジャ

COMPARE	SORT (必須)
CORR	STANDARD
FREQ	SUMMARY
MEANS	TABULATE

PRINT	TRANSPOSE
RANK	UNIVARIATE
REPORT (非ウィンドウ環境のみ)	ODSTEXT
ODSLIST	ODSTABLE

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

関連項目

"BY ステートメント" (SAS DATA ステップステートメント: リファレンス)

CLASS ステートメント

値が分析対象のサブグループの組み合わせを定義する変数を指定します。

別名: CLASSES

注: オプションのない CLASS ステートメントは ORDER=INTERNAL(デフォルト)、または PROC MEANS ステートメントの ORDER=オプションで指定された値を使用します。たとえば、次のコードでは、変数 c と d は ORDER=INTERNAL を使用します。PROC MEANS ステートメントで ORDER=オプションが指定されている場合、変数 c と d は PROC MEANS ステートメントの ORDER=オプションで指定された値を使用します。

```
class a b / order=data;
 class c d;
```

ヒント: 複数の CLASS ステートメントを使用できます。

一部の CLASS ステートメントオプションも PROC MEANS ステートメントで使用できます。これらは、すべての CLASS 変数に影響します。CLASS ステートメントで指定するオプションは、CLASS ステートメントの変数にのみ適用されます。

参照項目: CLASS ステートメントによるフォーマットされた値のグループ化方法については、"フォーマットされた値" (62 ページ)を参照してください。

例:
"例 2: 分類変数を使用した記述統計量の計算" (1294 ページ)
"例 3: BY ステートメントを分類変数とともに使用する" (1296 ページ)
"例 4: CLASSDATA=データセットを分類変数とともに使用する" (1299 ページ)
"例 5: 値のマルチラベル出力形式を分類変数とともに使用する" (1302 ページ)
"例 6: 事前に読み込まれた出力形式を分類変数とともに使用する" (1307 ページ)
"例 7: 平均の信頼限界の計算" (1310 ページ)

構文

CLASS *variable(s)* </ *options*>;

必須引数

variable(s)
プロシジャがデータのグループ化に使用する 1 つ以上の変数を指定します。CLASS ステートメントの変数は、*分類変数*といいます。分類変数は、数値または文字です。分類変数には連続値を含めることができますが、通常は変数の水準を定義するいくつかの不連続値が含まれます。データを分類変数別に並べ替える必要はありません。

操作 TYPES ステートメントまたは WAYS ステートメントを使用して、PROC MEANS がデータのグループ化に使用する分類変数を制御します。

ヒント 分類変数水準の数を減らすには、FORMAT ステートメントを使用して、変数値をまとめます。出力形式が複数の内部値を 1 つのフォーマットされた値にまとめると、PROC MEANS は最小内部値を出力します。

参照項目: “分類変数の使用 ” (1226 ページ)

オプション引数

ASCENDING
分類変数水準を昇順に並べ替えるように指定します。

別名 ASCEND

操作 ASCENDING と DESCENDING の両方を指定すると、PROC MEANS は警告メッセージを発行し、両方のオプションを無視します。

参照項目: “例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

DESCENDING
分類変数水準を降順で並べ替えるように指定します。

別名 DESCEND

操作 ASCENDING と DESCENDING の両方を指定すると、PROC MEANS は警告メッセージを発行し、両方のオプションを無視します。

EXCLUSIVE
ユーザー定義出力形式のすでに読み込まれた範囲にない分類変数のすべての組み合わせを分析から除外します。

要件 PRELOADFMT を指定して、分類変数出力形式を事前に読み込む必要があります。

参照項目: “例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

GROUPINTERNAL
PROC MEANS が値をグループ化して分類変数の組み合わせを作成する場合は、出力形式を分類変数に適用しないように指定します。

操作 PRELOADFMT オプションを指定すると、PROC MEANS は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ORDER=FORMATTED オプションを指定すると、PROC MEANS は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ヒント このオプションは、数値分類変数に不連続値が含まれている場合にコンピューターリソースを節約します。

参照項目: “コンピューターリソース” (1259 ページ)

MISSING
欠損値を分類変数水準に有効な値とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING を省略すると、PROC MEANS は欠損分類変数値を含むオブザベーションを分析から除外します。

参照項目: 特別な意味を持つ欠損値の説明については、*SAS プログラマガイド: エッセンシャル*を参照してください。

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

MLF
PROC MEANS により、マルチラベル出力形式が分類変数に割り当てられている場合に指定の範囲や重複する範囲の 1 番目と 2 番目の出力形式ラベルを使用して、サブグループの組み合わせを作成できるようになります。

要件 VALUE ステートメントで PROC FORMAT と MULTILABEL オプションを使用して、マルチラベル出力形式を作成する必要があります。

操作 OUTPUT ステートメントと MLF を使用すると、分類変数にはフォーマットされた値に対応する文字列が含まれます。フォーマットされた値が内部値になるため、この変数の長さは最長の出力形式ラベルの文字数です。

MLF と ORDER=FREQ を使用すると、フォーマットされた値に使用する順序が生成されない場合があります。MLF を CLASSDATA および EXCLUSIVE と使用した場合に期待する結果が得られない場合があります。MLF 処理で各 TYPE が別個に計算されるように要求されるためです。NWAY 以外の種類には、予想よりも多くの水準が含まれています。

注 フォーマットされた値が重複する場合、1 つの内部分類変数値が複数の分類変数サブグループの組み合わせにマップします。そのため、すべてのサブグループの N 統計量の合計がデータセットのオブザベーション数(N 統計量全体)を超えます。

ヒント MLF を省略すると、PROC MEANS は 1 番目の出力形式ラベルを使用します。このアクションは、最初の外部出力形式値を使用したサブグループの組み合わせの決定に一致します。

参照項目: FORMAT プロシジャの VALUE ステートメントにおける MULTILABEL オプション“オプション引数” (885 ページ)。

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
出力の分類変数の水準をグループ化する順序を指定します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 PRELOADFMT を使用すると、各分類変数の値の順序が、PROC FORMAT が関連するユーザー定義出力形式の値の格納に使用する順序と一致します。PROC ステートメントで CLASSDATA=オプションを使用すると、PROC MEANS は CLASSDATA=データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC MEANS はまずユーザー定義出力形式を使用して、出力を並べ替えます。PROC ステートメントで EXCLUSIVE を省略すると、PROC MEANS はユーザー定義出力形式と CLASSDATA=値の後に入力データセットの分類変数の一意の値を発生順に基づいて追加します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

参照項目: “例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。この順序は、使用している動作環境によって異なります。出力形式がクラス変数に割り当てられていない場合、デフォルトの出力形式 BEST12 が使用されます。

別名 FMT

EXTERNAL

参照項目: “例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

FREQ
最も多くのオブザベーションを含む水準が最初に表示されるように、値を度数カウントの降順で並べ替えます。

操作 分類変数の多重組み合わせに対し、PROC MEANS は個別の分類変数度数から水準の順序を決定します。

ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

参照項目: “例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

UNFORMATTED
値をフォーマットされていない値で並べ替えます。PROC SORT と同じ順序になります。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を年代順に表示する場合に特に便利です。

別名 UNFMT

INTERNAL

デフォルト UNFORMATTED

ヒント デフォルトでは、FREQ 以外のすべての順序は昇順です。降順の順序の場合は、DESCENDING オプションを使用します。

参照項目: “分類値の並べ替え” (1227 ページ)

PRELOADFMT
分類変数のすべての出力形式がすでに読み込まれていることを示します。

要件 COMPLETETYPES、EXCLUSIVE、または ORDER=DATA を指定し、出力形式を分類変数に割り当てない限り、PRELOADFMT は影響しません。

操作 PRELOADFMT と COMPLETETYPES の組み合わせは、入力データセット内に表示されない可能性のある値を生成します。場合によっては、文字クラス変数の場合、非常に低いまたは高い sentinel 値が生成されます。これらの値は、セッションエンコーディングの有効な文字を表していない可能性があります。これらの値は、生の形式で印刷するのではなく、関連する形式を使用して印刷する必要があります。出力形式定義で OTHER=値を使用すると、フォーマットを使用するときに生の sentinel 値が印刷されないようにすることができます。

PROC TABULATE 出力を入力データセットに存在するフォーマットされた分類変数値の組み合わせに制限するには、CLASS ステートメントで EXCLUSIVE オプションを使用します。

度数がゼロの場合でもユーザー定義出力形式のすべての範囲と値を出力に含めるには、PROC ステートメントで COMPLETETYPES を使用します。

注 文字出力形式をプリロードする場合、ラベルを生成する代表的な未加工値が少なくとも 1 つ存在しなければなりません。範囲に未加工値がない場合、到達不能と判断され、エラーが発生します。たとえば、Other=と Low-High の両方の範囲を持つマルチラベル文字出力形式をプリロードすることはできません。Other は、すべてのラベルが定義された後に残る値または範囲を参照する特別なキーワードです。文字出力形式の場合、low 範囲に欠損値が含まれるため、その他の域を占める未加工値が残りません。

参照項目: “例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

詳細

BY ステートメントと CLASS ステートメントの比較

BY ステートメントを使用することは、PROC MEANS が各 BY グループを入力データの独立したサブセットとして要約する、CLASS ステートメントと NWAY オプションを使用することと似ています。そのため、入力データの全体の要約は使用できません。ただし、CLASS ステートメントとは異なり、BY ステートメントでは BY ステートメントの変数でデータを事前に並べ替えるかインデックス付けする必要があります。

NWAY オプションを使用すると、PROC MEANS がすべての分類変数を要約するにはメモリが不足することがあります。一部の分類変数を BY ステートメントに移動できます。最大活用するには、すでに並べ替えられた、または最も多い一意の値を含む BY ステートメントに分類変数を移動します。

CLASS ステートメントと BY ステートメントを使用して、データを BY グループ内の分類変数の水準別に分析できます。“例 3: BY ステートメントを分類変数とともに使用する” (1296 ページ)を参照してください。

分類変数の欠損値の PROC MEANS による処理方法

デフォルトでは、オブザベーションに分類変数の欠損値が含まれている場合、PROC MEANS はそのオブザベーションを分析から除外します。PROC ステートメントで MISSING オプションを指定すると、プロシジャは欠損値を分類変数の組み合わせに有効な水準とみなします。

CLASS ステートメントで MISSING オプションを指定すると、個別の分類変数に対する欠損値の受け入れを制御できます。

コンピューターリソース

PROC MEANS で使用できる一意の分類変数値の合計は、使用可能なコンピューターメモリの量によって異なります。詳細については、“コンピューターリソース” (1228 ページ)を参照してください。

GROUPINTERNAL オプションは、コンピューターパフォーマンスを向上させることができます。グループ化プロセスが分類変数の内部値に基づいているためです。数

値分類変数に出力形式が割り当てられておらず、GROUPINTERNAL を指定しない場合、PROC MEANS はデフォルト出力形式、BEST12.を使用して文字列として数値をフォーマットします。PROC MEANS はこれらの数値変数を文字値別にグループ化します。これにはより多くの時間とコンピューターメモリが必要になります。

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。"出力形式の取り消し" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、"PUT ステートメント: リスト" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

FREQ ステートメント

各オブザベーションの度数を含む数値変数を指定します。

別名: FREQUENCY

構文

FREQ *variable*;

必須引数

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

度数変数の合計は、オブザベーションの合計数を表します。

注: FREQ 変数は、OUTPUT ステートメントで IDGROUP 構文を使用するときの PROC MEANS による複数の極値の識別方法に影響しません。

ID ステートメント

追加の変数を出力データセットに含めます。

参照項目: Iid-group-specification の説明については、“OUTPUT ステートメント” (1264 ページ)を参照してください。

構文

ID *variable(s)*;

必須引数

variable(s)
PROC MEANS がそのオブザベーショングループの最大値を出力データセットに含める 1 つ以上の変数を入力データセットから識別します。

操作 PROC ステートメントで IDMIN を使用して、ID 変数の最小値を出力データセットに含めます。

ヒント PROC ステートメントで PRINTIDVARS オプションを使用して、ID 変数の値を表示出力に含めます。

詳細

ID 変数の値の選択

ID ステートメントで変数を 1 つだけ指定する場合、指定されているオブザベーションに対する ID 変数の値が、入力データセットの対応するオブザベーショングループで検出される最大(最小)値となります。ID ステートメントで複数の変数を指定すると、PROC MEANS は ID ステートメントの変数を表示されている順序で処理し、最大値を選択します。PROC MEANS は最初の ID 変数の値を比較して、使用するオブザベーションをすべての ID 変数から決定します。複数のオブザベーションに同一の最大(最小) ID 値が含まれている場合、PROC MEANS は 2 番目と後続の ID 変数値を“タイブレーカー”として使用します。いかなる場合でも、すべての ID 値は、指定されている BY グループまたは種類内の分類水準に対する同一のオブザベーションから取得されます。

PROC MEANS による最大値を決定するための文字値の比較方法については、“文字変数の並べ替え順序 ” (2031 ページ)を参照してください。

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=’ ‘>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限 ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

事項 label N = 'Number = ';

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

label Name = "Owner's Last Name";

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: “文字定数と文字変数” (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

OUTPUT ステートメント

統計量を新しい SAS データセットに書き込みます。

ヒント: 複数の OUTPUT ステートメントを使用して、複数の OUT=データセットを作成できます。

例:
“例 8: 出力統計量を計算する” (1313 ページ)
“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1315 ページ)
“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)
“例 11: 出力統計量を使用し、極値を識別する” (1320 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

構文

OUTPUT <OUT=*SAS-data-set*>
<*statistic-specifications*>
<*id-group-specifications*> <*maximum-id-specifications*>
<*minimum-id-specifications*> </ *options*>;

オプション引数

OUT=*SAS-data-set*
新しい出力データセットに名前を付けます。*SAS-data-set* が存在しない場合、PROC MEANS が作成します。OUT=を省略すると、データセットの名前は DATA*n* となります。*n* は、名前を一意のものにする最小整数です。

デフォルト DATA*n*

ヒント データセットオプションを OUT=オプションと使用できます。

statistic-keyword<(***variable-list***)><=*name(s)*>
OUT=データセットに格納する統計量を指定し、その統計量を含む 1 つ以上の変数に名前を付けます。

statistic-keyword
出力データセットに格納する統計量を指定します。

使用可能な統計量キーワードは、“統計量キーワード” (1271 ページ)にリストされています。

デフォルトでは、出力データセットの統計量は分析変数の出力形式、入力形式、ラベルを自動的に継承します。ただし、N、NMISS、SUMWGT、USS、CSS、VAR、CV、T、PROBT、PRT、SKEWNESS、KURTOSIS に対して計算された統計量は、分析変数の出力形式を継承しません。この出力形式がこれらの統計量に無効な場合があるためです(ドル出力形式、日時出力形式など)。

デフォルト N、MEAN、STD、MIN、MAX

制限事項 *variable* と *name(s)*を省略すると、AUTONAME オプションも使用しない限り、PROC MEANS は *statistic-keyword* を単一の OUTPUT ステートメントで一度だけ使用できるようにします。

参照項目: キーワードの定義と関連する統計量の式については、“キーワードと式” (2410 ページ)を参照してください。

“例 8: 出力統計量を計算する” (1313 ページ)

“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1315 ページ)

“例 11: 出力統計量を使用し、極値を識別する” (1320 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

variable-list

統計量を出力データセットに格納する 1 つ以上の数値分析変数の名前を指定します。

デフォルト すべての 数値分析変数

names

分析変数統計量を含む出力データセットの変数に対し 1 つ以上の名前を指定します。たとえば、最初の名前には最初の分析変数に対する統計量が含まれます。2 番目の名前には 2 番目の分析変数の統計量が含まれます。

デフォルト 分析変数名。AUTONAME を指定する場合、デフォルトは分析変数名と *statistic-keyword* の組み合わせです。*output-statistic-specification* を使用せずに CLASS ステートメントと OUTPUT ステートメントを使用すると、出力データセットには分類変数の各組み合わせに対し、N、MIN、MAX、MEAN および STD の値という 5 つのオブザベーションが含まれます。WEIGHT ステートメント、または VAR ステートメントで WEIGHT オプションを使用すると、出力データセットには分類変数の各組み合わせに対する重みの合計(SUMWGT)を含むオブザベーションも含まれます。

操作 *variable-list* を指定する場合、PROC MEANS は分析変数を指定する順序を使用して、統計量を出力データセット変数に格納します。

ヒント AUTONAME オプションを使用すると、PROC MEANS は複数の変数と統計量に対し、一意の名前を生成します。

参照項目: “例 8: 出力統計量を計算する” (1313 ページ)

IDGROUP (<**MIN** (***variable-list-1***) | **MAX** (***variable-list-1***) <...**MIN** (***variable-list-n***) | **MAX** (***variable-list-n***) >> <**MISSING**> <**OBS**> <**LAST**> OUT <[*n*]> (*id-variable-list*)=<*names*>)

機能を組み合わせ、ID ステートメント、PROC ステートメントの IDMIN オプション、OUTPUT ステートメントの MAXID オプションと MINID オプションを拡張して、複数の極値を識別する OUT=データセットを作成します。

LAST

OUT=データセットに最後のオブザベーション(または *n* が指定されている場合は最後の *n* オブザベーション)からの値が含まれるように指定します。

LAST を指定しない場合、OUT=データセットには、最初のオブザベーション(または *n* が指定されている場合は最初の *n* オブザベーション)からの値が含まれます。OUT=データセットには、複数のオブザベーションが含まれている場合があります。これは、最後の(最初の)オブザベーションの値に加え、OUT=データセット値には分類変数値の組み合わせによって定義される各サブグループ水準の最後の(最初の)オブザベーションからの値が含まれるためです。

操作 MIN または MAX を指定し、複数のオブザベーションに同じ極値が含まれている場合、PROC MEANS はオブザベーション番号を使用して、OUT=データセットに保存するオブザベーションを解決します。LAST を指定すると、PROC MEANS は関係の解決に後のオブザベーションを使用します。LAST を使用しない場合、PROC MEANS は関係の解決に前のオブザベーションを使用します。

MAX(*variable-list*)

選択基準を指定し、*variable-list* で指定された 1 つ以上の入力データセット変数の最大極値を決定します。

複数の選択変数を指定すると、*n* 極値の選択に対するオブザベーションの並べ替えが、PROC SORT が複数の BY 変数を含むデータを並べ替えるのと同じ方法で行われます。PROC MEANS は、変数値を単一のキーにまとめます。MAX (*variable-list*)選択基準は、PROC SORT と、BY ステートメントで DESCENDING オプションを選択することに類似しています。

デフォルト MIN または MAX を指定しない場合、PROC MEANS はオブザベーション番号をオブザベーションを出力するための選択基準として使用します。

制限事項 矛盾した基準を指定すると、PROC MEANS は最初の選択基準のみ使用します。

操作 複数のオブザベーションですべての MIN 変数または MAX 変数に同じ極値が含まれている場合、PROC MEANS はオブザベーション番号を使用して、出力に書き込むオブザベーションを解決します。デフォルトでは、PROC MEANS は最初のオブザベーションを使用して、関係を解決します。ただし、LAST オプションを指定すると、PROC MEANS は最後のオブザベーションを使用して関係を解決します。

MIN(*variable-list*)

選択基準を指定し、*variable-list* で指定された 1 つ以上の入力データセット変数の最小極値を決定します。

複数の選択変数を指定すると、*n* 極値の選択に対するオブザベーションの並べ替えが、PROC SORT が複数の BY 変数を含むデータを並べ替えるのと同じ方法で行われます。PROC MEANS は、変数値を単一のキーにまとめます。

デフォルト　MIN または MAX を指定しない場合、PROC MEANS はオブザベーション番号をオブザベーションを出力するための選択基準として使用します。

制限事項　矛盾した基準を指定すると、PROC MEANS は最初の選択基準のみ使用します。

操作　複数のオブザベーションですべての MIN 変数または MAX 変数に同じ極値が含まれている場合、PROC MEANS はオブザベーション番号を使用して、出力に書き込むオブザベーションを解決します。デフォルトでは、PROC MEANS は最初のオブザベーションを使用して、関係を解決します。ただし、LAST オプションを指定すると、PROC MEANS は最後のオブザベーションを使用して関係を解決します。

MISSING

欠損値が選択基準で使用されるように指定します。

別名　MISS

OBS

極値が見つかった入力データセットにオブザベーション数を含む OUT=データセットに_OBS_変数を含めます。

操作　WHERE 処理を使用すると、_OBS_の値は、入力データセットのオブザベーションの場所に一致しないことがあります。

n]を使用して複数の極値を出力に書き込む場合、PROC MEANS は *n*_OBS_変数を作成し、接尾辞 *n* を使用して変数名を作成します。*n* は、1 から *n* までの連続した整数です。

[*n*]

OUT=データセットに含める *id-variable-list* の各変数に対する極値の数を指定します。PROC MEANS は *n* の新規変数を作成し、接尾辞_*n* を使用して変数名を作成します。*n* は、1 から *n* までの連続した整数です。

デフォルトでは、PROC MEANS は、要求された各種類の水準ごとに 1 つの極値を決定します。*n* が 1 より大きい場合、*n* 極値が各種類の水準ごとに生成されます。*n* が 1 より大きく、極値選択を要求する場合、時間計算量は $O(T * N \log_2 n)$になります。Tは要求された種類の数であり、Nは入力データセットのオブザベーションの数です。比較すると、データセット全体をグループ化するための時間計算量は $O(N \log_2 N)$になります。

デフォルト　1

範囲　1 から 100 までの整数

例　たとえば、変数ごとに 2 つの最小極値を生成するには、次を使用します。

```
idgroup(min(x) out[2](x y z)=MinX MinY MinZ);
```

OUT=データセットには、変数 MinX_1、MinX_2、MinY_1、MinY_2、MinZ_1、MinZ_2 が含まれます。

(*id-variable-list*)
PROC MEANS がその値を OUT=データセットに含める 1 つ以上の入力データセット変数を識別します。PROC MEANS は、指定する選択基準(MIN、MAX および LAST)によって生成するオブザベーションを決定します。

別名 IDGRP

要件 まず MIN | MAX 選択基準を選択し、OUT(*id-variable-list*)=をサブオプション MISSING、OBS、LAST の後に指定する必要があります。

ヒント ID ステートメントの動作を模倣するには、*id-group-specification* を使用し、また OUTPUT ステートメントで *maximum-id-specification* または *minimum-id-specification* を使用することができます。

出力データセットに極値とその他の ID 変数を含める場合、それらを *id-variable-list* に含めると個別の統計量を求めるよりも効率的です。たとえば、ステートメント output idgrp(max(x) out(x a b)=);は、ステートメント output idgrp(max(x) out(a b)=) max(x)=;より効率的です。

参照項目: “例 8: 出力統計量を計算する” (1313 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

names
OUT=データセットの変数に対し 1 つ以上の名前を指定します。

デフォルト *name* を省略すると、PROC MEANS は *id-variable-list* の変数名を使用します。

ヒント AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注意
IDGROUP 構文を使用して、同じ名前の出力変数を作成できます。 このアクションが発生するのは、出力データセットに最初の変数が表示される場合だけです。AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注: 指定する新しい変数名が分析変数と ID 変数の組み合わせよりも少ない場合、その他の出力変数は、PROC MEANS が新しい変数名のリストを使い果たすとすぐに ID 変数の対応する名前を使用します。

別名 IDGRP

MAXID <(***variable-1*** <(***id-variable-list-1***)> <...***variable-n*** <(***id-variable-list-n***)>>)>
= *names*
1 つ以上の ID 変数が分析変数の最大値と関連付けられるように指定します。

variable
PROC MEANS が最大値を決定する数値分析変数を識別します。PROC MEANS は、1 つの変数に対して複数の最大値を決定できます。これは、全体の最大値に加え、分類変数値の組み合わせによって定義されるサブグループ水準にも最大値が含まれているためです。

ヒント ID ステートメントを使用し、*variable* を省略すると、PROC MEANS はすべての分析変数を使用します。

id-variable-list
値が分析変数の最大値を含むオブザベーションを識別する 1 つ以上の変数を識別します。

デフォルト ID ステートメント変数

names
各分析変数の最大値と関連付けられている ID 変数の値を含む新しい変数の名前を指定します。

注 複数のオブザベーションに分類水準内の最大値が含まれている場合、PROC MEANS は出力データセットのこれらのオブザベーションのうち最初のものだけに対する ID 変数の値を保存します。

ヒント ID ステートメントを使用し、*variable* と *id-variable* を省略すると、PROC MEANS はすべての ID ステートメント変数と各分類変数を関連付けます。このため、各分析変数に対し、出力データセットで作成される変数の数は ID ステートメントで指定する変数の数と同じです。

AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

参照項目: “例 11: 出力統計量を使用し、極値を識別する” (1320 ページ)

注意
MAXID 構文を使用して、同じ名前の出力変数を作成できます。 このアクションが発生するのは、出力データセットに最初の変数が表示される場合だけです。AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注: 指定する新しい変数名が分析変数と ID 変数の組み合わせよりも少ない場合、その他の出力変数は、PROC MEANS が新しい変数名のリストを使い果たすとすぐに ID 変数の対応する名前を使用します。

MINID<(***variable-1*** <(***id-variable-list-1***)> <...***variable-n*** <(***id-variable-list-n***)>>)> = ***name(s)***
maximum-id-specification の説明を参照してください。このオプションは、PROC MEANS が最大値ではなく最小値を決定することを除けば、まったく同じように作動します。

MINID は明示的な変数リストなしで使用される場合、次のさらに詳細な IDGROUP 構文例と類似しています。

IDGRP(min(x) missing out(id_variable)=idminx) idgrp(min(y) missing out(id_variable)=idminy)

1 つ以上の分類変数に欠損値が含まれている場合、id_variable 値は、MIN 統計量に対する値を含むオブザベーションではなく、欠損値を含むオブザベーションに対応します。

スラッシュ区切り文字の後に使用するオプション

AUTOLABEL
PROC MEANS が統計量名を変数ラベルの終わりに追加することを指定します。分析変数にラベルがない場合、PROC MEANS は統計量名を分析変数名に追加してラベルを作成します。

参照項目: “例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

AUTONAME
OUTPUT ステートメントで変数名を割り当てない場合、PROC MEANS は出力統計量に対して一意の変数名を作成することを指定します。このアクションは、統計量の派生元入力変数名の終わりに *statistic-keyword* を追加して実行されます。

たとえば、ステートメント output min(x)=/autoname;では、出力データセットに x_Min 変数が生成されます。

AUTONAME は、SAS の内部メカニズムを使用して出力データセットの変数名の競合問題を自動的に解決します。変数が重複しても、エラーは発生しません。

その結果、ステートメント output min(x)= min(x)=/autoname;では、出力データセットに 2 つの変数 x_Min と x_Min2 が生成されます。

新しい変数名が 32 文字を超えると、variable-name 部分が切り捨てられます。

参照項目: “例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

KEEPLEN
出力データセットの統計量は、それらの派生に使用される分析変数の長さを継承することを指定します。

注意
分析変数の長さが原因で PROC MEANS が統計量の値を切り捨てるか丸めると、数値精度を永久に失います。ただし、統計量の精度は入力の精度と一致します。

LEVELS
出力データセットに_LEVEL_という名前の変数を含めます。この変数には、分類変数の値(_TYPE_変数の値)の一意の組み合わせを示す 1 から *n* までの値が含まれます。

参照項目: “出力データセット” (1290 ページ)

“例 8: 出力統計量を計算する” (1313 ページ)

NOINHERIT
統計量を含む出力データセットの変数は、それらを派生するときに使用される分析変数の属性(ラベルと出力形式)を継承しないことを指定します。

操作 オプションが使用されない場合(暗黙の INHERIT)、統計量は入力分析変数の属性、ラベル、出力形式を継承します。OUTPUT ステートメントで INHERIT オプションが使用されると、統計量は入力分析変数の長さ、ラベルと出力形式を継承します。

ヒント デフォルトでは、出力データセットに、各分析変数と、 N、MIN、MAX、MEAN、STDDEV を含む 5 つのオブザベーションに対する 1 つの出力変数が含まれます。NOINHERIT を指定しない場合、この変数は分析変数の出力形式を継承します。この出力形式は N 統計量に対して無効である場合があります(日時出力形式など)。

WAYS
出力データセットに_WAY_という名前の変数を含めます。この変数には、PROC MEANS が TYPE 値を作成するために組み合わせる分類変数の数を指定する、1 から分類変数の最大数までの値が 1 つ含まれます。

参照項目: “出力データセット” (1290 ページ)

“WAYS ステートメント” (1277 ページ)

“例 8: 出力統計量を計算する” (1313 ページ)

統計量キーワード

CLM
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

CSS
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

KURTOSIS
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

別名 KURT

LCLM
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

MAX
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

MEAN
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

MIN
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

MODE
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

N
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

NMISS
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

RANGE
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

SKEWNESS
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

別名 SKEW

STDDEV
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

別名 STD

STDERR
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

SUM
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

SUMWGT
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

UCLM
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

USS
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

VAR
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

MEDIAN
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

別名 P50

P1
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P5
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P10
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P20
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P40
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P70
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

Q1
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

別名 P25

Q3
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

別名 P75

P90
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P95
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P99
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P30
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (1264 ページ)を参照してください。

P60
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

P80
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

QRANGE
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

PROBT
仮説検定キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

別名 PBT

T
仮説検定キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (1264 ページ)を参照してください。

TYPES ステートメント

生成する分類変数の可能な組み合わせを決定します。

要件 CLASS ステートメント

参照項目: “出力データセット” (1290 ページ)

例: “例 2: 分類変数を使用した記述統計量の計算” (1294 ページ)
“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

構文

TYPES *request(s)*;

必須引数

request(s)
PROC MEANS が種類の作成に使用する分類変数の2^kの組み合わせを指定します。このkは分類変数の数です。要求には、1 つの分類変数名、アスタリスクまたは()で区切られた複数の分類変数名が含まれます。

分類変数の組み合わせをすばやく要求するには、複数の変数をかっこで囲み、他の変数または変数の組み合わせを結合してグループ化構文を使用します。たとえば、次のステートメントはグループ化構文を示しています。

要求	等しい
types A*(B C);	types A*B A*C;
types (A B)*(C D);	types A*C A*D B*C B*D;
types (A B C)*D;	types A*D B*D C*D;

操作 CLASSDATA=オプションは、NWAY 種類に制約を設定します。PROC MEANS は、他のすべての種類を結果の NWAY 種類から派生したように生成します。

ヒント ( )を使用して全体合計を要求します(_TYPE_=0)。

出力データセットですべての種類が必要でない場合、WHERE 句をデータセットに適用するのではなく、TYPES ステートメントを使用して特定のサブタイプを指定します。これを行うことにより、時間とコンピューターメモリが節約されます。

詳細

出力における分析の順序

分析は、PROC MEANS によって計算される、_TYPE_変数の値が増える順序で出力に書き込まれます。_TYPE_変数には、分類変数の各組み合わせに対し一意の値が含まれています。値は TYPES ステートメントではなく、CLASS ステートメントの指定方法によって決定されます。そのため、

```
class A B C;
types (A B)*C;
```

を指定すると、B*C 分析(_TYPE_=3)が最初に書き込まれ、A*C 分析(_TYPE_=5)が後に続きます。ただし、

```
class B A C;
types (A B)*C;
```

を指定すると、A*C 分析が最初に実行されます。

_TYPE_変数は、出力データセットが要求されなくても計算されます。_TYPE_変数の詳細については、“出力データセット” (1290 ページ)を参照してください。

VAR ステートメント

分析変数を識別し、出力における順序を識別します。

別名: VARS
VARIABLES

デフォルト: VAR ステートメントを省略すると、PROC MEANS はその他のステートメントでリストされないすべての数値変数を標準化します。すべての変数が文字変数の場合、PROC MEANS はオブザベーションの単純なカウントを生成します。

ヒント: 複数の VAR ステートメントを使用できます。

参照項目: 60 章, “SUMMARY プロシジャ,” (2141 ページ)

例: “例 1: 特定の記述統計量の計算” (1292 ページ)

構文

VAR *variable(s)* </ WEIGHT=*weight-variable*>;

必須引数

variable(s)
分析変数を識別し、結果での順序を識別します。

オプション引数

WEIGHT=*weight-variable*
その値が VAR ステートメントで指定される変数の値に重みを付ける数値変数を指定します。重み変数が整数である必要がないのは、次のテーブルに、PROC MEANS による WEIGHT 変数のさまざまな値の処理方法を説明します。

重み値	PROC MEANS 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 より小さい	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

重み変数は、プロシジャによる範囲、極値、欠損値数の決定方法を変更しません。

制限事項 重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。

歪度と尖度は、WEIGHT オプションと使用できません。

注 バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

ヒント WEIGHT オプションを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。VARDEF=の説明を参照してください。

複数の VAR ステートメントで WEIGHT オプションを使用して、分析変数に対し異なる重みを指定します。

WAYS ステートメント

分類変数の一意の組み合わせを作成するための方法の数を指定します。

ヒント: TYPES ステートメントを使用して、分類変数の追加の組み合わせを指定します。

例: “例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

構文

WAYS *list*;

必須引数

list
分類変数のすべての一意の組み合わせを作成するために組み合わせる分類変数の数を定義する 1 つ以上の整数を指定します。たとえば、可能なすべてのペアに対して 2 を指定し、可能なすべてのトリプルに対して 3 を指定できます。*list* は次の方法で指定できます。

- *m*
- *m1 m2 ... mn*
- *m1,m2,...,mn*
- *m* TO *n* <BY *increment*>
- *m1,m2* TO *m3* <BY *increment*>, *m4*

範囲 0 から分類変数の最大数

参照項目: WAYS オプション

例 次のコードは、分類変数 A、B、C に対する 2 次元の種類を作成する例です。この WAYS ステートメントは、TYPES ステートメントで a*b、a*c、b*c を指定する場合と同じです。

```
class A B C ; ways 2;
```

WEIGHT ステートメント

統計量計算の分析変数に対し重みを指定します。WEIGHT ステートメントと FREQ ステートメントは、両方のステートメントをサポートする任意のプロシジャの同一ステップ内で使用できます。

別名: WGT

制限事項: 重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。

歪度と尖度は、WEIGHT ステートメントと使用できません。

PROC MEANS は、重み変数がアクティブな場合、MODE を計算しません。かわりに、MODE の計算が必要で、重み変数がアクティブな場合、UNIVARIATE プロシジャを使用しようとします。

操作: VAR ステートメントで WEIGHT=オプションを使用して重み変数を指定すると、代わりに PROC MEANS はこの変数を使用してこれらの VAR ステートメント変数を重み付けます。

構文

WEIGHT *variable*;

必須引数

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

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

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

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

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

注意

単一の極値重み値により、不正確な結果が生成される可能性があります。 1 つの(唯一の)重み値がその他の重み値より桁違いに大きい場合(49 重み値が 1、1 重み値が 1×10^{14} など)、特定の統計量は可能な正しい制限内にないことがあります。影響を受ける統計量は、2 次モーメント(標準偏差、修正平方和、分散、平均値の標準誤差など)に基づいています。特定の状況下では、警告は SAS ログに書き込まれません。

プロシジャでは、w_iを WEIGHT 変数の値で置換します。これは、“キーワードと式” (2410 ページ)に記載されています。

例

WEIGHT ステートメントをサポートするプロシジャ

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

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

重み付き統計量の計算

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

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

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

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

重み付き統計量の使用

WEIGHT ステートメントの例として、30cm 幅の対象物のサイズを推定するよう 20 人に求めたとします。対象物からの距離が異なる場所にそれぞれを配置します。対象物からの距離が増加するにつれて、推定値の精度は低下します。SAS データセット SIZE には、メートル単位の各距離(Distance)におけるセンチメートル単位の推定値(ObjectSize)、および各推定値の精度(Precision)が含まれます。最大偏差(20cm の過大見積もり)が最長距離(対象物から 7.5 メートル)で発生していることに注意してください。精度(1/Distance)として、対象物に近い位置での推定値ほど重みを増やし、遠い位置での推定値ほど重みを減らします。PROC MEANS ステップでは、重みを無視して、物体サイズの平均推定値が計算されます。WEIGHT 変数がなければ、PROC MEANS では、すべてのオブザベーションに対してデフォルトの重み 1 が使用されます。したがって、すべての距離の物体サイズ推定値に、等しい重みが与えられます。物体サイズの平均推定値は、実際のサイズを 3.55cm 上回ります。

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

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

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

```
run;
```

出力

Unweighted Analysis of the SIZE Data Set

The MEANS Procedure

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

PROC MEANS で精度のある WEIGHT ステートメントを使用

次の 2 つの PROC MEANS ステップでは、WEIGHT ステートメントで精度 (Precision)を使用し、値の異なる VARDEF=オプションの使用効果を示します。最初の PROC ステップでは、分散と標準偏差を含む出力データセットが作成されます。遠い位置での推定値の重みを減らすと、物体サイズの重み付き平均推定値が実際のサイズに近づきます。*i* 番目のオブザベーションの分散が$var(x_i) = \sigma^2/w_i$で、w_iは *i* 番目のオブザベーションの重みです。最初の PROC MEANS ステップでは、計算された分散はσ^2の推定値です。2 番目の PROC MEANS ステップでは、計算された分散は$(n-1/n)\,\sigma^2/\overline{w}$の推定値です。このとき、$\overline{w}$は平均重みです。n が大きい場合、この値は平均重み付きのオブザベーションの分散の概算推定値です。

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

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

出力

Weighted Analysis Using Default VARDEF=DF

The MEANS Procedure

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

Weighted Analysis Using VARDEF=WEIGHT

The MEANS Procedure

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

各オブザベーション値の重み付き分散と重み付き標準偏差を使用してデータセットを作成する

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

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

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

出力

Weighted Statistics

Distance=1.5

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

Distance=3

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

Distance=4.5

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

Distance=6

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

Distance=7.5

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

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント　次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

使用法: MEANS プロシジャ

モーメント統計量の計算

PROC MEANS は、モーメント統計量(平均値、分散、歪度、尖度など)の計算にシングルパスアルゴリズムを使用します。統計式については、“キーワードと式” (2410 ページ)を参照してください。

信頼限界、仮説検定統計量、四分位範囲統計量に関する計算詳細について、次に示します。

信頼限界

キーワード CLM、LCLM、UCLM を使用して、平均値に対する信頼限界を計算できます。信頼限界はサンプル統計量の値の周囲に構成される範囲で、反復サンプリング時に指定される確率(ALPHA=)を含む対応する真の母集団値が含まれます。

平均値に対する両側$100(1-\alpha)$%信頼区間には、上限と下限があります。

$$\overline{x} \pm t_{(1-\alpha/2;n-1)}\frac{s}{\sqrt{n}}$$

ここで、sは$\sqrt{\frac{1}{n-1}\Sigma\left(x_i-\overline{x}\right)^2}$であり、$t_{(1-\alpha/2;n-1)}$は自由度が$n-1$であるスチューデントの t 統計量の$(1-\alpha/2)$ 限界値です。

片側$100(1-\alpha)$%信頼区間は、次のように計算されます。

$$\overline{x} + t_{(1-\alpha;n-1)}\frac{s}{\sqrt{n}} \quad \text{(upper)}$$

$$\overline{x} - t_{(1-\alpha;n-1)}\frac{s}{\sqrt{n}} \quad \text{(lower)}$$

WEIGHT ステートメント、または VAR ステートメントで WEIGHT=を使用し、VARDEF=のデフォルト値、DF を使用する場合、重み付き平均に対する$100(1-\alpha)$ %信頼区間には、上限と下限があります。

$$\overline{y}_w \pm t_{(1-\alpha/2)}\frac{s_w}{\sqrt{\sum_{i=1}^{n} w_i}}$$

ここで、$\overline{y}_w$加重平均です、s_wは加重標準偏差であり、w_iの重量ですithオブザベーション、そして$t_{(1-\alpha/2)}(1-\alpha/2)$自由度が$n-1$のスチューデントの t 分布の臨界値です。

スチューデントの t 検定

PROC MEANS は、t 統計量を次のように計算します。

$$t = \frac{\overline{x}-\mu_0}{s/\sqrt{n}}$$

ここで、$\overline{x}$はサンプルの平均、nは、変数の非欠損値の数、そしてsはサンプルの標準偏差です。帰無仮説では、母集団の平均値はμ_0です。データ値が正規分布に近似している場合、観測値(*p* 値)と同じ、またはそれ以上の極値である t 統計量の帰無仮説での確率は、t 分布(自由度が$n-1$)から取得されます。大きなnの場合、t 統計量は漸近的に z 検定と等しくなります。

WEIGHT ステートメント、または VAR ステートメントで WEIGHT=を使用し、VARDEF=のデフォルト値、DF を使用する場合、スチューデントの t 統計量は次のように計算されます。

$$t_w = \frac{\bar{y}_w - \mu_0}{s_w / \sqrt{\sum_{i=1}^{n} w_i}}$$

ここで、$\bar{y}_w$は重み付き平均、s_wは重み付き標準偏差、そしてw_iはithオブザベーションに対する重みになります。t_w統計量は、スチューデントの t 分布と$n-1$自由度を含むものとして扱われます。PROC ステートメントで EXCLNPWGT オプションを指定すると、nは、WEIGHT 変数の値が正の場合の非欠損オブザベーション数です。デフォルトでは、nが、WEIGHT 変数に対する非欠損オブザベーション数です。

分位点

オプション QMETHOD=、QNTLDEF=、QMARKERS=により、PROC MEANS の分位点計算方法が決定されます。QNTLDEF=は、分位点の数学的定義を扱います。“分位数と関連統計量” (2415 ページ)を参照してください。QMETHOD=は、PROC MEANS の入力データ処理方法の手法を扱います。次の 2 つの方法があります。

OS
: すべてのデータをメモリに読み込み、一意の値別に並べ替えます。

P2
: すべてのデータを分位点の概算に使用される固定サンプルサイズに累積します。

データセット A に数値変数 X に対する一意の値が 100、データセット B に数値変数 X に対する一意の値が 1000 含まれている場合、データセット B の QMETHOD=OS には、データセット A のものよりも 10 倍のメモリが必要になります。QMETHOD=P2 の場合、A と B の両方のデータセットには、分位点の生成に同じメモリ空間が必要です。

QMETHOD=P2 の手法は、Jain と Chlamtac (1985)によって考案された piecewise-parabolic (P^2)アルゴリズムに基づいています。P^2は、大きなデータセットに対し分位点を決定するためのワンパスアルゴリズムです。種類内の各水準の各変数に対し、一定量のメモリが必要です。ただし、シミュレーションスタディを 使用した、一部の分位点(P1、P5、P95、P99)の信頼性のある推定は、裾の重い分布または歪度が高い分布を含むデータセットなど、一部のデータセットには可能でないことがあります。

オブザベーション数が QMARKERS=値未満である場合、QMETHOD=P2 の結果は QNTLDEF=5 のときの QMETHOD=OS と同じになります。重み付き分位点を計算するには、QMETHOD=OS を使用する必要があります。

結果: MEANS プロシジャ

欠損値

PROC MEANS は、統計量を計算する前に分析変数の欠損値を除外します。分析変数はそれぞれ個別に処理されます。1 つの変数のオブザベーションに対する欠損値はその他の変数の計算に影響しません。ステートメントは、欠損値を次のように処理します。

- 分類変数にオブザベーションに対する欠損値が含まれている場合、PROC ステートメントまたは CLASS ステートメントで MISSING オプションを使用しない限り、PROC MEANS はそのオブザベーションを分析から除外します。
- BY 変数値または ID 変数値が欠損している場合、PROC MEANS はそれをその他の BY 変数値または ID 変数値と同様に処理します。欠損値により、別の BY グループが形成されます。
- FREQ 変数値が欠損している、または非正の場合、PROC MEANS はそのオブザベーションを分析から除外します。
- WEIGHT 変数値が欠損している場合、PROC MEANS はそのオブザベーションを分析から除外します。

PROC MEANS は、欠損値の数を表形式で示します。欠損値の数を表形式で示す前に、PROC MEANS は FREQ ステートメントの使用時に度数が非正のオブザベーションと、WEIGHT ステートメントの使用時(EXCLNPWGT オプションの使用時)に重みが欠損している、または非正のオブザベーションを除外します。この情報をプロシジャ出力にレポートするには、PROC ステートメントで NMISS 統計量キーワードを使用します。

出力の列幅

PROC ステートメントで FW=オプションを使用して、表示される統計量の列幅を制御します。出力形式を数値分類変数または ID 変数に割り当てない限り、PROC MEANS は FW=オプションの値を使用します。出力形式を数値分類変数または ID 変数に割り当てると、PROC MEANS は列幅を出力形式から直接決定します。CLASS ステートメントで PRELOADFMT オプションを使用すると、PROC MEANS は割り当てられている出力形式から分類変数の列幅を決定します。

N Obs 統計量

デフォルトで CLASS ステートメントを使用すると、PROC MEANS は N Obs という追加の統計量を表示します。この統計量は、オブザベーションの合計数、または PROC MEANS が分類水準ごとに処理する FREQ 変数のオブザベーションの合計をレポートします。PROC MEANS は、オブザベーションをこの合計から省略することがあります。1 つ以上の分類変数に欠損値があるため、あるいは EXCLUSIVE オプションを PRELOADFMT オプションまたは CLASSDATA=オプションと使用する場合の EXCLUSIVE オプションの影響のためです。このアクションと、WEIGHT 変数に欠損値が含まれている場合のオブザベーションの除外により、N Obs、N および NMISS の間に常に直接関係があるわけではありません。

出力データセットでは、N Obs の値は_FREQ_変数に格納されます。PROC ステートメントで NONOBS オプションを使用して、表示出力でこの情報を非表示にします。

出力データセット

PROC MEANS は、1 つ以上の出力データセットを作成できます。プロシジャは出力データセットを印刷しません。PROC PRINT、PROC REPORT または別の SAS レポートツールを使用して、出力データセットを表示します。

注: デフォルトでは、出力データセットの統計量は、分析変数の出力形式とラベルを自動的に継承します。ただし、N、NMISS、SUMWGT、USS、CSS、VAR、CV、T、PROBT、PRT、SKEWNESS および KURTOSIS に対して計算される統計量は、分析変数の出力形式を継承しません。この出力形式がこれらの統計量に無効である可能性があるためです。OUTPUT ステートメントで NOINHERIT オプションを使用して、その他の統計量が出力形式とラベル属性を継承しないようにします。

出力データセットには、次の変数が含まれている可能性があります。

- BY ステートメントで指定された変数。
- ID ステートメントで指定された変数。
- CLASS ステートメントで指定された変数。
- 分類変数に関する情報が含まれている変数_TYPE_.デフォルトでは、_TYPE_は数値変数です。PROC ステートメントで CHARTYPE を指定する場合、_TYPE_は文字変数になります。32 を超える分類変数を使用する場合、_TYPE_は自動的に文字変数になります。
- 指定出力水準が表すオブザベーション数を含む変数_FREQ_。
- 出力統計量と極値が含まれている OUTPUT ステートメントで要求される変数。
- 統計量キーワードを省略する場合にデフォルトの統計量の名前が含まれている変数_STAT_。
- LEVEL オプションを指定する場合の変数_LEVEL_。

- WAYS オプションを指定する場合の変数_WAY_。

_TYPE_の値は、PROC MEANS が統計量の計算に使用する分類変数の組み合わせを示します。_TYPE_の文字値は一連のゼロと 1 です。1 の値はそれぞれ種類のアクティブな分類変数を示します。たとえば、3 つの分類変数を使用して、PROC MEANS は種類 1 を 001、種類 5 を 101 として表します。

通常、出力データセットには、種類別の水準ごとに 1 つのオブザベーションが含まれています。ただし、OUTPUT ステートメントで統計量キーワードを省略すると、出力データセットには水準ごとに 5 つ(WEIGHT 変数を指定する場合は 6 つ)のオブザベーションが含まれます。そのため、出力データセットのオブザベーションの合計数は、適用可能な場合に 1、5 または 6 を乗算して求めるすべての種類の水準の合計に等しくなります。

CLASS ステートメント(_TYPE_= 0)を省略すると、BY グループごとに出力水準が 1 つだけ常に存在します。CLASS ステートメントを使用すると、要求する各種類の水準数の上限は入力データセットのオブザベーション数と同じになります。デフォルトでは、PROC MEANS はすべての 可能な種類を生成します。この場合、各 BY グループの水準の合計数の上限は、次と同じになります。

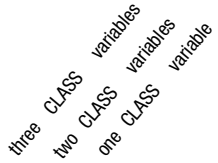
$$m \cdot \left(2^k - 1\right) \cdot n + 1$$

ここで、kは分類変数の数、nは入力データセットの指定 BY グループに対するオブザベーション数、そしてmは 1、5 または 6 です。

PROC MEANS は、アクティブな各分類変数の一意の組み合わせ数から、指定種類に対する水準の実際の数を決定します。単一の水準は、フォーマットされた分類値が一致するすべての 入力オブザベーションから構成されます。

次の図に、分類変数を 1 つ、2 つ、3 つ指定したときの_TYPE_の値とデータセットのオブザベーション数を示します。

図 35.1 OUTPUT データセットにおける分類変数の影響



C	B	A	_WAY_	_TYPE_	Subgroup defined by	Number of observations of this _TYPE_ and _WAY_ in the data set	Total number of observations in the data set
0	0	0	0	0	Total	1	
0	0	1	1	1	A	a	1+a
0	1	0	1	2	B	b	
0	1	1	2	3	A*B	a*b	1+a+b+a*b
1	0	0	1	4	C	c	
1	0	1	2	5	A*C	a*c	
1	1	0	2	6	B*C	b*c	1+a+b+a*b+c
1	1	1	3	7	A*B*C	a*b*c	+a*c+b*c+a*b*c
Character equivalent of binary _TYPE_ (CHARTYPE option)					A, B, C=CLASS variables	a, b, c,=number of levels of A, B, C, respectively	

例: MEANS プロシジャ

例 1: 特定の記述統計量の計算

要素: PROC MEANS ステートメントオプション
統計量キーワード
FW=
VAR ステートメント

データセット: CAKE

詳細

この例では、次を行います。

- 分析変数を指定します。
- 指定キーワードに対する統計量を計算し、順番に表示します。
- 統計量のフィールド幅を指定します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data cake;
  input LastName $ 1-12 Age 13-14 PresentScore 16-17
      TasteScore 19-20 Flavor $ 23-32 Layers 34 ;
  datalines;
Orlando     27 93 80  Vanilla    1
Ramey       32 84 72  Rum        2
Goldston    46 68 75  Vanilla    1
Roe         38 79 73  Vanilla    2
Larsen      23 77 84  Chocolate  .
Davis       51 86 91  Spice      3
Strickland  19 82 79  Chocolate  1
Nguyen      57 77 84  Vanilla    .
Hildenbrand 33 81 83  Chocolate  1
Byron       62 72 87  Vanilla    2
Sanders     26 56 79  Chocolate  1
```

```
Jaeger     43 66 74          1
Davis      28 69 75  Chocolate  2
Conrad      69 85 94  Vanilla   1
Walters    55 67 72  Chocolate  2
Rossburger  28 78 81  Spice     2
Matthew    42 81 92  Chocolate  2
Becker     36 62 83  Spice     2
Anderson   27 87 85  Chocolate  1
Merritt    62 73 84  Chocolate  1
;

proc means data=cake n mean max min range std fw=8;

  var PresentScore TasteScore;

  title 'Summary of Presentation and Taste Scores';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=60;
```

CAKE データセットを作成します。 CAKE には、ケーキ作りコンテストの次のデータが含まれています。各参加者の姓、年齢、出来ばえのスコア、味のスコア、ケーキの風味、ケーキの層の数。ケーキの層の数が 2 つのオブザベーションにありません。ケーキの風味が別のオブザベーションにありません。

```
data cake;
  input LastName $ 1-12 Age 13-14 PresentScore 16-17
      TasteScore 19-20 Flavor $ 23-32 Layers 34 ;
  datalines;
Orlando    27 93 80  Vanilla   1
Ramey      32 84 72  Rum       2
Goldston   46 68 75  Vanilla   1
Roe        38 79 73  Vanilla   2
Larsen     23 77 84  Chocolate .
Davis      51 86 91  Spice     3
Strickland  19 82 79  Chocolate  1
Nguyen     57 77 84  Vanilla   .
Hildenbrand 33 81 83  Chocolate  1
Byron      62 72 87  Vanilla   2
Sanders    26 56 79  Chocolate  1
Jaeger     43 66 74          1
Davis      28 69 75  Chocolate  2
Conrad      69 85 94  Vanilla   1
Walters    55 67 72  Chocolate  2
Rossburger  28 78 81  Spice     2
Matthew    42 81 92  Chocolate  2
Becker     36 62 83  Spice     2
Anderson   27 87 85  Chocolate  1
Merritt    62 73 84  Chocolate  1
;
```

分析と分析オプションを指定します。 統計量キーワードは、統計量と出力での順序を指定します。FW=は、フィールド幅 8 を使用して統計量を表示します。

```
proc means data=cake n mean max min range std fw=8;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が PresentScore 変数と TasteScore 変数の統計量を計算するように指定します。

```
var PresentScore TasteScore;
```

タイトルを指定します。

```
title 'Summary of Presentation and Taste Scores';
run;
```

例 2: 分類変数を使用した記述統計量の計算

要素: PROC MEANS ステートメントオプション
MAXDEC=
CLASS ステートメント
TYPES ステートメント

データセット: GRADE

詳細

この例では、次を行います。

- 分類変数の 2 次元組み合わせのデータとすべてのオブザベーションにわたるデータを分析します。
- 表示される統計量に対する小数点以下桁数を制限します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data grade;
  input Name $ 1-8 Gender $ 11 Status $13 Year $ 15-16
      Section $ 18 Score 20-21 FinalGrade 23-24;
  datalines;
Abbott   F 2 97 A 90 87
Branford  M 1 98 A 92 97
Crandell  M 2 98 B 81 71
Dennison  M 1 97 A 85 72
Edgar    F 1 98 B 89 80
Faust    M 1 97 B 78 73
Greeley   F 2 97 A 82 91
Hart     F 1 98 B 84 80
```

```
Isley     M 2 97 A 88 86
Jasper    M 1 97 B 91 93
;

proc means data=grade maxdec=3;

  var Score;

  class Status Year;

  types () status*year;

  title 'Final Exam Grades for Student Status and Year of Graduation';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=60;
```

GRADE データセットを作成します。 GRADE には、各生徒の姓、性別、大学生(1)または大学院生(2)のいずれかの区分、卒業見込年度、クラスセクション(A または B)、期末試験のスコア、クラスの最終成績が含まれています。

```
data grade;
  input Name $ 1-8 Gender $ 11 Status $13 Year $ 15-16
       Section $ 18 Score 20-21 FinalGrade 23-24;
  datalines;
Abbott    F 2 97 A 90 87
Branford  M 1 98 A 92 97
Crandell  M 2 98 B 81 71
Dennison  M 1 97 A 85 72
Edgar     F 1 98 B 89 80
Faust     M 1 97 B 78 73
Greeley   F 2 97 A 82 91
Hart      F 1 98 B 84 80
Isley     M 2 97 A 88 86
Jasper    M 1 97 B 91 93
;
```

デフォルトの統計量を生成して、分析オプションを指定します。 PROC MEANS ステートメントで統計量が指定されていないため、すべてのデフォルトの統計量(N、MEAN、STD、MIN、MAX)が生成されます。MAXDEC=は、表示される統計量を小数点以下 3 桁に制限します。

```
proc means data=grade maxdec=3;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が Score 変数の統計量を計算するように指定します。

```
  var Score;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは分析をサブグループに分割します。Status と Year の一意の値の組み合わせはそれぞれサブグループを表します。

```
    class Status Year;
```

分析するサブグループを指定します。 TYPES ステートメントは、GRADE データセットのすべてのオブザベーション、および結果が 4 つのサブグループとなる(Status と Year にはそれぞれ 2 つの一意の値が含まれているため)Status と Year の 2 次元組み合わせで分析が実行されるように要求します。

```
    types () status*year;
```

タイトルを指定します。

```
    title 'Final Exam Grades for Student Status and Year of Graduation';
run;
```

出力

PROC MEANS は、すべてのオブザベーション(_TYPE_=0)のデフォルト統計量および Status と Year の組み合わせの 4 つの分類水準(Status=1, Year=97; Status=1, Year=98; Status=2, Year=97; Status=2, Year=98)を表示します。

アウトプット 35.4　期末試験の成績

Final Exam Grades for Student Status and Year of Graduation

The MEANS Procedure

Analysis Variable : Score					
N Obs	N	Mean	Std Dev	Minimum	Maximum
10	10	86.000	4.714	78.000	92.000

Analysis Variable : Score							
Status	Year	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	97	3	3	84.667	6.506	78.000	91.000
	98	3	3	88.333	4.041	84.000	92.000
2	97	3	3	86.667	4.163	82.000	90.000
	98	1	1	81.000	.	81.000	81.000

例 3: BY ステートメントを分類変数とともに使用する

要素:　PROC MEANS ステートメントオプション
　　統計量キーワード
　BY ステートメント

CLASS ステートメント
SORT プロシジャ

データセット: GRADE

詳細

この例では、次を行います。

- BY 値内の分類変数の組み合わせに対する分析を分割する
- BY ステートメントに対する並べ替え順序要件を表示する
- 最小値、最大値、中央値を計算する

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc sort data=Grade out=GradeBySection;
  by section;
run;

proc means data=GradeBySection min max median;

  by Section;

  var Score;

  class Status Year;

  title1 'Final Exam Scores for Student Status and Year of Graduation';
  title2 ' Within Each Section';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=60;
```

GRADE データセットを並べ替えます。 PROC SORT はオブザベーションを変数 Section 別に並べ替えます。並べ替えは、Section を BY 変数として PROC MEANS ステップで使用するために必要です。

```
proc sort data=Grade out=GradeBySection;
  by section;
run;
```

分析を指定します。 統計量キーワードは、統計量と出力での順序を指定します。

```
proc means data=GradeBySection min max median;
```

データセットを BY グループに分割します。 BY ステートメントは、Section の各値に対し別の分析を生成します。

```
by Section;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が Score 変数の統計量を計算するように指定します。

```
var Score;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Status と Year の値別に分割します。このプログラムには TYPES ステートメントがないため、分析は各 BY グループ内の各サブグループに対して実行されます。

```
class Status Year;
```

タイトルを指定します。

```
title1 'Final Exam Scores for Student Status and Year of Graduation';
title2 ' Within Each Section';
run;
```

出力

アウトプット 35.5　期末試験のスコア

Final Exam Scores for Student Status and Year of Graduation Within Each Section

The MEANS Procedure

Section=A

Analysis Variable : Score					
Status	Year	N Obs	Minimum	Maximum	Median
1	97	1	85.0000000	85.0000000	85.0000000
	98	1	92.0000000	92.0000000	92.0000000
2	97	3	82.0000000	90.0000000	88.0000000

Section=B

Analysis Variable : Score					
Status	Year	N Obs	Minimum	Maximum	Median
1	97	2	78.0000000	91.0000000	84.5000000
	98	2	84.0000000	89.0000000	86.5000000
2	98	1	81.0000000	81.0000000	81.0000000

例 4: CLASSDATA=データセットを分類変数とともに使用する

要素:

PROC MEANS ステートメントオプション
- CLASSDATA=
- EXCLUSIVE
- FW=
- MAXDEC=
- PRINTALLTYPES

CLASS ステートメント

データセット:

CAKE

CAKETYPE

詳細

この例では、次を行います。

- 表示される統計量のフィールド幅と小数点以下桁数を指定します。
- CLASSDATA=データセットの値のみを分類変数の組み合わせの水準として使用します。
- 範囲、中間値、最小値、最大値を計算します。
- 分析に分類変数のすべての組み合わせを表示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data caketype;
  input Flavor $ 1-10  Layers 12;
  datalines;
Vanilla    1
Vanilla    2
Vanilla    3
Chocolate  1
Chocolate  2
Chocolate  3
;

proc means data=cake range median min max fw=7 maxdec=0
        classdata=caketype exclusive printalltypes;

  var TasteScore;


  class flavor layers;

  title 'Taste Score For Number of Layers and Cake Flavor';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=60;
```

CAKETYPE データセットを作成します。 CAKETYPE には、PROC MEANS 出力に表示する必要があるケーキの風味と層の数が含まれています。

```
data caketype;
  input Flavor $ 1-10  Layers 12;
```

```
  datalines;
Vanilla   1
Vanilla   2
Vanilla   3
Chocolate 1
Chocolate 2
Chocolate 3
;
```

分析と分析オプションを指定します。 FW=オプションはフィールド幅 7、MAXDEC=オプションは小数点以下桁数ゼロをそれぞれ使用して統計量を表示します。CLASSDATA=と EXCLUSIVE は、分類水準を CAKETYPE データセットにある値に制限します。PRINTALLTYPES は、出力に分類変数のすべての組み合わせを表示します。

```
proc means data=cake range median min max fw=7 maxdec=0
      classdata=caketype exclusive printalltypes;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が TasteScore 変数の統計量を計算するように指定します。

```
  var TasteScore;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Flavor と Layers の値別に分割します。これらの変数、およびこれらの変数のみが CAKETYPE データセットに表示される必要があります。

```
  class flavor layers;
```

タイトルを指定します。

```
  title 'Taste Score For Number of Layers and Cake Flavor';
run;
```

出力

PROC MEANS は、13 個のチョコレートケーキとバニラケーキに対する統計量を計算します。CLASSDATA=データセットに Layers の値として 3 が含まれているため、PROC MEANS は度数がゼロの場合でも 3 を分類値として使用します。

アウトプット 35.6　味のスコア

Taste Score For Number of Layers and Cake Flavor

The MEANS Procedure

Analysis Variable : TasteScore				
N Obs	Range	Median	Minimum	Maximum
13	22	80	72	94

Analysis Variable : TasteScore					
Layers	N Obs	Range	Median	Minimum	Maximum
1	8	19	82	75	94
2	5	20	75	72	92
3	0	.	.	.	.

Analysis Variable : TasteScore					
Flavor	N Obs	Range	Median	Minimum	Maximum
Chocolate	8	20	81	72	92
Vanilla	5	21	80	73	94

Analysis Variable : TasteScore						
Flavor	Layers	N Obs	Range	Median	Minimum	Maximum
Chocolate	1	5	6	83	79	85
	2	3	20	75	72	92
	3	0	.	.	.	.
Vanilla	1	3	19	80	75	94
	2	2	14	80	73	87
	3	0	.	.	.	.

例 5: 値のマルチラベル出力形式を分類変数とともに使用する

要素:　PROC MEANS ステートメントオプション
- 統計量キーワード
- FW=
- NONOBS

CLASS ステートメントオプション
MLF
ORDER=
TYPES ステートメント
FORMAT プロシジャ
FORMAT ステートメント

データセット: CAKE

詳細

この例では、次を行います。

- 指定キーワードに対する統計量を計算し、順番に表示します。
- 統計量のフィールド幅を指定します。
- オブザベーションの合計数を含む列を非表示にします。
- ケーキの風味の 1 次元組み合わせと、ケーキの風味と参加者の年齢の 2 次元組み合わせに対しデータを分析します。
- ユーザー定義出力形式を分類変数に割り当てます。
- マルチラベル出力形式を分類変数の水準として使用します。
- ケーキの風味の水準を度数カウントの降順で並べ替え、年齢の水準をフォーマットされた値の昇順で並べ替えます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=64;

proc format;
  value $flvrfmt
            'Chocolate'='Chocolate'
            'Vanilla'='Vanilla'
            'Rum','Spice'='Other Flavor';
  value agefmt (multilabel)
              15 - 29='below 30 years'
              30 - 50='between 30 and 50'
              51 - high='over 50 years'
              15 - 19='15 to 19'
              20 - 25='20 to 25'
              25 - 39='25 to 39'
              40 - 55='40 to 55'
              56 - high='56 and above';
run;

proc means data=cake fw=6 n min max median nonobs;

  class flavor/order=data;
  class  age /mlf order=fmt;
```

```
 types flavor flavor*age;
 var TasteScore;
 format age agefmt. flavor $flvrfmt.;
 title 'Taste Score for Cake Flavors and Participant''s Age';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=64;
```

$FLVRFMT.出力形式と AGEFMT.出力形式を作成します。 PROC FORMAT は、ユーザー定義出力形式を作成して、ケーキの風味と参加者の年齢を分類します。MULTILABEL は、Age に対しマルチラベル出力形式を作成します。マルチラベル出力形式は、複数のラベルを同じ値に割り当てることができる出力形式です。この場合は、範囲が重複するためです。各値は、発生する範囲ごとに出力に表示されます。

```
proc format;
  value $flvrfmt
          'Chocolate'='Chocolate'
          'Vanilla'='Vanilla'
          'Rum','Spice'='Other Flavor';
  value agefmt (multilabel)
            15 - 29='below 30 years'
            30 - 50='between 30 and 50'
            51 - high='over 50 years'
            15 - 19='15 to 19'
            20 - 25='20 to 25'
            25 - 39='25 to 39'
            40 - 55='40 to 55'
            56 - high='56 and above';
run;
```

分析と分析オプションを指定します。 FW=は、フィールド幅 6 を使用して統計量を表示します。統計量キーワードは、統計量と出力での順序を指定します。NONOBS は、N Obs 列を非表示にします。

```
proc means data=cake fw=6 n min max median nonobs;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Flavor と Age の値別に分割します。ORDER=DATA は、値を入力データセットの順序に従って並べ替えます。ORDER=FMT は、Age の水準をフォーマットされた値の昇順で並べ替えます。MLF は、マルチラベル値出力形式が Age に使用されるように指定します。

```
 class flavor/order=data;
 class  age /mlf order=fmt;
```

分析するサブグループを指定します。 TYPES ステートメントは、Flavor の 1 次元の組み合わせと、Flavor と Age の 2 次元の組み合わせに対し分析を要求します。

```
 types flavor flavor*age;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が TasteScore 変数の統計量を計算するように指定します。

```
var TasteScore;
```

出力をフォーマットします。 FORMAT ステートメントは、ユーザー定義出力形式をこの分析の対象となる Age 変数と Flavor 変数に割り当てます。

```
format age agefmt. flavor $flvrfmt.;
```

タイトルを指定します。

```
title 'Taste Score for Cake Flavors and Participant''s Age';
run;
```

出力

分類変数の 1 次元の組み合わせは、2 次元の組み合わせの前に表示されます。フィールド幅 6 は、統計量を小数点以下 4 桁に切り捨てます。Age と Flavor の 2 次元の組み合わせの場合、オブザベーションの合計数は Flavor の 1 次元の組み合わせよりも大きくなります。このような状況が発生するのは、1 つの内部値を複数のフォーマットされた値にマップする Age のマルチラベル出力形式のためです。Flavor の水準の順序は、各水準の度数カウントに基づいています。Age の水準の順序は、ユーザー定義出力形式の順序に基づいています。

アウトプット 35.7　ケーキの風味別の味のスコア

Taste Score for Cake Flavors and Participant's Age

The MEANS Procedure

Analysis Variable : TasteScore				
Flavor	N	Minimum	Maximum	Median
Vanilla	6	73.00	94.00	82.00
Other Flavor	4	72.00	91.00	82.00
Chocolate	9	72.00	92.00	83.00

Analysis Variable : TasteScore					
Flavor	Age	N	Minimum	Maximum	Median
Vanilla	25 to 39	2	73.00	80.00	76.50
	40 to 55	1	75.00	75.00	75.00
	56 and above	3	84.00	94.00	87.00
	below 30 years	1	80.00	80.00	80.00
	between 30 and 50	2	73.00	75.00	74.00
	over 50 years	3	84.00	94.00	87.00
Other Flavor	25 to 39	3	72.00	83.00	81.00
	40 to 55	1	91.00	91.00	91.00
	below 30 years	1	81.00	81.00	81.00
	between 30 and 50	2	72.00	83.00	77.50
	over 50 years	1	91.00	91.00	91.00
Chocolate	15 to 19	1	79.00	79.00	79.00
	20 to 25	1	84.00	84.00	84.00
	25 to 39	4	75.00	85.00	81.00
	40 to 55	2	72.00	92.00	82.00
	56 and above	1	84.00	84.00	84.00
	below 30 years	5	75.00	85.00	79.00
	between 30 and 50	2	83.00	92.00	87.50
	over 50 years	2	72.00	84.00	78.00

例 6: 事前に読み込まれた出力形式を分類変数とともに使用する

要素:

PROC MEANS ステートメントオプション
- COMPLETETYPES
- FW=
- MISSING
- NONOBS

CLASS ステートメントオプション
- EXCLUSIVE
- ORDER=
- PRELOADFMT

WAYS ステートメント

FORMAT プロシジャ

FORMAT ステートメント

データセット: CAKE

詳細

この例では、次を行います。

- 統計量のフィールド幅を指定します。
- オブザベーションの合計数を含む列を非表示にします。
- 度数がゼロの場合でも、分類変数値のすべての 可能な組み合わせを分析に含めます。
- 欠損値を有効な分類水準とします。
- 分類変数の 1 次元および 2 次元の組み合わせを分析します。
- ユーザー定義出力形式を分類変数に割り当てます。
- ユーザー定義出力形式のすでに読み込まれている範囲のみ分類変数の水準として使用します。
- 結果をフォーマットされたデータの値別に並べ替えます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=64;

proc format;
  value layerfmt 1='single layer'
```

```
            2-3='multi-layer'
            .='unknown';
  value $flvrfmt (notsorted)
            'Vanilla'='Vanilla'
            'Orange','Lemon'='Citrus'
            'Spice'='Spice'
            'Rum','Mint','Almond'='Other Flavor';
run;

proc means data=cake fw=7 completetypes missing nonobs;
  class flavor layers/preloadfmt exclusive order=data;
  ways 1 2;
  var TasteScore;
  format layers layerfmt. flavor $flvrfmt.;
  title 'Taste Score For Number of Layers and Cake Flavors';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=64;
```

**LAYERFMT.出力形式と$FLVRFMT.出力形式を作成します。** PROC FORMAT は、ユーザー定義出力形式を作成して、ケーキの層の数とケーキの風味を分類します。NOTSORTED は$FLVRFMT を並べ替えず、出力形式値の元の順序を維持します。

```
proc format;
  value layerfmt 1='single layer'
            2-3='multi-layer'
            .='unknown';
  value $flvrfmt (notsorted)
            'Vanilla'='Vanilla'
            'Orange','Lemon'='Citrus'
            'Spice'='Spice'
            'Rum','Mint','Almond'='Other Flavor';
run;
```

デフォルトの統計量を生成して、分析オプションを指定します。 FW=は、フィールド幅 7 を使用して統計量を表示します。COMPLETETYPES には、度数がゼロの分類水準が含まれています。MISSING は、欠損値をすべての分類変数に有効な値とみなします。NONOBS は、N Obs 列を非表示にします。特定の分析が要求されないため、すべてのデフォルトの分析がすべて実行されます。

```
proc means data=cake fw=7 completetypes missing nonobs;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Flavor と Layers の値別に分割します。PRELOADFMT と EXCLUSIVE は、水準をユーザー定義出力形式のすでに読み込まれた値に制限します。ORDER=DATA は、Flavor と Layer の水準をフォーマットされたデータ値別に並べ替えます。

```
  class flavor layers/preloadfmt exclusive order=data;
```

分析するサブグループを指定します。 WAYS ステートメントは、分析変数の 1 次元および 2 次元の組み合わせを要求します。

```
ways 1 2;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が TasteScore 変数の統計量を計算するように指定します。

```
var TasteScore;
```

出力をフォーマットします。 FORMAT ステートメントは、ユーザー定義出力形式をこの分析の対象となる Flavor 変数と Layers 変数に割り当てます。

```
format layers layerfmt. flavor $flvrfmt.;
```

タイトルを指定します。

```
title 'Taste Score For Number of Layers and Cake Flavors';
run;
```

出力

分類変数の 1 次元の組み合わせは、2 次元の組み合わせの前に表示されます。PROC MEANS は、オブザベーションの度数がゼロ(この場合はかんきつ類)の場合でも、ユーザー定義出力形式のすでに読み込まれた範囲に表示される水準値のみレポートします。PROC MEANS は、指定オブザベーションの単一の分類値の除外に基づいてオブザベーション全体をリジェクトします。そのため、層の数が不明な場合、統計量は 1 つのオブザベーションについてのみ計算されます。チョコレート風味が Flavor 用のすでに読み込まれているユーザー定義出力形式に含まれていないため、その他のオブザベーションは除外されます。水準の順序は、ユーザー定義出力形式の順序に基づいています。PROC FORMAT は自動的に Layers 出力形式を並べ替え、Flavor 出力形式を並べ替えませんでした。

アウトプット 35.8　層の数別の味のスコア

Taste Score For Number of Layers and Cake Flavors

The MEANS Procedure

Analysis Variable : TasteScore					
Layers	N	Mean	Std Dev	Minimum	Maximum
unknown	1	84.000	.	84.000	84.000
single layer	3	83.000	9.849	75.000	94.000
multi-layer	6	81.167	7.548	72.000	91.000

Analysis Variable : TasteScore					
Flavor	N	Mean	Std Dev	Minimum	Maximum
Vanilla	6	82.167	7.834	73.000	94.000
Citrus	0	.	.	.	.
Spice	3	85.000	5.292	81.000	91.000
Other Flavor	1	72.000	.	72.000	72.000

Analysis Variable : TasteScore						
Flavor	Layers	N	Mean	Std Dev	Minimum	Maximum
Vanilla	unknown	1	84.000	.	84.000	84.000
	single layer	3	83.000	9.849	75.000	94.000
	multi-layer	2	80.000	9.899	73.000	87.000
Citrus	unknown	0	.	.	.	.
	single layer	0	.	.	.	.
	multi-layer	0	.	.	.	.
Spice	unknown	0	.	.	.	.
	single layer	0	.	.	.	.
	multi-layer	3	85.000	5.292	81.000	91.000
Other Flavor	unknown	0	.	.	.	.
	single layer	0	.	.	.	.
	multi-layer	1	72.000	.	72.000	72.000

例 7: 平均の信頼限界の計算

要素:　　PROC MEANS ステートメントオプション

ALPHA=
FW=
MAXDEC=

CLASS ステートメント

データセット: Charity

詳細

この例では、次を行います。

- 統計量のフィールド幅と小数点以下桁数を指定します。
- 3 年間のデータの MoneyRaised と HoursVolunteered の平均値に対する両側 90%信頼限界を計算します。

このデータがより大きいボランティアの母集団を表す場合、信頼限界は真の母平均に対し可能性のある値の範囲を提供します。

プログラム

```
data charity;
  input School $ 1-7 Year 9-12 Name $ 14-20 MoneyRaised 22-26
      HoursVolunteered 28-29;
  datalines;
Monroe  2016 Allison 31.65 19
Monroe  2016 Barry   23.76 16
Monroe  2016 Candace 21.11  5

   . . . more data lines . . .
Kennedy 2018 Sid     27.45 25
Kennedy 2018 Will    28.88 21
Kennedy 2018 Morty   34.44 25
;

proc means data=charity fw=8 maxdec=2 alpha=0.1 clm mean std;

  class Year;

  var MoneyRaised HoursVolunteered;

  title 'Confidence Limits for Fund Raising Statistics';
  title2 '2016-18';
run;
```

プログラムの説明

CHARITY データセットを作成します。 CHARITY には、高校生の慈善事業のボランティア活動に関する情報が含まれています。変数は、高校名、募金活動の年度、各

生徒の名前、各学生が集めた金額、各学生がボランティア活動をした時間数を提供します。DATA ステップは、次のデータセットを作成します。

```
data charity;
  input School $ 1-7 Year 9-12 Name $ 14-20 MoneyRaised 22-26
      HoursVolunteered 28-29;
  datalines;
Monroe  2016 Allison 31.65 19
Monroe  2016 Barry   23.76 16
Monroe  2016 Candace 21.11  5

   . . . more data lines . . .
Kennedy 2018 Sid     27.45 25
Kennedy 2018 Will    28.88 21
Kennedy 2018 Morty   34.44 25
;
```

分析と分析オプションを指定します。 FW=はフィールド幅 8 を、MAXDEC=は小数点以下 2 桁をそれぞれ使用して、統計量を表示します。ALPHA=0.1 は 90%の信頼限界を指定し、CLM キーワードは両側信頼限界を要求します。MEAN と STD は、平均値と標準偏差をそれぞれ要求します。

```
proc means data=charity fw=8 maxdec=2 alpha=0.1 clm mean std;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Year の値別に分割します。

```
  class Year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が MoneyRaised 変数と HoursVolunteered 変数の統計量を計算するように指定します。

```
  var MoneyRaised HoursVolunteered;
```

タイトルを指定します。

```
  title 'Confidence Limits for Fund Raising Statistics';
  title2 '2016-18';
run;
```

出力

PROC MEANS は、各年度の両方の変数に対し、下限信頼限界と上限信頼限界を表示します。

アウトプット 35.9 信頼限界

Confidence Limits for Fund Raising Statistics
2016-18

The MEANS Procedure

Year	N Obs	Variable	Lower 90% CL for Mean	Upper 90% CL for Mean	Mean	Std Dev
2016	34	MoneyRaised	25.69	32.29	28.99	11.37
		HoursVolunteered	17.68	22.73	20.21	8.71
2017	29	MoneyRaised	24.61	31.61	28.11	11.09
		HoursVolunteered	15.67	20.19	17.93	7.17
2018	46	MoneyRaised	26.73	33.78	30.26	14.23
		HoursVolunteered	19.68	22.63	21.15	5.96

例 8: 出力統計量を計算する

要素:

PROC MEANS ステートメントオプション
- NOPRINT

CLASS ステートメント

OUTPUT ステートメントオプション
- 統計量キーワード
- IDGROUP
- LEVELS
- WAYS

PRINT プロシジャ

データセット: GRADE

詳細

この例では、次を行います。

- PROC MEANS 出力の表示を非表示にします。
- 平均の最終成績を新しい変数に格納します。
- 期末試験のスコアが最高の生徒名を新しい変数に格納します。
- 組み合わせる分類変数の数を_WAY_変数に格納します。
- 分類水準の値を_LEVEL_変数に格納します。
- 出力データセットを表示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc means data=Grade noprint;

  class Status Year;

  var FinalGrade;

  output out=sumstat mean=AverageGrade
       idgroup (max(score) obs out (name)=BestScore)
       / ways levels;
run;

proc print data=sumstat noobs;
  title1 'Average Undergraduate and Graduate Course Grades';
  title2 'For Two Years';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=60;
```

分析オプションを指定します。 NOPRINT は、すべての PROC MEANS 出力の表示を非表示にします。

```
proc means data=Grade noprint;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Status と Year の値別に分割します。

```
  class Status Year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が FinalGrade 変数の統計量を計算するように指定します。

```
  var FinalGrade;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは SUMSTAT データセットを作成し、最終成績の平均値を新しい変数 AverageGrade に書き込みます。IDGROUP は、試験のスコアが最上位の生徒の名前を変数 BestScore と、上位のスコアが含まれたオブザベーション番号に書き込みます。WAYS と LEVELS は、分類変数の組み合わせ方法に関する情報を書き込みます。

```
  output out=sumstat mean=AverageGrade
       idgroup (max(score) obs out (name)=BestScore)
       / ways levels;
run;
```

出力データセット WORK.SUMSTAT を印刷します。 NOOBS オプションは、オブザベーション番号を非表示にします。

```
proc print data=sumstat noobs;
  title1 'Average Undergraduate and Graduate Course Grades';
```

```
    title2 'For Two Years';
run;
```

出力

最初のオブザベーションには、2 年間にわたるクラスの平均成績と最高スコアの学生の名前が含まれます。次の 4 つのオブザベーションには、各分類変数値に対する値が含まれています。その他の 4 つのオブザベーションには、Year と Status の組み合わせに対する値が含まれています。変数_WAY_、_TYPE_、および_LEVEL_は、PROC MEANS が分類変数の組み合わせを作成した方法を示します。変数_OBS_には、試験の最高スコアが含まれた GRADE データセットのオブザベーション番号が含まれています。

アウトプット 35.10　Average Course Grades

Average Undergraduate and Graduate Course Grades
For Two Years

Status	Year	_WAY_	_TYPE_	_LEVEL_	_FREQ_	AverageGrade	BestScore	_OBS_
		0	0	1	10	83.0000	Branford	2
	97	1	1	1	6	83.6667	Jasper	10
	98	1	1	2	4	82.0000	Branford	2
1		1	2	1	6	82.5000	Branford	2
2		1	2	2	4	83.7500	Abbott	1
1	97	2	3	1	3	79.3333	Jasper	10
1	98	2	3	2	3	85.6667	Branford	2
2	97	2	3	3	3	88.0000	Abbott	1
2	98	2	3	4	1	71.0000	Crandell	3

例 9: 複数の変数に対して、異なる出力統計量を計算する

要素:

PROC MEANS ステートメントオプション
- DESCEND
- NOPRINT

CLASS ステートメント

OUTPUT ステートメントオプション
- 統計量キーワード

PRINT プロシジャ

WHERE=データセットオプション

データセット: GRADE

詳細

この例では、次を行います。

- PROC MEANS 出力の表示を非表示にします。
- 分類水準と、WHERE=によって指定される分類変数の組み合わせを出力データセットに格納します。
- 出力データセットのオブザベーションを_TYPE_値の降順で並べ替えます。
- 新しい変数名を割り当てずに、試験スコアの平均値と最終成績の平均値を格納します。
- 最終成績の中央値を新しい変数に格納します。
- 出力データセットを表示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc means data=Grade noprint descend;

  class Status Year;

  var Score FinalGrade;

  output out=Sumdata (where=(status='1' or _type_=0))
       mean= median(finalgrade)=MedianGrade;
run;

proc print data=Sumdata;
  title 'Exam and Course Grades for Undergraduates Only';
  title2 'and for All Students';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=60;
```

分析オプションを指定します。 NOPRINT は、すべての PROC MEANS 出力の表示を非表示にします。DESCEND は、_TYPE_値の降順で OUT=データセットのオブザベーションを並べ替えます。

```
proc means data=Grade noprint descend;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Status と Year の値別に分割します。

```
class Status Year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が Score 変数と FinalGrade 変数の統計量を計算するように指定します。

```
var Score FinalGrade;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは、Score と FinalGrade の平均値を同じ名前の変数に書き込みます。最終成績の中間値が、変数 MedianGrade に書き込まれます。WHERE=データセットオプションは、SUMDATA のオブザベーションを制限します。1 つのオブザベーションには、全体の統計量 (_TYPE_=0)が含まれています。その他の区分は、1 である必要があります。

```
output out=Sumdata (where=(status='1' or _type_=0))
     mean= median(finalgrade)=MedianGrade;
run;
```

出力データセット WORK.SUMDATA を印刷します。

```
proc print data=Sumdata;
  title 'Exam and Course Grades for Undergraduates Only';
  title2 'and for All Students';
run;
```

出力

最初の 3 つのオブザベーションには、区分 1 の分類変数水準に対する統計量が含まれています。最後のオブザベーションには、すべてのオブザベーションに対する統計量が含まれています(サブグループなし)。Score にはテストスコアの平均値が、FinalGrade には最終成績の平均値がそれぞれ含まれています。

アウトプット 35.11　試験とクラスの成績

Exam and Course Grades for Undergraduates Only and for All Students

Obs	Status	Year	_TYPE_	_FREQ_	Score	FinalGrade	MedianGrade
1	1	97	3	3	84.6667	79.3333	73
2	1	98	3	3	88.3333	85.6667	80
3	1		2	6	86.5000	82.5000	80
4			0	10	86.0000	83.0000	83

例 10: 分類変数の欠損値を使用し、出力統計量を計算する

要素:

PROC MEANS ステートメントオプション
- CHARTYPE
- NOPRINT
- NWAY

CLASS ステートメントオプション
- ASCENDING
- MISSING
- ORDER=

OUTPUT ステートメント

PRINT プロシジャ

データセット: CAKE

詳細

この例では、次を行います。

- PROC MEANS 出力の表示を非表示にします。
- 欠損値を 1 つの分類変数にのみ有効な水準値とみなします。
- 出力データセットのオブザベーションを 1 つの分類変数に対する度数の昇順で並べ替えます。
- _TYPE_の値が最高のオブザベーションを格納します。
- _TYPE_をバイナリ文字値として格納します。
- 味のスコアの最大値を新しい変数に格納します。
- 出力データセットを表示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc means data=cake chartype nway noprint;

  class flavor /order=freq ascending;
  class layers /missing;

  var TasteScore;

  output out=cakestat max=HighScore;
run;
```

```
proc print data=cakestat;
  title 'Maximum Taste Score for Flavor and Cake Layers';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=60;
```

分析オプションを指定します。 NWAY は、_TYPE_値が最高のオブザベーションを印刷します。NOPRINT は、すべての PROC MEANS 出力の表示を非表示にします。

```
proc means data=cake chartype nway noprint;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Flavor と Layers の値別に分割します。ORDER=FREQ と ASCENDING は、度数の昇順で Flavor の水準を並べ替えます。MISSING は、Layers の欠損値を有効な分類水準値として使用します。

```
  class flavor /order=freq ascending;
  class layers /missing;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が TasteScore 変数の統計量を計算するように指定します。

```
  var TasteScore;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは、CAKESTAT データセットを作成し、味のスコアの最大値を新しい変数 HighScore に出力します。

```
  output out=cakestat max=HighScore;
run;
```

出力データセット WORK.CAKESTAT を印刷します。

```
proc print data=cakestat;
  title 'Maximum Taste Score for Flavor and Cake Layers';
run;
```

出力

CAKESTAT 出力データセットには、2 つの分類変数、Flavor と Layers の組み合わせに対するオブザベーションのみ含まれています。そのため、_TYPE_の値は、すべてのオブザベーションの場合は 11 となります。オブザベーションは、Flavor の度数の昇順で並べ替えられます。Layers の欠損値は、この分類変数に有効な値です。PROC MEANS は、風味が欠損しているオブザベーションを除外します。これは Flavor に無効な値であるためです。

アウトプット 35.12　味の最大スコア

Maximum Taste Score for Flavor and Cake Layers

Obs	Flavor	Layers	_TYPE_	_FREQ_	HighScore
1	Rum	2	11	1	72
2	Spice	2	11	2	83
3	Spice	3	11	1	91
4	Vanilla	.	11	1	84
5	Vanilla	1	11	3	94
6	Vanilla	2	11	2	87
7	Chocolate	.	11	1	84
8	Chocolate	1	11	5	85
9	Chocolate	2	11	3	92

例 11: 出力統計量を使用し、極値を識別する

要素:
- CLASS ステートメント
- OUTPUT ステートメントオプション
  - 統計量キーワード
  - MAXID
- PRINT プロシジャ

データセット: Charity

詳細

この例では、次を行います。

- 2 つの変数に対する最大値を含むオブザベーションを識別します。
- 最大値に対し新しい変数を作成します。
- 出力データセットを表示します。

プログラム

```
proc means data=Charity n mean range chartype;
```

```
  class School Year;

  var MoneyRaised HoursVolunteered;

  output out=Prize maxid(MoneyRaised(name)
       HoursVolunteered(name))= MostCash MostTime
       max= ;


  title 'Summary of Volunteer Work by School and Year';
run;

proc print data=Prize;
  title 'Best Results: Most Money Raised and Most Hours Worked';
run;
```

プログラムの説明

分析を指定します。 統計量キーワードは、統計量と出力での順序を指定します。CHARTYPE は、_TYPE_値をバイナリ文字として出力データセットに書き込みます。

```
proc means data=Charity n mean range chartype;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を School と Year 別に分割します。

```
  class School Year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が MoneyRaised 変数と HoursVolunteered 変数の統計量を計算するように指定します。

```
  var MoneyRaised HoursVolunteered;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは、最も多くの金額を集めた生徒と最も多くの時間をボランティアに充てた生徒の名前がそれぞれ含まれている新しい変数、MostCash と MostTime を PRIZE データセットに書き込みます。

```
  output out=Prize maxid(MoneyRaised(name)
       HoursVolunteered(name))= MostCash MostTime
       max= ;
```

タイトルを指定します。

```
  title 'Summary of Volunteer Work by School and Year';
run;
```

WORK.PRIZE 出力データセットを印刷します。

```
proc print data=Prize;
  title 'Best Results: Most Money Raised and Most Hours Worked';
run;
```

出力

出力の最初のページには、PROC MEANS からの出力が 6 つの分類水準に対する統計量とともに表示されます。これは、2007 年、2008 年、2009 年の Monroe High に対するものと、同じ 3 年間の Kennedy High に対するものです。

アウトプット 35.13　ボランティア活動の概要

Summary of Volunteer Work by School and Year

The MEANS Procedure

School	Year	N Obs	Variable	N	Mean	Range
Kennedy	2007	18	MoneyRaised	18	29.3811111	39.7500000
			HoursVolunteered	18	21.4444444	30.0000000
	2008	17	MoneyRaised	17	28.1564706	23.5600000
			HoursVolunteered	17	19.4117647	20.0000000
	2009	18	MoneyRaised	18	31.5794444	65.4400000
			HoursVolunteered	18	24.2777778	15.0000000
Monroe	2007	16	MoneyRaised	16	28.5450000	48.2700000
			HoursVolunteered	16	18.8125000	38.0000000
	2008	12	MoneyRaised	12	28.0500000	52.4600000
			HoursVolunteered	12	15.8333333	21.0000000
	2009	28	MoneyRaised	28	29.4100000	73.5300000
			HoursVolunteered	28	19.1428571	26.0000000

PROC PRINT からの出力には、MoneyRaised と HoursVolunteered の最大値と、担当した学生の名前が表示されます。最初のオブザベーションには全体の結果が含まれ、次の 3 つには年度別の結果、次の 2 つには学校別の結果、最後の 6 つには School と Year 別の結果がそれぞれ含まれています。

アウトプット 35.14　最高結果

Best Results: Most Money Raised and Most Hours Worked

Obs	School	Year	_TYPE_	_FREQ_	MostCash	MostTime	MoneyRaised	HoursVolunteered
1		.	00	109	Willard	Tonya	78.65	40
2		2007	01	34	Tonya	Tonya	55.16	40
3		2008	01	29	Cameron	Amy	65.44	31
4		2009	01	46	Willard	L.T.	78.65	33
5	Kennedy	.	10	53	Luther	Jay	72.22	35
6	Monroe	.	10	56	Willard	Tonya	78.65	40
7	Kennedy	2007	11	18	Thelma	Jay	52.63	35
8	Kennedy	2008	11	17	Bill	Amy	42.23	31
9	Kennedy	2009	11	18	Luther	Che-Min	72.22	33
10	Monroe	2007	11	16	Tonya	Tonya	55.16	40
11	Monroe	2008	11	12	Cameron	Myrtle	65.44	26
12	Monroe	2009	11	28	Willard	L.T.	78.65	33

例 12: 出力統計量を使用し、上位 3 位の極値を識別する

要素:

PROC MEANS ステートメントオプション
- NOPRINT

CLASS ステートメント

OUTPUT ステートメントオプション
- 統計量キーワード
- AUTOLABEL
- AUTONAME
- IDGROUP

TYPES ステートメント

FORMAT プロシジャ

FORMAT ステートメント

PRINT プロシジャ

RENAME=データセットオプション

データセット: Charity

詳細

この例では、次を行います。

- PROC MEANS 出力の表示を非表示にします。
- 分類変数の 1 次元組み合わせのデータとすべてのオブザベーションにわたるデータを分析します。
- 募金金額の合計と平均を新しい変数に格納します。
- 新しい変数に上位 3 位の募金金額、それを集めた 3 人の学生の名前、その年度、学生が通った学校を格納します。
- 名前が出力データセットの新しい変数に割り当てられているときの変数名の競合問題を自動的に解決します。
- 分析変数に対して計算された統計量を含む出力データセットの変数のラベルに統計量名を追加します。
- この変数から計算される統計量が出力データセットの属性を継承するように、出力形式を分析変数に割り当てます。
- 出力データセットの_FREQ_変数の名前を変更します。
- 出力データセットとその内容を表示します。

プログラム

```
proc format;
  value yrFmt . = " All";
  value $schFmt ' ' = "All    ";
run;

proc means data=Charity noprint;

  class School Year;

  types () school year;

  var MoneyRaised;

  output out=top3list(rename=(_freq_=NumberStudents))sum= mean=
       idgroup( max(moneyraised) out[3] (moneyraised name
        school year)=)/autolabel autoname;

  label MoneyRaised='Amount Raised';
  format year yrfmt. school $schfmt.
       moneyraised dollar8.2;
 run;

proc print data=top3list;
  title1 'School Fund Raising Report';
  title2 'Top Three Students';
run;

proc datasets library=work nolist;
```

```
  contents data=top3list;
  title1 'Contents of the PROC MEANS Output Data Set';
run;
```

プログラムの説明

YRFMT.出力形式と$SCHFMT.出力形式を作成します。 PROC FORMAT は、**All** の値を分類変数の欠損水準に割り当てるユーザー定義出力形式を作成します。

```
proc format;
  value yrFmt . = " All";
  value $schFmt ' ' = "All    ";
run;
```

デフォルトの統計量を生成して、分析オプションを指定します。 NOPRINT は、すべての PROC MEANS 出力の表示を非表示にします。

```
proc means data=Charity noprint;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を School と Year の値別に分割します。

```
  class School Year;
```

分析するサブグループを指定します。 TYPES ステートメントは、すべてのオブザベーションにわたる分析と、School と Year の各 1 次元の組み合わせに対する分析を要求します。

```
  types () school year;
```

分析変数を指定します。 VAR ステートメントは、PROC MEANS が MoneyRaised 変数の統計量を計算するように指定します。

```
  var MoneyRaised;
```

出力データセットオプションを指定します。 OUTPUT ステートメントは、TOP3LIST データセットを作成します。RENAME=は、各分類水準に対する度数カウントを含む_FREQ_変数の名前を変更します。SUM=と MEAN=は、分析変数(MoneyRaised)の合計と平均値が出力データセットに書き込まれるように指定します。IDGROUP は、上位 3 位の募金金額、対応する 3 人の生徒、学校および年度を含む 12 の変数を書き込みます。AUTOLABEL は、分析変数の名前を合計と平均値を含む出力変数のラベルに追加します。AUTONAME は、これらの変数の名前の競合問題を解決します。

```
  output out=top3list(rename=(_freq_=NumberStudents))sum= mean=
      idgroup( max(moneyraised) out[3] (moneyraised name
       school year)=)/autolabel autoname;
```

出力をフォーマットします。 LABEL ステートメントは、ラベルを分析変数 MoneyRaised に割り当てます。FORMAT ステートメントは、ユーザー定義出力形式を Year 変数と School 変数に、SAS ドル出力形式を MoneyRaised 変数にそれぞれ割り当てます。

```
  label MoneyRaised='Amount Raised';
  format year yrfmt. school $schfmt.
      moneyraised dollar8.2;
run;
```

出力データセット WORK.TOP3LIST を印刷します。

```
proc print data=top3list;
  title1 'School Fund Raising Report';
  title2 'Top Three Students';
run;
```

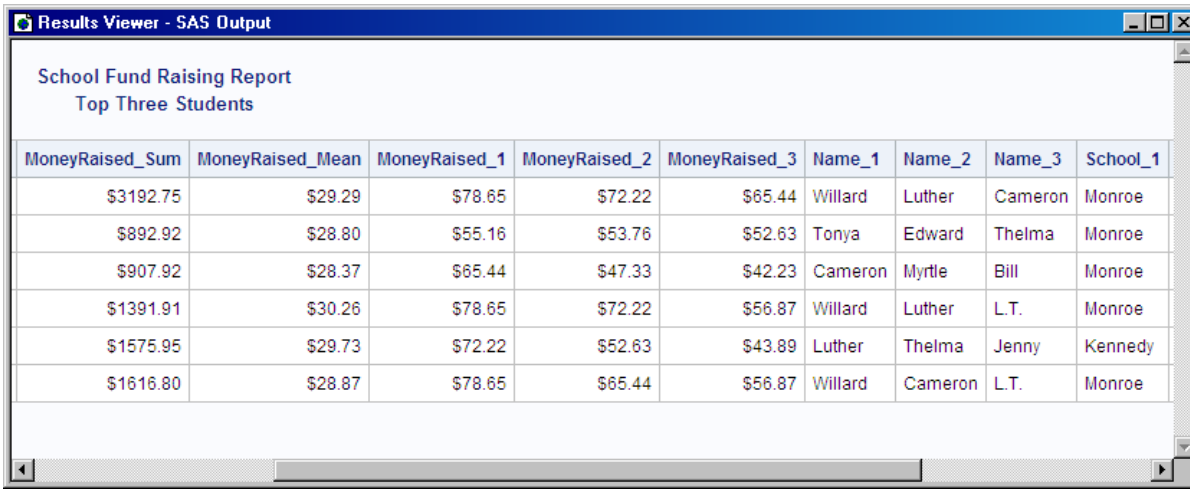
TOP3LIST データセットに関する情報を表示します。 PROC DATASETS は、TOP3LIST データセットのコンテンツを表示します。NOLIST は、WORK データライブラリのディレクトリリストの表示を非表示にします。

```
proc datasets library=work nolist;
  contents data=top3list;
  title1 'Contents of the PROC MEANS Output Data Set';
run;
```

出力

PROC PRINT からの出力には、MoneyRaised の上位 3 位の値、これらの金額を集めた学生の名前、その学生が通う学校、募金が行われた年度が表示されます。最初のオブザベーションには全体の結果が含まれ、次の 3 つには年度別の結果、最後の 2 つには学校別の結果が含まれています。School と Year の欠損分類水準は、値 ALL と置き換えられます。MoneyRaised から計算された統計量を含む変数のラベルには、ラベルの最後に統計量名が含まれます。

アウトプット 35.15　学校基金調達


Results Viewer - SAS Output

School Fund Raising Report
Top Three Students

MoneyRaised_Sum	MoneyRaised_Mean	MoneyRaised_1	MoneyRaised_2	MoneyRaised_3	Name_1	Name_2	Name_3	School_1
$3192.75	$29.29	$78.65	$72.22	$65.44	Willard	Luther	Cameron	Monroe
$892.92	$28.80	$55.16	$53.76	$52.63	Tonya	Edward	Thelma	Monroe
$907.92	$28.37	$65.44	$47.33	$42.23	Cameron	Myrtle	Bill	Monroe
$1391.91	$30.26	$78.65	$72.22	$56.87	Willard	Luther	L.T.	Monroe
$1575.95	$29.73	$72.22	$52.63	$43.89	Luther	Thelma	Jenny	Kennedy
$1616.80	$28.87	$78.65	$65.44	$56.87	Willard	Cameron	L.T.	Monroe

この出力データセットに対するカスタムテーブルテンプレートの作成方法の例については、*SAS Output Delivery System: ユーザーガイド*の TEMPLATE プロシジャを参照してください。

アウトプット 35.16 PROC MEANS 出力データセット

School Fund Raising Report
Top Three Students

Obs	School	Year	_TYPE_	NumberStudents	MoneyRaised_Sum	MoneyRaised_Mean	MoneyRaised_1	MoneyRaised_2	MoneyRaised_3	Name_1
1	All	All	0	109	$3192.75	$29.29	$78.65	$72.22	$65.44	Willard
2	All	2007	1	34	$985.58	$28.99	$55.16	$53.76	$52.63	Tonya
3	All	2008	1	29	$815.26	$28.11	$65.44	$47.33	$42.23	Cameron
4	All	2009	1	46	$1391.91	$30.26	$78.65	$72.22	$56.87	Willard
5	Kenn	All	2	53	$1575.95	$29.73	$72.22	$52.63	$43.89	Luther
6	Monr	All	2	56	$1616.80	$28.87	$78.65	$65.44	$56.87	Willard

Contents of the PROC MEANS Output Data Set

The DATASETS Procedure

Data Set Name	WORK.TOP3LIST	**Observations**	6
Member Type	DATA	**Variables**	18
Engine	V9	**Indexes**	0
Created	Tuesday, July 05, 2011 10:24:55 AM	**Observation Length**	144
Last Modified	Tuesday, July 05, 2011 10:24:55 AM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	12288
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	85
Obs in First Data Page	6
Number of Data Set Repairs	0
Filename	C:\DOCUME~1\lirezn\LOCALS~1\Temp\SAS Temporary Files_TD3456_d21560_\top3list.sas7bdat
Release Created	9.0301M0
Host Created	XP_PRO

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
7	MoneyRaised_1	Num	8	DOLLAR8.2	Amount Raised
8	MoneyRaised_2	Num	8	DOLLAR8.2	Amount Raised
9	MoneyRaised_3	Num	8	DOLLAR8.2	Amount Raised
6	MoneyRaised_Mean	Num	8	DOLLAR8.2	Amount Raised_Mean
5	MoneyRaised_Sum	Num	8	DOLLAR8.2	Amount Raised_Sum
10	Name_1	Char	7		
11	Name_2	Char	7		
12	Name_3	Char	7		
4	NumberStudents	Num	8		
1	School	Char	7	$SCHFMT.	
13	School_1	Char	7	$SCHFMT.	
14	School_2	Char	7	$SCHFMT.	
15	School_3	Char	7	$SCHFMT.	
2	Year	Num	8	YRFMT.	
16	Year_1	Num	8	YRFMT.	
17	Year_2	Num	8	YRFMT.	
18	Year_3	Num	8	YRFMT.	
3	_TYPE_	Num	8		

例 13: STACKODSOUTPUT オプションによるデータの制御

要素: PROC PRINT
PROC CONTENTS

最初の例では、STACKODSOUTPUT オプションは使用しません。2 つ目の例では、STACKODSOUTPUT オプションを使用します。

プログラム

```
proc means data=sashelp.class;
class sex;
var weight height;
ods output summary=default;
run;

proc print data=default; run;
proc contents data=default; run;
```

プログラムの説明

このコードは、STACKODSOUTPUT オプションを使用せずにデータを処理します。

```
proc means data=sashelp.class;
class sex;
var weight height;
ods output summary=default;
run;
```

PROC PRINT を使用してデータを印刷します。PROC CONTENTS を使用してプロシジャのコンテンツを出力します。

```
proc print data=default; run;
proc contents data=default; run;
```

出力

次の出力に、STACKODSOUTPUT オプションを使用した場合と使用しない場合のデータ処理の違いを示します。

この出力は、STACKODSOUTPUT オプションを使用せずに生成されます。

アウトプット 35.17　STACKODSOUTPUT オプションを使用しない出力

The SAS System

The MEANS Procedure

Sex	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
F	9	Weight	9	90.1111111	19.3839137	50.5000000	112.5000000
		Height	9	60.5888889	5.0183275	51.3000000	66.5000000
M	10	Weight	10	108.9500000	22.7271864	83.0000000	150.0000000
		Height	10	63.9100000	4.9379370	57.3000000	72.0000000

The SAS System

Obs	Sex	NObs	VName_Weight	Weight_N	Weight_Mean	Weight_StdDev	Weight_Min	Weight_Max	VName_H
1	F	9	Weight	9	90.111111111	19.38391372	50.5	112.5	Height
2	M	10	Weight	10	108.95	22.727186363	83	150	Height

The SAS System

The CONTENTS Procedure

Data Set Name	WORK.DEFAULT	**Observations**	2
Member Type	DATA	**Variables**	14
Engine	V9	**Indexes**	0
Created	Sunday, January 23, 2011 02:48:10 PM	**Observation Length**	104
Last Modified	Sunday, January 23, 2011 02:48:10 PM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label	Summary statistics		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	12288
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	117
Obs in First Data Page	2
Number of Data Set Repairs	0
Filename	C:\DOCUME~1\lirezn\LOCALS~1\Temp\SAS Temporary Files_TD3828_d21560_\default.sas7bdat
Release Created	9.0301B0
Host Created	XP_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
14	Height_Max	Num	8	BEST12.	Maximum
11	Height_Mean	Num	8	BEST12.	Mean
13	Height_Min	Num	8	BEST12.	Minimum
10	Height_N	Num	8	BEST2.	N
12	Height_StdDev	Num	8	BEST12.	Std Dev
2	NObs	Num	8	BEST2.	N Obs
1	Sex	Char	1		
9	VName_Height	Char	6		Variable
3	VName_Weight	Char	6		Variable
8	Weight_Max	Num	8	BEST12.	Maximum
5	Weight_Mean	Num	8	BEST12.	Mean
7	Weight_Min	Num	8	BEST12.	Minimum
4	Weight_N	Num	8	BEST2.	N
6	Weight_StdDev	Num	8	BEST12.	Std Dev

このコードは、STACKODSOUTPUT オプションを使用してデータを処理します。

```
proc means data=sashelp.class STACKODSOUTPUT;
```

```
class sex;
var weight height;
ods output summary=stacked;
run;
```

PROC PRINT を使用してデータを印刷します。PROC CONTENTS を使用してプロシジャのコンテンツを出力します。

```
proc print data=stacked; run;
proc contents data=stacked; run;
```

この出力は、STACKODSOUTPUT オプションを使用して生成されます。

アウトプット 35.18　STACKODSOUTPUT オプションを使用した出力

The SAS System

The MEANS Procedure

Sex	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
F	9	Weight	9	90.111111	19.383914	50.500000	112.500000
		Height	9	60.588889	5.018328	51.300000	66.500000
M	10	Weight	10	108.950000	22.727186	83.000000	150.000000
		Height	10	63.910000	4.937937	57.300000	72.000000

The SAS System

Obs	Sex	NObs	_control_	Variable	N	Mean	StdDev	Min	Max
1	F	9		Weight	9	90.111111	19.383914	50.500000	112.500000
2	F	9		Height	9	60.588889	5.018328	51.300000	66.500000
3	M	10	1	Weight	10	108.950000	22.727186	83.000000	150.000000
4	M	10		Height	10	63.910000	4.937937	57.300000	72.000000

The SAS System

The CONTENTS Procedure

Data Set Name	WORK.STACKED	**Observations**	4
Member Type	DATA	**Variables**	9
Engine	V9	**Indexes**	0
Created	Sunday, January 23, 2011 03:19:07 PM	**Observation Length**	56
Last Modified	Sunday, January 23, 2011 03:19:07 PM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label	Summary statistics		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	145
Obs in First Data Page	4
Number of Data Set Repairs	0
Filename	C:\DOCUME~1\lirezn\LOCALS~1\Temp\SAS Temporary Files_TD5560_d21560_\stacked.sas7bdat
Release Created	9.0301B0
Host Created	XP_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
9	Max	Num	8	D12.3	Maximum
6	Mean	Num	8	D12.3	
8	Min	Num	8	D12.3	Minimum
5	N	Num	8	BEST2.	
2	NObs	Num	8	BEST2.	N Obs
1	Sex	Char	1		
7	StdDev	Num	8	D12.3	Std Dev
4	Variable	Char	6		
3	_control_	Char	1		

参考文献

Jain R., and I. Chlamtac. 1985. “The P² Algorithm for Dynamic Calculation of Quantiles and Histograms without Sorting Observations.” *Communications of the Association of Computing Machinery* 28 (10): 1076-0185.

36

MIGRATE プロシジャ

概要: MIGRATE プロシジャ

MIGRATE プロシジャの機能

このプロシジャでは、ライブラリのメンバーを現在のリリースの SAS ソフトウェアに移行します。ソースライブラリとターゲットライブラリは、V9 エンジンライブラリでなければなりません。

このプロシジャは、次のライブラリメンバーを移行します。

- 代替照合順序、監査証跡、圧縮、作成日時と変更日時、削除されたオブザベーション、暗号化、拡張属性、世代、インデックス、一貫性制約、パスワードを含むデータセット
- 多くの場合、ビュー、カタログ、アイテムストア、多次元データベース(MDDB)(概念: MIGRATE プロシジャ (1336 ページ)を参照)

このプロシジャは保存されたコンパイル済み DATA ステッププログラムや保存されたコンパイル済みマクロをサポートしていません。そこではコンパイルと保存が可能です)。このプロシジャは SAS プログラムファイルをサポートしていません。このプロシジャは、スケーラブルパフォーマンスデータ(SPD)エンジンデータセットをサポートしていません。*SAS Scalable Performance Data Engine: リファレンス*を参照してください。このプロシジャは、拡張オブザベーションカウント属性をサポートしていません。

概念: MIGRATE プロシジャ

データセット

PROC MIGRATE は、代替照合順序、圧縮、作成日時と変更日時、削除されたオブザベーション、暗号化、拡張属性、インデックス、一貫性制約、パスワードを保持します。構文の先頭 (1339 ページ)と各オプションの構文に記述されているいくつかの重要な制限事項を参照してください。

監査証跡と世代も移行されます。インデックスと一貫性制約は、ターゲットライブラリのメンバーで再作成されます。“データセットの移行(監査証跡、世代、インデックス、または一貫性制約)” (1345 ページ)を参照してください。

移行されたデータセットでは、ターゲットライブラリのデータ表現とエンコーディング属性が採用されます。データセットを、文字がより多くのバイトで表現されるエンコーディングに移行する場合、列の長さがより大きな文字サイズに対応しない場合、切り捨てが発生する可能性があります。たとえば、ある文字が Wlatin1 エンコーディングでは 1 バイトとして表示されるが、UTF-8 エンコーディングでは 2 バイトとして表示される場合があります。最善の解決策は、移行中に CVP エンジンを使用して、ライブラリ内のデータセットの列の長さを拡張することです。“切り捨てを回避するための CVP エンジンによる移行” (1343 ページ)を参照してください。

暗号化されたデータセットの場合、作成および変更された日時の値は保持されません。現在の日時値が使用されます。AES または AES2 で暗号化されたデータセットの場合、ENCRYPTKEY=値を提供するには、SAS をインタラクティブに実行する必要があります。PROC MIGRATE は、AES または AES2 暗号化キーに自動的にアクセスできません。SAS 独自の暗号化データセット、またはパスワードで保護されたデータセットの場合、SAS をバッチで実行できます。PROC MIGRATE は自動的にパスワードにアクセスできます。いずれの場合も、移行が成功すると、暗号化、暗号化キー、およびパスワードがターゲットライブラリに保持されます。

言語的に並べ替えられたデータセットの場合、International Components for Unicode (ICU)のバージョン番号は、現在のセッションのものとは異なる場合があります。その場合、PROC MIGRATE は OUT=出力ライブラリでのデータセットの並べ替え順序を保持します。ただし、そのデータセットから並べ替え済みのマーキングがなくなり、SAS ログに警告メッセージが書き込まれます。言語的な並べ替えの詳細については、55 章, “SORT プロシジャ,” (2027 ページ)を参照してください。

ビュー

データファイルと同様に、移行されたデータビューでは、ターゲットライブラリのデータ表現とエンコーディング属性が採用されます。

DATA ステップビュー
: DATA ステップビューを作成する際に、SOURCE=オプションを指定すると、DATA ステップコードをビューと一緒に保存できます。PROC MIGRATE では、コードの保存された DEAT ステップビューがサポートされています。保存されたコードは、DATA ステップビューがターゲット環境で初めて SAS にアクセスされる際に、再コンパイルされます。PROC MIGRATE では、SAS 8 より前に作成された DATA ステップビューや、コードが保存されていない DATA ステップビューはサポートされていません。コードの保存されていない DATA ステップビューでは、ソースセッションで DESCRIBE ステートメントを使用して DATA ステップコードを復元します。次に、ターゲットセッションでその DATA ステップコードをサブミットして、再コンパイルします。

PROC SQL ビュー
: ビューに関連付けられている埋め込まれたライブラリ参照名は移行時に更新されないので注意してください。次の例で、この問題について説明します。この例では、Lib1.MyView にデータセット Lib1.MyData のビューが含まれます。

```
data Lib1.MyData;x=1;
```

```
run;
proc sql;
  create view Lib1.MyView as select * from Lib1.MyData;
quit;
```

Lib1 を Lib2 に移行すると、Lib2.MyView と Lib2.MyData が得られます。ただし、Lib2.MyView は、当初の作成時に埋め込まれたライブラリ参照名が Lib1 であったため、現在もデータセット Lib2.MyData ではなく、Lib1.MyData を参照します。次の例では、処理が失敗して、Lib1 が見つからないというエラーメッセージが出ます。

```
proc print data=Lib2.MyView;
run;
```

カタログ

カタログを移行するために、PROC MIGRATE は PROC CPORT と PROC CIMPORT を呼び出します。移行時に、CPORT と CIMPORT のノートが SAS ログに書き込まれる場合があります。PROC CPORT と CIMPORT の制限が適用されます。カタログに保存されている保存済みでコンパイル済みのマクロはサポートされません(かわりに、ソースコードをターゲットに移動します。そこではコンパイルと保存が可能です)。移行後にカタログエントリの更新が必要になる可能性があります(たとえば、ハードコードされたパス名を含んでいるコード)。

ターゲット環境とは異なるエンコーディングまたはデータ表現でカタログが作成された場合、移行中にトランスコーディングエラーが発生する可能性があります。ベストプラクティスは、カタログが作成されたセッションのエンコードとデータ表現を把握することです。カタログはエンコーディングまたはデータ表現に関するメタデータを維持しないため、PROC DATASETS を使用してカタログのこれらの属性を学習することはできません。これらのトランスコーディングエラーを回避するために、次のオプションを使用できます。

- 現在のセッションのエンコーディングが UTF-8 で、ユーザー定義出力形式のカタログが UTF-8 以外のセッションで作成されていて、各国語文字を含んでいる場合は、PROC MIGRATE で ENCODING=オプションを使用できます。このオプションにより、PROC MIGRATE はユーザー定義出力形式のカタログエントリをトランスコードできます。切り捨てを避けるために、必要に応じてフォーマットの長さが拡張されます。他のライブラリメンバーは移行されますが、このオプションはそれらのメンバーには適用されません。カタログが作成されたセッションのエンコーディングを指定する必要があります。“ENCODING=*encoding-value*” (1341 ページ)と“例 4: カタログが異なるエンコーディングからのものである場合の移行” (1352 ページ)を参照してください。
- データ表現がターゲットセッションと互換性がない場合は、SAS/CONNECT も必要です。IN=オプションまたは SLIBREF=オプションでサーバーを指定します。参照: “SAS/CONNECT サーバーの使用” (1344 ページ).SAS/CONNECT ソフトウェアにアクセスできない場合は、“サポートしていないカタログへの追加のステップ” (1346 ページ)で説明されているプロセスを使用します。
- ライブラリにデータセットが存在する場合は、CVP エンジンを使用してデータの長さを拡張できます。CVP エンジンは、カタログエントリをトランスコードした

り、フォーマットの長さを拡張したりしません。参照: “切り捨てを回避するための CVP エンジンによる移行” (1343 ページ).

SAS ライブラリにカタログが含まれているかどうかを判断する 2 つの方法を次に示します。

- オペレーティングシステムツールを使用して、ファイルシステムを調べます。SAS カタログのファイル拡張子は、.sas7bcat です 。
- MEMTYPE=CAT を指定して DATASETS プロシジャをサブミットします。カタログのリストについては、SAS ログを調べてください。

MDDB

PROC MIGRATE では、既知の問題がない MDDB がサポートされています。

アイテムストア

PROC MIGRATE では、32 ビット環境から 64 ビット環境への移行でなければ、アイテムストアがサポートされます。この場合、SAS ログにエラーメッセージが書き込まれますが、アイテムストアがターゲットライブラリでは正しく機能しない場合があります。

したがって、PROC MIGRATE は通常ベストプラクティスではありません。詳細については、“SAS Files That Can and Cannot Be Migrated” (*SAS V9 LIBNAME Engine: Reference*)を参照してください。

サポート対象外

PROC MIGRATE は保存されたコンパイル済み DATA ステッププログラムや保存されたコンパイル済みマクロをサポートしていません。そこではコンパイルと保存が可能です)。PROC MIGRATE では SAS プログラムファイルはサポートされていません。PROC MIGRATE では SPD Engine データセットはサポートされていません。(*SAS Scalable Performance Data Engine: リファレンス*を参照してください。)また、各メンバーの種類については前述の制限事項を参照してください。

構文: MIGRATE プロシジャ

制限事項: SAS データセットオプションは、PROC MIGRATE ではサポートされていません。

PROC MIGRATE がターゲットライブラリでメンバーを作成した場合、そのメンバーではソースライブラリの権限は保持されません。PROC MIGRATE 実行者のユーザー ID に関連付けられた権限が含まれます。

PROC MIGRATEIN=*libref-1*OUT=*libref-2*<*options*>;

ステートメント	タスク	例
“PROC MIGRATE ステートメント”	SAS ライブラリのメンバーを現在の SAS バージョンに移行	Ex. 1, Ex. 2, Ex. 3, Ex. 4

PROC MIGRATE ステートメント

SAS ライブラリをより新しい現在の SAS ソフトウェアのリリースに移行します。

構文

PROC MIGRATEIN=*libref-1*OUT=*libref-2*
<BUFSIZE=KEEPSIZE | *n* | *n*K | *n*M | *n*G>
< ENCODING=*encoding-value*>
< MOVE>
< SLIBREF=*libref*>
< KEEPNODUPKEY>;

必須引数

IN=*libref-1*
メンバーの移行元のソース SAS ライブラリの名前を指定します。 IN=オプションまたは SLIBREF=オプションで SAS/CONNECT ライブラリ参照名を指定する必要があるかどうかを判断するには、“SAS/CONNECT サーバーの使用” (1344 ページ)を参照してください。

制限事項
連結ライブラリはサポートされていません。https://support.sas.com/kb/24/539.html の SAS Note 24539 を参照してください。

要件
IN=ソースライブラリを OUT=ターゲットライブラリとは別の物理的な場所に割り当てます。

OUT=*libref-2*
移行したメンバーを含めるターゲット SAS ライブラリの名前を指定します。

制限事項
OUT=ターゲットライブラリを SAS/CONNECT サーバーに割り当てないでください。REMOTE エンジンでは、ターゲットライブラリはサポートされていません。次の例では、エラーが引き起こされます。

```
libname Lib1 'source-library-pathname';
libname Lib2 server=server-id;
```

```
proc migrate in=Lib1 out=Lib2;
run;
```

要件 OUT=ターゲットライブラリは、IN=ソースライブラリとは別の物理的な場所に割り当てます。

操作 PROC MIGRATE では、DATA、VIEW、ACCESS、MDDB、DMDB というメンバーの種類に対して LIBNAME オプション OUTREP=を使用できます。OUTREP=オプションを指定する場合は、EXTENDOBSCOUNTER=オプションも指定することがあります。LIBNAME ステートメントでは、これらのオプションは OUT=ライブラリに適しています。*SAS DATA ステップステートメント: リファレンス*を参照してください。

ヒント ターゲットライブラリを空の場所に割り当てます。ターゲットライブラリに、名前とメンバーの種類がソースライブラリのメンバーと同じメンバーがすでに存在する場合、そのメンバーは移行されません。SAS ログにエラーメッセージが書き込まれ、PROC MIGRATE は次のメンバーの処理を続行します。

オプション引数

BUFSIZE=KEEPSIZE | *n* | *n*K | *n*M | *n*G
ターゲットライブラリに書き込まれるメンバーのバッファーページサイズを指定します。 たとえば、**10000** の値では 10,000 バイトのページサイズ、**4k** の値では 4096 バイトのページサイズが指定されます。デフォルト値は **0** です。ページサイズを設定すると、SAS のパフォーマンスを最適化できます。

BUFSIZE=データセットオプションまたはシステムオプションの詳細については、動作環境に関するドキュメントを参照してください。

KEEPSIZE
ソースライブラリのメンバーのページサイズを保持(コピー)します。

n
バイト数を指定します。

***n*K**
キロバイト数を指定します。

***n*M**
メガバイト数を指定します。

***n*G**
ギガバイト数を指定します。

デフォルト 現在のセッションのバッファーページサイズ

ENCODING=*encoding-value*
フォーマットカタログの作成時に使用されたセッションエンコーディングを指定します。

ターゲットセッションとは異なるエンコーディングで作成されたユーザー定義出力形式のカタログに対して、このオプションを指定します。このオプションにより、PROC MIGRATE はユーザー定義出力形式のカタログエントリをトランスコードできます。切り捨てを避けるために、必要に応じてフォーマットの長さが拡張されます。他のライブラリメンバーは移行されますが、このオプションはそ

れらのメンバーには適用されません。“例 4: カタログが異なるエンコーディングからのものである場合の移行” (1352 ページ)を参照してください。

制限事項 フォーマット名または関数名での DBCS 文字はサポートされていません。

SAS バージョン 6 ユーザー定義出力形式はサポートされません。

要件 現在のセッションのエンコーディングは、UTF-8 でなければなりません。

操作 データ表現が現在のセッションと互換性がない場合は、SAS/CONNECT も必要です。参照: “SAS/CONNECT サーバーの使用” (1344 ページ).

MOVE
ソースライブラリから元のメンバーを削除します。 ターゲットライブラリにすでにメンバーが存在する場合、メンバーはソースライブラリから削除されず、SAS ログにメッセージが送信されます。ターゲットライブラリにすでにカタログが存在する場合、カタログはソースライブラリから削除されず、ログにメッセージが送信されます。データセットに参照一貫性制約がある場合、このデータセットがソースライブラリから削除され、ログにエラーが送信されます。

操作 CVP エンジンは読み取り専用であるため、MOVE オプションをサポートしていません。

ヒント システムのスペースが制限されている場合に限り、MOVE オプションを使用します。メンバーを削除する前に、移行の確認をすることをお勧めします。

SLIBREF=*libref*
SAS/CONNECT サーバー経由で割り当てられるライブラリ参照名を指定します。

IN=オプションまたは SLIBREF=オプションで SAS/CONNECT ライブラリ参照名を指定する必要があるかどうかを判断するには、“SAS/CONNECT サーバーの使用” (1344 ページ)を参照してください。

制限事項 カタログが SAS 6 または SAS 8 で作成された場合、SLIBREF=は SAS 8 サーバー経由で割り当てる必要があります。

要件 サーバーが稼働する動作環境の種類は、ソースライブラリと同じにする必要があります。たとえば、ソースセッションが Linux で稼働している場合、サーバーも Linux で稼働している必要があります。

操作 ライブラリ内のメンバーの種類が CATALOG だけのときに SLIBREF=オプションを使用する場合は、IN=オプションを省略します。

KEEPNODUPKEY
指定すると、NODUPKEY 並べ替え順序が保持されます。 “NODUPKEY ソートインジケーターを含む SAS データセットの移行” (1346 ページ)を参照してください。

使用法: MIGRATE プロシジャ

切り捨てを回避するための CVP エンジンによる移行

より多くのバイトを使用して文字を表現するエンコーディングでファイルを処理するときに、可変長がより大きな文字サイズに対応できない場合、切り捨てが発生する可能性があります。たとえば、ある文字が wlatin1 エンコーディングでは 1 バイトとして表示されるが、UTF-8 エンコーディングでは 2 バイトとして表示される場合があります。切り捨てにより、出力にガベージ文字または不正な文字が生じることもあります。さらに、スマート引用符などの一部の Microsoft Word 文字は、UTF-8 で 1 バイト以上を必要とします。これらの文字が 1 バイトエンコーディングからトランスコードされるときに、データバッファーが十分に大きくない場合は、切り捨てが発生する可能性があります。

PROC MIGRATE は、CVP エンジンを使用して SAS データセット内の文字変数の長さを拡張することをサポートしています。IN=ライブラリに対する LIBNAME ステートメントで CVP エンジンを指定します。

デフォルトでは、CVP エンジンは乗数値を自動的に選択します。自動的な値は、通常は切り捨てを回避するのに十分な値です。CVPMULTIPLIER=オプションを使用して、自分で乗数値を指定することもできます。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*で文字データ切り捨ての回避に関する説明を参照してください。

PROC MIGRATE で CVP エンジンを使用する場合の制限は次のとおりです。

- SAS カタログは移行されますが、CVP エンジンはカタログエントリをトランスコードしたり、フォーマットで定義された長さを拡張したりしません。カタログの移行方法については、“カタログ” (1338 ページ)を参照してください。
- このエンジンは、V6 または TAPE エンジンで作成されたデータをサポートしていません。
- エンジンは読み取り専用であるため、MOVE オプションはサポートされていません。
- エンジンは、SAS/CONNECT を使用するライブラリ割り当てでは指定できません。PROC MIGRATE で CVP と SAS/CONNECT を一緒に使用する方法の詳細については、“IN=オプションまたは SLIBREF=オプションでサーバーを指定する場合” (1344 ページ)を参照してください。

CVP エンジンの詳細については、“CVP エンジンの使用による文字データ切り捨ての回避” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

SAS/CONNECT サーバーの使用

SAS/CONNECT サーバーを使用する場合

SAS/CONNECT サーバーを使用する 2 つの理由は次のとおりです。

- 直接アクセスできない場合は、サーバーを使用してコンピューター間を移行できます。
- 次の 2 つの条件が両方とも満たされる場合は、サーバーを使用する必要があります。
  - ソースライブラリにはカタログが含まれています。
  - ソースライブラリは、ターゲットセッションと互換性のないデータ表現で作成されました。Linux for x64 上の SAS Viya プラットフォームには、LINUX_X86_64 の SAS データ表現があります。次の 3 つのデータ表現は、SAS カタログと互換性があります: Linux for x64 (LINUX_X86_64)、Linux on the Power Architecture (LINUX_POWER_64)、および Solaris for x64 (SOLARIS_X86_64)。

カタログの移行方法については、“カタログ” (1338 ページ)を参照してください。

IN=オプションまたは SLIBREF=オプションでサーバーを指定する場合

- CVP エンジンを使用している場合は、SLIBREF=オプションで SAS/CONNECT サーバーのライブラリ参照名を指定します。IN=オプションで CVP ライブラリ参照名を指定します。SAS/CONNECT ライブラリ割り当てでは CVP を指定できないため、2 つのオプションの両方が必要です。参照: “例 3: 切り捨てを回避するために CVP エンジンを使用して移行する” (1350 ページ).
- CVP エンジンを使用していない場合は、IN=オプションで SAS/CONNECT サーバーのライブラリ参照名を指定します。この構文は、ソースライブラリに SAS Viya 製品または SAS 9.1.3 以降で作成されたカタログが含まれている場合にサポートされます。参照: “例 2: SAS/CONNECT を使用して SAS®9 から移行する” (1349 ページ).
- ソースライブラリにカタログが含まれていない場合は、IN=オプションで SAS/CONNECT サーバーライブラリ参照名を指定します。一部の顧客は、直接 NFS アクセスを使用する代わりに、SAS/CONNECT を使用することを選択します。
- ソースライブラリに SAS9.1.3 より前のリリースで作成されたカタログが含まれている場合は、SLIBREF=オプションで SAS/CONNECT サーバーのライブラリ参照名を指定します。

SAS/CONNECT サーバーの要件

SAS/CONNECT サーバーが稼働する動作環境とエンコーディングの種類は、ソースライブラリと同じにする必要があります。たとえば、ソースセッションが Latin1 エンコーディングを使用した Linux で稼働している場合、サーバーも Latin1 エンコーディングを使用した Linux で稼働している必要があります。

IN=オプションで指定するサーバーは、SAS 9.1.3 以降にする必要があります。

これらの要件を満たせない場合は、“サポートしていないカタログへの追加のステップ” (1346 ページ)で説明している代替方法を使用してください。

データセットの移行(監査証跡、世代、インデックス、または一貫性制約)

すべての場合において、まずデータセットが移行してから、監査証跡、世代、インデックス、一貫性制約のいずれかに処理が適用されます。エラーは次の方法で処理されます。

- 移行したデータセットのインデックス作成時にエラーが発生した場合、データセットがインデックスなしで移行されるか、処理が停止することがあります。メッセージが SAS ログに書き込まれます。インデックスの移行が失敗した場合は、エラーを解決して、ターゲットライブラリにインデックスを再作成します。

  ソースライブラリにインデックスがない場合は、環境とシステムオプションに応じて、インデックスの再作成による移行時のデータセットの修復が試行されます。データセットに、セッションエンコーディングやデータ表現との互換性がない場合は、インデックスを再作成するとエラーが発生します。エラーを解決するには、元のオペレーティングシステムを使用してソースライブラリにインデックスを再作成するか、元の場所からインデックスを移動して、PROC MIGRATE を再度サブミットします。カスタマーが OS を使用してデータセットを移動し、移動時にインデックスを含めるのに失敗すると、エラーが発生する場合があります。

- 移行したデータファイルに一貫性制約が適用されるか、または監査証跡や世代が移行されたときに、エラーが発生すると、データセットがターゲットライブラリから削除されます。ノートが SAS ログに書き込まれます。MOVE オプションを指定した場合、ソースライブラリのデータセットは削除されません。

- 参照一貫性制約を含むデータセットの場合、MOVE オプションは、移行が正常終了しても、ソースライブラリのメンバーを削除しません。メンバーを削除するには、参照一貫性制約を削除しておく必要があります。エラーメッセージが SAS ログに書き込まれます。

NODUPKEY ソートインジケーターを含む SAS データセットの移行

NODUPKEY オプションを使用して並べ替えられた SAS データセットを移行する際は、デフォルトの動作を使用することも、KEEPNODUPKEY オプションを指定することもできます。

デフォルトの動作(KEEPNODUPKEY オプションなし)では、ターゲットライブラリで SAS データセットのソートインジケーターが保持されます。ただし、NODUPKEY 属性は削除され、SAS ログに警告メッセージが書き込まれます。これがデフォルトの動作となるのは、前のリリースで NODUPKEY オプションを使用して並べ替えられた SAS データセットに、重複キーを含むオブザベーションが保持されている可能性があるためです。PROC SORT のキー変数を基準にして移行した SAS データセットをもう一度並べ替えると、キーの重複したオブザベーションを削除し、正しい属性を記録できます。

KEEPNODUPKEY オプションを指定する場合は、移行したデータを検証して、キーの重複したオブザベーションが存在するかどうかを判断する必要があります。存在する場合は、SAS データセットをもう一度並べ替えて、データと NODUPKEY 並べ替え属性をマッチさせる必要があります。

サポートしていないカタログへの追加のステップ

追加ステップを使用する場合

PROC MIGRATE では、次の状況でのカタログの移行はサポートされていません。

- 互換性のないデータ表現からカタログを移行するには、PROC MIGRATE を備えた SAS/CONNECT ソフトウェアを使用する必要があります。(詳細については、“SAS/CONNECT サーバーの使用” (1344 ページ)を参照してください。)SAS/CONNECT ソフトウェアにアクセスできない場合は、“互換性のないデータ表現からのカタログの移送ファイルの使用” (1347 ページ)の手順に従います。
- PROC MIGRATE は、カタログに保存されている保存済みでコンパイル済みのマクロを移行しません。かわりに、ソースコードをターゲットに移動し、そこでコンパイルして保存します。ベストプラクティスとして、マクロ定義のソースコードを保存するには、常に%MACRO ステートメントで SOURCE オプションを指定します。ソースコードを別のファイル(テキストファイルなど)に保存することもできます。

サポートされていないカタログが唯一の問題である場合は、カタログのみに追加の手順を使用できます。ライブラリの他のメンバーに対して PROC MIGRATE を使用すると、それらのメンバーの属性を保持できます。

互換性のないデータ表現からのカタログの移送ファイルの使用

1 ソースセッションで、PROC CPORT を使用して移送ファイルを作成します。カタログのみを含める場合は、MEMTYPE=CATALOG を指定します。(14 章, “CPORT プロシジャ,” (335 ページ)を参照してください。)

2 移送ファイルをターゲット環境に移動します。カタログの移動に RLS (SAS/CONNECT ソフトウェアの機能)を使用しないでください。使用するとエラーが発生します。バイナリ FTP、DOWNLOAD プロシジャ、NFS (Network File System)など、ファイルに直接アクセスする手法を使用する必要があります。

3 ターゲットセッションで、PROC CIMPORT を使用して移送ファイルをインポートします。(10 章, “CIMPORT プロシジャ,” (173 ページ)を参照してください。)

例: MIGRATE プロシジャ

例 1: ダイレクトアクセスで移行する

要素: PROC MIGRATE ステートメントのオプション
IN=
OUT=

詳細

この例では、次のような場合について説明します。

- ターゲット環境は、ソースライブラリに(たとえば、NFS を使用して)直接アクセスできます。
- カタログが存在する場合は、追加のオプションが必要になることがあります。これらのオプションは、この例では使用していません。

プログラム

```
libname source 'source-library-pathname';
libname target 'target-library-pathname';

proc migrate in=source out=target options;
run;
```

プログラムの説明

ソースライブラリとターゲットライブラリを割り当てます。 ライブラリパスを置き換えます。これらの場所は、SAS Viya プラットフォームのプログラミング環境である SAS Compute Server からアクセスできる必要があります。

```
libname source 'source-library-pathname';
libname target 'target-library-pathname';
```

PROC MIGRATE をサブミットし、必要なオプションを含めます。

```
proc migrate in=source out=target options;
run;
```

ログメッセージ

次の SAS ログメッセージは、CEDA が呼び出されることを示しています。通常、CEDA はインデックスをサポートしていません。その制限を回避するために、PROC MIGRATE はターゲットセッションでインデックスを再定義します。

```
NOTE: Migrating SOURCE.INVENTORY to TARGET.INVENTORY (memtype=DATA).
NOTE: Data file SOURCE.INVENTORY.DATA is in a format that is native to another host, or the
      file encoding does not match the session encoding.Cross Environment Data Access will be
      used, which might require additional CPU resources and might reduce performance.
NOTE: Simple index PRODTYPE has been defined.
NOTE: Simple index QUARTER has been defined.
NOTE: There were 23040 observations read from the data set SOURCE.INVENTORY.
NOTE: The data set TARGET.INVENTORY has 23040 observations and 11 variables.
```

次のログメッセージには CEDA メッセージが含まれていません。ソースライブラリのデータセットのデータ表現とエンコードは、ターゲットセッションと同じです。

```
NOTE: Migrating SOURCE.MYDATASETUTF8 to TARGET.MYDATASETUTF8 (memtype=DATA).
NOTE: There were 2 observations read from the data set SOURCE.MYDATASETUTF8.
NOTE: The data set TARGET.MYDATASETUTF8 has 2 observations and 4 variables.
```

次のログメッセージは、カタログが移行されず、処理が停止したことを示しています。ただし、ライブラリの他のメンバーは移行されます。

```
ERROR: File SOURCE.FORMATS.CATALOG was created for a different operating system.
WARNING: No data is available.
NOTE: The SAS System stopped processing this step because of errors.
```

例 2: SAS/CONNECT を使用して SAS®9 から移行する

要素: PROC MIGRATE ステートメントのオプション
IN=
OUT=

詳細

この例では、スポーナーは、SAS/CONNECT サーバーでセッションを開始して、リモート SAS データにアクセスします。サーバーを使用する理由については、“SAS/CONNECT サーバーの使用” (1344 ページ)を参照してください。

プログラム

```
options comamid=tcp;
%let myserver=host.name.com;
signon myserver.__1234 user=userid password='mypw';

libname source 'source-library-pathname' server=myserver.__1234;
libname target 'target-library-pathname';

proc migrate in=source out=target <options>;
run;
signoff myserver.__1234;
```

プログラムの説明

次のコードをサブミットして SAS/CONNECT スポーナーを呼び出します。
COMAMID=システムオプションは、通信アクセス方法として TCP/IP を指定します。myserver マクロ変数は、リモート SAS/CONNECT サーバーのホスト名に割り当てられます。SIGNON ステートメントは myserver マクロ変数と、その後に続く、SAS/CONNECT スポーナーがリッスンしているポート番号を参照します。ポート番号またはサービス名がマクロ変数で定義されている場合は、それを SIGNON から省略します。

```
options comamid=tcp;
%let myserver=host.name.com;
```

signon myserver.__*1234* user=*userid* password='*mypw*';

ソースライブラリとターゲットライブラリを割り当てます。 ライブラリパスを置き換えて、ソースライブラリの SERVER=に myserver マクロ変数を指定します。SIGNON ステートメントで指定されている場合は、ポート番号を含めます。ターゲットライブラリの場所は、計算サーバーからアクセスできる必要があります。

libname source '*source-library-pathname*' server=myserver.__*1234*;
libname target '*target-library-pathname*';

PROC MIGRATE をサブミットし、必要なオプションを含めます。

proc migrate in=source out=target <*options*>;
run;
signoff myserver.__*1234*;

ログメッセージ

次の SAS ログメッセージは、データセットとフォーマットカタログが移行されたことを示しています。PROC MIGRATE は、CPORT および CIMPORT プロシジャを呼び出して、カタログを移行します。通常、CEDA はカタログをサポートしていませんが、SAS/CONNECT サーバーを使用すると CEDA の制限を回避できます。サーバーは、ソースライブラリ内のデータと同じデータ表現およびエンコーディングを備えている必要があります。

```
NOTE: Migrating SOURCE.TEST to TARGET.TEST (memtype=DATA).
NOTE: There were 3 observations read from the data set SOURCE.TEST.
NOTE: The data set TARGET.TEST has 3 observations and 2 variables.
NOTE: PROC CPORT begins to transport catalog SOURCE.FORMATS
NOTE: Entry MYFMT.FORMAT has been transported.
NOTE: Entry MYABC.FORMATC has been transported.
```

例 3: 切り捨てを回避するために CVP エンジンを使用して移行する

要素: CVP engine

PROC MIGRATE ステートメントのオプション
- IN=
- OUT=
- SLIBREF=

注: “切り捨てを回避するための CVP エンジンによる移行” (1343 ページ)も参照してください。

詳細

CVP エンジンは、移行中に変数の長さを増やすことで切り捨てを防ぎます。より多くのバイトを使用して文字を表すエンコーディングに移行すると、変数の長さがより大きな文字サイズに対応できない場合、切り捨てが発生する可能性があります。

この例では、次のような場合について説明します。

- SLIBREF=引数が、ソースライブラリのカタログにアクセスする。SLIBREF=引数は、カタログのデータ表現と互換性のある SAS/CONNECT スポーナーを指定します。サーバーを使用する理由については、“SAS/CONNECT サーバーの使用” (1344 ページ)を参照してください。
- この例ではフォーマットカタログが移行されますが、フォーマットで定義された長さが十分であることを確認する必要があります。CVP エンジンは、カタログエントリをトランスコードしたり、フォーマットで定義された長さを拡張したりしません。 カタログ内のユーザー定義出力形式をトランスコードして展開するには、“例 4: カタログが異なるエンコーディングからのものである場合の移行” (1352 ページ)を参照してください。
- IN=引数は、ソースライブラリでサポートされている残りのファイルの種類にアクセスします。IN=引数は、CVP エンジンに割り当てられるライブラリ参照名を指定し、Network File System (NFS)経由でファイルに直接アクセスします。ライブラリの割り当てでは、SAS/CONNECT スポーナーは使用されません。SAS/CONNECT を使用するライブラリ割り当てでは、CVP エンジンを指定できません。NFS は、UNIX 動作環境の標準プロトコルです。NFS と動作環境に関するドキュメントを参照してください。
- 詳細については、“切り捨てを回避するための CVP エンジンによる移行” (1343 ページ)を参照してください。

次のコードをサブミットして SAS/CONNECT スポーナーを呼び出します。 COMAMID=システムオプションは、通信アクセス方法として TCP/IP を指定します。myserver マクロ変数は、リモート SAS/CONNECT サーバーのホスト名に割り当てられます。SIGNON ステートメントは myserver マクロ変数と、その後に続く、SAS/CONNECT スポーナーがリッスンしているポート番号を参照します。ポート番号またはサービス名がマクロ変数で定義されている場合は、それを SIGNON から省略します。

```
options comamid=tcp;
%let myserver=host.name.com;
signon myserver.__1234 user=userid password='mypw';
```

svrlib ライブラリ、ソースライブラリ、およびターゲットライブラリを割り当てます。 ライブラリパスを置き換えて、svrlib ライブラリの SERVER=に myserver マクロ変数を指定します。SIGNON ステートメントで指定されている場合は、ポート番号を含めます。ソースライブラリは CVP エンジンを使用し、svrlib ライブラリと同じ場所にアクセスします。ターゲットライブラリの場所は、計算サーバーからアクセスできる必要があります。

```
libname svrlib 'source-library-pathname' server=myserver.__1234;
libname source cvp 'source-library-pathname';
libname target 'target-library-pathname';
```

PROC MIGRATE をサブミットし、必要なオプションを含めます。

```
proc migrate in=source slibref=svrlib out=target <options>;
run;
signoff myserver.__1234;
```

例 4: カタログが異なるエンコーディングからのものである場合の移行

要素:

CVP engine

PROC MIGRATE ステートメントのオプション

- IN=
- OUT=
- ENCODING=

詳細

ENCODING=オプションは、ターゲットセッションとは異なるエンコーディングで作成されたユーザー定義出力形式のカタログを移行します。ENCODING=オプションを使用して、カタログが作成されたセッションのエンコードを指定します。詳細については、“ENCODING=*encoding-value*” (1341 ページ)および“カタログ” (1338 ページ)を参照してください。

このサンプルコードは SAS/CONNECT の使用方法を示すものではないことに注意してください。この例では、カタログは現在のセッションと互換性のあるデータ表現で作成されている必要があります。

サンプルデータセットとユーザー定義出力形式を作成します。 この形式が後で PRINT プロシジャで使用されると、cafe という単語が café として印刷され、latté という単語が espresso として印刷されます。形式が適切に移行されない場合、é 文字は印刷されません。このコードを latin1 セッションでサブミットします。

```
/* Create an example library. Submit this code in a latin1 session. */
libname mylib 'library-pathname';
data mylib.test;
   input drink $;
   datalines;
cafe
latté
;
run;

proc format library=mylib;
   value $translate
         'cafe' = 'café'
         'latté' = 'espresso'
   ;
run;
```

ライブラリを移行します。 ENCODING=オプションを使用すると、SAS はユーザー定義出力形式のカタログを latin1 から utf8 にトランスコードします。UTF-8 でエンコードされたターゲットセッションでこのコードをサブミットします。この例の場合、CVP エンジンは不要です。ここに示したコードは、データセット内のデータ値を拡張する必要があるユーザーを対象としています。

```
/* Migrate the library. Submit this code in a UTF-8 session. */
libname mytest.cvp 'source-library-pathname';
libname outlib 'target-library-pathname';

proc migrate in=mytest out=outlib encoding='latin1';
run;
```

移行の成功を実証します。 FMTSEARCH=オプションをサブミットして、移行されたライブラリを指定します。PROC PRINT で FORMAT ステートメントを使用して、フォーマットを適用します。

```
options fmtsearch=(outlib);

proc print data=outlib.test;
   format drink $translate.;
run;
```

フォーマットされたデータを示す Outlib.Test データセット

Obs	drink
1	café
2	espresso

37

OPTIONS プロシジャ

概要: OPTIONS プロシジャ

OPTIONS プロシジャの機能

OPTIONS プロシジャは、SAS システムオプションの現在の設定を SAS ログにリストします。

SAS システムオプションでは、SAS 出力形式による出力の制御、ファイルの処理、データセットの処理、動作環境との交互作用、単一の SAS プログラムや SAS データセットに固有ではないその他のタスクが実行されます。OPTIONS プロシジャを使用すると、オプションやオプショングループについての情報を得られます。
OPTIONS プロシジャが提供する情報の一部を次に示します。

- オプションの現在の値とその設定方法
- オプションの説明
- オプションの有効な構文、有効なオプション値および値の範囲
- システムオプションの設定場所
- サイト管理者によるオプションの制限の可否
- オプションが制限されているかどうか
- システムオプショングループに属しているシステムオプション
- 動作環境に固有のシステムオプション
- オプション値が INSERT または APPEND システムオプションによって変更されているかどうか
- OPTSAVE プロシジャで保存可能なシステムオプション

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

構文: OPTIONS プロシジャ

注: SAS システムオプションは、いくつかのドキュメントに記載されています。"他の SAS ドキュメントで説明されている SAS システムオプション" (*SAS システムオプション: リファレンス*)からすべてのシステムオプションにアクセスできます 。

PROC OPTIONS <*options*>;

ステートメント	タスク	例
"PROC OPTIONS ステートメント"	現在のシステムオプション設定を SAS ログにリストします。	Ex. 1, Ex. 2, Ex. 3, Ex. 4

PROC OPTIONS ステートメント

SAS システムオプションの現在の設定を SAS ログにリストします。

例: "例 1: 簡易形式のオプションリストを作成する" (1371 ページ)
"例 2: 単一オプションの設定を表示する" (1372 ページ)

“例 3: 拡張パス環境変数の表示” (1373 ページ)
“例 4: INSERT オプションと APPEND オプションで指定可能なオプションのリスト” (1374 ページ)

構文

PROC OPTIONS <*options*>;

オプション引数の要約

LISTGROUPS
: システムオプショングループならびに各グループの説明を表示します。

NOEXPAND
: 環境値のパスを返します。

Choose the format of the listing

DEFINE
: オプション、オプショングループおよびオプションの種類の概要説明を表示します。

EXPAND
: 指定された環境変数の値を返します。

HEXVALUE
: システムオプション文字値を 16 進値で表示します。

LOGNUMBERFORMAT
: 数値システムオプション値の表示に句読点を含めるか、または省略します。

LONG
: 各システムオプションを説明付きで別の行にリストします。

NOLOGNUMBERFORMAT
: 数値システムオプション値の表示で句読点を省略します。

SHORT
: オプションの圧縮されたリストを説明を省いて表示するよう指定します。

VALUE
: オプションの値とスコープ、さらにその値が設定された方法を表示します。

Restrict the number of options displayed

GROUP=*group-name* | GROUP=*(group-name-1 ... group-name-n)*
: *group-name* で指定した 1 つ以上のグループのオプションを表示します。

HOST
: ホストオプションのみを表示します。

LISTINSERTAPPEND
: INSERT および APPEND システムオプションで値の変更が可能なシステムオプションをリストにします。

LISTOPTSAVE

PROC OPTSAVE とともに保存できるシステムオプションをリストします。

LISTRESTRICT
サイト管理者によって制限可能なシステムオプションをリストします。

NOHOST
ポータブルオプションのみを表示します。

OPTION=*option-name* | OPTION=(*option-name-1*...*option-name-n*)
1 つ以上のシステムオプションについての情報を表示します。

RESTRICT
サイト管理者によって更新が制限されているシステムオプションを表示します。

オプション引数

DEFINE
オプション、オプショングループおよびオプションの種類の概要説明を表示します。 SAS では、オプションをいつ設定できるか、オプションを制限できるかどうか、オプションの有効値、OPTSAVE プロシジャがオプションを保存するかどうかについての情報が表示されます。

操作 SHORT を指定すると、このオプションは無視されます。

例 “例 2: 単一オプションの設定を表示する” (1372 ページ)

EXPAND
指定された環境変数の値を返します。 文字オプションの表示時に、EXPAND はオプション値の環境変数を環境変数の値に置き換えます。オプションがブール値オプション(CENTER や NOCENTER など)の場合、またはオプションの値が数値の場合、EXPAND は無視されます。

パスの表示時に、環境変数の値ではなく環境変数を使用してパスを表示します。

デフォルト NOEXPAND

制限事項 変数展開は、Windows および UNIX 動作環境でのみ有効です。

ヒント デフォルトでは、展開された変数と一緒に表示されるオプション値もあります。その他のオプション値では、PROC OPTIONS ステートメントに EXPAND オプションが必要です。オプション値がデフォルトで変数を展開するのか、EXPAND オプションが必要なのかを確認するには、PROC OPTIONS ステートメントで DEFINE オプションを使用します。PROC OPTIONS DEFINE からの出力に次の情報が表示された場合、変数値を展開するには EXPAND オプションを使用する必要があります。

```
Expansion: Environment variables, within the option value,
      are not expanded
```

参照項目: “NOEXPAND” (1360 ページ).

例 “例 3: 拡張パス環境変数の表示” (1373 ページ)

GROUP=*group-name* | GROUP=*(group-name–1 ... group-name-n*)
group-name で指定した 1 つ以上のグループのオプションを表示します。

要件 2 つ以上のグループを指定する場合は、グループ名をかっこで囲み、スペースでグループ名を区切ります。

参照項目: “システムオプショングループの情報を表示する” (1365 ページ)

HEXVALUE
システムオプション文字値を 16 進値で表示します。

HOST
ホストオプションのみを表示します。 HOST はホストオプションのみを表示します。

NOHOST はポータブルオプションのみを表示します。

参照項目: “NOHOST” (1361 ページ).

LISTINSERTAPPEND
INSERT および APPEND システムオプションで値の変更が可能なシステムオプションをリストにします。 INSERT オプションは、システムオプション値リストの最初の値として挿入される値を指定します。APPEND オプションは、システムオプション値リストの最終値として追加される値を指定します。どのシステムオプションで、値を値リストの最初に挿入したり最後に追加したりできるのかを表示するには、LISTINERTAPPEND を使用します。

参照項目: “INSERT= システムオプション” (*SAS システムオプション: リファレンス*)

“APPEND= システムオプション” (*SAS システムオプション: リファレンス*)

例 “例 4: INSERT オプションと APPEND オプションで指定可能なオプションのリスト” (1374 ページ)

LISTGROUPS
システムオプショングループならびに各グループの説明を表示します。

参照項目: “システムオプショングループの情報を表示する” (1365 ページ)

LISTOPTSAVE
PROC OPTSAVE とともに保存できるシステムオプションをリストします。

LISTRESTRICT
サイト管理者によって制限可能なシステムオプションをリストします。

参照項目: サイト管理者によって制限されているオプションをリストする “RESTRICT” (1361 ページ)オプション

LOGNUMBERFORMAT
数値システムオプション値の表示に句読点を含めるか、または省略します。 LOGNUMBERFORMAT は、ロケール固有の句読点を使用して数値システムオプション値を表示します。

NOLOGNUMBERFORMAT は、カンマやピリオドなどの句読点を使用せずに数値システムオプション値を表示します。

デフォルト NOLOGNUMBERFORMAT

参照項目: “NOLOGNUMBERFORMAT” (1361 ページ).

例 “例 2: 単一オプションの設定を表示する” (1372 ページ)

LONG
各システムオプションを説明付きで別の行にリストします。 これがデフォルトです。あるいは、説明なしの簡潔なリストを作成することもできます。

参照項目: 説明なしの簡潔なリストを作成する“SHORT” (1361 ページ)オプション

例 “例 1: 簡易形式のオプションリストを作成する” (1371 ページ)

OPTION=*option-name* | OPTION=(*option-name-1*...*option-name-n*)
概要説明、および(存在する場合は)*option-name* で指定したオプションの値を表示します。DEFINE オプションと VALUE オプションは、オプションについての詳細情報を提供します。

option-name
プロシジャへの入力として使用するオプションを指定します。

要件 SAS システムオプションで等号(PAGESIZE=など)が使用されている場合、OPTION=に対するオプションの指定時にはその等号を含めないでください。

例 “例 2: 単一オプションの設定を表示する” (1372 ページ)

NOEXPAND
環境値のパスを返します。 文字オプションの表示時に、EXPAND はオプション値の環境変数を環境変数の値に置き換えます。オプションがブール値オプション(CENTER や NOCENTER など)の場合、またはオプションの値が数値の場合、EXPAND は無視されます。

パスの表示時に、環境変数の値ではなく環境変数を使用してパスを表示します。

デフォルト NOEXPAND

制限事項 変数展開は、Windows および UNIX 動作環境でのみ有効です。

ヒント デフォルトでは、展開された変数と一緒に表示されるオプション値もあります。その他のオプション値では、PROC OPTIONS ステートメントに EXPAND オプションが必要です。オプション値がデフォルトで変数を展開するのか、EXPAND オプションが必要なのかを確認するには、PROC OPTIONS ステートメントで DEFINE オプションを使用します。PROC OPTIONS DEFINE からの出力に次の情報が表示された場合、変数値を展開するには EXPAND オプションを使用する必要があります。

```
Expansion: Environment variables, within the option value,
      are not expanded
```

参照項目: “EXPAND” (1358 ページ)

例 “例 3: 拡張パス環境変数の表示” (1373 ページ)

NOHOST
ポータブルオプションのみを表示します。 HOST はホストオプションのみを表示します。

NOHOST はポータブルオプションのみを表示します。

別名 PORTABLE

PORT

参照項目: “HOST” (1359 ページ)

NOLOGNUMBERFORMAT
数値システムオプション値の表示で句読点を省略します。
LOGNUMBERFORMAT は、ロケール固有の句読点を使用して数値システムオプション値を表示します。

NOLOGNUMBERFORMAT は、カンマやピリオドなどの句読点を使用せずに数値システムオプション値を表示します。

デフォルト NOLOGNUMBERFORMAT

参照項目: “LOGNUMBERFORMAT” (1359 ページ)

例 “例 2: 単一オプションの設定を表示する” (1372 ページ)

RESTRICT
サイト管理者によって制限オプション構成ファイルに設定されたシステムオプションを表示します。ユーザーはこれらのオプションを変更できません。
RESTRICT オプションは、制限されているオプションごとに、オプションの値、スコープおよび設定内容を表示します。

サイト管理者が一切のオプションを制限していない場合は、SAS ログに次のメッセージが表示されます。

Your Site Administrator has not restricted any SAS options.

参照項目: サイト管理者が制限できるオプションをリストする“LISTRESTRICT” (1359 ページ)オプション

SHORT
オプションの圧縮されたリストを説明を省いて表示するよう指定します。

参照項目: オプションの説明があるリストを作成する“LONG” (1360 ページ)オプション

VALUE
オプションの値とスコープ、さらにその値が設定された方法を表示します。 構成ファイルを使用してこの値が設定された場合、SAS ログにはその構成ファイルの名前が表示されます。INSERT システムオプションまたは APPEND システムオ

プションを使用してこのオプションが設定された場合、SAS ログには挿入または追加された値が表示されます。

操作 SHORT を指定すると、このオプションは無効になります。

このオプションが Threaded Kernel(TK)システムオプションのグループ内にあるときは、**How option value set** の値が **Internal** のように表示されます

注 EMAILPW などのパスワードである SAS オプションは、実際のパスワードではなく値 xxxxxxxx を返します。

例 “例 2: 単一オプションの設定を表示する” (1372 ページ)

使用法: OPTIONS プロシジャ

システムオプションリストの表示

PROC OPTIONS の実行によって生じるログは、すべての動作環境で利用可能 なオプションに対する、および単一の動作環境に特化したオプションに対するシステムオプションを表示できます。すべての動作環境で利用可能なオプションはポータブルオプションといいます。単一の動作環境に特化したオプションはホストオプションといいます。

次の例では、ポータブルオプションの設定を表示しているログの一部を示します。

```
proc options;
run;
```

例のコード 37.1 SAS システムオプションのリストの一部を示す SAS ログ

```
Portable Options:

 NOACCESSIBLECHECK Do not detect and log ODS output that is not accessible.
 NOACCESSIBLEGRAPH Do not create accessible ODS graphics by default.
 NOACCESSIBLEPDF   Do not create accessible PDF files by default.
 NOACCESSIBLETABLE Do not create accessible tables for enabled procedures, by default.

 ANIMATION=STOP    Specifies whether to start or stop animation.
 ANIMDURATION=MIN  Specifies the number of seconds that each animation frame displays.
 ANIMLOOP=YES      Specifies the number of iterations that animated images repeat.
 ANIMOVERLAY       Specifies that animation frames are overlaid in order to view all frames.
 APPEND=           Specifies an option=value pair to insert the value at the end of the existing
                   option value.
 APPLETLOC=site-specific-path
                   Specifies the location of Java applets, which is typically a URL.
 ARMAGENT=         Specifies an ARM agent (which is an executable module or keyword, such as
                   LOG4SAS) that contains a specific implementation of the ARM API.
 ARMLOC=ARMLOG.LOG Specifies the location of the ARM log.
 ARMSUBSYS=(ARM_NONE)
                   Specifies the SAS ARM subsystems to enable or disable.
 AUTOCORRECT       Automatically corrects misspelled procedure names and keywords, and global
                   statement names.
```

proc options;をサブミットすると、このログにはポータブルオプションとホストオプションの両方が表示されます。

ホストオプションのみを閲覧するには、このバージョンの OPTIONS プロシジャを使用してください。

```
proc options host;
run;
```

例のコード 37.2 ホストオプションリストの一部を表示する SAS ログ

```
Host Options:

 ALIGNSASIOFILES   Aligns SAS files on a page boundary for improved performance.
 ALTLOG=           Specifies the location for a copy of the SAS log when SAS is running in batch
                   mode.
 ALTPRINT=         Specifies the location for a copy of the SAS procedure output when SAS is
                   running in batch mode.
 AUTHPROVIDERDOMAIN=
                   Specifies the authentication provider that is associated with a domain.
 AUTHSERVER=       Specifies the domain server that finds and authenticates secure server logins.
```

1 つ以上のオプションの情報の表示

1 つ以上の特定オプションの設定を表示するには、PROC OPTIONS ステートメントで OPTION=オプションと DEFINE オプションを使用します。次の例は、PROC OPTIONS が単一 SAS システムオプションに対して作成するログを示しています。

```
proc options option=errorcheck define;
```

```
run;
```

例のコード 37.3　単一 SAS システムオプションの設定

```
 ERRORCHECK=NORMAL
Option Definition Information for SAS Option ERRORCHECK
   Group= ERRORHANDLING
   Group Description: Error messages and error conditions settings
   Description: Specifies whether SAS enters syntax-check mode when errors are found in the
            LIBNAME, FILENAME,  %INCLUDE, and LOCK statements.
   Type: The option value is of type CHARACTER
      Maximum Number of Characters: 10
      Casing: The option value is retained uppercased
      Quotes: If present during "set", start and end quotes are removed
      Parentheses: The option value does not require enclosure within parentheses.If
      present, the parentheses are retained.
      Expansion: Environment variables, within the option value, are not expanded
      Number of valid values: 2
         Valid value: NORMAL
         Valid value: STRICT
   When Can Set: Startup or anytime during the SAS Session
   Restricted: Your Site Administrator can restrict modification of this option
   Optsave: PROC Optsave or command Dmoptsave will save this option
```

2 つ以上のオプションの設定を表示するには、オプションをかっこで囲み、スペースでオプションを区切ります。

```
proc options option=(pdfsecurity pdfpassword) define;
run;
```

例のコード 37.4　2 つの SAS システムオプションの設定

```
 PDFSECURITY=NONE
Option Definition Information for SAS Option PDFSECURITY
   Group= PDF
   Group Description: PDF settings
   Group= SECURITY
   Group Description: Security settings
   Description: Specifies the level of encryption to use for PDF documents.
   Type: The option value is of type CHARACTER
      Maximum Number of Characters: 4
      Casing: The option value is retained uppercased
      Quotes: If present during "set", start and end quotes are removed
      Parentheses: The option value does not require enclosure within parentheses.If
      present, the parentheses are retained.
      Expansion: Environment variables, within the option value, are not expanded
      Number of valid values: 3
         Valid value: HIGH
         Valid value: LOW
         Valid value: NONE
   When Can Set: Startup or anytime during the SAS Session
   Restricted: Your Site Administrator can restrict modification of this option
   Optsave: PROC Optsave or command Dmoptsave will save this option
```

```
 PDFPASSWORD=xxxxxxxx
Option Definition Information for SAS Option PDFPASSWORD
   Group= PDF
   Group Description: PDF settings
   Group= SECURITY
   Group Description: Security settings
   Description: Specifies the password to use to open a PDF document and the password used by a
            PDF document owner.
   Type: The option value is of type CHARACTER
      Maximum Number of Characters: 2048
      Casing: The option value is retained with original casing
      Quotes: If present during "set", start and end quotes are removed
      Parentheses: The option value must be enclosed within parentheses.The parentheses are
      retained.
      Expansion: Environment variables, within the option value, are not expanded
      Number of valid values: 2
         Valid value: OPEN
         Valid value: OWNER
   When Can Set: Startup or anytime during the SAS Session
   Restricted: Your Site Administrator cannot restrict modification of this option
   Optsave: PROC Optsave or command Dmoptsave will not save this option
```

システムオプショングループの情報を表示する

各 SAS システムオプションは 1 つ以上のグループに属しています。このグループ分けは、エラー処理や並べ替えなどの機能に基づいています。システムオプショングループ、およびそのグループの 1 つ以上に属するシステムオプションのリストが表示されます。

システムオプショングループのリストを表示するには、LISTGROUPS オプションを使用します。

```
proc options listgroups;
run;
```

例のコード 37.5　SAS システムオプショングループのリスト

```
Option Groups
  GROUP=ANIMATION           Animation
  GROUP=CAS                 CAS Options
  GROUP=CODEGEN             Code generation
  GROUP=COMMUNICATIONS      Networking and encryption
  GROUP=DATACOM             Datacom
  GROUP=DATAQUALITY         Data Quality
  GROUP=DB2                 DB2
  GROUP=EMAIL               E-mail
  GROUP=ENVDISPLAY          Display
  GROUP=ENVFILES            Files
  GROUP=ERRORHANDLING       Error handling
  GROUP=EXECMODES           Initialization and operation
  GROUP=EXTFILES            External files
  GROUP=GRAPHICS            Driver settings
  GROUP=HELP                Help
  GROUP=IDMS                IDMS
  GROUP=IMS                 IMS
  GROUP=INPUTCONTROL        Data Processing
  GROUP=INSTALL             Installation
  GROUP=ISPF                ISPF
  GROUP=LANGUAGECONTROL     Language control
  GROUP=LISTCONTROL         Procedure output
  GROUP=LOGCONTROL          SAS log
  GROUP=LOG_LISTCONTROL     SAS log and procedure output
  GROUP=MACRO               SAS macro
  GROUP=MEMORY              Memory
  GROUP=META                Metadata
  GROUP=ODSPRINT            ODS Printing
  GROUP=PDF                 PDF
  GROUP=PERFORMANCE         Performance
  GROUP=REXX                REXX
  GROUP=SASFILES            SAS Files
  GROUP=SECURITY            Security
  GROUP=SMF                 SMF
  GROUP=SORT                Procedure options
  GROUP=SQL                 SQL
  GROUP=SVG                 SVG
  GROUP=TK                  TK
```

特定のグループに属するシステムオプションを表示するには、GROUP=オプションを使用します。1 つ以上のグループを指定できます。

```
proc options group=(svg graphics);
run;
```

例のコード 37.6 GROUP=オプションを使用したサンプル出力

```
Group=SVG
 ANIMATION=STOP    Specifies whether to start or stop animation.
 ANIMDURATION=MIN  Specifies the number of seconds that each animation frame displays.
 ANIMLOOP=YES      Specifies the number of iterations that animated images repeat.
 ANIMOVERLAY       Specifies that animation frames are overlaid in order to view all frames.
 SVGAUTOPLAY       Starts animation when the page is loaded in the browser.
 NOSVGCONTROLBUTTONS
               Does not display the paging control buttons and an index in a multipage SVG
               document.
 SVGFADEIN=0       Specifies the number of seconds for the fade-in effect for a graph.
 SVGFADEMODE=OVERLAP
               Specifies whether to use sequential frames or to overlap frames for the
               fade-in effect of a graph.
 SVGFADEOUT=0      Specifies the number of seconds for a graph to fade out of view.
 SVGHEIGHT=        Specifies the height of the viewport.Specifies the value of the height
               attribute of the outermost SVG element.
 NOSVGMAGNIFYBUTTON
               Disables the SVG magnifier tool.
 SVGPRESERVEASPECTRATIO=
               Specifies whether to force uniform scaling of SVG output.Specifies the
               preserveAspectRatio attribute on the outermost SVG element.
 SVGTITLE=         Specifies the text in the title bar of the SVG output.Specifies the value of
               the TITLE element in the SVG file.
 SVGVIEWBOX=       Specifies the coordinates, width, and height that are used to set the viewBox
               attribute on the outermost SVG element.
 SVGWIDTH=         Specifies the width of the viewport.Specifies the value of the width
               attribute of the outermost SVG element.
 SVGX=             Specifies the x-axis coordinate of one corner of the rectangular region for
               an embedded SVG element.Specifies the x attribute in the outermost SVG
               element.
 SVGY=             Specifies the y-axis coordinate of one corner of the rectangular region for
               an embedded SVG element.Specifies the y attribute in the outermost SVG
               element.

Group=GRAPHICS
 DEVICE=           Specifies the device driver to which SAS/GRAPH sends procedure output.
 GSTYLE            Uses ODS styles to generate graphs that are stored as GRSEG catalog entries.
 GWINDOW           Displays SAS/GRAPH output in the GRAPH window.
 MAPS=("!sasroot/path-to-maps")
               Specifies the location of SAS/GRAPH map data sets.
 MAPSGFK=(         "!sasroot/path-to-maps"  )
               Specifies the location of GfK maps.
 MAPSSAS=(         "!sasroot/path-to-maps"  )
               Specifies the location of SAS map data sets.
```

次のグループ名を GROUP=オプションの値として使用すると、グループ内のシステムオプションを表示できます。

ANIMATION	GRAPHICS	META
CAS	HELP	ODSPRINT
COMMUNICATIONS	INPUTCONTROL	PDF
DATAQUALITY	INSTALL	PERFORMANCE
EMAIL	LANGUAGECONTROL	SASFILES
ENVDISPLAY	LISTCONTROL	SECURITY
ENVFILES	LOGCONTROL	SORT
ERRORHANDLING	LOG_LISTCONTROL	SQL
EXECMODES	MACRO	SVG

EXTFILES	MEMORY	TK

次のグループを使用して動作環境固有の値一覧を表示できます。この値は、GROUP=オプションを PROC OPTIONS と共に使用すると取得できる場合があります。

ADABAS	IDMS	REXX
CODEGEN	IMS	SMF
DATACOM	ISPF	
DB2	ORACLE	

動作環境の情報: これらのホスト固有のオプションの詳細については、動作環境に関する SAS ドキュメントを参照してください。

制限オプションの表示

サイト管理者は一部のシステムオプションを制限して、サイトに対して設定されたオプションに SAS セッションを適合させることができます。制限オプションを変更できるのはサイト管理者だけです。OPTIONS プロシジャでは、制限オプションについての情報を表示するオプションが 2 つ提供されます。RESTRICT オプションは、サイト管理者が制限したシステムオプションをリストします。LISTRESTRICT オプションは、サイト管理者による制限が可能なオプションをリストします。制限できないオプションのリストについては、*“制限されたオプション” (SAS システムオプション: リファレンス)*を参照してください。

次の SAS ログには、RESTRICT オプション指定時の出力、および LISTRESTRICT オプション指定時の出力の一部が表示されています。

```
proc options restrict;
  run;
```

例のコード 37.7　サイト管理者によって制限されているオプションのリスト

```
Option Value Information For SAS Option CMPOPT
   Option Value: (NOEXTRAMATH NOMISSCHECK NOPRECISE NOGUARDCHECK
NOGENSYMNAMES NOFUNCDIFFERENCING)
   Option Scope: SAS Session
   How option value set:  Site Administrator Restricted
```

```
proc options listrestrict;
  run;
```

例のコード 37.8 制限可能なオプションをリストするログの一部

```
Your Site Administrator can restrict the ability to modify the following Portable Options:

 ACCESSIBLECHECK    Detect and log ODS output that is not accessible.
 ACCESSIBLEGRAPH    Create accessible ODS graphics by default.
 ACCESSIBLEPDF      Create accessible PDF files by default.
 ACCESSIBLETABLE    Create accessible tables for enabled procedures, by default.
 ANIMATION          Specifies whether to start or stop animation.
 ANIMDURATION       Specifies the number of seconds that each animation frame displays.
 ANIMLOOP           Specifies the number of iterations that animated images repeat.
 ANIMOVERLAY        Specifies that animation frames are overlaid in order to view all frames.
 APPLETLOC          Specifies the location of Java applets, which is typically a URL.
 ARMAGENT           Specifies an ARM agent (which is an executable module or keyword, such
                    as LOG4SAS) that contains a specific implementation of the ARM API.
 ARMLOC             Specifies the location of the ARM log.
 ARMSUBSYS          Specifies the SAS ARM subsystems to enable or disable.
 AUTOCORRECT        Automatically corrects misspelled procedure names and keywords, and
                    global statement names.
 AUTOSAVELOC        Specifies the location of the Program Editor auto-saved file.
```

保存可能オプションの表示

PROC OPTSAVE を使用すれば、多くのシステムオプションを保存できます。これらのオプションは、後で、PROC OPTLOAD を使用して復元できます。PROC OPTIONS の LISTOPTSAVE オプションを使用すれば、保存して後で復元できるシステムオプションを表示できます。

注: PROC OPTSAVE は、次の種類のシステムオプションを保存しません。

- SAS 起動オプション
- SAS 環境変数オプション
- パスワードを含む SAS システムオプション
- 読み取り専用で設定できない SAS システムオプション

次の SAS ログは、PROC OPTSAVE を使用して保存できるオプションのリストの一部を示しています。

```
proc options listoptsave;
   run;
```

例のコード 37.9　保存可能なシステムオプションのリストの一部

```
Core options that can be saved with OPTSAVE

 ACCESSIBLECHECK    Detect and log ODS output that is not accessible.
 ACCESSIBLEGRAPH    Create accessible ODS graphics by default.
 ACCESSIBLEPDF      Create accessible PDF files by default.
 ACCESSIBLETABLE    Create accessible tables for enabled procedures, by default.
 ANIMATION          Specifies whether to start or stop animation.
 ANIMDURATION       Specifies the number of seconds that each animation frame displays.
 ANIMLOOP           Specifies the number of iterations that animated images repeat.
 ANIMOVERLAY        Specifies that animation frames are overlaid in order to view all frames.
 APPLETLOC          Specifies the location of Java applets, which is typically a URL.
 AUTOCORRECT        Automatically corrects misspelled procedure names and keywords, and
                    global statement names.
 AUTOSAVELOC        Specifies the location of the Program Editor auto-saved file.
 AUTOSIGNON         Enables a SAS/CONNECT client to automatically submit the SIGNON command
                    remotely with the RSUBMIT command.
 BINDING            Specifies the binding edge type of duplexed printed output.
 BOMFILE            Writes the byte order mark (BOM) prefix when a Unicode-encoded file is
                    written to an external file.
 BOTTOMMARGIN       Specifies the size of the margin at the bottom of a printed page.
 BUFNO              Specifies the number of buffers for processing SAS data sets.
 BUFSIZE            Specifies the size of a buffer page for output SAS data sets.
```

オプションが設定された場所を特定する

PROC OPTIONS ステートメントで VALUE オプションを指定して、オプションのスコープと、オプションの設定に使用されたメソッドを識別することができます。オプションの値が複数のメソッドを使用して設定されている場合(たとえば、OPTIONS ステートメント内または複数の構成ファイル内)、VALUE オプションは、有効な値を決定するメソッドを識別します。メソッドが構成ファイルの場合、VALUE オプションはファイルパスを識別します。

次の PROC OPTIONS ステートメントを使用して、VALUE オプションを指定します。

```
proc options option=option-name value;
```

詳細については、37 章, “OPTIONS プロシジャ,” (1355 ページ)を参照してください。

結果: OPTIONS プロシジャ

SAS ログに出力された PROC OPTIONS を表示する

SAS では、SAS ログにオプションリストが書き込まれます。

option | NO*option* という形式の SAS システムオプションは、現在の設定に応じて、*option* か NO*option* のどちらかとしてリストされます。これらは常に肯定形式で並べ替えられます。たとえば、NOCAPS は C にリストされます。

OPTIONS プロシジャは、SAS ログでパスワードを表示する際、実際の長さにかかわらず 8 個の X で表示します。

動作環境の情報: PROC OPTIONS は、SAS システムの実行環境に固有の詳細情報を作成します。その詳細と、ホスト固有のオプションの説明については、動作環境に関する SAS ドキュメントを参照してください。

例: OPTIONS プロシジャ

例 1: 簡易形式のオプションリストを作成する

要素: PROC OPTIONS ステートメントオプション
SHORT

詳細

この例では、SAS システムオプション設定の簡易形式のリストの生成方法を示します。この簡易形式を“システムオプションリストの表示” (1362 ページ)に示された長い形式と比較してください。

プログラム

すべてのオプションとその設定をリストします。 SHORT は、SAS システムオプションとその設定を説明なしでリストします。

```
proc options short;
run;
```

ログ

例のコード 37.10　SHORT オプションのリストの一部

```
Portable Options:

 NOACCESSIBLECHECK NOACCESSIBLEGRAPH NOACCESSIBLEPDF NOACCESSIBLETABLE ANIMATION=STOP
ANIMDURATION=MIN ANIMLOOP=YES ANIMOVERLAY APPEND=
APPLETLOC=//bb03smb01/sasgen/dev/mva-v940m6/avdobj/jar/wx6no ARMAGENT= ARMLOC=ARMLOG.LOG
ARMSUBSYS=(ARM_NONE) AUTOCORRECT AUTOEXEC= AUTOSAVELOC= NOAUTOSIGNON BINDING=DEFAULT BOMFILE
BOTTOMMARGIN=0.000 IN BUFNO=1 BUFSIZE=0 BYERR BYLINE BYSORTED NOCAPS NOCARDIMAGE CASAUTHINFO=
CASDATALIMIT=100M CASHOST= CASLIB= CASNCHARMULTIPLIER=1 CASNWORKERS=ALL CASPORT=0 CASSESSOPTS=
CASTIMEOUT=60 CASUSER= CATCACHE=0 CBUFNO=0 CENTER CGOPTIMIZE=3 NOCHARCODE NOCHKPTCLEAN CLEANUP
NOCMDMAC CMPLIB= CMPMODEL=BOTH CMPOPT=(NOEXTRAMATH NOMISSCHECK NOPRECISE NOGUARDCHECK
NOGENSYMNAMES NOFUNCDIFFERENCING SHORTCIRCUIT NOPROFILE) NOCOLLATE COLOPHON= COLORPRINTING
```

例 2: 単一オプションの設定を表示する

要素:

PROC OPTIONS ステートメントオプション
- OPTION=
- DEFINE
- LOGNUMBERFORMAT
- VALUE

詳細

この例では、単一 SAS システムオプションの設定の表示方法を示します。ログには、SAS システムオプション MAXMEMQUERY の現在の設定が表示されています。DEFINE オプションと VALUE オプションは、詳細情報を表示します。LOGNUMBERFORMAT は、カンマを使用して値を表示します。

プログラム

MAXMEMQUERY SAS システムオプションを指定します。
OPTION=MAXMEMQUERY は、オプション値情報を表示します。DEFINE と VALUE は、詳細情報を表示します。LOGNUMBERFORMAT は、カンマを使用した出力形式を値に適用するように指定します。

```
proc options option=maxmemquery define value lognumberformat;
run;
```

ログ

例のコード 37.11 MAXMEMQUERY オプションの指定によるログ出力

```
Option Value Information For SAS Option MAXMEMQUERY
   Value: 268,435,456
   Scope: SAS Session
   How option value set: Shipped Default
Option Definition Information for SAS Option MAXMEMQUERY
   Group= MEMORY
   Group Description: Memory settings
   Description: For certain procedures, specifies the maximum amount of memory that can be allocated per request.
   Type: The option value is of type LONG
      Range of Values: The minimum is 0 and the maximum is 9007199254740992
      Valid Syntax(any casing): MIN|MAX|n|nK|nM|nG|nT|hexadecimal
   Numeric Format: Usage of LOGNUMBERFORMAT impacts the value format
   When Can Set: Startup or anytime during the SAS Session
   Restricted: Your Site Administrator can restrict modification of this option
   Optsave: PROC Optsave or command Dmoptsave will save this option
```

例 3: 拡張パス環境変数の表示

要素:

PROC OPTIONS ステートメントオプション
- OPTION=
- EXPAND
- NOEXPAND
- HOST

詳細

この例では、パスを表示する際の環境変数の値を示します。

プログラム

環境変数の値を表示します。 EXPAND オプションでは、環境変数の代わりに環境変数の値が表示されます。NOEXPAND オプションでは、環境変数が表示されます。この例では、環境変数は!sasroot です。

```
proc options option=msg expand;
run;
proc options option=msg noexpand;
run;
```

ログ

例のコード 37.12　OPTIONS プロシジャによる展開パス名と非展開パス名の表示

```
82   proc options option=msg expand;
83   run;
   SAS (r) Proprietary Software Release
 MSG=/opt/sas/viya/home/SASFoundation/sasmsg
             Specifies the path to the library that contains SAS messages.
NOTE: PROCEDURE OPTIONS used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds

84   proc options option=msg noexpand;
85   run;
   SAS (r) Proprietary Software Release
 MSG=!SASROOT/sasmsg
             Specifies the path to the library that contains SAS messages.
```

例 4: INSERT オプションと APPEND オプションで指定可能なオプションのリスト

要素:　PROC OPTIONS ステートメントオプション
　　LISTINSERTAPPEND

詳細

この例では、INSERT および APPEND システムオプションで指定可能なオプションの表示方法を示します。

プログラム

SAS Viya プラットフォームで INSERT オプションと APPEND オプションによって指定できるオプションをすべてリストします。 LISTINSERTAPPEND オプションは、これらのオプションのリストと説明を提供します。

```
proc options listinsertappend;
run;
```

ログ

例のコード 37.13　INSERT オプションと APPEND オプションで指定可能なオプションの表示

```
Core options that can utilize INSERT and APPEND
 AUTOEXEC          Specifies the location of the SAS AUTOEXEC files.
 CMPLIB           Specifies one or more SAS data sets that contain compiler subroutines to include during compilation.
 FMTSEARCH         Specifies the order in which format catalogs are searched.
 MAPSGFK          Specifies the location of GfK maps.
 SASAUTOS          Specifies the location of one or more autocall libraries.
 SASHELP          Specifies the location of the Sashelp library.
 SASSCRIPT         Specifies one or more locations of SAS/CONNECT server sign-on script files.
Host options that can utilize INSERT and APPEND
 MSG              Specifies the path to the library that contains SAS messages.
 SET              Defines an environment variable.
```

38

OPTLOAD プロシジャ

概要: OPTLOAD プロシジャ

OPTLOAD プロシジャの動作について

OPTLOAD プロシジャは、SAS レジストリや SAS データセットに保存されている SAS システムオプション設定を読み取り、有効化します。

PROC OPTLOAD ステートメントを使用して、SAS データセットまたはレジストリキーから SAS システムオプション設定をロードできます。

オプションがサイト管理者によって制限されており、なおかつ PROC OPTLOAD によって設定されるオプション値がサイト管理者の設定したオプション値と異なる場合、ログに警告メッセージが出力されます。

構文: OPTLOAD プロシジャ

PROC OPTLOAD <*options*>;

ステートメント	タスク	例
“PROC OPTLOAD ステートメント”	SAS レジストリや SAS データセットに保存されている SAS システムオプション設定を使用します。	Ex. 1

PROC OPTLOAD ステートメント

SAS レジストリや SAS データセットに保存されている SAS システムオプションの保存設定をロードします。

構文

PROC OPTLOAD <*options*>;

オプション引数の要約

DATA=*libref.dataset*
: 既存のデータセットから SAS システムオプション設定をロードします。

KEY="*SAS registry key*"
: 既存のレジストリキーから SAS システムオプション設定をロードします。

オプション引数

DATA=*libref.dataset*
: SAS システムオプション設定のロード元のライブラリとデータセットの名前を指定します。SAS 変数 OPTNAME には SAS システムオプション名の文字値が含まれ、SAS 変数 OPTVALUE には SAS システムオプション設定の文字値が含まれます。

  デフォルト
  : DATA=オプションと KEY=オプションを省略すると、プロシジャはデフォルトの SAS ライブラリとデータセットを使用します。デフォルトライブラリは、現在のユーザープロファイルが存在する場所です。ライブラリを指定しなければ、デフォルトライブラリは SASUSER です。SASUSER が別のアクティブな SAS セッションで使用されている場合は、一時 WORK ライブラリが、データセットのロード元のデフォルトの場所になります。デフォルトのデータセット名は MYOPTS です。

  要件
  : SAS ライブラリとデータセットが存在する必要があります。

KEY="*SAS registry key*"
: 保存された SAS システムオプション設定の SAS レジストリの場所を指定します。レジストリは SASUSER で保持されます。SASUSER が使用できない場合は、

一時 WORK ライブラリが使用されます。たとえば、KEY="OPTIONS"は、OPTIONS レジストリキーからシステムオプションをロードします。

要件 "*SAS registry key*"は、既存の SAS レジストリキーにする必要があります。

"*SAS registry key*"名は引用符で囲んでください。複数のキー名を並べる場合は、バックスラッシュ(\)で名前を区切ります。たとえば、KEY="CORE\OPTIONS"は、CORE\OPTIONS レジストリキーからシステムオプションをロードします。

例: 保存済みシステムオプションのデータセットのロード

要素: PROC OPTLOAD ステートメントのオプション
DATA=

詳細

この例では、OPTSAVE プロシジャを使用して現在のシステムオプション設定を保存し、YEARCUTOFF システムオプションを変更して、元のシステムオプションセットをロードします。

プログラム

```
libname mysas "/your-file-path";

proc options option=yearcutoff;
run;

proc optsave out=mysas.options;
run;

options yearcutoff=2000;

proc options option=yearcutoff;
run;

proc optload data=mysas.options;
run;


proc options option=yearcutoff;
run;
```

プログラムの説明

YEARCUTOFF オプションの表示を可能にするには、これらのステートメントとプロシジャを SAS プログラムとして実行するのではなく 1 つずつサブミットします。

ライブラリ参照名を割り当てます。

```
libname mysas "/your-file-path";
```

YEARCUTOFF=システムオプションの値を表示します。

```
proc options option=yearcutoff;
run;
```

現在のシステムオプション設定を mysas.options に保存します。

```
proc optsave out=mysas.options;
run;
```

OPTIONS ステートメントを使用して、YEARCUTOFF=システムオプションを値 2000 に設定します。

```
options yearcutoff=2000;
```

YEARCUTOFF=システムオプションの値を表示します。

```
proc options option=yearcutoff;
run;
```

保存したシステムオプション設定をロードします。

```
proc optload data=mysas.options;
run;
```

YEARCUTOFF=システムオプションの値を表示します。 保存したシステムオプション設定をロードすると、YEARCUTOFF=オプションの値が元の値に戻ります。

```
proc options option=yearcutoff;
run;
```

ログ

例のコード 38.1 PROC OPTLOAD を使用してオプションをロードした後の YEARCUTOFF=値を示す SAS ログ

```
1    libname mysas "/your-file-path";
NOTE: Libref MYSAS was successfully assigned as follows:
      Engine:       V9
      Physical Name: /your-file-path

2    proc options option=yearcutoff;
3    run;

    SAS (r) Proprietary Software Release

 YEARCUTOFF=1940   Specifies the first year of a 100-year span that is used by date informats
                   and functions to read a two-digit year.

NOTE: PROCEDURE OPTIONS used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds


4    proc optsave out=mysas.options;
5    run;

NOTE: The data set MYSAS.OPTIONS has 259 observations and 2 variables.
NOTE: PROCEDURE OPTSAVE used (Total process time):
      real time          0.03 seconds
      cpu time           0.03 seconds

6    options yearcutoff=2000;

7    proc options option=yearcutoff;
8    run;

    SAS (r) Proprietary Software Release

 YEARCUTOFF=2000   Specifies the first year of a 100-year span that is used by date informats
                   and functions to read a two-digit year.

NOTE: PROCEDURE OPTIONS used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds


9    proc optload data=mysas.options;
10   run;

NOTE: PROCEDURE OPTLOAD used (Total process time):
      real time          0.06 seconds
      cpu time           0.01 seconds
```

```
11   proc options option=yearcutoff;
12   run;

    SAS (r) Proprietary Software Release

 YEARCUTOFF=1940   Specifies the first year of a 100-year span that is used by date informats
                   and functions to read a two-digit year.

NOTE: PROCEDURE OPTIONS used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

39

OPTSAVE プロシジャ

概要: OPTSAVE プロシジャ

OPTSAVE プロシジャの動作について

PROC OPTSAVE は、SAS レジストリや SAS データセットに現在の SAS システムオプション設定を保存します。

SAS システムオプションは、SAS セッションをまたいで保存できます。PROC OPTSAVE ステートメントを使用して、SAS システムオプションの設定を SAS データセットまたはレジストリキーに保存できます。

構文: OPTSAVE プロシジャ

PROC OPTSAVE <*options* >;

ステートメント	タスク	例
“PROC OPTSAVE ステートメント”	現在の SAS システムオプション設定を SAS レジストリや SAS データセットに保存します。	Ex. 1

PROC OPTSAVE ステートメント

現在の SAS システムオプション設定を SAS レジストリや SAS データセットに保存します。

構文

PROC OPTSAVE <*options* >;

オプション引数の要約

KEY="*SAS registry key*"
: SAS システムオプション設定をレジストリキーに保存します。

OUT=*libref.dataset*
: SAS システムオプション設定を SAS データセットに保存します。

オプション引数

KEY="*SAS registry key*"
: 保存された SAS システムオプション設定の SAS レジストリの場所を指定します。レジストリは SASUSER で保持されます。SASUSER が使用できない場合は、一時 WORK ライブラリが使用されます。たとえば、KEY="OPTIONS"により、システムオプションが OPTIONS レジストリキーに保存されます。

制限事項
: 複数行にわたる“*SAS registry key*”名は使用できません。

要件
: 複数のキー名を並べる場合は、バックスラッシュ(\)で名前を区切ります。個々のキー名には、バックスラッシュ以外の任意の文字を含められます。

キー名の長さは 255 文字(バックスラッシュを含む)を超えないようにしてください。

"*SAS registry key*"名は引用符で囲んでください。

ヒント サブキーを指定するには、ルートキーで始まる複数のキー名を入力します。

注意 **キーがすでに存在する場合は、上書きされます。** 指定したキーが現在の SAS レジストリに存在していない場合は、オプション設定が SAS レジストリに保存されるときに、自動的にキーが作成されます。

OUT=*libref.dataset*
SAS システムオプション設定の保存先のライブラリとデータセットの名前を指定します。SAS 変数 OPTNAME には、SAS システムオプション名の文字値が含まれます。SAS 変数 OPTVALUE には、SAS システムオプション設定の文字値が含まれます。

デフォルト OUT=オプションと KEY=オプションを省略すると、プロシジャはデフォルトの SAS ライブラリとデータセットを使用します。デフォルト SAS ライブラリは、現在のユーザープロファイルが存在する場所です。SAS ライブラリを指定しなければ、デフォルトライブラリは SASUSER です。SASUSER が別のアクティブな SAS セッションで使用されている場合は、一時 WORK ライブラリが、データセットを保存するデフォルトの場所になります。デフォルトのデータセット名は MYOPTS です。

注意 **データセットがすでに存在する場合は、上書きされます。**

使用法: OPTSAVE プロシジャ

単一オプションが保存可能かを指定する

オプションを保存できるかどうかを確認するには、OPTIONS プロシジャに DEFINE を定義します。ログの出力では、**Optsave:**で始まる行は、オプションを保存できることを示しています。

```
proc options option=pageno define;
run;
```

例のコード 39.1　OPTIONS プロシジャの DEFINE オプションの出力を表示している SAS ログ

```
Option Definition Information for SAS Option PAGENO
   Group= LISTCONTROL
   Group Description: Procedure output and display settings
   Description: Resets the SAS output page number.
   Type: The option value is of type LONG
        Range of Values: The minimum is 1 and the maximum is 2147483647
        Valid Syntax(any casing): MIN|MAX|n|nK|nM|nG|nT|hexadecimal
   Numeric Format: Usage of LOGNUMBERFORMAT impacts the value format
   When Can Set: Startup or anytime during the SAS Session
   Restricted: Your Site Administrator can restrict modification of this option
   Optsave: PROC Optsave or command Dmoptsave will save this option
```

保存可能なオプションのリストの作成

一部のシステムオプションは保存できません。

注: PROC OPTSAVE は、次の種類のシステムオプションを保存しません。

- SAS 起動オプション
- SAS 環境変数オプション
- パスワードを含む SAS システムオプション
- 読み取り専用で設定できない SAS システムオプション

保存可能なオプションのリストを作成するには、次の SAS コードをサブミットします。

```
proc options listoptsave;
run;
```

保存可能なオプションのリストの一部をここに示します。

例のコード 39.2 保存可能なオプションのリストの一部

```
Core options that can be saved with OPTSAVE

 ACCESSIBLECHECK    Detect and log ODS output that is not accessible.
 ACCESSIBLEGRAPH    Create accessible ODS graphics by default.
 ACCESSIBLEPDF      Create accessible PDF files by default.
 ACCESSIBLETABLE    Create accessible tables for enabled procedures, by default.
 ANIMATION          Specifies whether to start or stop animation.
 ANIMDURATION       Specifies the number of seconds that each animation frame displays.
 ANIMLOOP           Specifies the number of iterations that animated images repeat.
 ANIMOVERLAY        Specifies that animation frames are overlaid in order to view all frames.
 APPLETLOC          Specifies the location of Java applets, which is typically a URL.
 AUTOCORRECT        Automatically corrects misspelled procedure names and keywords, and
                    global statement names.
 AUTOSAVELOC        Specifies the location of the Program Editor auto-saved file.
 AUTOSIGNON         Enables a SAS/CONNECT client to automatically submit the SIGNON command
                    remotely with the RSUBMIT command.
 BINDING            Specifies the binding edge type of duplexed printed output.
 BOMFILE            Writes the byte order mark (BOM) prefix when a Unicode-encoded file is
                    written to an external file.
 BOTTOMMARGIN       Specifies the size of the margin at the bottom of a printed page.
 BUFNO              Specifies the number of buffers for processing SAS data sets.
 BUFSIZE            Specifies the size of a buffer page for output SAS data sets.
 BYERR              SAS issues an error message and stops processing if the SORT procedure
                    attempts to sort a _NULL_ data set.
 BYLINE             Prints the BY line above each BY group.
 BYSORTED           Requires observations in one or more data sets to be sorted in
                    alphabetic or numeric order.
 CAPS               Converts certain types of input, and all data lines, into uppercase
                    characters.
 CARDIMAGE          Processes SAS source code and data lines as 80-byte records.
 CBUFNO             Specifies the number of extra page buffers to allocate for each open SAS
                    catalog.
 CENTER             Center SAS procedure output.
```

例: データセットのシステムオプションの保存

要素: PROC OPTSAVE ステートメントのオプション
OUT=

詳細

この例では、OPTSAVE プロシジャを使用して現在のシステムオプション設定を保存します。

プログラム

```
libname mysas "/your-file-path";

proc optsave out=mysas.options;
run;
```

プログラムの説明

ライブラリ参照名を作成します。

```
libname mysas "/your-file-path";
```

現在のシステムオプション設定を保存します。

```
proc optsave out=mysas.options;
run;
```

ログ

例のコード 39.3　PROC OPTSAVE の処理を示す SAS ログ

```
1    libname mysas "/your-file-path";
NOTE: Libref MYSAS was successfully assigned as follows:
      Engine:       V9
      Physical Name: /your-file-path

2    proc optsave out=mysas.options;
3    run;

NOTE: The data set MYSAS.OPTIONS has 289 observations and 2 variables.
NOTE: PROCEDURE OPTSAVE used (Total process time):
      real time          0.03 seconds
      cpu time           0.03 seconds
```

40

PRINT プロシジャ

概要: PRINT プロシジャ

PRINT プロシジャの機能

PRINT プロシジャはすべての 変数または一部の変数を使用して、SAS データセットの行またはテーブルのオブザベーションを印刷します。簡易テーブルから、データの分類および数値変数に関する合計や小計の計算を行う高度にカスタマイズされたレポートに至るまで、さまざまなレポートの作成が可能です。

単純なレポート

次の出力に、生成可能な最も単純な種類のレポートを示します。出力を生成するステートメントは次のとおりです。“例 2: 印刷する変数の選択” (1444 ページ)では、データセット EXPREV を作成します。

```
options obs=10;
proc print data=exprev;
run;
```

> ヒント　OBS=システムオプションは、現在の SAS セッション中のすべてのステップに対して、または設定を変更するまで有効です。1 つの PROC ステップのオブザベーション数を設定するには、OBS=データセットオプションを使用します。
>
> ```
> proc print data=exprev(obs=10);
> ```
>
> 詳細については、“OBS= データセットオプション” (*SAS データセットオプション: リファレンス*)および“OBS= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください

Obs	Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
1	Antarctica	99999999	1/1/18	1/7/18	Internet	2	92.6	20.70
2	Puerto Rico	99999999	1/1/18	1/5/18	Catalog	14	51.2	12.10
3	Virgin Islands (U.S.)	99999999	1/1/18	1/4/18	In Store	25	31.1	15.65
4	Aruba	99999999	1/1/18	1/4/18	Catalog	30	123.7	59.00
5	Bahamas	99999999	1/1/18	1/4/18	Catalog	8	113.4	28.45
6	Bermuda	99999999	1/1/18	1/4/18	Catalog	7	41.0	9.25
7	Belize	120458	1/2/18	1/2/18	In Store	2	146.4	36.70
8	British Virgin Islands	99999999	1/2/18	1/5/18	Catalog	11	40.2	20.20
9	Canada	99999999	1/2/18	1/5/18	Catalog	100	11.8	5.00
10	Cayman Islands	120454	1/2/18	1/2/18	In Store	20	71.0	32.30

次の例では、Sashelp.cars データセットのサブセットとして CAS テーブル Mycas.Cars を作成します。

```
options cashost="cloud.example.com" casport=5555;
cas casauto;
libname mycas cas;

proc casutil outcaslib="casuser";
load data=sashelp.cars replace;
run;

data mycas.cars;
  set mycas.cars(where=(weight>6000));
  keep make model type;
run;

proc print data=mycas.cars;
  title "Cars Greater Than 6000 Pounds";
run;
```

Cars Greater Than 6000 Pounds

Obs	Make	Model	Type
1	Ford	Excursion 6.8 XLT	SUV
2	Hummer	H2	SUV
3	GMC	Yukon XL 2500 SLT	SUV

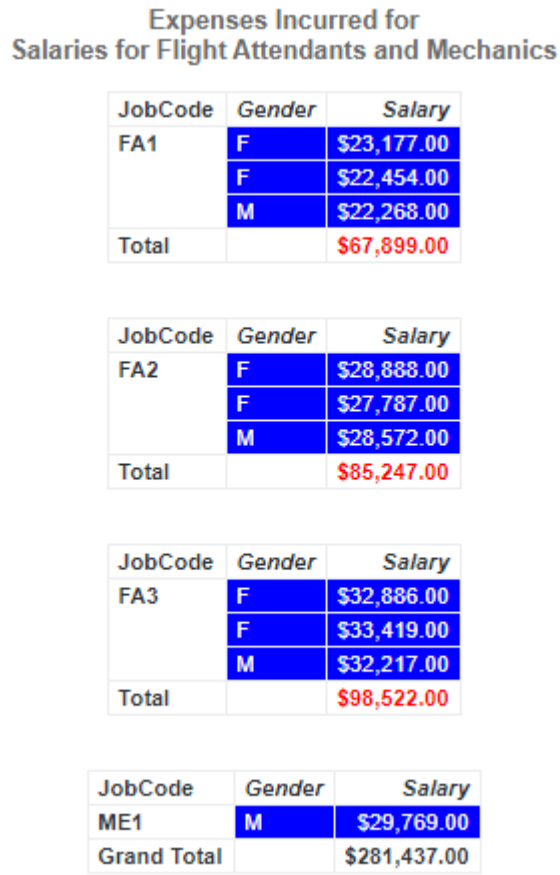
レポートのカスタマイズ

次の HTML レポートは、ODS を使用して PROC PRINT によって生成されるカスタマイズされたレポートです。このレポートを作成するステートメントを使用して、次のことを行います。

- タイトルと列ヘッダーをカスタマイズする
- レポートの表示をカスタマイズする
- 数値出力でドル記号とカンマを使用する
- 変数の順序をレポートに選択的に含めて制御する
- JobCode 別にデータをグループ化する
- 各ジョブコードおよびすべてのジョブコードの給与の値を合計する
- 要約行と総計行のラベルを追加します。

このレポートを作成するプログラムの説明については、“プログラム: STYLE オプションを使用して HTML5 レポートを作成する” (1479 ページ)を参照してください。

図 40.1　ODS を使用して PROC PRINT によって作成された、カスタマイズされたレポート



Expenses Incurred for
Salaries for Flight Attendants and Mechanics

JobCode	Gender	Salary
FA1	F	$23,177.00
	F	$22,454.00
	M	$22,268.00
Total		$67,899.00

JobCode	Gender	Salary
FA2	F	$28,888.00
	F	$27,787.00
	M	$28,572.00
Total		$85,247.00

JobCode	Gender	Salary
FA3	F	$32,886.00
	F	$33,419.00
	M	$32,217.00
Total		$98,522.00

JobCode	Gender	Salary
ME1	M	$29,769.00
Grand Total		$281,437.00

構文: PRINT プロシジャ

操作: 一般的なプラクティスは、PROC PRINT BY ステートメントを使用する前に PROC SORT を使用してデータセットを並べ替えることです。

PROC SORT を使用して VARCHAR 変数を含む CAS テーブルを並べ替えると、VARCHAR 変数は CHAR 変数に変換されます。

注: CAS テーブルでは、PROC PRINT で VARCHAR データ型がサポートされています。

ヒント: 各パスワードと暗号化キーオプションは別個の行でコード化し、ログに適切に書き込まれるようにしてください。

Output Delivery System をサポートします。詳細については、“Output Delivery System: 基本概念” (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

ATTRIB、FORMAT、LABEL、TITLE、WHERE ステートメントを使用できます。

SAS には、視覚障害者が PROC PRINT 出力にアクセスできることを確認するためのチェックが含まれています。ACCESSIBLECHECK システムオプションを設定して、出力にアクセスできるかどうかを SAS に確認させることができます。アクセス可能な出力の作成に関するベストプラクティスについては、 ODS および ODS Graphics を使用した SAS Viya プラットフォームのアクセス可能な出力の作成を参照してください。

PROC PRINT <*options*>;
  ATTRIB *variable-list(s) attribute-list(s)*;
  BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* ...>
   <NOTSORTED>;
    PAGEBY *BY-variable*;
    SUMBY *BY-variable*;
  FORMAT *variable-1* <...*variable-n*> <*format*> <DEFAULT=*default-format*>;
  ID *variable(s)*
   </ STYLE <(*location(s)*)>=<*style-override*>>;
  LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;
  SUM *variable(s)*
   </ STYLE <(*location(s)*)>=<*style-override*>>;
  VAR *variable(s)*
   </ STYLE <(*location(s)*)>=<*style-override*> >;
    WHERE *where-expression-1*
     <*logical-operator where-expression-n*>;

ステートメント	タスク	例
“PROC PRINT ステートメント”	データセットのオブザベーションを印刷します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 6, Ex. 9

ステートメント	タスク	例
“BY ステートメント”	BY グループごとに別のレポートセクションを生成します	Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 9
“ID ステートメント”	オブザベーション番号ではなく、リストする変数のフォーマットされた値によってオブザベーションを識別します	Ex. 8
“PAGEBY ステートメント”	ページが埋まる前に発生するページ排出を制御します	Ex. 4
“SUMBY ステートメント”	レポートに表示する合計数を制限します	Ex. 5, Ex. 6, Ex. 7, Ex. 9
“SUM ステートメント”	数値変数の合計値	Ex. 7
“VAR ステートメント”	レポートに表示する変数を選択し、その順序を決定します	Ex. 2, Ex. 3, Ex. 9

PROC PRINT ステートメント

一部の変数またはすべての変数を使用して SAS データセットのオブザベーションを印刷します。

構文

PROC PRINT <*options*>;

オプション引数の要約

CONTENTS=*link-text* <#BYLINE> <#BYVAL> <#BYVAR>
: 目次のリンク用テキストを指定します。

DATA=*SAS-data-set*
: 印刷する SAS データセットを指定します。

Control column format

GRANDTOTAL_LABEL='*label*'
: 総計行にラベルを表示します。

HEADING=HORIZONTAL | VERTICAL
: 列ヘッダーの方向を制御します。

LABEL
: 変数のラベルを列ヘッダーとして使用するように指定します。

SPLIT='*split-character*'
: 列ヘッダーの改行を制御する区切り文字を指定します。

STYLE <(*location(s)*)>=<*style-override(s)*>
: レポートの特定の領域のデフォルトのスタイル要素および属性を変更するために、ODS スタイルのオーバーライドを 1 つ以上指定します。

SUMLABEL

NOSUMLABEL

SUMLABEL='*label*'
: BY グループの要約行へのラベル表示の有無を指定します。

Control general format

BLANKLINE=*n*

BLANKLINE=(COUNT=*n* <STYLE=[*style-attibute-specification(s)*]>)
: *n* オブザベーションの後にブランクを書き込みます。

DOUBLE
: オブザベーション間にブランク行を書き込みます。

N<="*string-1*"> <"*string-2*">
: データセット、BY グループのいずれか、またはその両方のオブザベーション数を印刷し、その数と印刷する説明テキストを指定します。

NOOBS
: 各オブザベーションを番号で識別する出力の列を非表示にします。

OBS="*column-header*"
: 各オブザベーションを番号で識別する列に対して列ヘッダーを指定します。

ROUND
: フォーマットされていない数値を小数点以下 2 桁に丸めます。

Control page format

ROWS=*page-format*
: 1 ページの行をフォーマットします。

UNIFORM
: 各変数のフォーマットされた幅をすべてのページで列幅として使用するように指定します。

WIDTH=FULL | MINIMUM | UNIFORM | UNIFORMBY
: 変数ごとの列幅を決定します。

オプション引数

BLANKLINE=*n*
BLANKLINE=(COUNT=*n* <STYLE=[*style-attibute-specification(s)*]>)
: 各 *n* オブザベーションの後にブランク行を挿入するように指定します。オブザベーション数は、すべての ODS 出力先に対する各 BY グループの開始時に 0 にリセットされます。

n
COUNT=*n*
: 後ろにブランク行が挿入されるオブザベーション番号を指定します。

STYLE=[*style-attribute-specification(s)*]
: ブランク行に使用するスタイル属性を指定します。

デフォルト DATA

ヒント BACKGROUNDCOLOR スタイル属性を使用して、オブザベーションを色によって視覚的に区別できます。

参照項目: STYLE=オプション (1402 ページ)(有効なスタイル属性)。

例 “例 2: 印刷する変数の選択” (1444 ページ)

別名 BLANK=

ヒント SAS には、視覚障害者が PROC PRINT 出力にアクセスできることを確認するためのチェックが含まれています。BLANKLINE=オプションを指定すると、PROC PRINT が作成する出力には、データではない 1 つ以上の行が含まれます。スクリーンリーダーとユーザーは、これらの行を誤って解釈する可能性があります。ACCESSIBLECHECK システムオプションを設定すると、SAS は出力に空白行を追加するために BLANKLINE オプションが使用されているかどうかを確認します。出力に空白行がある場合、SAS は警告メッセージを SAS ログに書き込みます。アクセス可能な出力の作成に関するベストプラクティスについては、 ODS および ODS Graphics を使用した SAS Viya プラットフォームのアクセス可能な出力の作成を参照してください。

CONTENTS=*link-text* <**#BYLINE**> <**#BYVAL**> <**#BYVAR**>
PROC PRINT ステートメントによって生成される出力へのテーブルコンテンツファイルのリンク用テキストを指定します。

link-text
目次で使用されるテキストを指定します。

#BYLINE
テキスト文字列に指定した#BYLINE を先頭と末尾にブランクを含まない BY 行全体で置き換えます。BY 行は、 *variable-name*=*value* の出力形式を使用します。

#BYVAL*n*
#BYVAL(*BY-variable-name*)
テキスト文字列に指定した#BYVAL を、指定した BY 変数の現在の値に置き換えます。この変数には次のいずれかの値を指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAL2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAL(YEAR)では、BY 変数 YEAR を指定します。*variable-name* では、大文字と小文字は区別されません。

#BYVAR*n*
#BYVAR(*BY-variable-name*)
テキスト文字列に指定した#BYVAR を BY 変数の名前または変数に関連付けられているラベルで置き換えます。この変数には次のいずれかの値を指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAR2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAR(SITES)では、BY 変数 SITES を指定します。*variable-name* では、大文字と小文字は区別されません。

制限事項 CONTENTS=は、HTML Body ファイルに影響しません。HTML コンテンツファイルにのみ影響します。

CONTENTS=は、ODS LISTING 出力先には無効です。

参照項目: HTML 出力の詳細については、HTML 出力先によって生成されるファイルおよび"ODS HTML5 ステートメント" (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

DATA=*SAS-data-set*
印刷する SAS データセットを指定します。

参照項目: "入力データセット" (24 ページ)

DOUBLE
オブザベーション間にブランク行を書き込みます。

別名 D

制限事項 DOUBLE は、ODS LISTING 出力先にのみ有効です。

例 "例 2: 印刷する変数の選択" (1444 ページ)

GRANDTOTAL_LABEL='*label*'
総計行にラベルを表示します。 #BYVAR 変数および#BYVAL 変数を'*label*'に含めることができます。

別名 GRAND_LABEL=

GRANDTOT_LABEL=

GTOT_LABEL=

GTOTAL_LABEL=

制限事項 #BYVAR 変数と#BYVAL 変数は LISTING 出力先には非対応です。

ヒント SAS には、視覚障害者が PROC PRINT 出力にアクセスできることを確認するためのチェックが含まれています。ACCESSIBLECHECK システムオプションを設定すると、SAS は SUMLABEL オプションと GRANDTOTAL_LABEL オプションの両方でラベルが使用可能かどうかを確認します。SAS は、出力に要約値と総計値のラベルがないことを検出すると、メッセージをログに書き込みます。アクセス可能な出力の作成に関するベストプラクティスについては、 ODS および ODS Graphics を使用した SAS Viya プラットフォームのアクセス可能な出力の作成を参照してください。

例 "例 6: 複数の BY 変数を使用して数値変数を合計する" (1464 ページ)

HEADING=HORIZONTAL | VERTICAL
列ヘッダーの方向を制御します。

HORIZONTAL
すべての列ヘッダーを横方向に印刷します。

別名 H

VERTICAL
すべての列ヘッダーを縦方向に印刷します。

別名 V

制限事項 LISTING 出力については、ページに対して列ヘッダーが長すぎる場合には、ラベルのかわりに変数名を使用します。

デフォルト ヘッダーはすべて 横方向、すべて縦方向のいずれかになります。HEADING=を省略すると、PROC PRINT は列ヘッダーの方向を次のように決定します。

LABEL を使用しない場合、スペースによって 列ヘッダーが縦方向または横方向であるかが決定されます。

LABEL を使用し、少なくとも 1 つの変数にラベルがある場合、すべてのヘッダーは横方向になります。

LABEL
変数のラベルを列ヘッダーとして使用するように指定します。

別名 L

デフォルト PROC PRINT は、次の 2 つの状況では変数名を列ヘッダーとして使用します。

1. PROC PRINT ステップに LABEL ステートメントが含まれている場合でも PROC PRINT ステートメントで LABEL オプションを省略する場合
2. 変数にラベルが含まれていない場合

操作 デフォルトでは、LABEL を指定し、少なくとも 1 つの変数にラベルが含まれている場合は PROC PRINT はすべての列ヘッダーを横方向に印刷します。そのため、LABEL を使用すると出力のページ数が増えることがあります。(PROC PRINT ステートメントで HEADING=VERTICAL を使用して、列ヘッダーを縦方向に印刷します。

PROC PRINT は、ラベルを複数行にわたって分割し、スペースを節約する場合があります。PROC PRINT ステートメントで SPLIT=を使用して、これらの分割が発生する箇所を制御します。SPLIT=を使用する場合、LABEL を使用する必要はありません。

注 SAS システムオプション LABEL は、プロシジャがラベルを使用するために有効である必要があります。詳細については、“LABEL システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

ヒント 変数に対しブランクの列ヘッダーを作成するには、PROC PRINT ステップでこの LABEL ステートメントを使用します。

```
label variable-name='00'x;
```

参照項目: LABEL ステートメントを使用してプロシジャで一時ラベルを作成する詳細については、“LABEL ステートメント” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

LABEL ステートメントを DATA ステップで使用して永久ラベルを作成する詳細については、“LABEL ステートメント” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

例 “例 4: オブザベーショングループごとに別々にレポートセクションを作成する” (1451 ページ)

N<=“*string-1*”> <“*string-2*”>
データセット、BY グループのいずれか、またはその両方のオブザベーション数を印刷し、その数と印刷する説明テキストを指定します。

N オプション使用	PROC PRINT アクション
BY ステートメントおよび SUM ステートメントと使用しない	レポートの最後にデータセットのオブザベーション数を印刷し、その数を *string-1* とラベル付けします。
BY ステートメントと使用	各 BY グループの最後に BY グループのオブザベーション数を印刷し、その数を *string-1* の値とラベル付けします。
BY ステートメントおよび SUM ステートメントと使用	各 BY グループの最後に BY グループのオブザベーション数を印刷し、レポートの最後にデータセットのオブザベーション数を印刷します。BY グループに対する数は *string-1* とラベル付けされます。データセット全体に対する数は *string-2* とラベル付けされます。

ヒント SAS には、視覚障害者が PROC PRINT 出力にアクセスできることを確認するためのチェックが含まれています。N=オプションを指定すると、PROC PRINT が作成する出力には、データではないテキスト行が含まれます。スクリーンリーダーは、このテキスト行をデータとして解釈する場合があります。ACCESSIBLECHECK システムオプションを設定すると、SAS は出力にアクセスできるかどうかを確認します。出力にデータではないテキストが含まれている場合、SAS は警告メッセージを SAS ログに書き込みます。アクセス可能な出力の作成に関するベストプラクティスについては、 ODS および ODS Graphics を使用した SAS Viya プラットフォームのアクセス可能な出力の作成を参照してください。

例 “例 3: 列ヘッダーテキストのカスタマイズ” (1447 ページ)

“例 4: オブザベーショングループごとに別々にレポートセクションを作成する” (1451 ページ)

“例 5: 1 つの BY グループを使用して数値変数を合計する” (1459 ページ)

NOOBS
各オブザベーションを番号で識別する出力の列を非表示にします。

例 “例 4: オブザベーショングループごとに別々にレポートセクションを作成する” (1451 ページ)

OBS=“*column-header*”
各オブザベーションを番号で識別する列に対して列ヘッダーを指定します。

ヒント OBS=は区切り文字を有効化します。(SPLIT=オプション (1402 ページ) の説明を参照してください。)

例 “例 3: 列ヘッダーテキストのカスタマイズ” (1447 ページ)

ROUND
フォーマットされていない数値を小数点以下 2 桁に丸めます。 (フォーマットされている値は、出力形式により指定小数点以下桁数にすでに丸められています。) フォーマットされている変数とフォーマットされていない変数について、PROC PRINT はこれらの丸められた値を使用してレポートの合計を計算します。

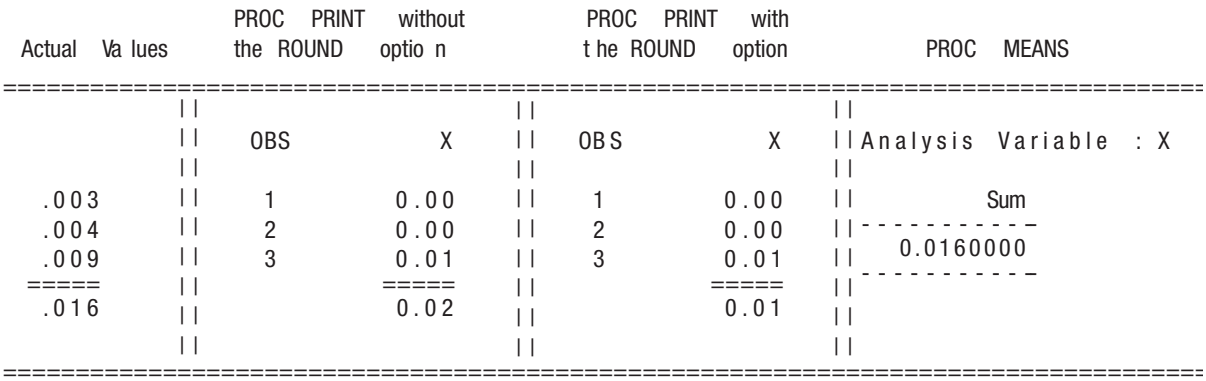
ROUND を省略すると、*PROC PRINT はフォーマットされた(丸められた)値を表示しますが*、行の実際の値を追加して合計を取得します。合計も出力形式によって丸められますが、この合計には丸めエラーが 1 つだけ含まれます。これは、実際の値の合計の丸めエラーです。ROUND オプションは値を合計する前に丸めるため、複数の丸めエラーがある場合があります。ROUND を使用しない方が結果はより正確なものになりますが、ROUND は合計が印刷された(丸められた)値の合計であることが重要なパブリッシュされたレポートに有用です。

ROUND オプションを使用した PROC PRINT の結果が、PROC MEANS、DATA ステップなどのその他の方法を使用して同じデータを合計した結果と異なる場合があることに注意してください。次が True である単純な場合について考えます。

- データセットには X に対する 3 つの値、.003、.004 および.009 が含まれています。
- X に出力形式 5.2 がある。

合計の計算方法に応じて、0.02、0.01 および 0.016 という 3 つの異なる答えが得られます。次の図に、PROC PRINT (ROUND オプションを使用しない場合と使用した場合)と PROC MEANS を使用した合計の計算結果を示します。

図 40.2 変数を合計する 3 つの方法



ROUND オプションを使用せずに生成される合計(.02)が、ROUND を使用して生成される合計(0.01)よりも実際の結果(0.16)に近いことに注意してください。ただし、ROUND を使用して生成される合計は、レポートに表示される数を反映しています。

別名 R

注意 **RROUND を PICTURE 出力形式と使用しないでください。** ROUND は、数値と使用します。SAS プロシジャは、ピクチャ出力形式の変数を文字変数として扱います。ROUND をそのような変数と使用すると、予期せぬ結果が発生することがあります。

ROWS=*page-format*
1 ページの行をフォーマットします。 現在、PAGE は *page-format* に使用できる唯一の値です。

PAGE
ページごとの各オブザベーションに対し 1 行の変数のみ印刷します。ROWS=PAGE を使用すると、PROC PRINT はページをセクションに分割せず、各ページにできるだけ多くのオブザベーションを印刷します。出力の最後のページを埋めるのに十分なオブザベーションがない場合、PROC PRINT は最後のページをセクションに分割し、最後の少数のオブザベーションに対しすべての変数を印刷します。

制限事項 ROWS=は、ODS LISTING 出力先にのみ有効です。そのため、ROWS=を使用する場合、PROC PRINT からの HTML 出力は同じです。

ヒント データセットに多くの変数とオブザベーションが含まれている場合、PAGE 値により出力のページ数を減らすことができます。ただし、データセットに含まれる変数の数が多く、オブザベーションの数が少ない場合、PAGE 値により出力のページ数を増やすことができます。

参照項目: “ページのサイズが制限されている場合のページレイアウト” (1439 ページ) (デフォルトレイアウトの説明)

例 “例 8: 多数の変数を使用したレポートのレイアウトを制御する” (1474 ページ)

SPLIT='*split-character*'
列ヘッダーの改行を制御する区切り文字を指定します。 また、ラベルを列ヘッダーとして使用します。PROC PRINT は区切り文字に達すると列ヘッダーを改行し、次の行ヘッダーを続けます。区切り文字は発生するたびラベルの最大 256 文字に考慮されますが、列ヘッダーの一部ではありません。

別名 S=

操作 SPLIT=はラベルの使用を意味するため、LABEL と SPLIT=の両方を使用する必要はありません。

OBS=オプションは区切り文字を有効化します。(*"OBS="column-header""* (1400 ページ)の説明を参照してください。)

注 PROC PRINT は、SPLIT=を指定しても各 BY グループまたは要約ラベルまたは総計レベルの前におかれるヘッダーの BY 変数のラベルを分割しません。かわりに、PROC PRINT は区切り文字をブランクと置き換えます。

例 "例 3: 列ヘッダーテキストのカスタマイズ" (1447 ページ)

STYLE <(*location(s)*)>=<*style-override(s)*>
レポートの特定の領域のデフォルトのスタイル要素および属性を変更するために、ODS スタイルのオーバーライドを 1 つ以上指定します。

スタイルのオーバーライドは、次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*

<*style-attribute-name-2=style-attribute-value-2 ...*>]

location
STYLE オプションが影響するレポートの部分を識別します。*location(s)*が指定されていない場合、スタイルのオーバーライドが適用される位置は、PROC PRINT によってステートメント、指定されたスタイル要素、およびスタイル属性に基づき決定されます。

次の図に、利用可能な場所と、それらを指定できるその他のステートメントを示します。

表 40.1 STYLE オプションでの場所の指定

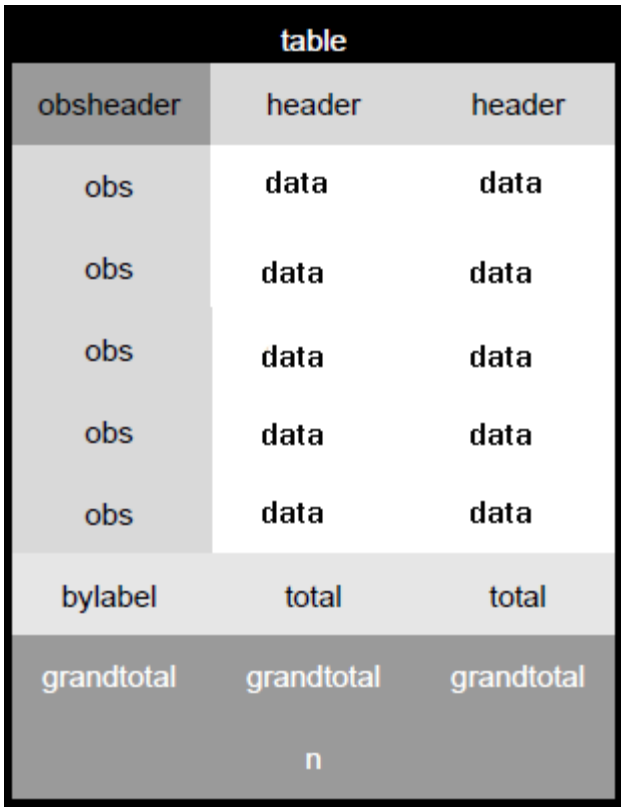
場所	場所の別名	影響を受けるレポート部分	次のステートメントでも使用可能
BYLABEL	BYSUMLABEL BYLBL BYSUMLBL	SUM 合計を含む行の BY 変数のラベル	なし

場所	場所の別名	影響を受けるレポート部分	次のステートメントでも使用可能
DATA	COLUMN COL	OBS 列または ID 列のデータを除くすべてのデータ または ID ステートメントの STYLE=オプションで DATA 位置が指定されているときの ID 列のデータ	VAR ID SUM
GRANDTOTAL	GRANDTOT GRAND GTOTAL GTOT	レポート全体の総計を含む SUM 行	SUM
HEADER	HEAD HDR	OBS 列または ID 列を除くすべてのヘッダー [1] または ステートメントの STYLE=オプションで HEADER 位置が指定されているときの ID 列の列ヘッダーすべて	VAR ID SUM
N	なし	N=テーブルおよびコンテンツ	なし
OBS	OBSDATA OBSCOLUMN OBSCOL	ID ステートメントの STYLE=オプションで DATA 位置が指定されていないときの OBS 列または ID 列のデータ	なし
OBSHEADER	OBSHEAD OBSHDR	ID ステートメントの STYLE=オプションで HEADER 位置が指定されていないときの OBS 列または ID 列のヘッダー [1]	なし

場所	場所の別名	影響を受けるレポート部分	次のステートメントでも使用可能
TABLE	REPORT	レポートの構造部分(境界の幅、セル間のスペースなどの設定に使用される基本テーブル)	なし
TOTAL	TOT BYSUMLINE BYLINE BYSUM	BY グループごとの合計を含む SUM 行	SUM

1 PROC PRINT ステートメントの STYLE=オプションの OBSHEADER 位置を使用して、OBS 列と ID 列のフォーマットします。既存プログラムにおける PROC PRINT ステートメントの STYLE=オプションには、HEADER 位置のみならず OBSHEADER 位置も含まれている必要があります。

図 40.3 PROC PRINT の領域および対応するステートメント



PROC PRINT ステートメントの OBSHEADER 位置と ID ステートメントの HEADER 位置に対して同一のスタイル属性が表示されていれば、OBSHEADER 属性よりも HEADER 位置属性が優先されます。PROC PRINT ステートメントと ID ステートメントの両方の ID 列に対する他のすべてのスタイル属性は、統合されてその ID 列のスタイルを作成します。たとえば PROC PRINT ステートメントでは、OBSHEADER の位置属性は{fontsize=5 fontweight=bold}になります。ID ステートメントでは、HEADER の位置属性は

[fontsize=6 fontstyle=italic]になります。その結果、ID 列のスタイルは[fontsize=6 fontweight=bold fontstyle=italic]となります。

```
proc print data=exprev style(obsheader)={fontsize=5 fontweight=bold};
  id country / style(header)=[fontsize=6 fontstyle=italic];
run;
```

Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
Bermuda	99999999	1/1/18	1/4/18	Catalog	7	41.0	9.25

PROC PRINT ステートメントの OBS 位置と ID ステートメントの DATA 位置に対して同一のスタイル属性が表示されていれば、OBS 属性よりも DATA 位置属性が優先されます。PROC PRINT ステートメントと ID ステートメントの両方の ID 列に対する他のすべてのスタイル属性は、統合されてその ID 列のスタイルを作成します。たとえば、PROC PRINT ステートメントでは、OBS の位置属性は{backgroundcolor=light gray color=blue}になります。ID ステートメントでは、DATA の位置属性は[color=white fontstyle=italic]になります。その結果、ID 列のスタイルは[backgroundcolor=light gray color=white fontstyle=italic]となります。

```
proc print data=exprev style(obs)={backgroundcolor=light gray color=blue};
  id country / style(data)=[color=white fontstyle=italic];
run;
```

Country	Emp_ID	Order_Date	Ship_Date	Sale_Type	Quantity	Price	Cost
Puerto Rico	99999999	1/1/18	1/5/18	Catalog	14	51.2	12.10
Aruba	99999999	1/1/18	1/4/18	Catalog	30	123.7	59.00
Bahamas	99999999	1/1/18	1/4/18	Catalog	8	113.4	28.45
Bermuda	99999999	1/1/18	1/4/18	Catalog	7	41.0	9.25
British Virgin Islands	99999999	1/2/18	1/5/18	Catalog	11	40.2	20.20
Canada	99999999	1/2/18	1/5/18	Catalog	100	11.8	5.00
Virgin Islands (U.S.)	99999999	1/1/18	1/4/18	In Store	25	31.1	15.65
Belize	120458	1/2/18	1/2/18	In Store	2	146.4	36.70
Cayman Islands	120454	1/2/18	1/2/18	In Store	20	71.0	32.30
Antarctica	99999999	1/1/18	1/7/18	Internet	2	92.6	20.70

style-element-name

Output Delivery System で登録されるスタイルテンプレートにおけるスタイル要素の名前です。SAS はいくつかの スタイルテンプレート を提供しています。ユーザーは TEMPLATE プロシジャを使用して、独自のスタイルテンプレートを作成できます。*SAS Output Delivery System: プロシジャガイド*を参照してください。

スタイル要素の処理時は、より詳細な特定のスタイル要素がより詳細でないものに優先します。各 PROC PRINT 位置のデフォルトのスタイル要素およびスタイル属性の表については、表 40.83 (1436 ページ)を参照してください。

ヒント スタイル要素名には、複合名やフォーマットを使用できます。複合スタイル要素名の使用例として、style(obsheader)=data.italic.red;があります。フォーマット要素名の使用例として、style=$cities があります。フォーマットの使用の詳細については、*SAS Output Delivery System: プロシジャガイド*を参照してください。

style-attribute-specification
変更するスタイル属性を記述します。*style-attribute-specification* にはそれぞれ、次の一般的な形式があります。

style-attribute-name=style-attribute-value

次のスタイル属性を TABLE 場所で設定できます。

BACKGROUNDCOLOR=	FONTWIDTH=[1]
BACKGROUNDIMAGE=	COLOR=[1]
BORDERCOLOR=	FRAME=
BORDERCOLORDARK=	HTMLCLASS=
BORDERCOLORLIGHT=	TEXTALIGN=
BORDERWIDTH=	OUTPUTWIDTH=
CELLPADDING=	POSTHTML=
CELLSPACING=	POSTIMAGE=
FONT=[1]	POSTTEXT=
FONTFAMILY=[1]	PREHTML=
FONTSIZE=[1]	PREIMAGE=
FONTSTYLE=[1]	PRETEXT=
FONTWEIGHT=[1]	RULES=

1 これらの属性は使用時に属性 PRETEXT=、POSTTEXT=、PREHTML=、POSTHTML=で指定されるテキストにのみ影響します。表に表示されるテキストの前景色またはフォントを変更するには、表ではなくセルに影響する場所で対応する属性を設定する必要があります。

次のスタイル属性を TABLE 以外のすべての場所で設定できます。

ASIS=	FONTWIDTH=
BACKGROUNDCOLOR=	HREFTARGET=
BACKGROUNDIMAGE=	CLASS=
BORDERCOLOR=	TEXTALIGN=
BORDERCOLORDARK=	NOBREAKSPACE=

BORDERCOLORLIGHT=	POSTHTML=
BORDERWIDTH=	POSTIMAGE=
HEIGHT=	POSTTEXT=
CELLWIDTH=	PREHTML=
FLYOVER=	PREIMAGE=
FONT=	PRETEXT=
FONTFAMILY=	PROTECTSPECIALCHARACTERS=
FONTSIZE=	TAGATTR=
FONTSTYLE=	URL=
FONTWEIGHT=	VERTICALALIGN=

制限事項 STYLE=は、ODS LISTING または ODS OUTPUT 出力先には無効です。

参照項目: PROC TABULATE, PROC REPORT および PROC PRINT とともに使用できるスタイル属性の表については、表 40.82 (1433 ページ)を参照してください。

各 PROC PRINT 位置のデフォルトのスタイル要素およびスタイル属性の表については、表 40.83 (1436 ページ)を参照してください。

PROC PRINT でのスタイルの使用の詳細については、"PROC PRINT で ODS スタイルを使用する" (1427 ページ)を参照してください。

スタイル属性および PROC TEMPLATE の詳細については、DEFINE スタイルステートメント(*SAS Output Delivery System: プロシジャガイド*)を参照してください。

SUMLABEL
NOSUMLABEL
SUMLABEL='*label*'
BY グループの要約行へのラベル表示の有無を指定します。

SUMLABEL
変数ラベルがあれば、変数名のかわりに要約ラインのラベルとして使用するよう指定します。

NOSUMLABEL
要約行のラベルを空白のままで残すよう指定します。または、SUMLABEL="" (間にスペースがない単一または二重引用符)を使用して要約行のブランクを表示することもできます。

SUMLABEL='*label*'
BY グループの要約行のラベルとしてテキストを使用するよう指定します。#BYVAR 変数および#BYVAL 変数を'*label*'に含めることができます。

制限事項 #BYVAR 変数と#BYVAL 変数は LISTING 出力先には非対応です。

デフォルト SUMLABEL を省略すると、PROC PRINT は要約行の BY 変数名を使用します。

ヒント SAS には、視覚障害者が PROC PRINT 出力にアクセスできることを確認するためのチェックが含まれています。ACCESSIBLECHECK システムオプションを設定すると、SAS は SUMLABEL オプションと GRANDTOTAL_LABEL オプションの両方でラベルが使用可能かどうかを確認します。SAS は、出力に要約値と総計値のラベルがないことを検出すると、メッセージをログに書き込みます。アクセス可能な出力の作成に関するベストプラクティスについては、ODS および ODS Graphics を使用した SAS Viya プラットフォームのアクセス可能な出力の作成を参照してください。

例 “例 5: 1 つの BY グループを使用して数値変数を合計する” (1459 ページ)

“例 6: 複数の BY 変数を使用して数値変数を合計する” (1464 ページ)

UNIFORM
WIDTH=UNIFORM (1408 ページ)を参照してください。

WIDTH=FULL | MINIMUM | UNIFORM | UNIFORMBY
変数ごとの列幅を決定します。

FULL
変数のフォーマットされた幅を列幅として使用します。変数にフィールド幅を明示的に指定する出力形式がない場合、PROC PRINT はデフォルトの幅を使用します。文字変数の場合、デフォルトの幅は、変数の長さになります。数値変数の場合、デフォルトの幅は 12 となります。WIDTH=FULL を使用すると、列幅がページによって異なることはありません。

ヒント WIDTH=FULL を使用すると、実行時間を削減できます。

MINIMUM
変数のすべての値に対応する最小列幅を各変数に使用します。

別名 MIN

UNIFORM
各変数のフォーマットされた幅をすべてのページで列幅として使用します。変数にフィールド幅を明示的に指定する出力形式が含まれていない場合、PROC PRINT は最大幅データ値を列幅として使用します。
WIDTH=UNIFORM を指定すると、PROC PRINT は通常データセットを二度読み込む必要があります。ただし、データセットのすべての変数にフィールド幅を明示的に指定する出力形式(たとえば BEST.ではなく BEST12.)が含まれている場合、PROC PRINT はデータセットを一度だけ読み込みます。

別名 U

制限事項 一部の変数に幅を明示的に指定する出力形式が含まれていない場合、別のユーザーがデータセットを同時に更新している場合は WIDTH=UNIFORM を同時アクセスをサポートするエンジンと使用できません。

ヒント データセットが大きく、一定のレポートが必要な場合、PROC PRINT がデータを一度だけ読み込むように、フィールド幅を明示的に指定する出力形式を使用してコンピューターリソースを節約できます。

WIDTH=UNIFORM は、UNIFORM と同じです。

UNIFORMBY
変数のフォーマットされた幅を列幅として使用して、BY グループ内のすべての列に出力形式を一様に適用します。変数にフィールド幅を明示的に指定する出力形式が含まれていない場合、PROC PRINT は最大幅データ値を列幅として使用します。

別名 UBY

制限事項 UNIFORMBY を順次データセットと使用できません。

デフォルト WIDTH=を省略し、UNIFORM オプションを指定しない場合、PROC PRINT は個別に出力の各ページを作成します。プロシジャはページごとにデータを分析し、最適な表示方法を決定します。そのため、列幅がページごとに異なる場合があります。

制限事項 WIDTH=は LISTING 出力先に対してのみ有効です

ヒント 列幅は、可変幅だけでなく、列ヘッダーの長さの影響も受けます。列ヘッダーが長いと、WIDTH=の有用性が低下することがあります。

参照項目: デフォルトの列幅の説明については、“列幅” (1442 ページ)を参照してください。

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント　SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント　この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント　この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

BY ステートメント

出力を BY グループ順に並べ替えます。

制限事項:　BY ステートメントではグループ変数を指定しないでください。BY 変数をグループ変数と同じにすることはできません。

例:
“例 4: オブザベーショングループごとに別々にレポートセクションを作成する” (1451 ページ)
“例 5: 1 つの BY グループを使用して数値変数を合計する” (1459 ページ)
“例 6: 複数の BY 変数を使用して数値変数を合計する” (1464 ページ)
“例 7: レポート内の合計数を制限する” (1469 ページ)
“例 9: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1476 ページ)

構文

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

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

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

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

詳細

BY グループ処理

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

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

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

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

- BY ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。
- PROC SORT の詳細については、55 章, "SORT プロシジャ," (2027 ページ)を参照してください。
- CAS での BY グループ処理の詳細については、"グループ化と順序付け" (*SAS Cloud Analytic Services: DATA ステッププログラミング*)を参照してください。
- インデックスの作成の詳細については、"INDEX CREATE ステートメント" (435 ページ)を参照してください。

BY 変数値のフォーマティング

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

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

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

2 つのデータセットで長さが異なる BY 変数

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

BY ステートメントの変数は、プライマリデータセットに出現している必要があります。

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

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

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

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

- A
- A、B
- A、B、C
- A、B、C、D

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

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブラ

ンクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデータセットの'ABCD'とセカンダリデータセットの'ABCD 'は一致します。セカンダリデータセットに'ABCDEF'が含まれている場合は、一致しません。

ID ステートメントを指定して BY ステートメントを使用

ID ステートメントの最初にすべての BY 変数を同じ順序で記述した場合、PROC PRINT で特殊レイアウトが使用されます。(参照: "例 9: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成" (1476 ページ)。)

NOBYLINE オプションを指定した BY ステートメントの使用

BY ステートメントを、BY グループ処理によって生成される出力に通常表示される BY 行を非表示にする SAS システムオプション NOBYLINE と使用すると、PROC PRINT は常に BY グループごとに新しいページを開始します。この動作により、BY グループ情報をタイトルに挿入し、デフォルトの BY 行を NOBYLINE で非表示にしてカスタマイズした BY 行を作成すると、タイトルの情報がページのレポートと一致するようになります。

並べ替えられていないデータの印刷時の BY 変数の使用

値が並べ替えられていない BY 変数を指定すると、最初の並べ替えられていないグループの処理時にデータセットの印刷が停止します。メッセージが SAS ログに書き込まれます。

BY ステートメントをサポートする Base SAS プロシジャ

COMPARE	SORT (必須)
CORR	STANDARD
FREQ	SUMMARY
MEANS	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE
REPORT (非ウィンドウ環境のみ)	ODSTEXT
ODSLIST	ODSTABLE

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

関連項目

"BY ステートメント" (*SAS DATA ステップステートメント: リファレンス*)

例

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT ステップで BY ステートメントを使用します。BY 変数 Year の値ごとに出力があります。

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

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

出力

Printing of Team Members by Year

Year=Freshman

Name	Gender	GPA
Capiccio	m	3.598
Tucker	m	3.901

Year=Sophomore

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

Year=Junior

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

Year=Senior

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

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。“出力形式の取り消し” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、“PUT ステートメント: リスト” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

ID ステートメント

オブザベーション番号ではなく、リストする変数のフォーマットされた値を使用して、オブザベーションを識別します。

例: “例 8: 多数の変数を使用したレポートのレイアウトを制御する” (1474 ページ)
“例 9: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1476 ページ)

構文

ID *variable(s)*
</ STYLE <(*location(s)*)>=<*style-override(s)*>>;

必須引数

variable(s)
レポートの各行の開始時にオブザベーション番号の代わりに印刷する 1 つ以上の変数を指定します。

制限事項 ID 変数が非常に多くのスペースを 占めているためにその行に少なくとも 1 つのその他の変数のためのスペースがない 場合、PROC PRINT は警告を SAS ログに書き込み、すべての ID 変数を ID 変数として扱いません。

操作 ID ステートメントの変数が VAR ステートメントにも表示される場合、出力にはその変数に対する 2 つの列が含まれます。

オプション引数

STYLE <(*location(s)*)>=<*style-override(s)*>
ID ステートメントで作成される ID 列に使用する 1 つ以上のスタイルのオーバーライドを指定します。

スタイルのオーバーライドは、次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2* ...>]

制限事項 OBSHEADER 位置のスタイル指定は ID ステートメントには無効です。

操作 ID ステートメントで STYLE(HEADER)=オプションを、PROC PRINT ステートメントで STYLE(OBSHEADER)=を指定すると、ID ステートメントのために指定されているスタイル属性が PROC PRINT ステートメントで指定したスタイル要素に優先します。その後、PROC PRINT ステートメントの STYLE(OBSHEADER)=オプションのスタイル属性は ID ステートメントの STYLE(HEADER)=オプションと結合し、ID 列ヘッダーの出力がレンダリングされます。

ヒント 異なる ID 列に対して異なるスタイルのオーバーライドを指定するには、各変数に別の ID ステートメントを使用して、異なる STYLE オプションを各 ID ステートメントに追加します。

参照項目: このオプションの引数およびその使用方法の詳細については、PROC PRINT ステートメントの STYLE= (1402 ページ)オプションを参照してください。

詳細

BY ステートメントと ID ステートメントとの併用

ID ステートメントの最初にすべての BY 変数を同じ順序で記述した場合、PROC PRINT で特殊レイアウトが使用されます。(“例 9: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1476 ページ)を参照してください。)

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=’ ‘>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項
ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変

数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: "文字定数と文字変数" (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

PAGEBY ステートメント

ページが埋まる前に発生するページ送りを制御します。

要件 BY ステートメント

例: “例 4: オブザベーショングループごとに別々にレポートセクションを作成する” (1451 ページ)

構文

PAGEBY *BY-variable*;

必須引数

BY-variable
PROC PRINT ステップの BY ステートメントに表示される変数を識別します。BY 変数の値が変わる、または BY ステートメントで前にくる BY 変数の値が変わると、PROC PRINT は新しいページの印刷を開始します。

操作 BY ステートメントを、BY グループ処理によって生成される出力に通常表示される BY 行を非表示にする SAS システムオプション NOBYLINE と使用すると、PROC PRINT は常に BY グループごとに新しいページを開始します。この動作により、BY グループ情報をタイトルに挿入し、デフォルトの BY 行を NOBYLINE で非表示にしてカスタマイズした BY 行を作成すると、タイトルの情報がページのレポートと一致するようになります。(“BY グループの情報を含むタイトルの作成 ” (55 ページ)を参照してください。)

SUM ステートメント

数値変数の値を総計します。

ヒント: SAS には、視覚障害者が PROC PRINT 出力にアクセスできることを確認するためのチェックが含まれています。SUM ステートメントを使用して ACCESSIBLECHECK システムオプションを設定すると、SAS は、PROCPRINT ステートメントの SUMLABEL オプションと GRANDTOTAL_LABEL オプションの両方にラベルが指定されているかどうかを確認します。出力に要約値と総計値のラベルがない場合、SAS は SAS ログにメッセージを書き込みます。アクセス可能な出力の作成に関するベストプラクティスについては、 ODS および ODS Graphics を使用した SAS Viya プラットフォームのアクセス可能な出力の作成を参照してください。

例: “例 5: 1 つの BY グループを使用して数値変数を合計する” (1459 ページ)
“例 6: 複数の BY 変数を使用して数値変数を合計する” (1464 ページ)
“例 7: レポート内の合計数を制限する” (1469 ページ)
“例 9: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1476 ページ)

構文

SUM *variable(s)*
</ STYLE <(*location(s)*)>=<*style-override(s)*> >;

必須引数

variable(s)
レポートで総計する数値変数を識別します。

オプション引数

STYLE <<(*location(s)*)>=<*style-override(s)*>
SUM ステートメントで作成される合計を含むセルに使用する 1 つ以上のスタイルのオーバーライドを指定します。

スタイルのオーバーライドは、次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2* ...>]

ヒント 合計をレポートする異なるセルに異なるスタイルのオーバーライドを指定するには、変数ごとに別の SUM ステートメントを使用して、異なる STYLE オプションを各 SUM ステートメントに追加します。

STYLE オプションが同じ場所に影響する複数の SUM ステートメントで使用される場合、最後の SUM ステートメントの STYLE オプションが使用されます。

参照項目: このオプションの引数およびその使用方法の詳細については、PROC PRINT ステートメントのオプション STYLE= (1395 ページ)を参照してください。

詳細

SUM ステートメントと BY ステートメントの併用

SUM ステートメントと、1 つの BY 変数を含む BY ステートメントを使用すると、PROC PRINT は複数のオブザベーションを含む BY グループごとに SUM 変数を合計し、すべての BY グループについて総計します。(参照: “例 5: 1 つの BY グループを使用して数値変数を合計する” (1459 ページ)。)

SUM ステートメントと、複数の BY 変数を含む BY ステートメントを使用すると、PROC PRINT は BY 変数を 1 つだけ使用する場合と同じように複数のオブザベーションを含む BY グループごとに SUM 変数を合計します。ただし、BY グループが変わると値が変わる BY 変数に対してのみ合計が出されます。(参照: “例 6: 複数の BY 変数を使用して数値変数を合計する” (1464 ページ)。)

注: BY 変数の値が変わると、SAS システムは BY ステートメントでその後にリストされるすべての 変数の値も変わるとみなします。

SUMBY ステートメント

レポートに表示される合計数を制限します。

要件 BY ステートメント

例: “例 7: レポート内の合計数を制限する” (1469 ページ)

構文

SUMBY *BY-variable*;

必須引数

BY-variable
PROC PRINT ステップの BY ステートメントに表示される変数を識別します。BY 変数の値が変わる、または BY ステートメントでその前にくる BY 変数の値が変わると、PROC PRINT は SUM ステートメントにリストされるすべての 変数の合計を印刷します。

詳細

要約される変数

SUM ステートメントを使用すると、PROC PRINT は SUM 変数のみ小計します。その他の場合は、PROC PRINT は ID ステートメントと BY ステートメントでリストされる変数を除く、データセットのすべての数値変数を小計します。

VAR ステートメント

レポートに表示される変数を選択し、その順序を決定します。

別名: VARIABLES

ヒント: VAR ステートメントを省略すると、PROC PRINT はデータセットのすべての 変数を印刷します。

例: “例 2: 印刷する変数の選択” (1444 ページ)

“例 9: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成” (1476 ページ)

構文

VAR *variable(s)*
</ STYLE <(*location(s)*)>=<*style-override(s)*> >;

必須引数

variable(s)
印刷する変数を識別します。PROC PRINT は、変数をリストする順序で印刷します。

操作 PROC PRINT 出力では、ID ステートメントにリストされる変数が VAR ステートメントにリストされる変数よりも前に置かれます。ID ステートメントの変数が VAR ステートメントにも表示される場合、出力にはその変数に対する 2 つの列が含まれます。

オプション引数

STYLE <(*location(s)*)>=<*style-override(s)*>
VAR ステートメントによって作成されるすべての列に使用する 1 つ以上のスタイルのオーバーライドを指定します。

スタイルのオーバーライドは、次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2* ...>]

ヒント 異なる列に異なるスタイルのオーバーライドを指定するには、別の VAR ステートメントを使用して変数ごとに列を作成し、異なる STYLE オプションを各 VAR ステートメントに追加します。

参照項目: このオプションの引数およびその使用方法の詳細については、PROC PRINT ステートメントのオプション STYLE= (1402 ページ)を参照してください。

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD

MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

使用法: PRINT プロシジャ

PROC PRINT で ODS スタイルを使用する

Base SAS プロシジャでのスタイルの使用

ODS をサポートする Base SAS プロシジャの多くは、1 つ以上のテーブルテンプレートを使用して出力オブジェクトを生成します。これらのテーブルテンプレートには、テーブル要素(列、ヘッダー、フッター)に対するテンプレートが含まれています。各テーブル要素で、出力のさまざまな部分に 1 つ以上のスタイル要素を使用することを指定できます。これらのスタイル要素はプロシジャの構文内で指定することはできませんが、使用する ODS 出力先に合わせてカスタマイズされたスタイルを使用できます。テーブルとスタイルのカスタマイズの詳細については、“TEMPLATE Procedure: Creating a Style Template” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

Base SAS レポートプロシジャの PROC PRINT、PROC REPORT および PROC TABULATE を使用してデータをすばやく分析し、読みやすいテーブルに整理することができます。これらのプロシジャステートメントで STYLE=オプションを使用し

てレポートの表示を変更できます。STYLE=オプションを使用すると、すべての出力のデフォルトスタイルを変更せずに、出力の各セションに変更を加えることができます。プロシジャ内の特定のステートメントで STYLE=オプションを指定して、プロシジャ出力の特定のセクションをカスタマイズすることができます。

次のプログラムでは、STYLE=オプションを使用して以下の PROC PRINT 出力の色を作成します。完全な入力データセットについては、“EXPREV” (2490 ページ)を参照してください。

```
ods _all_ close;
ods html5 file="file-path/customStyle.html";
proc print data=exprev noobs sumlabel='Total' GRANDTOTAL_LABEL="Grand Total"
    style(table)=[frame=box rules=groups]
    style(bysumline)=[background=red foreground=linen]
    style(grandtotal)=[foreground=green]
    style(header)=[font_style=italic background=orange];
  by sale_type order_date;
  sum price quantity;
  sumby sale_type;
  label sale_type='Sale Type' order_date='Sale Date';
format price dollar10.2 cost dollar10.2;
  var Country / style(data)=[font_face=arial font_weight=bold background=linen];
  var Price / style(data)=[font_style=italic background=yellow];
  var Cost / style(data)=[foreground=hgt. background=lightgreen];
title 'Retail and Quantity Totals for Each Sale Type';
run;
ods _all_ close;
```

アウトプット 40.1　拡張された PROC PRINT 出力

Retail and Quantity Totals for Each Sale Type

Sale Type=Catalog Sale Date=1/1/12

Country	Price	Cost	Quantity
Puerto Rico	$51.20	$12.10	14
Aruba	$123.70	$59.00	30
Bahamas	$113.40	$28.45	8
Bermuda	$41.00	$9.25	7

Sale Type=Catalog Sale Date=1/2/12

Country	Price	Cost	Quantity
British Virgin Islands	$40.20	$20.20	11
Canada	$11.80	$5.00	100
Total	$381.30		170

Sale Type=In Store Sale Date=1/1/12

Country	Price	Cost	Quantity
Virgin Islands (U.S.)	$31.10	$15.65	25

Sale Type=In Store Sale Date=1/2/12

Country	Price	Cost	Quantity
Belize	$146.40	$36.70	2
Cayman Islands	$71.00	$32.30	20
Total	$248.50		47

Sale Type=Internet Sale Date=1/1/12

Country	Price	Cost	Quantity
Antarctica	$92.60	$20.70	2
Grand Total	$722.40		219

スタイル、スタイル要素およびスタイル属性

スタイル、スタイル要素、およびスタイル属性について

SAS 出力の外観は、ODS スタイルテンプレート(ODS スタイル)によって制御されます。ODS スタイルは、PROC TEMPLATE スタイルの構文で記述されたコンパイル済みの STYLE テンプレートから生成されます。ODS スタイルテンプレートは、SAS 出力に特定の視覚的属性を提供するスタイル要素のコレクションです。

- スタイル要素は、出力の特定の部分に適用されるスタイル属性の名前付きコレクションです。ODS 出力の各領域には、スタイル要素名が関連付けられています。スタイル要素名は、スタイル属性が適用される場所を指定します。たとえば、スタイル要素には、列見出しの表示やセル内のデータの表示に関する指示が含まれる場合があります。スタイル要素は、スタイルを使用する出力のデフォルトの色とフォントを指定する場合もあります。
- スタイル属性は、予約済みの名前と値を使用して ODS で定義される色、フォントプロパティ、線の特性などの視覚的なプロパティです。Style 属性は、スタイルテンプレート内のスタイル要素によってまとめて参照されます。各*スタイル属性*は、プレゼンテーションの 1 つの側面の値を指定します。たとえば、BACKGROUNDCOLOR=属性は、HTML テーブルの背景色または印刷出力の色付きテーブルの色を指定します。FONTSTYLE=属性は、Roman フォントとイタリックフォントのどちらを使用するかを指定します。

注: スタイルはデータの表示を制御するため DOCUMENT または OUTPUT の出力先に送られる出力オブジェクトには影響しません。

利用可能なスタイルは SASHELP.TMPLMST アイテムストアにあります。PROC TEMPLATE で LIST ステートメントを使用して、使用可能なスタイルテンプレートのリストを表示できます。

```
proc template;
  list styles / store=sashelp.tmplmst;
run;
```

ODS スタイルの表示の詳細については、“Viewing ODS Styles Supplied by SAS” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

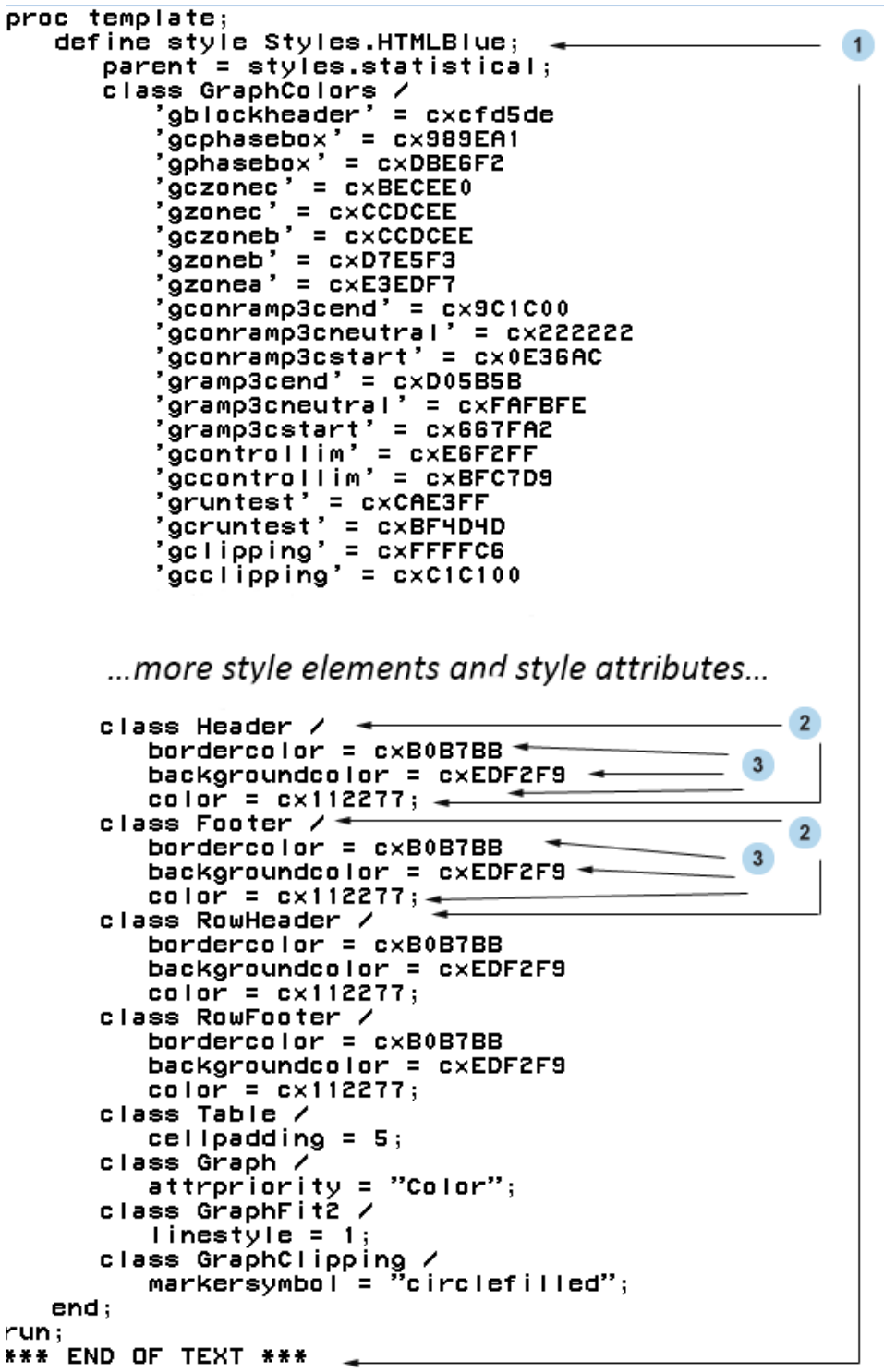
HTML5 出力は、HTMLEncore スタイルテンプレートを使用します。スタイル、スタイル要素、およびスタイル属性に慣れるために、それらの関係に注目してください。

PROC TEMPLATE で SOURCE ステートメントを使用して、スタイルテンプレートの構造を表示できます。次のコードは、HTMLBlue スタイルテンプレートの構造を SAS ログに出力します。

```
proc template;
  source styles.HTMLBlue;
run;
```

次の図は、スタイルの構造を示しています。この図は、スタイル、スタイル要素、およびスタイル属性の関係を示しています。

図 40.4 HtmlBlue スタイルの図



次のリストは、前の図の番号付きの項目に対応しています。

1 Styles.HtmlBlue は*スタイル*です。スタイルは、SAS 出力のプレゼンテーションの側面 (色、フォント、フォント サイズなど) を表示する方法を記述します。スタイルは、それを使用する ODS ドキュメントの全体的な外観を決定します。HTML 出力のデフォルトのスタイルは、HtmlBlue です。各スタイルは、スタイル要素で構成されています。

"DEFINE STYLE Statement" (*SAS Output Delivery System: Procedures Guide*)を使用して新しいスタイルを作成できます。 新しいスタイルは、独立して作成することも、既存のスタイルから作成することもできます。"PARENT= Statement" (*SAS Output Delivery System: Procedures Guide*)を使用して既存のスタイルから新しいスタイルを作成することができます。ODS スタイルの表示の詳細については、"Style Templates" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

2 ヘッダーとフッターは*スタイル要素*の例です。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。たとえば、スタイル要素には、列見出しの表示やテーブルセル内のデータの表示に関する指示が含まれる場合があります。スタイル要素は、そのスタイルを使用する出力のデフォルトの色とフォントを指定する場合もあります。スタイル要素はスタイル内に存在し、1 つ以上のスタイル属性で構成されます。スタイル要素は、ユーザーが定義するか、SAS によって提供されます。ユーザー定義のスタイル要素は、"STYLE Statement" (*SAS Output Delivery System: Procedures Guide*)で作成できます。

注: HTML およびマークアップ言語に使用されるデフォルトのスタイル要素とその継承のリストについては、"Style Elements" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

3 BORDERCOLOR=、BACKGROUNDCOLOR=、および COLOR=は、*スタイル属性*の例です。スタイル属性は、そのスタイル要素が適用される出力領域の 1 つの側面の値を指定します。たとえば、COLOR=属性は、フォントの色に値 **cx112277** を指定します。SAS が提供するスタイル属性のリストについては、"Style Attributes" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

スタイル属性は、スタイル参照で参照できます。スタイル参照の詳細については、"style-reference" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

次の表は、PROC PRINT、PROC TABULATE、および PROC REPORT で STYLE=オプションを使用して設定できる、一般的に使用されるスタイル属性を示しています。これらの属性のほとんどは、セル以外の表の部分に適用されます (たとえば、表の境界線、列と行の間の線など)。すべての出力先ですべての属性が有効であるとは限らないことに注意してください。これらのスタイル属性、その有効な値、およびその適用可能な出力先の詳細については、"Style Attributes Tables" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

表 40.2 PROC REPORT、PROC TABULATE および PROC PRINT のスタイル属性

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
ASIS=	X	X		X		X
BACKGROUNDCOLOR=	X	X	X	X	X	X
BACKGROUNDIMAGE=	X	X	X	X	X	X
BORDERBOTTOMCOLOR=	X	X		X		
BORDERBOTTOMSTYLE=	X	X	X	X		
BORDERBOTTOMWIDTH=	X	X	X	X		
BORDERLEFTCOLOR=	X	X		X		
BORDERLEFTSTYLE=	X	X	X	X		
BORDERLEFTWIDTH=	X	X	X	X		
BORDERCOLOR=	X	X		X	X	X
BORDERCOLORDARK=	X	X	X	X	X	X
BORDERCOLORLIGHT=	X	X	X	X	X	X
BORDERRIGHTCOLOR=	X	X		X		
BORDERRIGHTSTYLE=	X	X	X	X		
BORDERRIGHTWIDTH=	X	X	X	X		

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
BORDERTOPCOLOR=	X	X		X		
BORDERTOPSTYLE=	X	X	X	X		
BORDERTOPWIDTH=	X	X	X	X		
BORDERWIDTH=	X	X	X	X	X	X
CELLPADDING=	X		X		X	
CELLSPACING=	X		X		X	
CELLWIDTH=	X	X	X	X		X
CLASS=	X	X	X	X	X	X
COLOR=	X	X	X			
FLYOVER=	X	X		X		X
FONT=	X	X	X	X	X	X
FONTFAMILY=	X	X	X	X	X	X
FONTSIZE=	X	X	X	X	X	X
FONTSTYLE=	X	X	X	X	X	X
FONTWEIGHT=	X	X	X	X	X	X
FONTWIDTH=	X	X	X	X		X
FRAME=	X		X		X	
HEIGHT=	X	X		X	X	X
HREFTARGET=		X		X		X
HTMLSTYLE=	X	X	X	X	X	
NOBREAKSPACE=[2]	X	X		X		X
OUTPUTWIDTH=	X	X	X	X	X	

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
POSTHTML=[1]	X	X	X	X	X	X
POSTIMAGE=	X	X	X	X	X	X
POSTTEXT=[1]	X	X	X	X	X	X
PREHTML=[1]	X	X	X	X	X	X
PREIMAGE=	X	X	X	X	X	X
PRETEXT=[1]	X	X	X	X	X	X
PROTECTSPECIALCHARS=		X		X	X	X
RULES=	X		X		X	
TAGATTR=	X	X	X	X	X	X
TEXTALIGN=	X	X	X	X	X	X
URL=		X		X		X
VERTICALALIGN=		X		X		X
WIDTH=	X	X	X	X	X	

1 これらの属性をこの場所で使用する場合には、属性 PRETEXT=、POSTTEXT=、PREHTML=、POSTHTML=で指定されるテキストにのみ影響します。表に表示されるテキストの前景色またはフォントを変更するには、表ではなくセルに影響する場所で対応する属性を設定する必要があります。スタイル属性とその値の詳細な説明については、*"Style Attributes" (SAS Output Delivery System: Procedures Guide)*を参照してください。

2 PROC REPORT を ODS RTF 出力先で使用する場合、長いテキスト文字列の予期せぬ折り返しを防ぐために、LINE ステートメントに影響する場所で NOBREAKSPACE=OFF を設定してください。NOBREAKSPACE=OFF 属性は、PROC REPORT コード内の LINE ステートメントまたは style(line)が指定されている PROC REPORT ステートメントのいずれかに設定する必要があります。

テーブル部分のデフォルトのスタイル要素およびスタイル属性

次のテーブルには PROC PRINT 出力のさまざまな場所におけるデフォルトのスタイル要素とスタイル属性がリストされます。テーブル内の場所は表 40.81 (1402 ページ)内の場所に対応します。

このテーブルには、最もよく使用される ODS 出力先のデフォルトを示します。HTML、PDF および RTF のデフォルトがリストされています。それぞれの出力先には、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルテンプレートが含まれます。ODS 出力先とそのデフォルトのスタイルに関する詳細なドキュメントは、“Style Templates” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

注:

SAS Studio での ODS 出力のデフォルトのスタイルは、SAS Studio がアクティブに使用しているテーマによって設定されます。テーマに応じて、デフォルトは Illuminate、Ignite、または HighContrast のいずれかです。詳細については、“Customizing SAS Studio” (*SAS Studio: User's Guide*)を参照してください。

- HTML5 出力のデフォルトスタイルは HTMLEncore です。
- PDF の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

表 40.3 レポート部分のデフォルトのスタイル要素およびスタイル属性

場所	スタイル要素	HTML5 スタイル属性	PDF スタイル属性	RTF スタイル属性
BYLABEL	Byline	FONTFAMILY = "'Avenir Next for SAS', Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BORDERWIDTH = NaN
GRANDTOTAL OBSHEADER HEADER TOTAL	Header	FONTFAMILY = "'Avenir Next for SAS', Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb

場所	スタイル要素	HTML5 スタイル属性	PDF スタイル属性	RTF スタイル属性
N	Linecontent	FONTFAMILY = "'Avenir Next for SAS', Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxfafbfe	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT =medium FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = roman COLOR =cx000000
OBS	Rowheader	FONTFAMILY = "'Avenir Next for SAS', Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb
DATA	Data	FONTFAMILY = "'Avenir Next for SAS', Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 8pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000 BORDERWIDTH = NaN COLOR = cx000000	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000
Titles	SystemTitle	FONTFAMILY = "'Avenir Next for SAS', Arial, 'Albany AMT', Helvetica, Helv"	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 11pt FONTWEIGHT = bold	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 13pt

場所	スタイル要素	HTML5 スタイル属性	PDF スタイル属性	RTF スタイル属性
		FONTSIZE = 3 FONTWEIGHT = bold FONTSTYLE = roman BACKGROUNDCOLOR = cxfafbfe	FONTSTYLE = roman COLOR = cx000000 BACKGROUNDCOLOR = cxffffff	FONTWEIGHT = bold FONTSTYLE = italic COLOR = cx000000
Footnotes	Footers	FONTFAMILY = "'Avenir Next for SAS', Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx000000 BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 13pt FONTWEIGHT = bold FONTSTYLE = italic COLOR = cx000000

PRINT プロシジャの出力でのエラー処理

PRINT プロシジャでエラーが発生した場合、または PRINT プロシジャが停止した場合は、エラー発生時までに処理されたオブザベーションに対して出力が作成される可能性があります。SAS はメッセージを SAS ログに書き込み、PRINT プロシジャを終了します。

結果: PRINT プロシジャ

PROC PRINT 出力について

デフォルトでは、PROC PRINT は HTML5 レポートを生成します。PRINT プロシジャステートメントである PROC PRINT、BY、PAGEBY、SUMBY、ID、SUM、VAR は、レポートの内容を管理するものです。各ステートメントのオプションによりレポートの体裁が管理されます。

レポートの ODS 出力先を変更するには、PROC PRINT ステートメントの前に ODS ステートメントを使用してください。HTML5 出力が不要な場合には、本プロシジャを運用する前に必ず ODS の HTML5 出力先を閉じてください。ODS の使用の詳細については、*SAS Output Delivery System: ユーザーガイド*を参照してください。

このプロシジャが生成するレポートの種類のサンプリングについては、PRINT プロシジャの例 (1444 ページ)を参照してください。

デフォルトの ODS 出力先である HTML のページレイアウト

ODS の HTML5 出力のページの縦や横の長さに制限はありません。従って、1 つのテーブル内の各オブザベーションは単一行に印刷され、レポートが印刷を指定するオブザベーションはすべて、HTML5 出力の 1 ページに表示されます。

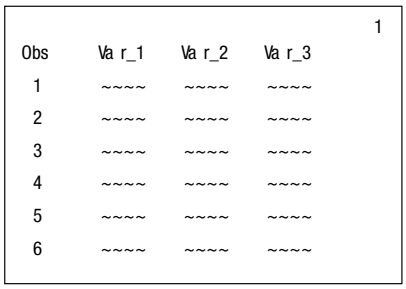
デフォルトでは、PROC PRINT が動作するたびに、SAS により出力の後に改ページが追加されます。横罫線で出力を分断すれば、改ページがレンダリングされます。詳細については、*"ODS HTML5 ステートメント" (SAS Output Delivery System: ユーザーガイド)*を参照してください。

ページのサイズが制限されている場合のページレイアウト

オブザベーション

PROC PRINT は、ページの縦横の長さが制限されている出力を生成する ODS 出力先の 1 ページ上の全オブザベーションに対しては、同一レイアウトを使用します。ODS 出力先には RTF、PDF、LISTING などがあります。まず、次の図のように、単一行のオブザベーションの印刷を試行します。

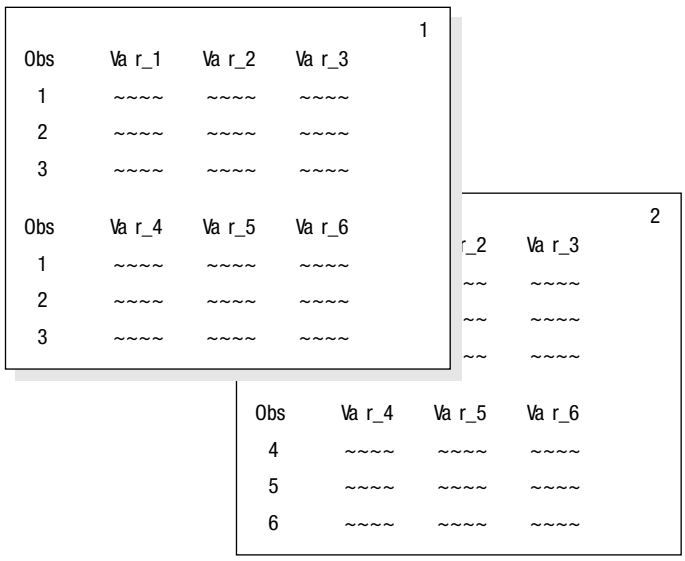
図 40.5　単一行のオブザベーションの印刷



				1
Obs	Var_1	Var_2	Var_3	
1	~~~~	~~~~	~~~~	
2	~~~~	~~~~	~~~~	
3	~~~~	~~~~	~~~~	
4	~~~~	~~~~	~~~~	
5	~~~~	~~~~	~~~~	
6	~~~~	~~~~	~~~~	

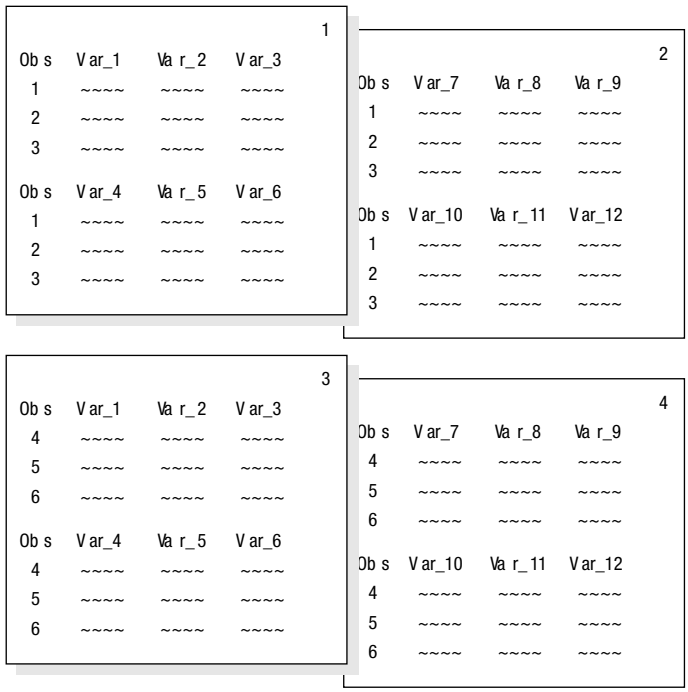
PROC PRINT は、単一行のすべての変数に適合しない場合、オブザベーションを 2 つ以上のセクションに分割し、各行の始めにオブザベーション番号または ID 変数を印刷します。たとえば、次の図では、PROC PRINT は各ページの最初のセクションの最初の 3 つの変数の値と、各ページの 2 番目のセクションの次の 3 つの変数の値を印刷します。

図 40.6　1 ページの複数のセクションへのオブザベーションの分割



PROC PRINT が 1 ページのすべての変数に適合しない場合、プロシジャはすべての変数を印刷するまで、同じオブザベーションを含む後続ページを印刷します。たとえば、次の図では、PROC PRINT は最初の 2 ページを使用して最初の 3 つのオブザベーションの値を印刷し、その次の 2 ページを使用してその他のオブザベーションの値を印刷します。

図 40.7 1 ページの複数のセクションへのオブザベーションの分割



1

Obs	Var_1	Var_2	Var_3
1	~~~~	~~~~	~~~~
2	~~~~	~~~~	~~~~
3	~~~~	~~~~	~~~~

Obs	Var_4	Var_5	Var_6
1	~~~~	~~~~	~~~~
2	~~~~	~~~~	~~~~
3	~~~~	~~~~	~~~~

2

Obs	Var_7	Var_8	Var_9
1	~~~~	~~~~	~~~~
2	~~~~	~~~~	~~~~
3	~~~~	~~~~	~~~~

Obs	Var_10	Var_11	Var_12
1	~~~~	~~~~	~~~~
2	~~~~	~~~~	~~~~
3	~~~~	~~~~	~~~~

3

Obs	Var_1	Var_2	Var_3
4	~~~~	~~~~	~~~~
5	~~~~	~~~~	~~~~
6	~~~~	~~~~	~~~~

Obs	Var_4	Var_5	Var_6
4	~~~~	~~~~	~~~~
5	~~~~	~~~~	~~~~
6	~~~~	~~~~	~~~~

4

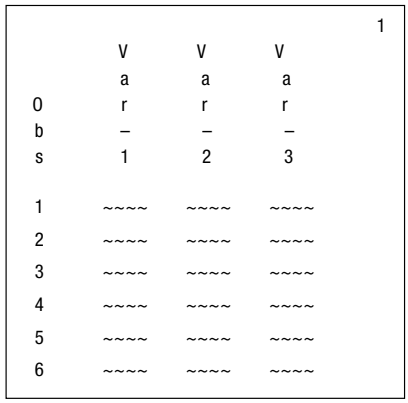
Obs	Var_7	Var_8	Var_9
4	~~~~	~~~~	~~~~
5	~~~~	~~~~	~~~~
6	~~~~	~~~~	~~~~

Obs	Var_10	Var_11	Var_12
4	~~~~	~~~~	~~~~
5	~~~~	~~~~	~~~~
6	~~~~	~~~~	~~~~

列ヘッダー

PROC PRINT が列ヘッダーを横方向または縦方向に印刷するかどうかは、スペースの数で指定します。図 40.29 (1439 ページ)、図 40.30 (1440 ページ)、および 図 40.31 (1441 ページ)は、すべて横方向のヘッダーを示します。次の図に、縦方向のヘッダーを示します。

図 40.8 縦方向のヘッダーの使用



1

Obs	Var_1	Var_2	Var_3
1	~~~~	~~~~	~~~~
2	~~~~	~~~~	~~~~
3	~~~~	~~~~	~~~~
4	~~~~	~~~~	~~~~
5	~~~~	~~~~	~~~~
6	~~~~	~~~~	~~~~

注: LABEL を使用し、少なくとも 1 つの変数にラベルがある場合、HEADING=VERTICAL を指定しない限り、PROC PRINT はすべての列ヘッダーを横方向に印刷します。

列幅

デフォルトでは、PROC PRINT は変数のフォーマットされた幅を列幅として使用します。)変数にフィールド幅を明示的に指定する出力形式が含まれていない場合、PROC PRINT はそのページの変数の最大幅データ値を列幅として使用します。

文字変数のフォーマットされた値またはフォーマットされていない文字変数のデータ幅がすべての ID 変数の長さを引いたページサイズを超える場合、PROC PRINT は値を切り捨てることがあります。次の状況について考えます。

- ページサイズが 80 です。
- IdNumber が長さ 10 の文字変数です。ID 変数として使用されます。
- State は長さ 2 の文字変数です。ID 変数として使用されます。
- コメントは長さ 200 の文字変数です。

PROC PRINT は、1 行にこれら 3 つの変数を印刷する際、2 つの ID 変数に 14 の印刷位置を、それぞれの後にスペースを使用します。この調整により COMMENT の印刷位置は、80–14 または 66 のままです。それよりも長い COMMENT の値は切り捨てられます。

注: 列幅は、可変幅だけでなく、列ヘッダーの長さの影響も受けます。列ヘッダーが長いと、WIDTH=の有用性が低下することがあります。

例: PRINT プロシジャ

例 1: CAS テーブルの印刷

要素: PROC PRINT DATA=*CAS-table*

CAS 言語要素:
- CAS ステートメント
- CAS エンジンの LIBNAME ステートメント
- PROC CASUTIL
- PROC MDSUMMARY

データセット: Sashelp.cars

詳細

この例は、次のタスクを示しています。

- CAS セッションを確立する
- Mycas libref を CAS Engine および CAS セッションに関連付ける
- CAS テーブル mycas.cars を作成する
- PROC MDSUMMARY を使用して cars データを要約する
- 要約された CAS テーブルのうち 15 行を印刷する

プログラム: SAS Compute Server で実行する

```
options cashost="cloud.example.com" casport=5555;
cas mysess sessopts=(caslib='casuser');
libname mycas cas sessref=mysess;

proc casutil outcaslib="casuser";
  load data=sashelp.cars replace;
run;

proc mdsummary data=mycas.cars;
  var mpg_highway;
  groupby origin type / out=mycas.mpghw_sum;
run;

options obs=15;
proc print data=mycas.mpghw_sum;
   var origin type _mean_;
   title "Average Highway Mileages";
run;
```

プログラムの説明

CAS サーバーを起動し、CAS セッションを設定し、CAS Engine の libref を作成して、そのエンジンを CAS セッションに接続します。 OPTIONS ステートメントは、SAS を CAS サーバーに接続します。CAS ステートメントでは、CASUSER caslib を使用して Mysess セッションが作成されます。LIBNAME ステートメントでは、CAS Engine の Mycas libref が作成されます。ここで、Mysess CAS セッションが使用されます。

```
options cashost="cloud.example.com" casport=5555;
cas mysess sessopts=(caslib='casuser');
libname mycas cas sessref=mysess;
```

テーブル Sashelp.cars を caslib Casuser にロードします。 OUTCASLIB=オプションでは、テーブルのロード先の caslib 名が指定されます。LOAD ステートメント

を使用して、Sashelp.cars からテーブルをロードします。ロードするテーブル名の指定における REPLACE オプションで、テーブルが置き換えられます。

```
proc casutil outcaslib="casuser";
  load data=sashelp.cars replace;
run;
```

PROC MDSUMMARY を使用してデータを要約します。 VAR ステートメントでは、結果の順序付けのための分析変数が指定されます。GROUPBY ステートメントでは、BY グループが作成され、テーブル Mycas.mpghw_sum に出力が保存されます。

```
proc mdsummary data=mycas.cars;
  var mpg_highway;
  groupby origin type / out=mycas.mpghw_sum;
run;
```

要約結果の最初の 15 行を印刷します。 OBS=15 を指定すると、PROC PRINT では、CAS テーブルのうち 15 行のみが印刷されます。VAR ステートメントでは、出力テーブルが、Origin、Type および_Mean_の 3 列に制限されます。

```
options obs=15;
proc print data=mycas.mpghw_sum;
   var origin type _mean_;
   title "Average Highway Mileages";
run;
```

Average Highway Mileages

Obs	Origin	Type	_Mean_
1	Asia	Hybrid	56
2	Asia	SUV	21.68
3	Asia	Sedan	29.968085106
4	Asia	Sports	26.647058824
5	Asia	Truck	22
6	Asia	Wagon	28.181818182
7	Europe	SUV	18.7
8	Europe	Sedan	27.115384615
9	Europe	Sports	25.130434783
10	Europe	Wagon	26.583333333
11	USA	SUV	20.04
12	USA	Sedan	28.544444444
13	USA	Sports	24.222222222
14	USA	Truck	20.5
15	USA	Wagon	29.714285714

例 2: 印刷する変数の選択

要素:

PROC PRINT ステートメントオプション
- BLANKLINE
- DOUBLE
- STYLE

VAR ステートメント
DATA ステップ
FOOTNOTE ステートメント
ODS HTML5 ステートメント
OPTIONS ステートメント
TITLE ステートメント

データセット: EXPREV

ODS 出力先: HTML5

詳細

この例は、次のタスクを示しています。

- レポートに 3 つの変数を選択する
- 変数ラベルを列ヘッダーとして使用する
- スタイルを指定した HTML5 レポートを作成する

プログラム: HTML5 レポートの作成

```
options obs=10;

proc print data=exprev;

  var country price sale_type;
  title 'Monthly Price Per Unit and Sale Type for Each Country';
  footnote '*prices in USD';
run;
```

プログラムの説明

HTML5 は、SAS Studio で SAS が開かれるときのデフォルトの出力先です。

OBS=システムオプションを設定して、10 のオブザベーションを処理します。

```
options obs=10;
```

出力を印刷します VAR ステートメントは、ランク付けする変数を指定します。

```
proc print data=exprev;

  var country price sale_type;
  title 'Monthly Price Per Unit and Sale Type for Each Country';
  footnote '*prices in USD';
run;
```

出力: HTML5

アウトプット 40.2　変数の選択: デフォルトの HTML5 出力

Monthly Price Per Unit and Sale Type for Each Country

Obs	Country	Price	Sale_Type
1	Antarctica	92.6	Internet
2	Puerto Rico	51.2	Catalog
3	Virgin Islands (U.S.)	31.1	In Store
4	Aruba	123.7	Catalog
5	Bahamas	113.4	Catalog
6	Bermuda	41.0	Catalog
7	Belize	146.4	In Store
8	British Virgin Islands	40.2	Catalog
9	Canada	11.8	Catalog
10	Cayman Islands	71.0	In Store

*prices in USD

プログラム: STYLE および BLANKLINE オプションを使用した HTML5 レポートの作成

```
options obs=5;
ods html5 file='your_file_styles.html';

proc print data=exprev
  style(header)={fontstyle=italic color= green}
  style(obs)={backgroundcolor=#a8a44ff8a color=blue}
  blankline=(count= 1 style={backgroundcolor=cx456789});
  var country price sale_type;
  title 'Monthly Price Per Unit and Sale Type for Each Country';
  footnote '*prices in USD';
run;
```

プログラムの説明

さらに、追加の形式設定を HTML5 出力に追加できます。次の例では STYLE オプションを使用して、シェーディングとスペースを HTML5 レポートに追加できます。

```
options obs=5;
ods html5 file='your_file_styles.html';
```

スタイルを指定した HTML5 出力を作成します。 最初の STYLE オプションは、列ヘッダーが緑色の斜体で書き込まれるように指定します。2 番目の STYLE オプションは、オブザベーション番号列の背景色が RGB カラー a8a44ff8a に、テキストの色が青になるように指定します。BLANKLINE オプションは、各オブザベーション間に黒

線を追加し、背景色に CMYK カラー cx456789 を使用するように指定します。OBSHEADER 位置にスタイルが定義されていないため、出力の Obs 列ヘッダーには緑ではなくデフォルトのスタイルカラーが使用されます。

```
proc print data=exprev
  style(header)={fontstyle=italic color= green}
  style(obs)={backgroundcolor=#a8a44ff8a color=blue}
  blankline=(count= 1 style={backgroundcolor=cx456789});
  var country price sale_type;
  title 'Monthly Price Per Unit and Sale Type for Each Country';
  footnote '*prices in USD';
run;
```

出力: スタイル付き出力の HTML5

アウトプット 40.3　変数の選択: スタイルを使用した HTML5 出力

Monthly Price Per Unit and Sale Type for Each Country

Obs	Country	Price	Sale_Type
1	Antarctica	92.6	Internet
2	Puerto Rico	51.2	Catalog
3	Virgin Islands (U.S.)	31.1	In Store
4	Aruba	123.7	Catalog
5	Bahamas	113.4	Catalog

*prices in USD

例 3: 列ヘッダーテキストのカスタマイズ

要素:
- PROC PRINT ステートメントオプション
  - N
  - OBS=
- VAR ステートメントオプション:
  - STYLE
- LABEL ステートメント
- ODS PDF ステートメント
- FORMAT ステートメント
- TITLE ステートメント

データセット: EXPREV

ODS 出力先: PDF

詳細

この例は、次のタスクを示しています。

- PDF 出力の変数の列ヘッダーに背景色を追加する
- 番号別にオブザベーションを識別する列の列ヘッダーをカスタマイズする
- レポートにオブザベーション数を表示する
- 変数 Price の値をドル記号とピリオドとともに書き込む

プログラム: PDF レポートの作成

```
options nodate pageno=1 linesize=80 pagesize=30 obs=10;
ods pdf file='your_file.pdf';
proc print data=exprev n obs='Observation Number';


  var country sale_type price;
      label country='Country Name'
          sale_type='Order Type'
          price='Price Per Unit in USD';
      format price dollar10.2;
      title 'Order Type and Price Per Unit in Each Country';
run;
ods pdf close;
```

プログラムの説明

次の例に示すように、ODS ステートメントをいくつか追加して、PDF 出力を簡単に作成できます。

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

```
options nodate pageno=1 linesize=80 pagesize=30 obs=10;
```

PDF 出力を作成し、出力を格納するファイルを指定します。 ODS PDF ステートメントは PDF 出力先を開き、PDF 出力を作成します。FILE=ステートメントで、PDF 出力を含む外部ファイルを指定します。

```
ods pdf file='your_file.pdf';
```

プロシジャオプションを設定します。 N オプションは、オブザベーション数をレポートの最後に印刷します。OBS=は、各オブザベーションを番号別に識別する列の列ヘッダーを指定します。

```
proc print data=exprev n obs='Observation Number';
```

データセットの変数を処理します。 VAR ステートメントは、ランク付けする変数を指定します。LABEL ステートメントは、変数名のかわりに印刷するテキストを作成します。FORMAT ステートメントは、DOLLARw.フォーマットを使用して価格変数をフォーマットするように指定します。TITLE ステートメントは、レポートのタイトルを作成します。

```
  var country sale_type price;
      label country='Country Name'
          sale_type='Order Type'
          price='Price Per Unit in USD';
      format price dollar10.2;
      title 'Order Type and Price Per Unit in Each Country';
run;
```

PDF 出力先を閉じます。 ODS PDF CLOSE ステートメントで、PDF 出力先を閉じます。

```
ods pdf close;
```

出力: PDF

アウトプット 40.4　列見出しのカスタマイズ: デフォルトの PDF 出力

Order Type and Price Per Unit in Each Country

Observation Number	Country	Sale_Type	Price
1	Antarctica	Internet	$92.60
2	Puerto Rico	Catalog	$51.20
3	Virgin Islands (U.S.)	In Store	$31.10
4	Aruba	Catalog	$123.70
5	Bahamas	Catalog	$113.40
6	Bermuda	Catalog	$41.00
7	Belize	In Store	$146.40
8	British Virgin Islands	Catalog	$40.20
9	Canada	Catalog	$11.80
10	Cayman Islands	In Store	$71.00
N = 10			

プログラム: STYLE オプションを使用した PDF レポートの作成

```
options obs=10;

ods pdf file='your_file.pdf';

proc print data=exprev n obs='Observation Number'
   style(n)={backgroundcolor=light blue fontstyle=italic}
   style(header obs obsheader)={backgroundcolor=light yellow color=blue
                        fontstyle=italic};
   style(data)={backgroundcolor=very light blue}

  var country sale_type price / style(data)=[backgroundcolor=very light blue];
  label country='Country Name'
       sale_type='Order Type'
       price='Price Per Unit in USD';
  format price dollar10.2;
run;

title 'Order Type and Price Per Unit in Each Country';

ods pdf close;
```

プログラムの説明

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;

ods pdf file='your_file.pdf';
```

スタイルを指定した PDF 出力を作成します。 最初の STYLE オプションは、N の値を含むセルの背景色が薄青色に、フォントスタイルが斜体にそれぞれ変更されるように指定します。2 番目の STYLE オプションは、オブザベーション列、オブザベーションヘッダー、その他の変数のヘッダーの背景色が薄黄色に、テキストの色が青色に、フォントスタイルが斜体にそれぞれ変更されるように指定します。

```
proc print data=exprev n obs='Observation Number'
   style(n)={backgroundcolor=light blue fontstyle=italic}
   style(header obs obsheader)={backgroundcolor=light yellow color=blue
                        fontstyle=italic};
   style(data)={backgroundcolor=very light blue}
```

スタイルを指定した PDF 出力を作成します。 STYLE オプションは、データを含むセルの色が非常に薄い青色に変更するものです。

```
  var country sale_type price / style(data)=[backgroundcolor=very light blue];
  label country='Country Name'
       sale_type='Order Type'
       price='Price Per Unit in USD';
  format price dollar10.2;
run;

title 'Order Type and Price Per Unit in Each Country';
```

PDF 出力先を閉じます。 ODS PDF CLOSE ステートメントで、PDF 出力先を閉じます。

```
ods pdf close;
```

出力: スタイル付き Report の PDF

アウトプット 40.5　列見出しのカスタマイズ: スタイルを使用した PDF

Observation Number	Country	Sale_Type	Price
1	Antarctica	Internet	$92.60
2	Puerto Rico	Catalog	$51.20
3	Virgin Islands (U.S.)	In Store	$31.10
4	Aruba	Catalog	$123.70
5	Bahamas	Catalog	$113.40
6	Bermuda	Catalog	$41.00
7	Belize	In Store	$146.40
8	British Virgin Islands	Catalog	$40.20
9	Canada	Catalog	$11.80
10	Cayman Islands	In Store	$71.00
N = 10			

例 4: オブザベーショングループごとに別々にレポートセクションを作成する

要素:
- PROC PRINT ステートメントオプション
  - LABEL
  - N=
  - NOOBS
  - STYLE
- BY ステートメント
- PAGEBY ステートメント
- SORT プロシジャ
- FORMAT ステートメント
- LABEL ステートメント
- ODS WORD ステートメント
- TITLE ステートメント

データセット: EXPREV

ODS 出力先: HTML5, WORD

詳細

この例では、次のタスクについて説明します。

- 各行の始めのオブザベーション番号の印刷を抑制する
- 種類ごとに販売データをレポートの別のセクションに表示する
- デフォルトの HTML5 レポートを作成する
- デフォルトの WORD レポートと様式的 RTF レポートを作成する

プログラム: HTML5 レポートの作成

```
options obs=10;

proc sort data=exprev;
  by sale_type order_date quantity;
run;

proc print data=exprev n='Number of observations for the month: '
        noobs label;

  var quantity cost price;

  by sale_type order_date;
  pageby order_date;

  label sale_type='Order Type' order_date='Order Date';

  format price dollar7.2 cost dollar7.2;
  title 'Prices and Cost Grouped by Date and Order Type';
  title2 'in USD';
run;

proc options option=bufno define;
run;
```

プログラムの説明

HTML5 出力先はデフォルトで開いています。ODS HTML5 ステートメントは必要ありません。

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;
```

EXPREV データセットを並べ替えます。 PROC SORT は、オブザベーションを Sale_Type、Order_Date、Quantity 別に並べ替えます。

```
proc sort data=exprev;
  by sale_type order_date quantity;
```

```
run;
```

レポートを印刷し、各 BY グループのオブザベーション数の合計を指定し、オブザベーション番号の印刷を抑制します。 N=は、BY グループのオブザベーション数を BY グループの最後に印刷します。N=オプションが提供する説明テキストは、その数字の前に表示されます。NOOBS は、行の始めのオブザベーション番号の印刷を抑制します。LABEL は、変数のラベルを列ヘッダーとして使用します。

```
proc print data=exprev n='Number of observations for the month: '
      noobs label;
```

レポートに含める変数を指定します。 VAR ステートメントは、Quantity、Cost、Price に対する列をこの順序で作成します。

```
var quantity cost price;
```

発注の種類ごとに別のセクションを作成し、Order_Date の BY グループごとに改ページを指定します。 BY ステートメントは、BY グループごとにレポートの別のセクションを生成し、それぞれの上にヘッダーを印刷します。PAGEBY ステートメントは、Order_Date の値が変わるたびに新しいページを開始します。

```
by sale_type order_date;
pageby order_date;
```

列ヘッダーを作成します。 LABEL ステートメントは、ラベルを PROC PRINT ステップの期間に対する変数 Sale_Type と Order_Date と関連付けます。PROC PRINT ステートメントで LABEL オプションを使用すると、プロシジャは列ヘッダーにラベルを使用します。

```
label sale_type='Order Type' order_date='Order Date';
```

数字を含む列をフォーマットして、タイトルとフットノートを指定します。 FORMAT ステートメントは、このレポートの Price と Cost に出力形式を割り当てます。TITLE ステートメントでタイトルを指定します。TITLE2 ステートメントは、2 番目のタイトルを指定します。

```
format price dollar7.2 cost dollar7.2;
title 'Prices and Cost Grouped by Date and Order Type';
title2 'in USD';
run;

proc options option=bufno define;
run;
```

出力: HTML5

アウトプット 40.6　オブザベーショングループごとに別々にレポートセクションを作成する: HTML5 出力

Prices and Cost Grouped by Date and Order Type in USD

Order Type=Catalog Order Date=1/1/18

Quantity	Cost	Price
7	$9.25	$41.00
8	$28.45	$113.40
14	$12.10	$51.20
30	$59.00	$123.70
Number of observations for the month: 4		

Prices and Cost Grouped by Date and Order Type in USD

Order Type=Catalog Order Date=1/2/18

Quantity	Cost	Price
11	$20.20	$40.20
100	$5.00	$11.80
Number of observations for the month: 2		

Prices and Cost Grouped by Date and Order Type in USD

Order Type=In Store Order Date=1/1/18

Quantity	Cost	Price
25	$15.65	$31.10
Number of observations for the month: 1		

Prices and Cost Grouped by Date and Order Type in USD

Order Type=In Store Order Date=1/2/18

Quantity	Cost	Price
2	$36.70	$146.40
20	$32.30	$71.00
Number of observations for the month: 2		

Prices and Cost Grouped by Date and Order Type in USD

Order Type=Internet Order Date=1/1/18

Quantity	Cost	Price
2	$20.70	$92.60
Number of observations for the month: 1		

プログラム: Word レポートの作成

```
options obs=10;

ods word file='your_file.docx' startpage=no;

proc sort data=exprev;
  by sale_type order_date quantity;
run;

proc print data=exprev n='Number of observations for each order type:'
          noobs label;
  var quantity cost price;
  by sale_type order_date;
  pageby order_date;
  label sale_type='Order Type' order_date='Order Date';
  format price dollar7.2 cost dollar7.2;
  title 'Price and Cost Grouped by Date and Order Type';
  title2 'in USD';
run;

ods word close;
```

プログラムの説明

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;
```

Microsoft Word に対する出力を作成し、出力を格納するファイルを指定します。 ODS WORD ステートメントは WORD 出力先を開いて、Microsoft Word に対しフォーマットされた出力を作成します。FILE=オプションは、WORD 出力を含む外部ファイルを指定します。STARTPAGE=NO オプションは、新しいページが各 BY グループの始めに明示的に挿入されないように指定します。

```
ods word file='your_file.docx' startpage=no;

proc sort data=exprev;
  by sale_type order_date quantity;
run;

proc print data=exprev n='Number of observations for each order type:'
        noobs label;
  var quantity cost price;
  by sale_type order_date;
  pageby order_date;
  label sale_type='Order Type' order_date='Order Date';
  format price dollar7.2 cost dollar7.2;
  title 'Price and Cost Grouped by Date and Order Type';
  title2 'in USD';
run;
```

WORD 出力先を閉じます。 ODS WORD CLOSE ステートメントで、WORD 出力先を閉じます。

```
ods word close;
```

出力: WORD

アウトプット 40.7　オブザベーショングループごとに別々にレポートセクションを作成する: デフォルト WORD 出力

Price and Cost Grouped by Date and Order Type in USD

Order Type=Catalog Order Date=1/1/18

Quantity	Cost	Price
7	$9.25	$41.00
8	$28.45	$113.40
14	$12.10	$51.20
30	$59.00	$123.70
Number of observations for each order type:4		

Order Type=Catalog Order Date=1/2/18

Quantity	Cost	Price
11	$20.20	$40.20
100	$5.00	$11.80
Number of observations for each order type:2		

Order Type=In Store Order Date=1/1/18

Quantity	Cost	Price
25	$15.65	$31.10
Number of observations for each order type:1		

Order Type=In Store Order Date=1/2/18

Quantity	Cost	Price
2	$36.70	$146.40
20	$32.30	$71.00
Number of observations for each order type:2		

Order Type=Internet Order Date=1/1/18

Quantity	Cost	Price
2	$20.70	$92.60
Number of observations for each order type:1		

プログラム: STYLE オプションを使用した Word レポートの作成

```
options obs=10;

ods word file='your_file.docx' startpage=no;

proc sort data=exprev;
  by sale_type order_date quantity;
```

```
run;

proc print data=exprev n='Number of observations for the month: '
        noobs label style(N)={backgroundcolor=very light gray};
  var quantity / style(header)=[backgroundcolor=light yellow];
  var cost / style(header)=[backgroundcolor=light blue foreground =
white];
  var price / style(header)=[backgroundcolor=light green];
  by sale_type order_date;
  pageby order_date;
  label sale_type='Order Type' order_date='Order Date';
  format price dollar7.2 cost dollar7.2;

title 'Prices and Cost Grouped by Date and Order Type';
title2 '*prices in USD';
run;

ods word close;
```

プログラムの説明

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;

ods word file='your_file.docx' startpage=no;

proc sort data=exprev;
  by sale_type order_date quantity;
run;
```

スタイルを指定した WORD レポートを作成します。 最初の STYLE オプションは、オブザベーションの数を含むセルの背景色が薄灰色に変更されるように指定します。2 番目の STYLE オプションは、変数 Quantity の列ヘッダーの背景色が薄黄色に変更されるように指定します。3 番目の STYLE オプションは、変数 Cost の列ヘッダーの背景色が薄青色に、フォントの色が白色にそれぞれ変更されるように指定します。4 番目の STYLE オプションは、変数 Price の列ヘッダーの背景色が薄緑色に変更されるように指定します。

```
proc print data=exprev n='Number of observations for the month: '
        noobs label style(N)={backgroundcolor=very light gray};
  var quantity / style(header)=[backgroundcolor=light yellow];
  var cost / style(header)=[backgroundcolor=light blue foreground =
white];
  var price / style(header)=[backgroundcolor=light green];
  by sale_type order_date;
  pageby order_date;
  label sale_type='Order Type' order_date='Order Date';
  format price dollar7.2 cost dollar7.2;

title 'Prices and Cost Grouped by Date and Order Type';
title2 '*prices in USD';
run;

ods word close;
```

出力: スタイル付きの Word

アウトプット 40.8　ブザベーショングループごとに別々にレポートセクションを作成する: WORD 出力スタイルの使用

Prices and Cost Grouped by Date and Order Type
***prices in USD**

Order Type=Catalog Order Date=1/1/18

Quantity	Cost	Price
7	$9.25	$41.00
8	$28.45	$113.40
14	$12.10	$51.20
30	$59.00	$123.70
Number of observations for the month: 4		

Order Type=Catalog Order Date=1/2/18

Quantity	Cost	Price
11	$20.20	$40.20
100	$5.00	$11.80
Number of observations for the month: 2		

Order Type=In Store Order Date=1/1/18

Quantity	Cost	Price
25	$15.65	$31.10
Number of observations for the month: 1		

Order Type=In Store Order Date=1/2/18

Quantity	Cost	Price
2	$36.70	$146.40
20	$32.30	$71.00
Number of observations for the month: 2		

Order Type=Internet Order Date=1/1/18

Quantity	Cost	Price
2	$20.70	$92.60
Number of observations for the month: 1		

例 5: 1 つの BY グループを使用して数値変数を合計する

要素:

PROC PRINT ステートメントオプション
- N=
- SUMLABEL

BY ステートメント

SUM ステートメント
ODS CSVALL ステートメント
SORT プロシジャ
TITLE ステートメント
　#BYVAL 指定
SAS システムオプション:
　BYLINE
　NOBYLINE

データセット: EXPREV

ODS 出力先: HTML5, CSV

詳細

この例は、次のタスクを示しています。

- リージョンごと、およびすべてのリージョンに対する経費と収益を合計します。
- 各 BY グループおよびレポート全体のオブザベーション数を表示する
- リージョン名を含む、カスタマイズされたタイトルを作成します。このタイトルは、BY グループごとのデフォルトの BY 行となります。
- デフォルト HTML5 ファイルを作成します。
- CSV ファイルを作成します。

プログラム: HTML5 レポートの作成

```
options obs=10 nobyline;

proc sort data=exprev;
  by sale_type;
run;

proc print data=exprev noobs label sumlabel
          n='Number of observations for the order type: '
          'Number of observations for the data set: ';

  var country order_date quantity price;

  label  sale_type='Sale Type'
        price='Total Retail Price* in USD'
        country='Country' order_date='Date' quantity='Quantity';

  sum price quantity;
  by sale_type;

  format price dollar7.2;

  title 'Retail and Quantity Totals for #byval(sale_type) Sales';
run;

options byline;
```

プログラムの説明

HTML5 出力先はデフォルトで開いています。このプログラムでは、HTML5 出力にデフォルトのファイル名が使用されます。ODS HTML5 ステートメントは必要ありません。

新しいページで BY グループをそれぞれ開始し、デフォルトの BY 行の印刷を抑制します。 SAS システムオプション NOBYLINE は、デフォルトの BY 行の印刷を抑制します。PROC PRINT を NOBYLINE オプションと使用すると、BY グループがそれぞれ新しいページを開始します。OBS=オプションは、処理するオブザベーション数を指定します。

```
options obs=10 nobyline;
```

データセットを並べ替えます。 PROC SORT は、オブザベーションを Sale_Type 別に並べ替えます。

```
proc sort data=exprev;
  by sale_type;
run;
```

レポートを印刷し、オブザベーション数の印刷を抑制し、選択した変数に対するオブザベーションの合計数を印刷します。 NOOBS は、行の始めのオブザベーション番号の印刷を抑制します。SUMLABEL は、それぞれの要約行に BY 変数ラベルを印刷します。N=は BY グループの最後に BY グループのオブザベーション数を印刷し、(SUM ステートメントにより)レポートの最後にデータセットのオブザベーション数を印刷します。N=が提供する最初の説明テキストが BY グループごとの数より前に置かれます。N=が提供する 2 番目の説明テキストがデータセット全体の数より前に置かれます。

```
proc print data=exprev noobs label sumlabel
        n='Number of observations for the order type: '
        'Number of observations for the data set: ';
```

レポートに含める変数を選択します。 VAR ステートメントは、Country、Order_Date、Quantity、Price に対する列をこの順序で作成します。

```
var country order_date quantity price;
```

変数のラベルを列ヘッダーとして割り当てます。 LABEL ステートメントは、ラベルを ROC PRINT ステップの期間の各変数と連付けます。

```
label  sale_type='Sale Type'
     price='Total Retail Price* in USD'
     country='Country' order_date='Date' quantity='Quantity';
```

選択されている変数の値を合計します。 SUM ステートメントは、それだけでデータセット全体の Price と Quantity の値を合計します。PROC PRINT ステップに BY ステートメントが含まれているため、SUM ステートメントも複数のオブザベーションを含む販売の種類ごとに Price と Quantity の値を合計します。

```
sum price quantity;
by sale_type;
```

指定列の数値をフォーマットします。 FORMAT ステートメントは、DOLLAR7.2.出力形式をこのレポートの Price に割り当てます。

```
format price dollar7.2;
```

動的な(または現在の)タイトルを指定して、フォーマットします。 TITLE ステートメントでタイトルを指定します。#BYVAL 指定は、タイトルに BY 変数 Sale_Type の現在の値を配置します。NOBYLINE が有効であるため、BY グループはそれぞれ新しいページで開始し、タイトルは BY 行として機能します。

```
    title 'Retail and Quantity Totals for #byval(sale_type) Sales';
run;
```

デフォルトの BY 行を生成します。 SAS システムオプション BYLINE は、デフォルトの BY 行の印刷をリセットします。

```
options byline;
```

出力: HTML5

アウトプット 40.9　1 つの BY グループ HTML5 出力を使用して数値変数を合計する

Retail and Quantity Totals for Catalog Sales

Country	Date	Quantity	Total Retail Price* in USD
Bermuda	1/1/18	7	$41.00
Bahamas	1/1/18	8	$113.40
Puerto Rico	1/1/18	14	$51.20
Aruba	1/1/18	30	$123.70
British Virgin Islands	1/2/18	11	$40.20
Canada	1/2/18	100	$11.80
Sale Type		**170**	**$381.30**

Number of observations for the order type: 6

Retail and Quantity Totals for In Store Sales

Country	Date	Quantity	Total Retail Price* in USD
Virgin Islands (U.S.)	1/1/18	25	$31.10
Belize	1/2/18	2	$146.40
Cayman Islands	1/2/18	20	$71.00
Sale Type		**47**	**$248.50**

Number of observations for the order type: 3

Retail and Quantity Totals for Internet Sales

Country	Date	Quantity	Total Retail Price* in USD
Antarctica	1/1/18	2	$92.60
		219	**$722.40**

Number of observations for the order type: 1
Number of observations for the data set: 10

プログラム: CSV ファイルの作成

```
options obs=10 nobyline;

ods csvall file='your_file.csv';

proc sort data=exprev;
  by sale_type;
run;

proc print data=exprev noobs label sumlabel
          n='Number of observations for the order type: '
          'Number of observations for the data set: ';

var country order_date quantity price;

  label  price='Total Retail Price* in USD'
        country='Country' order_date='Date' quantity='Quantity';

  sum price quantity;
  by sale_type;

  format price dollar7.2;

  title 'Retail and Quantity Totals for #byval(sale_type) Sales';
run;

options byline;

ods csvall close;
```

プログラムの説明

```
options obs=10 nobyline;
```

CSV 形式の出力を生成し、格納するファイルを指定します。 ODS CSVALL ステートメントは CSVALL 出力先を開き、表形式出力を含むファイルをタイトル、説明、BY 行で作成します。FILE=引数は、CSV 出力を含む外部ファイルを指定します。

```
ods csvall file='your_file.csv';

proc sort data=exprev;
  by sale_type;
run;

proc print data=exprev noobs label sumlabel
          n='Number of observations for the order type: '
          'Number of observations for the data set: ';

var country order_date quantity price;

  label  price='Total Retail Price* in USD'
        country='Country' order_date='Date' quantity='Quantity';

  sum price quantity;
  by sale_type;

  format price dollar7.2;

  title 'Retail and Quantity Totals for #byval(sale_type) Sales';
run;

options byline;
```

CSVALL 出力先を閉じます。 ODS CSVALL CLOSE ステートメントで、CSVALL 出力先を閉じます。

```
ods csvall close;
```

出力: CSV ファイル

アウトプット 40.10　1 つの BY グループを使用した数値変数の合計: Microsoft Excel で表示された CSV 出力

	A	B	C	D	E	F
1	Retail and Quantity Totals for Catalog Sales					
2						
3	Country	Date	Quantity	Total Retail Price* in USD		
4	Bermuda	1/1/2018	7	$41.00		
5	Bahamas	1/1/2018	8	$113.40		
6	Puerto Ri	1/1/2018	14	$51.20		
7	Aruba	1/1/2018	30	$123.70		
8	British Vir	1/2/2018	11	$40.20		
9	Canada	1/2/2018	100	$11.80		
10	Sale_Type		170	$381.30		
11	Number of observations for the order type: 6					
12						
13	Retail and Quantity Totals for In Store Sales					
14						
15	Country	Date	Quantity	Total Retail Price* in USD		
16	Virgin Isla	1/1/2018	25	$31.10		
17	Belize	1/2/2018	2	$146.40		
18	Cayman Is	1/2/2018	20	$71.00		
19	Sale_Type		47	$248.50		
20	Number of observations for the order type: 3					
21						
22	Retail and Quantity Totals for Internet Sales					
23						
24	Country	Date	Quantity	Total Retail Price* in USD		
25	Antarctica	1/1/2018	2	$92.60		
26			219	$722.40		
27	Number of observations for the order type: 1					
28	Number of observations for the data set: 10					
29						

例 6: 複数の BY 変数を使用して数値変数を合計する

要素:

PROC PRINT ステートメントオプション
- GRANDTOTAL_LABEL=
- N=
- NOOBS
- SUMLABEL=

BY ステートメント

SUM ステートメント

ODS HTML5 ステートメント
LABEL ステートメント
FORMAT ステートメント
SORT プロシジャ
TITLE ステートメント

データセット: EXPREV

ODS 出力先: HTML5

詳細

この例は、次のタスクを示しています。

- 次の項目に対する数量と販売価格を合計する:
  - 各注文日
  - レポートに複数行が存在する販売の種類
  - レポートのすべての行
- 各 BY グループおよびレポート全体のオブザベーション数を表示します
- 要約行の BY グループ変数名のかわりにカスタマイズしたラベルを表示する
- 総計行にカスタマイズしたラベルを表示する
- デフォルトの HTML5 レポートを作成する
- スタイルを指定した HTML5 レポートを作成する

プログラム: HTML5 レポートの作成

```
options obs=10;

proc sort data=exprev;
  by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel='Totals' grandtotal_label='Grand Total';
  by sale_type order_date;
  sum price quantity cost;

  label  sale_type='Sale Type' order_date='Sale Date';
  format price dollar10.2 cost dollar10.2;
  title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;
```

プログラムの説明

```
options obs=10;
```

HTML5 出力を生成します。 SORT プロシジャは、まず EXPREV データセットを sale_type および order_date で並べ替えます。次に、PROC PRINT ステップは並べ替えられたデータを出力し、同じ変数ごとに出力をグループ化して、price、quantity、および cost 列の小計と総計を表示します。

```
proc sort data=exprev;
  by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel='Totals' grandtotal_label='Grand Total';
  by sale_type order_date;
  sum price quantity cost;

  label  sale_type='Sale Type' order_date='Sale Date';
  format price dollar10.2 cost dollar10.2;
  title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;
```

出力: HTML5

アウトプット 40.11 複数の BY 変数を含む数値変数の合計: 店舗販売: デフォルトの HTML5 出力

Retail and Quantity Totals for Each Sale Date and Sale Type

Sale Type=Catalog Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Bermuda	99999999	1/4/18	7	$41.00	$9.25
Bahamas	99999999	1/4/18	8	$113.40	$28.45
Puerto Rico	99999999	1/5/18	14	$51.20	$12.10
Aruba	99999999	1/4/18	30	$123.70	$59.00
Totals			**59**	**$329.30**	**$108.80**

N = 4

Sale Type=Catalog Sale Date=1/2/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
British Virgin Islands	99999999	1/5/18	11	$40.20	$20.20
Canada	99999999	1/5/18	100	$11.80	$5.00
Totals			**111**	**$52.00**	**$25.20**
Totals			**170**	**$381.30**	**$134.00**

N = 2

Sale Type=In Store Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Virgin Islands (U.S.)	99999999	1/4/18	25	$31.10	$15.65

N = 1

Sale Type=In Store Sale Date=1/2/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Belize	120458	1/2/18	2	$146.40	$36.70
Cayman Islands	120454	1/2/18	20	$71.00	$32.30
Totals			**22**	**$217.40**	**$69.00**
Totals			**47**	**$248.50**	**$84.65**

N = 2

Sale Type=Internet Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Antarctica	99999999	1/7/18	2	$92.60	$20.70
Grand Total			**219**	**$722.40**	**$239.35**

N = 1
Total N = 10

プログラム: STYLE オプションを使用して HTML5 レポートを作成する

```
options obs=10;

proc sort data=exprev;
  by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel='Totals' grandtotal_label='Grand Total';
```

```
  by sale_type order_date;
sum price / style(GRANDTOTAL)=[backgroundcolor=white color=blue];
sum quantity / style(TOTAL)=[backgroundcolor=dark blue color=white];

  label  sale_type='Sale Type' order_date='Sale Date';
  format price dollar10.2 cost dollar10.2;
  title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;
```

プログラムの説明

```
options obs=10;

proc sort data=exprev;
  by sale_type order_date;
run;

proc print data=exprev n noobs sumlabel='Totals' grandtotal_label='Grand Total';
```

スタイルを指定した HTML5 出力を作成します。 最初の SUM ステートメントの STYLE オプションは、変数 Price の総計を含むセルの背景色が白に、フォントの色が青に変更されるように指定します。2 番目の SUM ステートメントの STYLE オプションは、変数 Quantity の合計を含むセルの背景色が紺青に、フォントの色が白に変更されるように指定します。

```
  by sale_type order_date;
sum price / style(GRANDTOTAL)=[backgroundcolor=white color=blue];
sum quantity / style(TOTAL)=[backgroundcolor=dark blue color=white];

  label  sale_type='Sale Type' order_date='Sale Date';
  format price dollar10.2 cost dollar10.2;
  title 'Retail and Quantity Totals for Each Sale Date and Sale Type';
run;
```

出力: スタイル付きの HTML5

アウトプット 40.12　複数の BY 変数を含む数値変数の合計: カタログ販売: スタイルを使用した HTML5 出力

Retail and Quantity Totals for Each Sale Date and Sale Type

Sale Type=Catalog Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Bermuda	99999999	1/4/18	7	$41.00	$9.25
Bahamas	99999999	1/4/18	8	$113.40	$28.45
Puerto Rico	99999999	1/5/18	14	$51.20	$12.10
Aruba	99999999	1/4/18	30	$123.70	$59.00
Totals			**59**	**$329.30**	

N = 4

Sale Type=Catalog Sale Date=1/2/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
British Virgin Islands	99999999	1/5/18	11	$40.20	$20.20
Canada	99999999	1/5/18	100	$11.80	$5.00
Totals			**111**	**$52.00**	
Totals			**170**	**$381.30**	

N = 2

Sale Type=In Store Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Virgin Islands (U.S.)	99999999	1/4/18	25	$31.10	$15.65

N = 1

Sale Type=In Store Sale Date=1/2/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Belize	120458	1/2/18	2	$146.40	$36.70
Cayman Islands	120454	1/2/18	20	$71.00	$32.30
Totals			**22**	**$217.40**	
Totals			**47**	**$248.50**	

N = 2

Sale Type=Internet Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Antarctica	99999999	1/7/18	2	$92.60	$20.70
Grand Total			**219**	**$722.40**	

N = 1
Total N = 10

例 7: レポート内の合計数を制限する

要素:

- BY ステートメント
- SUM ステートメント
- SUMBY ステートメント
- FORMAT ステートメント
- LABEL ステートメント
- ODS PDF ステートメント
- SORT プロシジャ

TITLE ステートメント

データセット: EXPREV

ODS 出力先: PDF

詳細

この例は、次のタスクを示しています。

- 販売の種類と販売日の組み合わせごとにレポートの別のセクションを作成する
- 個別の日付ではなく、販売の種類別およびすべての販売の種類に対する数量と販売価格のみを合計する
- PDF ファイルを作成する

プログラム: PDF ファイルの作成

```
options obs=10;

ods html5 close;
ods pdf file='your_file.pdf';

proc sort data=exprev;
  by sale_type order_date;
run;

proc print data=exprev noobs sumlabel='Total' grandtotal_label='Grand Total';

  by sale_type order_date;
  sum price quantity;
  sumby sale_type;

  label sale_type='Sale Type' order_date='Sale Date';

  format price dollar10.2 cost dollar10.2;
  title 'Retail and Quantity Totals for Each Sale Type';
run;

ods pdf close;
```

プログラムの説明

OBS=システムオプションで、10 個のオブザベーションを処理することを指定します。

```
options obs=10;
```

PDF 出力を作成し、出力を格納するファイルを指定します。 ODS HTML5 CLOSE ステートメントはデフォルトの出力先を閉じます。ODS PDF ステートメントは PDF 出力先を開いて、PDF 出力を含むファイルを作成します。FILE=ステートメントで、PDF 出力を含む外部ファイルを指定します。

```
ods html5 close;
ods pdf file='your_file.pdf';
```

データセットを並べ替えます。 PROC SORT は、オブザベーションを Sales_Type と Order_Date 別に並べ替えます。

```
proc sort data=exprev;
  by sale_type order_date;
run;
```

レポートを印刷し、オブザベーション番号を削除します。 NOOBS は、行の始めのオブザベーション番号の印刷を抑制します。SUMLABEL は、各 BY グループの要約行の BY 変数のラベルを使用します。

```
proc print data=exprev noobs sumlabel='Total' grandtotal_label='Grand Total';
```

リージョンごとの値を合計します。 SUM ステートメントと BY ステートメントを使用して、BY グループごと、レポート全体の Price と Quantity の値を合計します。SUMBY ステートメントは、小計を販売の種類ごとに 1 つに制限します。

```
  by sale_type order_date;
  sum price quantity;
  sumby sale_type;
```

ラベルを特定の変数に割り当てます。 LABEL ステートメントは、ラベルを ROC PRINT ステップの期間に対する変数 Sale_Type と Order_Date と連付けます。これらのラベルは、BY グループタイトルまたは要約行で使用されます。

```
  label sale_type='Sale Type' order_date='Sale Date';
```

出力形式を必要な変数に割り当て、タイトルを指定します。 FORMAT ステートメントは、このレポートの Cost と Price に COMMA10.出力形式を割り当てます。TITLE ステートメントでタイトルを指定します。

```
  format price dollar10.2 cost dollar10.2;
  title 'Retail and Quantity Totals for Each Sale Type';
run;
```

PDF 出力先を閉じます。 ODS PDF CLOSE ステートメントで、PDF 出力先を閉じます。

```
ods pdf close;
```

出力: PDF

アウトプット 40.13　レポート内の合計数を制限する: PDF 出力

Retail and Quantity Totals for Each Sale Type

Sale Type=Catalog Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Bermuda	99999999	1/4/18	7	$41.00	$9.25
Bahamas	99999999	1/4/18	8	$113.40	$28.45
Puerto Rico	99999999	1/5/18	14	$51.20	$12.10
Aruba	99999999	1/4/18	30	$123.70	$59.00

Sale Type=Catalog Sale Date=1/2/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
British Virgin Islands	99999999	1/5/18	11	$40.20	$20.20
Canada	99999999	1/5/18	100	$11.80	$5.00
Total			**170**	**$381.30**	

Sale Type=In Store Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Virgin Islands (U.S.)	99999999	1/4/18	25	$31.10	$15.65

Sale Type=In Store Sale Date=1/2/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Belize	120458	1/2/18	2	$146.40	$36.70
Cayman Islands	120454	1/2/18	20	$71.00	$32.30
Total			**47**	**$248.50**	

Sale Type=Internet Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Antarctica	99999999	1/7/18	2	$92.60	$20.70
Grand Total			**219**	**$722.40**	

プログラム: STYLE オプションを使用した PDF レポートの作成

```
options obs=10;

ods pdf file='your_file.pdf';

proc sort data=exprev;
  by sale_type order_date;
run;

proc print data=exprev noobs sumlabel='Total' grandtotal_label='Grand Total';

  by sale_type order_date;
```

```
  sum quantity price / style(TOTAL)=[backgroundcolor=light blue color=white];
  sum quantity price / style(GRANDTOTAL)=[backgroundcolor=green color=white];
  sumby sale_type;

  label sale_type='Sale Type' order_date='Sale Date';

  format price dollar10.2 cost dollar10.2;
  title 'Retail and Quantity Totals for Each Sale Type';
run;

ods pdf close;
```

プログラムの説明

```
options obs=10;

ods pdf file='your_file.pdf';

proc sort data=exprev;
  by sale_type order_date;
run;

proc print data=exprev noobs sumlabel='Total' grandtotal_label='Grand Total';

  by sale_type order_date;
```

スタイルを指定した PDF 出力を作成します。 最初の SUM ステートメントの STYLE オプションは、変数 Price の合計を含むセルの背景色が薄青色に、フォントの色が白色に変更されるように指定します。2 番目の SUM ステートメントの STYLE オプションは、Quantity 変数の総計を含むセルの背景色が緑に、フォントの色が白に変更されるように指定します。

```
  sum quantity price / style(TOTAL)=[backgroundcolor=light blue color=white];
  sum quantity price / style(GRANDTOTAL)=[backgroundcolor=green color=white];
  sumby sale_type;

  label sale_type='Sale Type' order_date='Sale Date';

  format price dollar10.2 cost dollar10.2;
  title 'Retail and Quantity Totals for Each Sale Type';
run;

ods pdf close;
```

出力: スタイル付きの PDF

アウトプット 40.14　レポート内の合計数を制限する: スタイルを使用した PostScript 出力

Retail and Quantity Totals for Each Sale Type

Sale Type=Catalog Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Bermuda	99999999	1/4/18	7	$41.00	$9.25
Bahamas	99999999	1/4/18	8	$113.40	$28.45
Puerto Rico	99999999	1/5/18	14	$51.20	$12.10
Aruba	99999999	1/4/18	30	$123.70	$59.00

Sale Type=Catalog Sale Date=1/2/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
British Virgin Islands	99999999	1/5/18	11	$40.20	$20.20
Canada	99999999	1/5/18	100	$11.80	$5.00
Total			170	$381.30	

Sale Type=In Store Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Virgin Islands (U.S.)	99999999	1/4/18	25	$31.10	$15.65

Sale Type=In Store Sale Date=1/2/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Belize	120458	1/2/18	2	$146.40	$36.70
Cayman Islands	120454	1/2/18	20	$71.00	$32.30
Total			47	$248.50	

Sale Type=Internet Sale Date=1/1/18

Country	Emp_ID	Ship_Date	Quantity	Price	Cost
Antarctica	99999999	1/7/18	2	$92.60	$20.70
Grand Total			219	$722.40	

例 8: 多数の変数を使用したレポートのレイアウトを制御する

要素:

PROC PRINT ステートメントオプション
　ROWS=

ID ステートメントオプション
　STYLE

ODS WORD ステートメント

SAS データセットオプション
OBS=

データセット: EMPDATA

ODS 出力先: WORD

詳細

この例では、多くの変数を含むデータセットを印刷するための 2 つの方法を示します。1 つはデフォルトで、多くの変数がある場合に複数行を印刷します。もう 1 つは ROWS=オプションを使用して 1 行を印刷します。ROWS=オプションは、LISTING 出力先にのみ有効です。 これらの 2 つのレポートのレイアウトに関する詳細説明については、オプション ROWS= (1401 ページ)と “ページのサイズが制限されている場合のページレイアウト” (1439 ページ)を参照してください。

これらのレポートには、異なるレイアウトの表示に役立つページサイズ 24、ページサイズ 64 が使用されます。

プログラム: WORD レポートの作成

```
options pageno=1;

ods word file='your_file.docx';

proc print data=empdata(obs=12);
  id idnumber;
  title 'Personnel Data';
run;

ods word close;
```

プログラムの説明

WORD 出力には、すべてのデータが 1 ページに表示されます。

```
options pageno=1;
```

Microsoft Word に対する出力を作成し、出力を格納するファイルを指定します。
ODS WORD ステートメントは WORD 出力先を開いて、Microsoft Word に対しフォーマットされた出力を作成します。FILE=引数は、WORD 出力を含む外部ファイルを指定します。

```
ods word file='your_file.docx';

proc print data=empdata(obs=12);
  id idnumber;
  title 'Personnel Data';
run;
```

WORD 出力先を閉じます。 ODS WORD CLOSE ステートメントで、WORD 出力先を閉じます。

```
ods word close;
```

出力: WORD

アウトプット 40.15　多数の変数を使用したレポートのレイアウト: WORD 出力

Personnel Data

IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary	Birth	Hired	HomePhone
1919	Adams	Gerald	Stamford	CT	M	TA2	34376	12SEP1990	04JUN2020	203/781-1255
1653	Ahmad	Azeem	Bridgeport	CT	F	ME2	35108	15OCT1984	09AUG2020	203/675-7715
1400	Alvarez	Gloria	New York	NY	M	ME1	29769	05NOV1987	16OCT2020	212/586-0808
1350	Arthur	Barbara	New York	NY	F	FA3	32886	31AUG1985	29JUL2020	718/383-1549
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822	13DEC1970	17NOV2020	201/732-8787
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025	26APR1974	07JUN2020	201/812-5665
1101	Basquez	Richardo	New York	NY	M	SCP	18723	06JUN1982	01OCT2020	212/586-8060
1333	Beaulieu	Armando	Stamford	CT	M	PT2	88606	30MAR1983	10FEB2020	203/781-1777
1402	Blalock	Ralph	New York	NY	M	TA2	32615	17JAN1983	02DEC2020	718/384-2849
1479	Bostic	Marie	New York	NY	F	TA3	38785	22DEC1988	05OCT2020	718/384-8816
1403	Bowden	Earl	Bridgeport	CT	M	ME1	28072	28JAN1989	21DEC2020	203/675-3434
1739	Boyce	Jonathan	New York	NY	M	PT1	66517	25DEC1984	27JAN2020	212/587-1247

例 9: BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成

要素:
- PROC PRINT ステートメントオプション
  - SUMLABEL
  - GRANDTOTAL_LABEL
- BY ステートメント
- ID ステートメント
- SUM ステートメント
- VAR ステートメント
- SORT プロシジャ

データセット: EMPDATA

ODS 出力先: HTML5

詳細

このカスタマイズされたレポートでは、次のタスクについて説明します。

- レポートに含める変数と、表示される順序を選択する
- レポートに含めるオブザベーションを選択する
- 選択したオブザベーションを JobCode 別にグループ化する
- 給与をジョブコードごと、およびすべてのジョブコードに対して合計する
- 数値データをカンマとドル記号とともに表示する

プログラム: HTML5 レポートの作成

```
proc sort data=empdata out=tempemp;
  by jobcode gender;
run;

ods html5 file='your_file.html';

proc print data=tempemp (obs=10) sumlabel='Total' grandtotal_lable='Grand Total';

 id jobcode;
  by jobcode;
  var gender salary;

  sum salary;

  label jobcode='Job Code'
       gender='Gender'
       salary='Annual Salary';

  format salary dollar11.2;
  where jobcode contains 'FA' or jobcode contains 'ME';
  title 'Salary Expenses';
run;
```

プログラムの説明

```
proc sort data=empdata out=tempemp;
  by jobcode gender;
run;
```

HTML5 出力を作成し、出力を格納するファイルを指定します。 HTML5 出力先は、デフォルトの ODS 出力先です。ODS HTML5 ステートメント FILE=オプションは、HTML5 出力を含む外部ファイルを指定します。

```
ods html5 file='your_file.html';
```

プロシジャのオプションを定義してください。 (obs=10)データセットのオプションは、処理するオブザベーション数を指定します。SUMLABEL オプションは、各 BY

グループの要約行でラベル'Total'を使用することを示します。
GRANDTOTAL_LABEL オプションは、レポート内のすべての BY グループの後、総計行でラベル'Grand Total'を使用することを示します。

```
proc print data=tempemp (obs=10) sumlabel='Total' grandtotal_lable='Grand Total';

 id jobcode;
  by jobcode;
  var gender salary;

  sum salary;

  label jobcode='Job Code'
       gender='Gender'
       salary='Annual Salary';

  format salary dollar11.2;
  where jobcode contains 'FA' or jobcode contains 'ME';
  title 'Salary Expenses';
run;
```

出力: HTML5

アウトプット 40.16　BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成: デフォルトの HTML5 出力

Salary Expenses

JobCode	Gender	Salary
FA1	F	$23,177.00
	F	$22,454.00
	M	$22,268.00
Total		**$67,899.00**

JobCode	Gender	Salary
FA2	F	$28,888.00
	F	$27,787.00
	M	$28,572.00
Total		**$85,247.00**

JobCode	Gender	Salary
FA3	F	$32,886.00
	F	$33,419.00
	M	$32,217.00
Total		**$98,522.00**

JobCode	Gender	Salary
ME1	M	$29,769.00
Grand Total		**$281,437.00**

プログラム: STYLE オプションを使用して HTML5 レポートを作成する

```
proc sort data=empdata out=tempemp;
  by jobcode gender;
run;

proc print data=tempemp (obs=10)sumlabel='Total' grandtotal_label='Grand Total'
  style(HEADER)={fontstyle=italic}
  style(DATA)={backgroundcolor=blue foreground=white};

 id jobcode;
  by jobcode;
  var gender salary;

  sum salary  / style(total)={color=red};

  label jobcode='Job Code'
        gender='Gender'
        salary='Annual Salary';

  format salary dollar11.2;
  where jobcode contains 'FA' or jobcode contains 'ME';
  title 'Expenses Incurred for';
  title2 'Salaries for Flight Attendants and Mechanics';
run;
```

プログラムの説明

```
proc sort data=empdata out=tempemp;
  by jobcode gender;
run;
```

スタイルを指定した HTML5 出力を作成します。 最初の STYLE オプションは、ヘッダーのフォントが斜体に変更されるように指定します。2 番目の STYLE オプションは、データを含むセルの背景が青色に、これらのセルの前景が白色に変更されるように指定します。SUMLABEL オプションは要約行で、GRANDTOTAL_LABEL オプションは総計行で、それぞれ変数名のかわりにラベルを使用します。

```
proc print data=tempemp (obs=10)sumlabel='Total' grandtotal_label='Grand Total'
  style(HEADER)={fontstyle=italic}
  style(DATA)={backgroundcolor=blue foreground=white};

 id jobcode;
  by jobcode;
  var gender salary;
```

赤で書き込まれる合計値を作成します。 STYLE オプションは、合計を含むセルの前景色が赤に変更されるように指定します。

```
  sum salary  / style(total)={color=red};

  label jobcode='Job Code'
        gender='Gender'
        salary='Annual Salary';

  format salary dollar11.2;
```

```
    where jobcode contains 'FA' or jobcode contains 'ME';
    title 'Expenses Incurred for';
    title2 'Salaries for Flight Attendants and Mechanics';
run;
```

出力: スタイル付きの HTML5

アウトプット 40.17　BY グループおよび ID 変数を使用したカスタマイズレイアウトの作成: HTML5 出力スタイルの使用

Expenses Incurred for
Salaries for Flight Attendants and Mechanics

JobCode	Gender	Salary
FA1	F	$23,177.00
	F	$22,454.00
	M	$22,268.00
Total		$67,899.00

JobCode	Gender	Salary
FA2	F	$28,888.00
	F	$27,787.00
	M	$28,572.00
Total		$85,247.00

JobCode	Gender	Salary
FA3	F	$32,886.00
	F	$33,419.00
	M	$32,217.00
Total		$98,522.00

JobCode	Gender	Salary
ME1	M	$29,769.00
Grand Total		$281,437.00

例 10: SAS ライブラリのすべてのデータセットを印刷する

要素: マクロ機能
DATASETS プロシジャ
PRINT プロシジャ

データセット: PROCLIB.DELAY と

PROCLIB.INTERNAT(生データと DATA ステップ付録)

ODS 出力先: HTML5

詳細

この例では、SAS ライブラリのすべてのデータセットを印刷します。同じプログラミング論理をどのプロシジャとも使用できます。例の最後辺りに PROC PRINT ステップと実行するプロシジャステップを置き換えるだけです。例では、マクロ言語を使用します。マクロ言語の詳細については、*SAS マクロ言語: リファレンス*を参照してください。

プログラム: ライブラリのすべてのデータセットを印刷する

```
libname printlib 'SAS-data-library';
libname proclib 'SAS-data-library';
options nodate pageno=1;

proc datasets library=proclib memtype=data nolist;
  copy out=printlib;
    select delay internat;
run;

%macro printall(libname,worklib=work);

  %local num i;

  proc datasets library=&libname memtype=data nodetails;
    contents out=&worklib..temp1(keep=memname) data=_all_ noprint;
  run;

  data _null_;
    set &worklib..temp1 end=final;
    by memname notsorted;
    if last.memname;
    n+1;
    call symput('ds'||left(put(n,8.)),trim(memname));

    if final then call symput('num',put(n,8.));

  run;

  %do i=1 %to &num;
    proc print data=&libname..&&ds&i noobs;
      title "Data Set &libname..&&ds&i";
    run;
  %end;
%mend printall;

%printall(printlib)
```

プログラムの説明

```
libname printlib 'SAS-data-library';
libname proclib 'SAS-data-library';
options nodate pageno=1;
```

必要なデータセットを WORK ライブラリから永久ライブラリにコピーします。 PROC DATASETS は 2 つのデータセットを WORK ライブラリから PRINTLIB ライブラリにコピーし、例に利用可能なデータセット数を制限します。

```
proc datasets library=proclib memtype=data nolist;
  copy out=printlib;
    select delay internat;
run;
```

マクロを作成して、パラメーターを指定します。 %MACRO ステートメントは、マクロ PRINTALL を作成します。マクロの呼び出し時には、1 つまたは 2 つのパラメーターを渡すことができます。最初のパラメーターは、データセットを印刷するライブラリの名前です。2 番目のパラメーターは、マクロによって使用されるライブラリです。このパラメーターを指定しない場合、WORK ライブラリがデフォルトとなります。

```
%macro printall(libname,worklib=work);
```

ローカルマクロ変数を作成します。 %LOCAL ステートメントは、ループで使用する 2 つのローカルマクロ変数、NUM と I を作成します。

```
%local num i;
```

出力データセットを生成します。 この PROC DATASETS ステップは、マクロの起動時にパラメーターとして指定するライブラリを読み込みます。CONTENTS ステートメントは、WORKLIB に TEMP1 という出力データセットを生成します。このデータセットには、ライブラリ LIBNAME の各データセットの変数ごとに 1 つのオブザベーションが含まれています。デフォルトでは、オブザベーションにはそれぞれ、変数が含まれるデータセット名および変数に関する情報が含まれます。ただし、KEEP=データセットオプションは、データセット名のみ TEMP1 に書き込みます。

```
proc datasets library=&libname memtype=data nodetails;
  contents out=&worklib..temp1(keep=memname) data=_all_ noprint;
run;
```

データセットの一意の値を指定し、それぞれにマクロ変数を割り当て、マクロ変数に DATA ステップ情報を割り当てます。 この DATA ステップは、最後のデータセット名を読み込むたびに N の値を増分します。(IF LAST.MEMNAME が True の場合) CALL SYMPUT ステートメントは N の現在の値を使用して、データセット TEMP1 の一意の値 MEMNAME に対してそれぞれマクロ変数を作成します。TRIM 関数は、後に続く PROC PRINT ステップの TITLE ステートメントの余分なブランクを削除します。

```
data _null_;
  set &worklib..temp1 end=final;
  by memname notsorted;
  if last.memname;
  n+1;
  call symput('ds'||left(put(n,8.)),trim(memname));
```

DATA ステップでオブザベーション数を決定します。 データセットの最後のオブザベーションの読み込み時に(FINAL が Ttrue のとき)、DATA ステップは N の値をマク

ロ変数 NUM に割り当てます。プログラムのこの時点で、N の値はデータセットのオブザベーション数です。

```
if final then call symput('num',put(n,8.));
```

DATA ステップを実行します。 RUN ステートメントは重要です。これにより DATA ステップが実行されるため、マクロ変数の作成が実行されます。このマクロ変数は、それらを使用する%DO ループより前に CALL SYMPUT ステートメントで使用されます。

```
run;
```

データセットを印刷し、マクロを終了します。 %DO ループは、データセットごとに PROC PRINT ステップを発行します。%MEND ステートメントはマクロを終了します。

```
%do i=1 %to &num;
  proc print data=&libname..&&ds&i noobs;
    title "Data Set &libname..&&ds&i";
  run;
%end;
%mend printall;
```

PRINTLIB ライブラリのすべてのデータセットを印刷します。 PRINTALL マクロを起動することによって、ライブラリ PRINTLIB のすべてのデータセットが印刷されます。

```
%printall(printlib)
```

出力: HTML5

アウトプット 40.18　データセット PRINTLIB.DELAY

Data Set PRINTLIB.DELAY

flight	date	orig	dest	delaycat	destype	delay
114	01MAR18	LGA	LAX	1-10 Minutes	Domestic	8
202	01MAR18	LGA	ORD	No Delay	Domestic	-5
219	01MAR18	LGA	LON	11+ Minutes	International	18
622	01MAR18	LGA	FRA	No Delay	International	-5
132	01MAR18	LGA	YYZ	11+ Minutes	International	14
271	01MAR18	LGA	PAR	1-10 Minutes	International	5
302	01MAR18	LGA	WAS	No Delay	Domestic	-2
114	02MAR18	LGA	LAX	No Delay	Domestic	0
202	02MAR18	LGA	ORD	1-10 Minutes	Domestic	5
219	02MAR18	LGA	LON	11+ Minutes	International	18
622	02MAR18	LGA	FRA	No Delay	International	0
132	02MAR18	LGA	YYZ	1-10 Minutes	International	5
271	02MAR18	LGA	PAR	1-10 Minutes	International	4
302	02MAR18	LGA	WAS	No Delay	Domestic	0
114	03MAR18	LGA	LAX	No Delay	Domestic	-1
202	03MAR18	LGA	ORD	No Delay	Domestic	-1
219	03MAR18	LGA	LON	1-10 Minutes	International	4
622	03MAR18	LGA	FRA	No Delay	International	-2
132	03MAR18	LGA	YYZ	1-10 Minutes	International	6
271	03MAR18	LGA	PAR	1-10 Minutes	International	2
302	03MAR18	LGA	WAS	1-10 Minutes	Domestic	5
114	04MAR18	LGA	LAX	11+ Minutes	Domestic	15
202	04MAR18	LGA	ORD	No Delay	Domestic	-5
219	04MAR18	LGA	LON	1-10 Minutes	International	3
622	04MAR18	LGA	FRA	11+ Minutes	International	30
132	04MAR18	LGA	YYZ	No Delay	International	-5
271	04MAR18	LGA	PAR	1-10 Minutes	International	5
302	04MAR18	LGA	WAS	1-10 Minutes	Domestic	7
114	05MAR18	LGA	LAX	No Delay	Domestic	-2
202	05MAR18	LGA	ORD	1-10 Minutes	Domestic	2
219	05MAR18	LGA	LON	1-10 Minutes	International	3
622	05MAR18	LGA	FRA	No Delay	International	-6
132	05MAR18	LGA	YYZ	1-10 Minutes	International	3
271	05MAR18	LGA	PAR	1-10 Minutes	International	5
114	06MAR18	LGA	LAX	No Delay	Domestic	-1
202	06MAR18	LGA	ORD	No Delay	Domestic	-3
219	06MAR18	LGA	LON	11+ Minutes	International	27
132	06MAR18	LGA	YYZ	1-10 Minutes	International	7
302	06MAR18	LGA	WAS	1-10 Minutes	Domestic	1
114	07MAR18	LGA	LAX	No Delay	Domestic	-1
202	07MAR18	LGA	ORD	No Delay	Domestic	-2
219	07MAR18	LGA	LON	11+ Minutes	International	15
622	07MAR18	LGA	FRA	11+ Minutes	International	21
132	07MAR18	LGA	YYZ	No Delay	International	-2
271	07MAR18	LGA	PAR	1-10 Minutes	International	4
302	07MAR18	LGA	WAS	No Delay	Domestic	0

アウトプット 40.19　データセット PRINTLIB.INTERNAT

Data Set PRINTLIB.INTERNAT

flight	date	dest	boarded
219	01MAR18	LON	198
622	01MAR18	FRA	207
132	01MAR18	YYZ	115
271	01MAR18	PAR	138
219	02MAR18	LON	147
622	02MAR18	FRA	176
132	02MAR18	YYZ	106
271	02MAR18	PAR	172
219	03MAR18	LON	197
622	03MAR18	FRA	180
132	03MAR18	YYZ	75
271	03MAR18	PAR	147
219	04MAR18	LON	232
622	04MAR18	FRA	137
132	04MAR18	YYZ	117
271	04MAR18	PAR	146
219	05MAR18	LON	160
622	05MAR18	FRA	185
132	05MAR18	YYZ	157
271	05MAR18	PAR	177
219	06MAR18	LON	163
132	06MAR18	YYZ	150
219	07MAR18	LON	241
622	07MAR18	FRA	210
132	07MAR18	YYZ	164
271	07MAR18	PAR	155

41

PRINTTO プロシジャ

概要: PRINTTO プロシジャ

PRINTTO プロシジャの機能

PRINTTO プロシジャでは、SAS プロシジャ出力と SAS ログの出力先(ODS 出力先以外)が定義されます。デフォルトでは、SAS プロシジャ出力と SAS ログは、操作方法に応じたデフォルトプロシジャ出力ファイルとデフォルト SAS ログファイルに送られます。PRINTTO プロシジャでは、ODS 出力先は定義されません。SAS ログとプロシジャ出力のデフォルト出力先については、次のテーブルを参照してください。

SAS ログやプロシジャ出力は、外部ファイルや SAS カタログエントリに保存できます。ファイルやカタログエントリに SAS 出力を書き込むには、ODS LISTING 出力先

を開く必要があります。追加プログラミングによって、同じジョブ内で SAS 出力を入力データとして使用できます。

表 41.1 SAS ログとプロシジャ出力のデフォルト出力先

SAS 実行方法	SAS ログのデフォルト出力先	プロシジャ出力のデフォルト出力先
ウィンドウ環境	**ログ**	**結果**
非対話型モードまたはバッチモード	ホスト OS に依存	動作環境に依存

構文: PRINTTO プロシジャ

PROC PRINTTO <*options*>;

ステートメント	タスク	例
"PROC PRINTTO ステートメント"	SAS プロシジャ出力と SAS ログの出力先(ODS 出力先以外)が定義されます。	Ex. 1, Ex. 2, Ex. 3

PROC PRINTTO ステートメント

SAS プロシジャ出力と SAS ログの出力先(ODS 出力先以外)が定義されます。

制限事項:

SAS ログとプロシジャ出力をプリンターに直接送るには、FILENAME ステートメントを PROC PRINTTO ステートメントとあわせて使用する必要があります。"SAS ログまたはプロシジャの出力を直接にプリンターに送る" (1493 ページ)を参照してください。"例 4: プリンターに送る" (1506 ページ)。

PRINTTO プロシジャでは、ODS 出力先は定義されません。

SAS がオブジェクトサーバーモードで起動されている場合、PRINTTO プロシジャはログメッセージを ALTLOG=システムオプションで指定したログに送りません。

注:

LOG=LOG および PRINT=PRINT は、ログおよびプロシジャの出力をデフォルト出力先に送ります。ただし、LOG=PRINT または PRINT=LOG を指定して、ログまたはプロシジャの出力を同じデフォルト出力先に送ることは無効です。

ヒント:

SAS ログとプロシジャ出力の出力先をデフォルトにリセットするには、PROC PRINTTO ステートメントをオプションなしで使用します。

SAS ログとプロシジャ出力を同じファイルに送るには、LOG=と PRINT=の両オプションで同じファイルを指定します。

例:
“例 1: 外部ファイルに送る” (1495 ページ)
“例 2: SAS カタログエントリに送る” (1499 ページ)
“例 3: プロシジャ出力を入力ファイルとして使用する” (1502 ページ)
“例 4: プリンターに送る” (1506 ページ)

構文

PROC PRINTTO <LABEL='*description*'><LOG= ログ | *file-specification* | *SAS-catalog-entry*><NEW ><PRINT= PRINT | *file-specification* | *SAS-catalog-entry*><UNIT=*nn*>;

引数なし

オプションを指定しなかった場合、PROC PRINTTO ステートメントでは次が実行されます。

- PROC PRINTTO ステートメントで開いたファイルをすべて閉じます。
- SAS ログと SAS プロシジャ出力の両方にデフォルト出力先を示します。
- LISTING 出力先を閉じます。

操作: 適切なファイルを閉じて、SAS ログまたはプロシジャ出力のみをデフォルト出力先に戻すには、LOG=LOG または PRINT=PRINT を使用します。

例:
“例 1: 外部ファイルに送る” (1495 ページ)
“例 2: SAS カタログエントリに送る” (1499 ページ)

オプション引数

LABEL='*description*'
SAS ログまたはプロシジャ出力を含むカタログエントリの説明が提供されます。

範囲 1–256 文字

操作 カタログエントリを LOG=または PRINT=オプションの値として指定する場合に限り、LABEL=オプションを使用します。

例 “例 2: SAS カタログエントリに送る” (1499 ページ)

LOG=LOG | *file-specification* | *SAS-catalog-entry*
SAS ログを 3 つのうちいずれかの場所に送ります。

LOG
SAS ログをデフォルト出力先に送ります。

file-specification
SAS ログを外部ファイルに送ります。*file-specification* は、次のいずれかになります。

'***external-file***'
引用符で囲んで指定する外部ファイル名。

制限事項 *external-file* は 1024 文字以内にしてください。

log-filename
引用符なしの英数字のテキスト文字列です。SAS で、*log-filename*.log をログファイル名として使用するログが作成されます。

fileref
事前に外部ファイルに割り当てられたファイル参照名。

SAS-catalog-entry
SAS ログを SAS カタログエントリに送ります。次の方法で *SAS-catalog-entry* を示します。

***libref.catalog.entry*<.LOG>**
指定された SAS ライブラリおよび SAS カタログに格納された SAS カタログエントリ。

fileref
SAS カタログエントリに事前に割り当てられたファイル参照名。
"FILENAME ステートメント: CATALOG アクセスメソッド" (*SAS グローバルステートメント: リファレンス*)を参照してください。

デフォルト LOG

操作 SAS ログとプロシジャ出力を、同時に同じカタログエントリに送ることはできません。

NEW オプションで、ファイルの既存コンテンツが新しいログに置き換えられます。そうでなければ、新しいログはファイルに追加されます。

SAS ログとプロシジャ出力を同じファイルに送るには、LOG=と PRINT=の両オプションで同じファイルを指定します。

ログを SAS カタログエントリに送る際、LABEL オプションを使用すると、カタログディレクトリのエントリの説明を提供できます。

ログがデフォルトログファイル以外のファイルに送られ、複数のソースからプログラムがサブミットされると、実時間と CPU 時間を含む最後の SAS システムメッセージがデフォルト SAS ログに書き込まれます。

ヒント 外部ファイルまたはカタログエントリにログを送った後で、SAS ログをデフォルト出力先に送り直すには、LOG を指定します。

SAS ログを送る際は、PROC PRINTTO ステートメントに RUN ステートメントを含めます。RUN ステートメントを省略した場合、その後に続く DATA または PROC ステップの最初の行は新しいファイルに送られません。(このような事象が発生するのは、ステップの境界を越えるまではステートメントが実行されないためです。

パスワードを含むマクロを作成して、パスワードを SAS ログに表示したくない場合、LOG=*file-specification* オプションを使用してログを外部ファイルにリダイレクトします。

LOG=を指定すると、SAS では&SYSPRINTTOLOG 自動マクロ変数に SAS ログファイルのパスが保存されます。このマクロ変数を使用して前の SAS ログファイルの場所を復元できます。詳細については、“前の SAS ログまたは LISTING 出力ファイルの場所の復元” (1494 ページ)を参照してください。

例 “例 1: 外部ファイルに送る” (1495 ページ)

“例 2: SAS カタログエントリに送る” (1499 ページ)

“例 3: プロシジャ出力を入力ファイルとして使用する” (1502 ページ)

NEW
ファイルに存在する情報をすべてクリアし、ファイルが SAS ログまたはプロシジャ出力を受信する準備をします。

デフォルト NEW を省略した場合、新しい情報は既存のファイルに追加されます。

操作 LOG=と PRINT=の両方を指定した場合、両方に NEW が適用されます。

例 “例 1: 外部ファイルに送る” (1495 ページ)

“例 2: SAS カタログエントリに送る” (1499 ページ)

“例 3: プロシジャ出力を入力ファイルとして使用する” (1502 ページ)

PRINT= PRINT | *file-specification* | *SAS-catalog-entry*
プロシジャ出力を 3 つのうちいずれかの場所に送ります。

PRINT
プロシジャ出力をデフォルト出力先に送ります。

ヒント 外部ファイルまたはカタログエントリに送った後で、後続のプロシジャ出力をデフォルト出力先に送るには、PRINT を指定します。

file-specification
プロシジャの出力を外部ファイルに送ります。*file-specification* は、次のいずれかになります。

'***external-file***'
引用符で囲んで指定する外部ファイル名。

制限事項 *external-file* は 1024 文字以内にしてください。

print-filename
引用符なしの英数字のテキスト文字列です。SAS で、*print-filename* を印刷ファイル名として使用する印刷ファイルが作成されます。

fileref
事前に外部ファイルに割り当てられたファイル参照名。

SAS-catalog-entry
プロシジャ出力を SAS カタログエントリに送ります。次の方法で *SAS-catalog-entry* を示します。

***libref.catalog.entry*<.OUTPUT>**
指定された SAS ライブラリおよび SAS カタログに格納された SAS カタログエントリ。

fileref
SAS カタログエントリに事前に割り当てられたファイル参照名。
“FILENAME ステートメント: CATALOG アクセスメソッド” (*SAS グローバルステートメント: リファレンス*)を参照してください。

別名
FILE=

NAME=

デフォルト
PRINT

操作
PRINT または FILE=または NAME=を指定して、LISTING 出力先が開いていないときは、PRINTTO プロシジャ出力の送信中に、このプロシジャにより LISTING 出力先が開きます。PRINT または FILE=または NAME=が指定される前に LISTING 出力先が開いた場合は、出力が LISTING 出力先に送られた後も、LISTING 出力先が開いたままになります。

プロシジャ出力と SAS ログを、同時に同じカタログエントリに送ることはできません。

NEW オプションで、ファイルの既存コンテンツが新しいプロシジャ出力に置き換えられます。NEW を省略した場合、新しい出力は既存のファイルに追加されます。

SAS ログとプロシジャ出力を同じファイルに送るには、LOG=と PRINT=の両オプションで同じファイルを指定します。

プロシジャ出力を SAS カタログエントリに送る際、LABEL オプションを使用すると、カタログディレクトリのエントリの説明を提供できます。

ヒント
PRINT=を指定すると、SAS では&SYSPRINTTOLIST 自動マクロ変数に LISTING 出力ファイルのパスが保存されます。このマクロ変数を使用して前の LISTING 出力ファイルの場所を復元できます。詳細については、“前の SAS ログまたは LISTING 出力ファイルの場所の復元” (1494 ページ)を参照してください。

例
“例 3: プロシジャ出力を入力ファイルとして使用する” (1502 ページ)

UNIT=*nn*
ファイル参照名 FT*nn*F001 で識別されるファイルに出力を送ります。このとき、*nn* は 1 から 99 までの整数です。

範囲 1–99、整数のみ

ヒント このファイル参照名は各自で定義できます。ただし、一部の OS では、このフォームで特定のファイル参照名が事前定義されます。

UNIT=を指定すると、SAS では&SYSPRINTTOLIST 自動マクロ変数に LISTING 出力ファイルのパスが保存されます。このマクロ変数を使用して前の LISTING 出力ファイルの場所を復元できます。詳細については、"前の SAS ログまたは LISTING 出力ファイルの場所の復元" (1494 ページ)を参照してください。

使用法: PRINTTO プロシジャ

SAS システムオプションを使用してページ番号を設定する

NUMBER SAS システムオプションが有効な場合、現在のジョブまたはセッションのすべての出力に対する単一のページ採番が行われます。NONUMBER が有効な場合、出力ページは採番されません。

OPTIONS ステートメントの PAGENO=を使用すると、現在作成している出力の開始ページ番号を指定できます。

SAS ログまたはプロシジャの出力を直接にプリンターに送る

SAS ログまたはプロシジャ出力を直接プリンターに送るには、FILENAME ステートメントでファイル参照名とプリンター名を関連付けて、そのファイル参照名を LOG=または PRINT=オプションで使用します。例については、"例 4: プリンターに送る" (1506 ページ)を参照してください。

詳細については、"FILENAME ステートメント" (*SAS グローバルステートメント: リファレンス*)を参照してください。

PRINTTO プロシジャでは、COLORPRINTING システムオプションはサポートされません。SAS ログまたはプロシジャ出力をカラープリンターに送っても、出力はカラー印刷されません。

前の SAS ログまたは LISTING 出力ファイルの場所の復元

PROC PRINTTO ステートメントで LOG=、PRINT=または UNIT=オプションを指定すると、SAS では自動マクロ変数に該当するファイルの場所が次のように保存されます。

SYSPRINTTOLOG	PRINTTO プロシジャによるリダイレクション前の SAS ログファイルの場所のパスを含む
SYSPRINTTOLIST	PRINTTO プロシジャによるリダイレクション前の出力ファイルの場所のパスを含む

前のファイルの場所を復元するには、LOG=、PRINT=または UNIT=オプションの値として適切な自動マクロ変数を指定します。次に、いくつかの例を示します。

```
/* Restore the previous log and the listing file locations. */
proc printto log=&sysprinttolog print=&sysprinttolist;
  run;

/* Restore the previous listing file location. */
proc printto unit=&sysprinttolist;
  run;
```

プログラム出力を別の場所に送信する

次のステートメントは、RUN ステートメントの後に生成される SAS ログエントリを、ファイル参照名 myfile に関連付けられた外部ファイルに送信します。

```
filename myfile '/users/myid/mydir/mylog';
proc printto log=myfile;
run;
```

myfile がファイル参照名として定義されていない場合、PROC PRINTTO はファイル myfile.log を現在のディレクトリに作成します。

次のステートメントは、RUN ステートメントの後に生成されるプロシジャ出力をファイル**/users/myid/mydir/myout** に送信します。

```
proc printto print='/users/myid/mydir/myout';
run;
```

次のステートメントは、CONTENTS プロシジャからのプロシジャ出力をシステムプリンターに直接送信します。

```
filename myfile printer;
proc printto print=myfile;
run;
proc contents data=oranges;
```

```
run;
```

SAS ログとプロシジャ出力を元のデフォルトの出力先にリダイレクトするには、オプションを指定せずに PROC PRINTTO を実行します。

```
proc printto;
run;
```

ファイル参照 myprint および mylog が定義されていない場合、次のステートメントは SAS プロシジャ出力を myprint.lst に送信し、すべてのログ出力を現在のディレクトリ内の mylog.log に送信します。

```
proc printto print=myprint log=mylog;
run;
```

ファイル参照 myprint および mylog が定義されている場合、出力はそれらのファイル参照に関連付けられているファイルに送られます。

例: PRINTTO プロシジャ

例 1: 外部ファイルに送る

要素:

オプションなしの PRINTTO ステートメント

PRINTTO ステートメントオプション

- LOG=
- NEW
- PRINT=

詳細

この例では、PROC PRINTTO を使用してログとプロシジャ出力を外部ファイルに送ってから、出力先を両方ともデフォルトにリセットします。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60 source;

proc printto log='log-file';
  run;

data numbers;
  input x y z;
```

```
  datalines;
 14.2   25.2   96.8
 10.8   51.6   96.8
  9.5   34.2  138.2
  8.8   27.6   83.2
 11.5   49.4  287.0
  6.3   42.0  170.7
;

proc printto print='output-file'
new;
run;

proc print data=numbers;
  title 'Listing of NUMBERS Data Set';
run;

proc printto;
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。SOURCE オプションはソースコード行を SAS ログのデフォルト出力先に書き込みます。

```
options nodate pageno=1 linesize=80 pagesize=60 source;
```

SAS ログを外部ファイルに送ります。 PROC PRINTTO では、LOG=オプションによって SAS ログが外部ファイルに送られます。デフォルトでは、このログは *log-file* の現在のコンテンツに追加されます。

```
proc printto log='log-file';
  run;
```

NUMBERS データセットを作成します。 DATA ステップでは、リスト入力によって NUMBERS データセットが作成されます。

```
data numbers;
  input x y z;
  datalines;
 14.2   25.2   96.8
 10.8   51.6   96.8
  9.5   34.2  138.2
  8.8   27.6   83.2
 11.5   49.4  287.0
  6.3   42.0  170.7
;
```

プロシジャ出力を外部ファイルに送ります。 PROC PRINTTO で、出力が外部ファイルに送られます。SAS 出力を外部ファイルに送るために LISTING 出力先を開く必要があるため、LISTING 出力先が開かれていなければ SAS により開かれます。ODS LISTING ステートメントを含める必要はありません。NEW を指定しているため、

output-file に出力が書き込まれると、ファイルの現在のコンテンツが上書きされます。

```
proc printto print='output-file'
new;
run;
```

NUMBERS データセットを印刷します。 PROC PRINT 出力が、指定した外部ファイルに書き込まれます。

```
proc print data=numbers;
  title 'Listing of NUMBERS Data Set';
run;
```

SAS ログとプロシジャ出力の出力先をデフォルトにリセットします。 PROC PRINTTO で、後続のログとプロシジャ出力がデフォルト出力先に送られ、現在のファイルが両方とも閉じられます。

```
proc printto;
run;
```

ログ

例のコード 41.1　デフォルト出力先に送られたログの一部

```
01   options nodate pageno=1 linesize=80 pagesize=60 source;
02
03   proc printto log='file-path/em/log1.log';
04   run;

NOTE: PROCEDURE PRINTTO used (Total process time):
      real time          0.01 seconds
      cpu time           0.00 seconds
```

例のコード 41.2　外部ファイルに送られたログの一部

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


5
6    data numbers;
7      input x y z;
8      datalines;

NOTE: The data set WORK.NUMBERS has 6 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


15   ;
16
17   proc printto print='print1.out' new;
18   run;

NOTE: Writing HTML Body file: sashtml.htm
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           4.04 seconds
      cpu time            0.62 seconds


19
20   proc print data=numbers;
21     title 'Listing of NUMBERS Data Set';
22   run;

NOTE: There were 6 observations read from the data set WORK.NUMBERS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.56 seconds
      cpu time            0.09 seconds


23
24   proc printto;
25   run;
```

出力

アウトプット 41.1　外部ファイルに送られたプロシジャ出力

Listing of NUMBERS Data Set

Obs	x	y	z
1	14.2	25.2	96.8
2	10.8	51.6	96.8
3	9.5	34.2	138.2
4	8.8	27.6	83.2
5	11.5	49.4	287.0
6	6.3	42.0	170.7

例 2: SAS カタログエントリに送る

要素: オプションなしの PRINTTO ステートメント
PRINTTO ステートメントオプション
LABEL=
LOG=
NEW
PRINT=

詳細

この例では、PROC PRINTTO を使用して SAS ログとプロシジャ出力を SAS カタログエントリに送ってから、出力先を両方ともデフォルトにリセットします。

プログラム

```
options source;

libname lib1 'SAS-library';

proc printto log=lib1.cat1.test.log label='Inventory program' new;
run;

data lib1.inventory;
  length Dept $ 4 Item $ 6 Season $ 6 Year 4;
  input dept item season year @@;
  datalines;
3070 20410  spring 2011 3070 20411  spring 2012
3070 20412  spring 2012 3070 20413  spring 2012
3070 20414  spring 2011 3070 20416  spring 2009
3071 20500  spring 2011 3071 20501  spring 2009
3071 20502  spring 2011 3071 20503  spring 2011
3071 20505  spring 2010 3071 20506  spring 2009
3071 20507  spring 2009 3071 20424  spring 2011
;

proc printto print=lib1.cat1.inventory.output
          label='Inventory program' new;
run;

proc report data=lib1.inventory nowindows headskip;
  column dept item season year;
  title 'Current Inventory Listing';
run;

proc printto;
run;
```

プログラムの説明

SAS システムオプションを設定します。 SOURCE オプションで、ソースステートメントを SAS ログに書き込むことが指定されます。

```
options source;
```

ライブラリ参照名を割り当てます。

```
libname lib1 'SAS-library';
```

SAS ログを SAS カタログエントリに送ります。 PROC PRINTTO は、SAS ログを LIB1.CAT1 カタログのカタログエントリ TEST.LOG に送ります。LABEL=で、カタログエントリの説明が割り当てられます。

```
proc printto log=lib1.cat1.test.log label='Inventory program' new;
run;
```

LIB1.INVENTORY データセットを作成します。 DATA ステップでは、永久 SAS データセットが作成されます。

```
data lib1.inventory;
  length Dept $ 4 Item $ 6 Season $ 6 Year 4;
  input dept item season year @@;
  datalines;
3070 20410  spring 2011 3070 20411  spring 2012
3070 20412  spring 2012 3070 20413  spring 2012
3070 20414  spring 2011 3070 20416  spring 2009
3071 20500  spring 2011 3071 20501  spring 2009
3071 20502  spring 2011 3071 20503  spring 2011
3071 20505  spring 2010 3071 20506  spring 2009
3071 20507  spring 2009 3071 20424  spring 2011
;
```

プロシジャ出力を SAS カタログエントリに送ります。 後続の PROC REPORT ステップから SAS カタログエントリ LIB1.CAT1.INVENTORY.OUTPUT まで、プロシジャ出力を送るために、PROC PRINTTO ルートにより LISTING 出力先が開かれます。LABEL=で、カタログエントリの説明が割り当てられます。プロシジャ出力が SAS カタログに送られた後、PROC PRINTTO により LISTING 出力先が閉じられます。

```
proc printto print=lib1.cat1.inventory.output
        label='Inventory program' new;
run;

proc report data=lib1.inventory nowindows headskip;
  column dept item season year;
  title 'Current Inventory Listing';
run;
```

SAS ログとプロシジャ出力をリセットしてデフォルトに戻し、ファイルを閉じます。 PROC PRINTTO によって、前の PROC PRINTTO ステップで開かれた現在のファイルが閉じられ、後続の SAS ログとプロシジャ出力がデフォルト出力先に送り直されます。

```
proc printto;
run;
```

ログ

例のコード 41.3　SAS カタログエントリ lib1.cat1.TEST.LOG に送られた SAS ログ

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


49
50   data lib1.inventry;
51      length Dept $ 4 Item $ 6 Season $ 6 Year 4;
52      input dept item season year @@;
53      datalines;

NOTE: SAS went to a new line when INPUT statement reached past the end of a
      line.
NOTE: The data set LIB1.INVENTORY has 14 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.03 seconds
      cpu time            0.00 seconds


61   ;
62   ods listing;
63   proc printto print=lib1.cat1.inventory.output
64                label='Inventory program' new;
65   run;

NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


66
67   proc report data=lib1.inventory nowindows headskip;
68      column dept item season year;
69      title 'Current Inventory Listing';
70   run;

NOTE: There were 14 observations read from the data set LIB1.INVENTRY.
NOTE: PROCEDURE REPORT used (Total process time):
      real time           0.09 seconds
      cpu time            0.04 seconds


71
72   proc printto;
73   run;
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


74   ods listing close;
```

出力

アウトプット 41.2　SAS カタログエントリ LIB1.CAT1.INVENTORY.OUTPUT に送られたプロシジャ出力

Current Inventory Listing

Dept	Item	Season	Year
3070	20410	spring	2011
3070	20411	spring	2012
3070	20412	spring	2012
3070	20413	spring	2012
3070	20414	spring	2011
3070	20416	spring	2009
3071	20500	spring	2011
3071	20501	spring	2009
3071	20502	spring	2011
3071	20503	spring	2011
3071	20505	spring	2010
3071	20506	spring	2009
3071	20507	spring	2009
3071	20424	spring	2011

例 3: プロシジャ出力を入力ファイルとして使用する

要素:

オプションなしの PRINTTO ステートメント

PRINTTO ステートメントオプション

- LOG=
- NEW
- PRINT=

詳細

この例では、PROC PRINTTO を使用してプロシジャ出力を外部ファイルに送り、そのファイルを DATA ステップへの入力として使用します。

プログラム

```
data test;
```

```
  do n=1 to 1000;
    x=int(ranuni(77777)*7);
    y=int(ranuni(77777)*5);
    output;
  end;
run;

filename routed 'output-filename';

proc printto print=routed new;
  run;

  proc freq data=test;
    tables x*y / chisq;
  run;

  proc printto print=print;
  run;

  data probtest;
    infile routed;
    input word1 $ @;
    if word1='Chi-Squa' then
      do;
        input df chisq prob;
        keep chisq prob;
        output;
      end;
  run;

  proc print data=probtest;
    title 'Chi-Square Analysis for Table of X by Y';
  run;
```

プログラムの説明

変数に対してランダム値を生成します。 DATA ステップで、RANUNI 関数を使用して、データセット A の変数 X と Y に対する値がランダム生成されます。

```
data test;
  do n=1 to 1000;
    x=int(ranuni(77777)*7);
    y=int(ranuni(77777)*5);
    output;
  end;
run;
```

ファイル参照名を割り当てて、プロシジャ出力を参照先ファイルに送ります。 FILENAME ステートメントでは、ファイル参照名が外部ファイルに割り当てられます。PROC PRINTTO で、後続のプロシジャ出力が、ファイル参照名 ROUTED で参照されるファイルに送られます。このプロシジャオプションを送っている間に、PROC PRINTTO により LISTING 出力先が開きます。後述の"参照名 ROUTED の外部ファイルに送られた PROC FREQ 出力"を参照してください。

```
filename routed 'output-filename';

proc printto print=routed new;
  run;
```

度数を作成します。 PROC FREQ で、データセット TEST の変数 X と Y の度数とカイ 2 乗分析が計算されます。この出力は ROUTED として参照されるファイルに送られます。

```
proc freq data=test;
  tables x*y / chisq;
run;
```

ファイルを閉じます。 次の DATA ステップでファイルの読み取りを可能にするために、もう 1 度 PROC PRINTTO を使用して、ファイル参照名 ROUTED で参照されるファイルを閉じる必要があります。また、このステップでは、後続のプロシジャ出力がデフォルト出力先に送られます。PRINT=を使用すると、ステップはプロシジャ出力にのみ影響し、SAS ログには影響しません。

```
proc printto print=print;
run;
```

データセット PROBTEST を作成します。 DATA ステップでは、ROUTED(PROC FREQ 出力を含むファイル)が入力ファイルとして使用され、データセット PROBTEST が作成されます。DATA ステップでは、ROUTED のレコードがすべて読み取られますが、**Chi-Squa** で始まるレコードからのみオブザベーションが作成されます。

```
data probtest;
  infile routed;
  input word1 $ @;
  if word1='Chi-Squa' then
    do;
      input df chisq prob;
      keep chisq prob;
      output;
    end;
run;
```

PROBTEST データセットを印刷します。 PROC PRINT で、データセット PROBTEST の簡単なリストが作成されます。この出力がデフォルト出力先に送られます。出力セクションの"デフォルト出力先に送られるデータセット PROBTEST の PROC PRINT 出力"を参照してください。

```
proc print data=probtest;
  title 'Chi-Square Analysis for Table of X by Y';
run;
```

出力

アウトプット 41.3　参照名 ROUTED の外部ファイルに送られた PROC FREQ 出力

The FREQ Procedure

Frequency
Percent
Row Pct
Col Pct

Table of x by y

x	y 0	1	2	3	4	Total
0	29 2.90 23.02 15.18	33 3.30 26.19 16.18	12 1.20 9.52 6.25	25 2.50 19.84 11.74	27 2.70 21.43 13.50	126 12.60
1	23 2.30 19.66 12.04	26 2.60 22.22 12.75	29 2.90 24.79 15.10	20 2.00 17.09 9.39	19 1.90 16.24 9.50	117 11.70
2	28 2.80 19.86 14.66	26 2.60 18.44 12.75	32 3.20 22.70 16.67	30 3.00 21.28 14.08	25 2.50 17.73 12.50	141 14.10
3	26 2.60 15.95 13.61	24 2.40 14.72 11.76	36 3.60 22.09 18.75	32 3.20 19.63 15.02	45 4.50 27.61 22.50	163 16.30
4	25 2.50 16.78 13.09	31 3.10 20.81 15.20	28 2.80 18.79 14.58	36 3.60 24.16 16.90	29 2.90 19.46 14.50	149 14.90
5	32 3.20 21.77 16.75	29 2.90 19.73 14.22	26 2.60 17.69 13.54	33 3.30 22.45 15.49	27 2.70 18.37 13.50	147 14.70
6	28 2.80 17.83 14.66	35 3.50 22.29 17.16	29 2.90 18.47 15.10	37 3.70 23.57 17.37	28 2.80 17.83 14.00	157 15.70
Total	191 19.10	204 20.40	192 19.20	213 21.30	200 20.00	1000 100.00

Statistics for Table of x by y

Statistic	DF	Value	Prob
Chi-Square	24	27.2971	0.2908
Likelihood Ratio Chi-Square	24	28.1830	0.2524
Mantel-Haenszel Chi-Square	1	0.6149	0.4330
Phi Coefficient		0.1652	
Contingency Coefficient		0.1630	
Cramer's V		0.0826	

Sample Size = 1000

アウトプット 41.4　デフォルト出力先に送られるデータセット PROBTEST の PROC PRINT 出力

Chi-Square Analysis for Table of X by Y

Obs	chisq	prob
1	27.2971	0.2908

例 4: プリンターに送る

要素:　PRINTTO ステートメントオプション
　PRINT=

詳細

この例では、PROC PRINTTO によって、プロシジャ出力が直接プリンターに送られます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

filename your_fileref printer
'printer-name';

proc printto print=your_fileref;
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=60;
```

ファイル参照名をプリンター名に関連付けます。 FILENAME ステートメントで、ファイル参照名が、指定したプリンター名に関連付けられます。ファイル参照名をデフォルトプリンターに関連付ける場合は、'*printer-name*'を省略します。

```
filename your_fileref printer
'printer-name';
```

プリンターに送るファイルを指定します。 PRINT=オプションで、PROC PRINTTO によってプリンターに送られるファイルが指定されます。

```
proc printto print=your_fileref;
run;
```

42

PROTO プロシジャ

概要: PROTO プロシジャ

PROTO プロシジャの機能

PROTO プロシジャを使用すると、C または C++プログラミング言語で記述される外部関数をバッチモードで登録できます。これらの関数は、SAS、C-language 構造および種類で使用できます。これらの C 言語関数は PROC PROTO に登録された後、FCMP プロシジャで宣言される SAS 関数またはサブルーチンから呼び出すことができます。

概念: PROTO プロシジャ

関数プロトタイプの登録

関数プロトタイプは PROTO プロシジャで登録(宣言)されます。詳細については、“FUNCTION-PROTOTYPE-N ステートメント” (1522 ページ)を参照してください。

サポート対象の C 言語の戻り値の種類

PROTO プロシジャでは、次の C 言語の戻り値の種類がサポートされています。

表 42.1　サポート対象の C 言語の戻り値の種類

関数プロトタイプ	SAS 変数の種類	C 変数の種類
short	数値	short, short *, short **
short *	numeric, array	short, short *, short **
int	数値	int, int *, int **

関数プロトタイプ	SAS 変数の種類	C 変数の種類
int *	numeric, array	int, int *, int **
long	数値	long, long *, long **
long *	numeric, array	long, long *, long **
double	数値	double, double *, double **
double *	numeric, array	double, double *, double **
char *	文字	char *, char **
struct *	struct	struct *, struct **
void		void

サポート対象の C 言語の引数の種類

PROTO プロシジャでは、次の C 言語の引数の種類がサポートされています。

表 42.2　サポート対象の C 言語の引数の種類

関数プロトタイプ	SAS 変数の種類	C 変数の種類
short	数値	short, short *, short **
short *	numeric, array	short, short *, short **
short **	array	short *, short **
int	数値	int, int *, int **
int *	numeric, array	int, int *, int **
int **	array	int *, int **
long	数値	long, long *, long **
long *	numeric, array	long, long *, long **
long **	array	long *, long **

関数プロトタイプ	SAS 変数の種類	C 変数の種類
double	数値	double, double *, double **
double *	numeric, array	double, double *, double **
double **	array	double *, double **
char *	文字	char *, char **
char **	文字	char *, char **
struct *	structure	struct *, struct **
struct **	structure	struct *, struct **

SAS の C 言語構造

基本概念

多くの C 言語ライブラリには、引数として構造体ポインターを有する関数が含まれています。SAS では、PROC PROTO でのみ構造を定義できます。定義後は、PROC PROTO に対応した多くのプロシジャ(PROC COMPILE など)内で、宣言、インスタンス化ができます。

C 構造体は、連続するメモリに適用されるテンプレートです。テンプレートの各エントリでは、名前と種類が指定されます。各要素の種類によって、各エントリに関連付けられるバイト数、および各エントリの使用法が決定されます。調整ルールとベースの種類のサイズがさまざまなため、SAS では、構造体のメモリの各エントリの場所決定は、現在のマシンコンパイラに依存しています。

SAS での構造体の宣言と参照

SAS における構造体宣言の構文は、C 非ポインター構造体宣言と同じです。構造宣言は、次の形式になります。

struct *structure_name structure_instance*;

各構造体は、宣言時にゼロ値に設定されます。構造体では、前のパスから、データによって次のパスが開始されるまで、値が保持されます。

構造要素は、C の静的ピリオド(.)表記を使用して参照されます。SAS にはポインター構文がありません。ある構造体が別の構造体を指す場合、指されている構造体を参

照する唯一の方法は、同じ型の宣言された構造体にポインターを割り当てることです。要素にアクセスするには、その宣言された構造体を使用します。

構造エントリが short、int または long 型で、なおかつ式で参照されている場合は、まず double 型にキャストしてから計算に使用します。構造エントリがベースの種類へのポインターの場合は、ポインターが間接参照され、値が返されます。ポインターが null の場合は、欠損値が返されます。変換に失敗した場合や、欠損値が二重構造以外のエンティティに割り当てられた場合は、PROC PROTO コードで行われた欠損値の割り当てが使用されます。

SAS で配列の長さがわかっている必要があります。これは、構造でディメンションが宣言されている限り、構造の配列エントリを SAS の配列と同様に使用できるようにするためです。この要件には、short、int および long 型の配列も含まれます。エントリが実際に double 型の配列へのポインターである場合は、SAS 配列にそのポインターを割り当てると、配列要素にアクセスできます。配列構文を使用すると、その他の型の配列へのポインターにはアクセスできません。

構造体の例

```
proc proto package =
work.mylib.struct
  label = "package of structures";

  #define MAX_IN 20;

  typedef char * ptr;
  struct foo {
    double hi;
    int mid;
    ptr buf1;
    long * low;
    struct {
      short ans[MAX_IN + 1];
      struct {    /* inner */
        int inner;
      } n2;
      short outer;
    } n;
  };

  typedef struct foo *str;

  struct foo2 {
    str tom;
  };

  str get_record(char *name, int userid);
run;

proc fcmp library = work.mylib;
  struct foo result;
  result = get_record("Mary", 32);
  put result=;
run;
```

SAS での列挙

列挙は、整数のニーモニックです。列挙によって、リテラルな名前を特定の数字として設定できるので、C プラグラムの読みやすさやサポートしやすさが促進されます。列挙は、C 言語ライブラリで、リターンコードを簡略化するために使用されます。C プログラムをコンパイルした後は、列挙名にアクセスできなくなります。

列挙の種類の例

次の例では、PROC PROTO で YesNoMaybeType と Tens という 2 つの列挙値の種類を設定する方法を示します。両方とも構造 ESTRUCTURE で参照されます。

```
proc proto package=work.mylib.str2
  label="package of structures";

  #define E_ROW 52;
  #define L_ROW 124;
  #define S_ROW 15;

  typedef double ExerciseArray[S_ROW][2];
  typedef double LadderArray[L_ROW];
  typedef double SamplingArray[S_Row];

  typedef enum
  {
    True, False, Maybe
  }  YesNoMaybeType;

  typedef enum {
    Ten=10, Twenty=20, Thirty=30, Forty=40, Fifty=50
    } Tens;

  typedef struct {
    short          rows;
    short          cols;
    YesNoMaybeType  type;
    Tens           dollar;
    ExerciseArray   dates;
   } EStructure;
run;
```

次の PROC FCMP の例では、これらの列挙の種類にアクセスする方法を示します。この例では、PROC PROTO で設定される列挙値が、SAS でマクロ変数として実装されます。したがって、&記号を使用してアクセスする必要があります。

```
proc fcmp library=work.mylib;
  struct EStructure mystruct;

  mystruct.type=&True;
  mystruct.dollar=&Twenty;
run;
```

SAS の C ソースコード

外部 C 関数をコンパイルするには、PROC PROTO を限定的な方法で使用します。次のように、PROC PROTO でソースコードを指定できます。

```
C-function-prototype;
externc function-name;
  ... C-source-statements ...
externcend;
```

関数名によって、PROC PROTO で、EXTERNC ステートメントと EXTERNCEND ステートメントの間にどの関数のソースコードが指定されているかがわかります。PROC PROTO でソースコードをコンパイルする際には、現在宣言されているすべての構造定義と C 関数プロトタイプが含まれます。ただし、typedef と#define は含まれません。

この機能は、先に存在する外部 C ライブラリへのインターフェイスを促進する単純な“Helper”関数の作成を可能にするために提供されます。有効な C ステートメントはすべて、#include ステートメントを除いて許可されます。C-stdlib 関数の限定サブセットのみ使用可能です。ただし、現在の PROC PROTO ステップ内ですでに宣言されているその他の C 関数を呼び出せます。

次の C-stdlib 関数がサポートされています。

表 42.3 サポート対象の stdlib 関数

関数	説明
double sin(double x)	x の正弦を返します(ラジアン)。
double cos(double x)	x の余弦を返します(ラジアン)。
double tan(double x)	x の正接を返します(ラジアン)。
double asin(double x)	x の逆正弦を返します(-pi/2 から pi/2 ラジアンまで)。
double acos(double x)	x の逆余弦を返します(0 から pi ラジアンまで)。
double atan(double x)	x の逆正接を返します(-pi/2 から pi/2 ラジアンまで)。
double atan2(double x, double y)	y/x の逆正接を返します(-pi から pi ラジアンまで)。
double sinh(double x)	x の双曲線正弦を返します(ラジアン)。
double cosh(double x)	x の双曲線余弦を返します(ラジアン)。

関数	説明
double tanh(double x)	x の双曲線正接を返します(ラジアン)。
double exp(double x)	x の指数値を返します。
double log(double x)	x の対数を返します。
double log2(double x)	底を 2 とする x の対数を返します。
double log10(double x)	底を 10 とする x の対数を返します。
double pow(double x, double y)	x の y 乗(x**y)を返します。
double sqrt(double x)	x の平方根を返します。
double ceil(double x)	x 以上の最小の整数を返します。
double fmod(double x, double y)	(x/y)の余りを返します。
double floor(double x)	x 以下の最大の整数を返します。
int abs(int x)	x の絶対値を返します。
double fabs(double)	x の絶対値を返します。
int min(int x, int y)	x と y の最小値を返します。
double fmin(double x, double y)	x と y の最小値を返します。
int max(int x, int y)	x と y の最大値を返します。
double fmax(double x, double y)	x と y の最大値を返します。
char* malloc(int x)	サイズ x のメモリを割り当てます。
void free(char*)	malloc で割り当てたメモリを解放します。

次の例は、PROC PROTO に直接書き込まれる単純な C 関数を示しています。

```
proc proto
package=work.mylib.foo;
  struct mystruct {
    short a;
    long b;
  };
  int fillMyStruct(short a, short b,
struct mystruct * s);
  externc  fillMyStruct;
    int fillMyStruct(short a, short b,
```

```
struct mystruct * s) {
        s ->a = a;
        s ->b = b;
        return(0);
    }
  externcend;
run;
```

C 言語指定の制限

PROTO プロシジャでの C 言語指定の制限は次のとおりです。

- #define ステートメントは、後ろにセミコロン(;)を指定し、値を数値にする必要があります。
- #define ステートメントの機能は、単純な置き換えと、ネストされていない式に制限されています。影響を受ける記号は、配列ディメンション参照のみです。
- C プリプロセッサステートメントの#include と#if はサポートされていません。SAS マクロ%INC を#include のかわりに使用できます。
- 構造要素では最大 2 レベルの間接指定が使用できます。"double ***"などの要素は使用できません。これらの要素の種類が構造体で必要にもかかわらず、SAS でアクセスされない場合は、プレースホルダーを使用できます。
- float 型はサポートされません。
- 現在サポートされている型指定子は Unsigned のみです。
- 構造変数の指定ビットサイズはサポートされません。
- 関数ポインターと関数ポインターの定義はサポートされません。
- union 型はサポートされません。ただし、union の 1 要素のみを使用する予定であれば、union の変数をその要素の型として宣言できます。
- 他の構造体へのポインター以外の参照は、使用前に定義する必要があります。
- 構造体では ENUM キーワードは使用できません。構造体で ENUM を指定するには、TYPEDEF キーワードを使用します。
- 同じ英数字の名前を持つ構造要素でも、大文字/小文字の表記が異なる場合(ALPHA、Alpha および alpha など)はサポートされません。SAS では、大文字と小文字は区別されません。したがって、大文字と個別の区別がないプログラムで比較する場合、すべての 構造要素を一意にする必要があります。

構文: PROTO プロシジャ

参照項目: *"PROTOLIBS システムオプション" (SAS システムオプション: リファレンス)*

PROC PROTO PACKAGE=*entry* <*options*>;

MAPMISS *type1=value1 type2=value2 ...*;

LINK *load-module* <NOUNLOAD>;

function–prototype–1 <*function-prototype-n* ...;>

ステートメント	タスク	例
“PROC PROTO ステートメント”	C または C++プログラミング言語で記述される外部関数をバッチモードで登録します。	Ex. 1
“LINK ステートメント”	名前、パス、および関数を含むロードモジュールを指定します。	
“MAPMISS ステートメント”	値が欠損している場合に関数に渡す代替値を型によって指定します。	
“FUNCTION-PROTOTYPE-N ステートメント”	PROTO プロシジャに関数プロトタイプを登録します。	

PROC PROTO ステートメント

C または C++プログラミング言語で記述される外部関数をバッチモードで登録します。

例: “例: 分割関数の例” (1533 ページ)

構文

PROC PROTO PACKAGE=*entry* <*options*>;

オプション引数の要約

ENCRYPT

HIDE
: XML データベースの場合のみ、データセット内でコードのエンコードを可能にします。

LABEL=*package-label*
: パッケージの説明やラベル付けのためのテキスト文字列を指定します。

NOSIGNALS
: パッケージ内の関数で例外を作成しないことを指定します。

STDCALL
: Windows PC プラットフォームでのみ、"_ stdcall"規則を使用して関数を呼び出すことを指定します。

STRUCTPACK
: 32 ビット Windows PC プラットフォームまたはで Linux でのみ実行する Power Architecture プラットフォームでは、パッケージ内のすべての構造

を、特定の N-BYTE パッキングプラグマを使用してコンパイルすることを指定します。

必須引数

PACKAGE=*entry*
プロトタイプ情報が保存されている SAS エントリを指定します。*Entry* は、次の形式をもつ 3 レベル名です: *library.dataset.package.Package* により、GUI でのグループ化を指定できます。

function-prototype-1
function-prototype-n
関数プロトタイプの C コードが含まれます。

オプション引数

ENCRYPT | HIDE
データベース内のエンコードの許可を指定します。

制限事項 このオプションは XML データベースにのみ使用可です。

LABEL=*package-label*
パッケージの説明やラベル付けのためのテキスト文字列を指定します。ラベルの最大長は 256 文字です。

NOSIGNALS
パッケージ内の関数で例外やシグナルを作成しないことを指定します。

STDCALL
Windows PC プラットフォームでのみ、パッケージ内のすべての関数を "_stdcall"規則を使用して呼び出すことを指定します。

STRUCTPACK
Linux でのみ実行する Power Architecture プラットフォームでは、このパッケージ内のすべての 構造を、指定の N-BYTE パッキングプラグマを使用してコンパイルすることを指定します。したがって、STRUCTURE4 では、パッケージ内のすべての構造を“#pragma pack(4)”オプションによりコンパイルすることを指定します。

別名 PACK

LINK ステートメント

関数を含むロードモジュールの名前を指定します。パスの指定はオプションです。

構文

LINK *load-module* <NOUNLOAD>;

必須引数

load-module
関数を含むロードモジュールを指定します。LINK ステートメントをさらに追加すると、プロトタイプに対して必要な数のライブラリを含められます。
Load-module は動作環境に応じて次の形式を取ります。

'c:\mylibs\xxx.dll';

'c:\mylibs\xxx';

'/users/me/mylibs/xxx';

ヒント　PROTO プロシジャによってモジュールをリンクするには、フルパス名の指定が最も安全かつ望ましい方法です。

オプション引数

NOUNLOAD

> 重要　NOUNLOAD オプションを使用して、特定のライブラリの実行時の非互換性が原因で PROTO プロシジャがクラッシュするのを防ぐことができます。

SAS セッション終了時に、選択したライブラリをロードしたままにしておくことを指定します。

詳細

拡張子と関数の指定

モジュールの拡張子を指定する必要はありません。SAS では、動作環境固有の拡張子が付いたモジュールがロードされます。

SAS での検索を可能にするために、ロードモジュールですべての 関数を外部宣言する必要があります。ほとんどのプラットフォームでは、外部宣言がコンパイラのデフォルト動作です。ただし、多くの C コンパイラでは、デフォルトでは関数名がエクスポートされません。次の例では、ほとんどの PC コンパイラに対して外部ロードのために関数を宣言する方法を示します。

```
_declspec(dllexport) int myfunc(int, double);
_declspec(dllexport) int price2(int a, double foo);
```

PROTOLIBS システムオプション

注意
セキュリティの問題 LINK ステートメントによって設定されたロードライブラリの有効なパスにサードパーティライブラリを呼び出すと、セキュリティの問題が発生する可能性があります。

サイトの SAS 管理者は、PROTOLIBS システムオプションを使用して、LINK ステートメントの機能を制御できます。管理者は、PROTOLIBS オプションを使用して、LINK ステートメントで指定されたモジュールをロードできる有効なパスのリストを指定できます。管理者はまた、PROTOLIBS オプションを使用して、LINK ステートメントを無効にすることもできます。

PROTOLIBS オプションは制限付きオプションです。サイトの SAS 管理者のみがその値を指定できます。PROTOLIBS オプションのデフォルト値は NONE です。これは、LINK ステートメントが無効になっていることを意味します。

PROC OPTIONS の VALUE 引数を使用して、ロードモジュールを登録できる場所を見つけることができます。

MAPMISS ステートメント

値が欠損している場合に関数に渡す代替値を型によって指定します。

構文

MAPMISS <POINTER=*NULL* > <CHARPOINTER=*NULL*> <INT=*integer-value* >
<DOUBLE=*double-value*>
< LONG=*long-value* > <SHORT=*short-value*>;

オプション引数

POINTER=*NULL*
数値が不足した場合に関数に NULL ポインタを渡すように指定します。

デフォルト null

CHARPOINTER=*NULL*

デフォルト null

制限事項 CHARPOINTER オプションは、2025.02 のリリースから利用可能になります。

文字が見つからない場合に関数に NULL ポインターを渡すように指定します。

INT=*integer-value*
欠損している整数値のかわりに関数に渡す整数値を指定します。

DOUBLE=*double-value*
欠損している double 値のかわりに関数に渡す double 値を指定します。

LONG=*long-value*
欠損している long 値のかわりに関数に渡す long 値を指定します。

SHORT=*short-value*
欠損している short 値のかわりに関数に渡す short 値を指定します。

詳細

MAPMISS ステートメントを使用して、データの種類またはポインター値によって、代替値を指定します。値が欠損している場合、これらの値が関数に渡されます。値は MAPMISS ステートメントの引数としてして指定されます。

POINTER=NULL または CHARPOINTER=NULL を設定した場合、欠損しているポインター変数のかわりに NULL 値ポインターが関数に渡されます。引数として使用される種類に対して関数へのマッピングを指定しない場合は、その型の数値引数が欠損していても、関数は呼び出されません。

パラメーターとして C 関数に渡されるときに、配列要素の欠損値チェックが行われないので、MAPMISS 値は配列に影響を及ぼしません。

FUNCTION-PROTOTYPE-N ステートメント

PROTO プロシジャに関数プロトタイプを登録します。

構文

function-prototype-1 <*function-prototype-n* ...> *return-type function-name* (*argument-type* <*argument-name*> / <*iotype*> <*argument-label*>, ...) <*options*>;

必須引数

return-type
戻り値の C 言語の種類を指定します。

ヒント *return-type* の前に UNSIGNED か EXCELDATE のどちらかの修飾子を付けられます。戻り値の種類が Microsoft Excel 日付の場合は、EXCELDATE を使用する必要があります。

参照項目: “サポート対象の C 言語の戻り値の種類” (1510 ページ)

function-name
登録する関数の名前を指定します。

ヒント 指定パッケージ内の関数名は、最初の 32 文字を一意にする必要があります。

argument-type
関数の引数に対して C 言語の型を指定します。

関数の引数リストで、引数ごとに *argument-type* を指定する必要があります。引数リストは、必ずかっこで囲んでください。引数が配列の場合は、配列サイズを含む角かっこが後ろに付いた引数を指定する必要があります(例:double A[10])。

サイズが不明な場合や、長さの検証を無効化する場合は、かわりに型**name* を使用します(例:double*A)。

ヒント *Argument-type* の前に UNSIGNED、CONST、EXCELDATE のいずれかの修飾子を付けられます。戻り値の種類が Microsoft Excel 日付の場合は、EXCELDATE を使用する必要があります。

参照項目: “サポート対象の C 言語の引数の種類” (1511 ページ)

argument-name
引数の名前を指定します。

iotype
引数の入出力の種類を指定します。入力(input)の場合は I、出力(output)の場合は O、更新(update)の場合は U を使用します。

別名 IO

ヒント プログラムコードでは、IOTYPE=I | O | U を使用します。

デフォルトでは、ポインターとなるパラメーターではすべて 入力の種類が U と見なされます。ポインター以外の値ではすべて入力の種類が I と見なされます。この動作は、C 言語のパラメーター渡しスキームに対応しています。

参照項目: “例: 分割関数の例” (1533 ページ) (*iotype* の使用例)。

argument-label
引数のラベルや説明を指定します。

LABEL="*text-string*"
関数の説明やラベルを指定します。テキスト文字列を引用符で囲みます。

注 LABEL オプションは PROTO プロシジャとともに使用できます。

KIND | GROUP=*group-type*
関数の属するグループを指定します。KIND=または GROUP=オプションを使用すると、パッケージの関数を簡単にグループ化できます。

引用符内の任意の文字列(40 文字以内)を使用して、類似する関数をグループ化できます。

注 KIND または GROUP オプションは、PROTO プロシジャとともに使用できます。

ヒント INPUT(金融商品の入力)、TRANS(リスクファクターの変換)、PRICING(金融商品の評価)、PROJECT は、リスクディメンション用に提供された特殊なケースで、引用符は必要ありません。デフォルトは PRICING です。

使用法: PROTO プロシジャ

基本的な C 言語の種類

SAS 言語では、文字と数値という 2 つのデータの種類がサポートされます。これらのデータの種類は、C プログラミング言語での character および double (倍精度浮動小数点)データ型の配列に対応しています。SAS 変数は、外部 C 関数に対する引き数として使用される場合、適切な型に変換(キャスト)されます。

文字変数の操作

文字変数は、"char *"値を必要とする引数に対してのみ使用できます。渡される文字列は、現在の文字列長で終了する null 文字列です。現在の文字列長は、文字列の割り当て長と、文字列に格納された最終値の長さの最小値です。文字列の割り当て長(デフォルトでは、32 バイト)は、LENGTH ステートメントを使用すると指定できます。"char *"を返す関数では、SAS 変数にコピーされる、null またはゼロ区切りの文字列が返される場合があります。現在の文字列長が割り当て長よりも短い場合、文字列にはブランクが埋め込まれます。

次の例では、*str* の割り当て長は 10 ですが、現在の長さは 5 です。文字列が割り当て長で NULL で終了する場合、"hello "が関数 xxx に渡されます。

```
length str $ 10;
str = "hello";
call xxx(str);
```

ブランクの埋め込みを回避するには、関数呼び出し内のパラメーターで SAS 関数 TRIM を使用します。

```
length str $ 10;
str = "hello";
call xxx(trim(str));
```

この場合、値"hello"が関数 xxx に渡されます。

数値変数の操作

short、int、long、double データ型を必要とする引数、ならびにそのデータ型へのポインターとして、数値変数を使用できます。数値変数は、自動的に必要な型に変換されます。変換に失敗すると、関数は呼び出されず、関数に対する出力は欠損に

設定されます。これらの型へのポインターが要求された場合は、変換値のアドレスが渡されます。値は、呼び出し先から返される際、double 型に変換し直され、SAS 変数に格納されます。SAS スカラー変数は、2 レベル以上の間接指定を要する引数として渡すことはできません。たとえば、SAS 変数は、"long **"型へのキャストを要する引数として渡すことはできません。

欠損値の操作

欠損値を含む SAS 変数は、PROTO プロシジャ使用時の呼び出し関数での欠損値のマッピング方法に従って変換されます。関数から返されたすべての値に対して、マップされた欠損値のチェックが行われ、SAS 欠損値に変換されます。

たとえば、関数に対する引数が欠損しており、なおかつその引数が整数への変換対象の場合、整数は-99 にマップされ、-99 が関数に渡されます。同じ関数で値-99 を含む整数が返された場合、この値の返し先の変数には欠損値が含まれます。

関数名

外部関数および FCMP 関数は、異なるパッケージに保存されている限り、同じ名前を保持できます。これらのパッケージのロード時には、ログの警告メッセージに、一定の関数に対するデフォルト定義を含むパッケージが表示されます。デフォルト定義以外の、パッケージの関数定義を使用するには、*package-name.function-name* を使用して関数を呼び出します。

function-name 複数の外部関数パッケージをロードするときは、すべての関数名が固有である必要があります。同一名の外部関数を 2 つ以上ロードする場合は、最初にロードされる関数を使用します。重複する外部関数は無視します。警告メッセージに、使用される関数を含むパッケージと、破棄された定義を含んでいたパッケージが表示されます。

外部 C 関数との連携

外部 C 関数との連携を容易にするために、PROTO に対応した多くのプロシジャが拡張され、その C 型のほとんどがサポートされるようになりました。

配列での作業時には、SAS 配列が EXTERNC ルーチンに一次元配列として受け渡されるため注意が必要です。SAS 配列は、FCMP 関数に戻るときには一次元配列として読み込まれます。

次の例は、配列とダブル**変数タイプの使用時に生成されるログ出力を示しています。

```
proc proto package=work.proto_ds.test;
  void idmat(double** a,int m,int n);
  externc idmat;
```

```
  void idmat(double** a,int m,int n)
  {
    int i=0,j=0;
    for(i=0;i<m;i++)
    {
      for(j=0;j<n;j++)
      {
       if(i==j)
         a[0][i*m+j]=1;
       else
         a[0][i*m+j]=0;
     }
   }
};
externcend;
run;

proc fcmp outlib=work.proto_ds.test inlib=work.proto_ds;
  subroutine sas_idmat(b[*,*]);
    outargs b;
    m=dim(b,1);
    n=dim(b,2);
    call idmat(b,m,n);
    put b=;
    put ' ';
  endsub;
quit;
run;

options cmplib=work.proto_ds;

data cmat;
  array ctmp[3,3] _temporary_;
  array c[3,3];
  output;
  call sas_idmat(ctmp);
  do i=1 to dim(c,1);
    do j=1 to dim(c,2);
      c[i,j]=ctmp[i,j];
    end;
  end;
  output;
  drop i j;
run;
```

SAS は次の結果をログに書き込みます。

```
b[1, 1]=1 b[1, 2]=0 b[1, 3]=0 b[2, 1]=0 b[2, 2]=1 b[2, 3]=0 b[3, 1]=0 b[3, 2]=0 b[3, 3]=1
```

SAS 変数でいずれかの種類にポインターを返して保存する方法はありません。ポインターは常に間接参照され、そのコンテンツが変換されて SAS 変数にコピーされます。

PROTO に対応したプロシジャで C 変数を指定するには、EXTERNC ステートメントを使用します。EXTERNC ステートメントの構文は、次の形式を取ります。

EXTERNC DOUBLE | INT | LONG | SHORT | CHAR <[*][*]> *var-1 <var-2 ... var-n>*;

次のテーブル(表 42.88 (1527 ページ))は、これらの変数が式の左側に位置する場合の処理を示しています。このテーブルでは、割り当ての右側で short 型に対して実行される自動キャストも示されます。(キャストと呼ばれる単項演算子によって、いずれの式でも明示的な型の変換が強制実行される場合があります。)テーブルには、SAS 変数に関連付けられた short 型の使用可能な組み合わせがすべてリストされています。

注: int、long および double 型のテーブルを作成するには、任意の型をこのテーブルの"short"と置き換えます。

ポインターのいずれかが null で、間接参照を要する場合は、結果変数に対して設定された欠損値があれば、結果は欠損に設定されます。詳細については、“MAPMISS ステートメント” (1521 ページ)を参照してください。

表 42.4 割り当てステートメントでの short データ型の自動型キャスト

割り当ての左側の型	割り当ての右側の型	実行されるキャスト
short	SAS 数値	y = (short) x
short	short	y = x
short	short *	y = * x
short	short **	y = ** x
short *	SAS 数値	* y = (short) x
short *	short	y = & x
short *	short *	y = x
short *	short **	y = * x
short **	SAS 数値	**y = (short) x
short **	short *	y = & x
short **	short **	y = x
SAS 数値	short	y = (double) x
SAS 数値	short *	y = (double) * x
SAS 数値	short **	y = (double) ** x

次のテーブルは、これらの変数が引数として外部 C 関数に渡されるときの処理を示しています。

表 42.5　外部 C 引数に使用できる種類

関数プロトタイプ	SAS 変数の種類	C 変数の種類
short	数値	short, short *, short **
short *	numeric, array	short, short *, short **
short **	array	short *, short **
int	数値	int, int *, int **
int *	numeric, array	int, int *, int **
int **	array	int *, int **
long	数値	long, long *, long **
long *	numeric, array	long, long *, long **
long **	array	long *, long **
double	数値	double, double *, double **
double *	numeric, array	double, double *, double **
double **	array	double *, double **
char *	文字	char *, char **
char **	文字	char *, char **
struct *	structure	struct *, struct **
struct **	structure	struct *, struct **

注: 2 つの異なる C 型間の自動変換は実行されません。

PROC PROTO のパッケージのスコープ

PROC PROTO パッケージは CMPLIB システムオプションに指定された順序でロードされ、コンテンツがグローバルに使用されます。PROC ステートメントオプションによってロードされたパッケージは、スコープにおいてはローカルと 見なされます。ローカル定義は最後にロードされ、いかなる場合でも、ローカルスコープによってグローバルスコープが 上書きされます。

名前が一意でも重複していても、定義がロードされます。特定の PROC PROTO 要素の複数定義(列挙名や関数プロトタイプなど)によって、名前が競合し、エラーが発生する可能性があります。PROC PROTO パッケージ間の名前の競合を防ぐには、列挙される型や関数定義などの要素の名前を一意にするようにします。

次の例では、3 つの PROC PROTO パッケージをロードし、typedef および#define ステートメントの相互上書き順序が示されます。

例のこの部分では、最初の 2 つの PROC PROTO パッケージがロードされます。

```
proc proto package = work.p1.test1;
   typedef struct { int a; int b; } AB_t;
   #define NUM 1;
   int p1(void);
   externc p1;
      int p1(void)
      {
         return NUM;
      }
   externcend;
run;

proc proto package = work.p2.test2;
   typedef struct { int a; int b; } AB_t;
   #define NUM 2;
   int p2(void);
   externc p2;
      int p2(void)
      {
         return NUM;
      }
   externcend;
run;

options CMPLIB = (work.p1 work.p2);

proc fcmp;
  x = p1();
  put "Should be 2: " x=;
run;
```

パッケージは順序どおりにロードされるので、前述のプログラムの実行結果は 2 です。

次の例では、前述の CMPLIB=システムオプション設定を保持したまま、PROC PROTO で 3 番目のパッケージを追加し、PROC FCMP でローカルに含めます。

```
proc proto package = work.p3.test3;
   typedef struct { int a; int b; } AB_t;
   #define NUM 3;
   int p3(void);
   externc p3;
      int p3(void)
      {
         return NUM;
      }
   externcend;
run;

proc fcmp libname = work.p3;
  x = p1();
  put "Should be 3: " x=;
run;
```

この例では、*work.p3* の NUM のローカル定義が、*work.p1* と *work.p2* によってロードされるグローバル定義のかわりに使用されます。

C Helper 関数と CALL ルーチン

C Helper 関数と CALL ルーチンについて

PROC FCMP で C 言語構造体を処理するために、パッケージとあわせて複数の Helper 関数と CALL ルーチンが提供されます。ほとんどの C 言語構造体は、参照したり PROC FCMP で使用する前に、PROC PROTO によって作成されるカタログパッケージで定義しておく必要があります。ISNULL 関数ならびに STRUCTINDEX および SETNULL CALL ルーチンが追加されて、SAS 言語が拡張され、そのままでは SAS 言語に適応しない C 言語構造体を処理できるようになりました。

次の C Helper 関数と CALL ルーチンが使用可能です。

表 42.6　C Helper 関数と CALL ルーチン

C Helper 関数または CALL ルーチン	説明
“ISNULL C Helper 関数” (1531 ページ)	構造体のポインター要素が NULL かどうかを決定します。
“SETNULL C Helper CALL ルーチン” (1531 ページ)	構造体のポインター要素を NULL に設定します。
“STRUCTINDEX C Helper CALL ルーチン” (1532 ページ)	構造体の配列の各構造体要素にアクセスできます。

ISNULL C Helper 関数

ISNULL 関数では、構造体のポインター要素が NULL かどうかが決定されます。関数は次の形式を取ります。

double **ISNULL** (*pointer-element*);

Pointer-element はポインター要素を指します。

次の例では、PROC PROTO を使用して、LINKLIST 構造と GET_LIST 関数が定義されます。GET_LIST 関数は、要求される数の要素を含むリンクリストを生成する外部 C ルーチンです。

```
struct linklist{
  double value;
  struct linklist * next;
};

struct linklist * get_list(int);
```

次の例では、ISNULL Helper 関数を使用して、GET_LIST 関数によって作成されるリンクされたリストにわたりループする方法を示します。

```
struct linklist list;

list = get_list(3);
put list.value=;

do while (^isnull(list.next));
  list = list.next;
  put list.value=;
end;
```

例のコード 42.1　リンクされたリストのループ化

```
LIST.value=0
LIST.value=1
LIST.value=2
```

SETNULL C Helper CALL ルーチン

SETNULL CALL ルーチンで、構造体のポインター要素が null に設定されます。これは次の形式を取ります。

CALL SETNULL(*pointer-element*);

Pointer-element は構造へのポインターです。

ポインター値のある変数(構造体エントリ)を指定すると、SETNULL でポインターが null に設定されます。

```
call setnull(12.next);
```

次の例では、PROC PROTO を使用して定義された LINKLIST 構造を前提としています。SETNULL CALL ルーチンを使用すると、次の要素を null に設定できます。

```
proc proto;
  struct linklist list;
  call setnull(list.next);
run;
```

STRUCTINDEX C Helper CALL ルーチン

STRUCTINDEX CALL ルーチンを使用すると、構造体の配列の各構造体要素にアクセスできます。構造体に構造体の配列が含まれている場合、STRUCTINDEX CALL ルーチンを使用すると、配列の各構造体要素にアクセスできます。STRUCTINDEX CALL は次の形式を取ります。

CALL STRUCTINDEX(*struct_array*, *index*, *struct_element*);

Struct_array は配列を指定します。*index* は、SAS 配列で使用される、1 つから始まるインデックスです。また、*struct_element* は、配列の要素を指します。

次の例は 2 つの部分で構成されています。例の 2 つの部分をコピーして SAS エディターに貼り付け、それらを 1 つの SAS プログラムとして実行します。

この例の最初の部分では、PROC PROTO を使用して、次の構造体と関数が定義されます。

```
options cmplib=(work.proto_ds work.fcmp_ds);
proc proto package=work.proto_ds.cfcns;
   struct POINT {
        short s;
        int i;
        long l;
        double d;
   };
   struct POINT_ARRAY {
        int length;
        struct POINT * p;
        char name[32];
   };
   struct POINT * struct_array( int );

   externc struct_array;
      struct POINT * struct_array( int num ) {
          return(malloc(sizeof(struct POINT) * num));
      }
   externcend;
run;
```

この例の 2 番目の部分の PROC FCMP コードセグメントでは、STRUCTINDEX CALL ルーチンを使用して、POINT_ARRAY 構造で P と呼ばれる配列の各 POINT 構造要素を取得、設定する方法が示されます。

```
proc fcmp;
struct point_array pntarray;
struct point pnt;
```

```
  /* Call struct_array to allocate an array of 2 POINT structures. */
pntarray.p = struct_array(2);
pntarray.plen = 2;
pntarray.name = "My funny structure";

  /* Get each element using the STRUCTINDEX CALL routine and set values. */
do i = 1 to 2;
  call structindex(pntarray.p, i, pnt);
  put "Before setting the" i "element: " pnt=;
  pnt.s = 1;
  pnt.i = 2;
  pnt.l = 3;
  pnt.d = 4.5;
  put "After setting the" i "element: " pnt=;
end;

run;
```

例のコード 42.2　STRUCTINDEX CALL ルーチンの結果

```
Before setting the 1 element: PNT {s=0, i=0, l=0, d=0}
After setting the 1 element: PNT {s=1, i=2, l=3, d=4.5}
Before setting the 2 element: PNT {s=0, i=0, l=0, d=0}
After setting the 2 element: PNT {s=1, i=2, l=3, d=4.5}
```

例: 分割関数の例

要素:

INT ステートメント
KIND=プロトタイプ引数
PROC FCMP

詳細

この例では、PROC PROTO を使用して、SPLIT と CASHFLOW という 2 つの外部 C 言語関数のプロトタイプを作成する方法を示します。これらの関数は、LINK ステートメントで指定された 2 つの共有ライブラリに含まれます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=40;

proc proto package =
```

```
sasuser.myfuncs.mathfun
        label = "package of math functions";


        link "link-library";
        link "link-library";

  int split(int x "number to split")
     label = "splitter function" kind=PRICING;

  int cashflow(double amt, double rate, int periods,
            double * flows / iotype=O)
     label = "cash flow function" kind=PRICING;

run;

proc fcmp libname=sasuser.myfuncs;
  array flows[20];
  a = split(32);
  put a;
  b = cashflow(1000, .07, 20, flows);
  put b;
  put flows;
run;

run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

関数パッケージ情報を保存するカタログエントリを指定します。 カタログエントリは 3 レベルの名前です。

```
proc proto package =
sasuser.myfuncs.mathfun
        label = "package of math functions";
```

SPLIT および CASHFLOW 関数を含むライブラリを指定します。 LINK ステートメントをさらに追加すると、プロトタイプに対して必要な数のライブラリを含められます。

```
        link "link-library";
        link "link-library";
```

SPLIT 関数のプロトタイプを作成します。 INT ステートメントで、SPLIT 関数のプロトタイプが作成され、その関数にラベルが割り当てられます。

```
  int split(int x "number to split")
     label = "splitter function" kind=PRICING;
```

CASHFLOW 関数のプロトタイプを作成します。 INT ステートメントで、CASHFLOW 関数のプロトタイプが作成され、その関数にラベルが割り当てられます。

```
  int cashflow(double amt, double rate, int periods,
               double * flows / iotype=O)
    label = "cash flow function" kind=PRICING;
```

PROTO プロシジャを実行します。 RUN ステートメントで、PROTO プロシジャが実行されます。

```
run;
```

SPLIT および CASHFLOW 関数を呼び出します。 PROC FCMP で、SPLIT および CASHFLOW 関数が呼び出されます。PROC FCMP からの出力が作成されます。

```
proc fcmp libname=sasuser.myfuncs;
  array flows[20];
  a = split(32);
  put a;
  b = cashflow(1000, .07, 20, flows);
  put b;
  put flows;
run;
```

FCMP プロシジャを実行します。 RUN ステートメントは、FCMP プロシジャを実行します。

```
run;
```

出力: 関数のプロトタイピング

アウトプット 42.1　SPLIT 関数および CASHFLOW 関数のプロトタイプ作成の結果

```
                                        The SAS System                      1


                                      The FCMP Procedure

16
12
70 105 128.33333333 145.83333333 159.83333333 171.5 181.5 190.25 198.02777778
205.02777778
211.39141414 217.22474747 222.60936286 227.60936286 232.27602953 236.65102953
240.76867658
244.65756547 248.341776 251.841776
```

43

PRTDEF プロシジャ

概要: PRTDEF プロシジャ

PRTDEF プロシジャの機能

PRTDEF プロシジャでは、個々のユーザーかサイトのすべての SAS ユーザーのどちらかに対して、バッチモードでプリンター定義が作成されます。システム管理者は、PROC PRTDEF で USESASHELP オプションを使用して、SAS レジストリにプリンター定義を作成し、サイトのすべての SAS ユーザーがそのプリンターを使用できるようにします。個々のユーザーは、PROC PRTDEF を使用して、SAS レジストリに個人用プリンター定義を作成します。

関連項目

44 章, “PRTEXP プロシジャ,” (1555 ページ)

構文: PRTDEF プロシジャ

PROC PRTDEF <*options*>;

ステートメント	タスク	例
“PROC PRTDEF ステートメント”	個々のユーザーかサイトのすべての SAS ユーザーのどちらかに対して、バッチモードでプリンター定義を作成します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5

PROC PRTDEF ステートメント

バッチモードでプリンター定義を作成します。

例:
“例 1: 複数プリンター定義の定義” (1546 ページ)
“例 2: SASUSER の PostScript プリンター出力をプレビューするために SASUSER に Ghostview プリンターを作成する” (1547 ページ)
“例 3: すべてのユーザーが使用可能な単一プリンター定義の作成” (1549 ページ)
“例 4: プリンター定義の追加、変更、削除” (1551 ページ)
“例 5: 単一プリンター定義の削除” (1553 ページ)

構文

PROC PRTDEF <*options*>;

オプション引数

DATA=*SAS-data-set*
プリンター属性を含む SAS データセットを指定します。

要件 指定の必要なプリンター属性変数は、DEST、DEVICE、MODEL、NAME ですが、変数 OPCODE の値が DELETE の場合は例外です。その場合は、NAME 変数のみ必須になります。

参照項目: “プリンター定義の作成に使用される入力データセット変数” (1540 ページ)

DELETE
デフォルト操作としてレジストリからプリンター定義を削除することを指定します。

操作 DELETE と REPLACE の両方を指定すると、DELETE がデフォルト操作になります。

ヒント ユーザー定義されたプリンター定義を削除しても、Sashelp カタログに管理者定義のプリンターが存在する場合は、そのプリンターを表示できます。

例 “例 5: 単一プリンター定義の削除” (1553 ページ)

FOREIGN
異なるホストにエクスポートするためにレジストリエントリを作成することを指定します。その結果として、いずれのホスト依存項目(TRANTAB など)のテストもスキップされます。

LIST
作成または置換するプリンターのリストをログに書き込むことを指定します。

例 “例 2: SASUSER の PostScript プリンター出力をプレビューするために SASUSER に Ghostview プリンターを作成する” (1547 ページ)

“例 4: プリンター定義の追加、変更、削除” (1551 ページ)

REPLACE
デフォルト操作として既存のプリンター定義を変更することを指定します。すでに存在するプリンター名はすべて、プリンター属性データセットの情報を使用して変更されます。存在しないプリンター名はすべて追加されます。

操作 REPLACE と DELETE の両方を指定すると、DELETE が実行されます。

例 “例 2: SASUSER の PostScript プリンター出力をプレビューするために SASUSER に Ghostview プリンターを作成する” (1547 ページ)

USESASHELP
プリンター定義を Sashelp ライブラリに置くことを指定します。ここでは、すべてのユーザーがこれらの定義を使用できます。USESASHELP オプションを指定しなければ、プリンター定義は現在の Sasuser ライブラリに置かれます。ここでは、ローカルユーザーのみこの定義を使用できます。

Windows 固有: Windows 動作環境で PROC PRTDEF を使用して、プリンター定義を作成できます。ただし、Windows ではデフォルトでユニバーサル印刷が無効になっているため、これらのプリンター定義は**印刷**ウィンドウに表示されません。ユニバーサル印刷が無効なときにプリンター定義を使用する場合は、PRINTERPATH システムオプションの一部としてプリンター定義を指定するか、または Output Delivery System (ODS)から、次のコードを発行します。

```
ODS PRINTER SAS PRINTER=myprinter;
```

この場合、*myprinter* は使用するプリンター定義の名前になります。

制限事項 USESASHELP オプションを使用するには、Sashelp カタログへの書き込み権限が必要です。

例 “例 3: すべてのユーザーが 使用可能な単一プリンター定義の作成” (1549 ページ)

使用法: PRTDEF プロシジャ

プリンター定義の作成に使用される入力データセット変数

有効変数の要約

プリンター定義を作成するには、適切なプリンター属性を含む変数を持つ SAS データセットを作成する必要があります。次のテーブルでは、このデータセットの必須変数とオプション変数の両方をリストにして説明を加えています。

表 43.1 プリンター定義レコード作成用の必須変数とオプション変数

変数名	変数説明
必須	
DEST	出力先
DEVICE	デバイス
MODEL	プロトタイプ
NAME	プリンター名
任意	
BOTTOM	デフォルト下余白
CHARSET	デフォルトフォント文字セット

変数名	変数説明
DESC	説明
FONTSIZE	デフォルトのフォントサイズ
HOSTOPT	ホストオプション
LEFT	デフォルト左余白
LRECL	出力バッファーサイズ
OPCODE	操作コード
PAPERIN	用紙トレイまたは入力トレイ
PAPEROUT	紙の出力先または出力トレイ
PAPERSIZ	用紙のサイズ
PAPERTYP	用紙の種類
PREVIEW	プレビュー
PROTOCOL	プロトコル
RES	デフォルトプリンター解像度
RIGHT	デフォルト右余白
STYLE	デフォルトのフォントスタイル
TOP	デフォルト上余白
TRANTAB	変換テーブル
TYPEFACE	デフォルトのフォント
UNITS	CM 単位または IN 単位
VIEWER	ビューアー
WEIGHT	デフォルトのフォントの太さ

必須変数

プリンターを作成または変更するには、NAME、MODEL、DEVICE、DEST 変数を指定する必要があります。その他のすべての変数では、MODEL 変数で指定されたプリンタープロトタイプのデフォルト値が使用されます。

プリンターを削除するには、必須の NAME 変数のみ指定します。

入力データセットでは、次の変数が必須です。

DEST
プリンターの出力先を指定します。

動作環境の情報: DEST では、一部のデバイスで大文字と小文字が区別されます。

制限事項 DEST は 1023 文字に制限されます。

DEVICE
プリンターへの出力送信時に使用する I/O デバイスの種類を指定します。有効なデバイスは、**プリンター定義**ウィザードと SAS レジストリエディターでリストされます。

制限事項 DEVICE は 31 文字までにしてください。

MODEL
プリンターの定義時に使用するプリンタープロトタイプを指定します。

プロトタイプまたはモデルの説明の有効なリストについては、SAS レジストリエディターで CORE\PRINTING\PROTOTYPES の下を参照してください。

制限事項 MODEL は 127 文字までにしてください。

ヒント 対話モード中は、REGEDIT コマンドでレジストリを呼び出せます。

NAME
プリンター定義のその他の属性に関連付けられるプリンター定義名を指定します。

名前は、指定レジストリ内では一意です。新しいプリンター定義に既存の名前が含まれている場合は、REPLACE オプションを指定しておくか、OPCODE 変数の値が **Modify** でなければ、レコードは処理されません。

制限事項 NAME は 127 文字までにしてください。また、ブランクではない文字が少なくとも 1 つ必要で、バックスラッシュは含められません。先頭と末尾のブランクは名前から切り捨てられます。

オプション変数

入力データセットでは、次の変数がオプションです。

BOTTOM
UNITS 変数で指定した単位でデフォルト下余白を指定します。

CHARSET
デフォルトフォント文字セットを指定します。

制限事項 値は、TYPEFACE 変数で指定したタイプフェイスの文字セット名のいずれかである必要があります。

DESC
プリンターの説明を指定します。

デフォルト DESC は、デフォルトで、プリンターの作成に使用するプロトタイプになります。

制限事項 説明には最大で 1023 文字まで使用できます。

FONTSIZE
デフォルトフォントのポイントサイズを指定します。

HOSTOPT
出力先の任意のホストオプションを指定します。ホストオプションの大文字と小文字は区別されません。

制限事項 ホストオプションには最大で 1023 文字まで使用できます。

LEFT
UNITS 変数で指定した単位でデフォルト左余白を指定します。

LRECL
プリンターへの出力送信時に使用するバッファーサイズとレコード長を指定します。

デフォルト 既存プリンターの変更時に LRECL がゼロより小さい場合、プリンターのバッファーサイズは、プリンタープロトタイプで指定するサイズにリセットされます。

OPCODE
プリンター定義に対して実行するアクション(追加、削除または変更)を指定する文字変数です。

Add
レジストリに新しいプリンター定義を作成します。REPLACE オプションを指定した場合、この操作でも既存のプリンター定義が変更されます。

Delete
既存のプリンター定義をレジストリから削除します。

制限事項 この操作では、NAME 変数のみ定義が必要です。その他の変数は無視されます。

Modify
レジストリの既存プリンター定義を変更するか、新しく追加します。

制限事項 OPTCODE は 8 文字までにしてください。

ヒント ユーザーが SASHELP ライブラリのプリンターの新しい属性を変更、保存すると、その変更は SASUSER ライブラリに格納されます。ユーザーの指定した値が、管理者の指定した値より優先されますが、置き換えられることはありません。

PAPERIN
デフォルトの用紙トレイまたは入力トレイを指定します。

制限事項 PAPERIN の値は、MODEL 変数で指定したプリンタープロトタイプの用紙トレイ名のいずれかである必要があります。

PAPEROUT
デフォルトの給紙先または出力トレイを指定します。

制限事項 PAPEROUT の値は、MODEL 変数で指定したプリンタープロトタイプの給紙先名のいずれかである必要があります。

PAPERSIZ
デフォルトの用紙トレイまたは入力トレイを指定します。

制限事項 PAPERSIZ の値は、MODEL 変数で指定するプリンタープロトタイプにリストされた用紙のサイズ名のいずれかである必要があります。

PAPERTYP
デフォルトの用紙の種類を指定します。

制限事項 PAPERTYP の値は、MODEL 変数で指定するプリンタープロトタイプにリストされた用紙トレイ名のいずれかである必要があります。

PREVIEW
印刷プレビューに使用するプリンターアプリケーションを指定します。

制限事項 PREVIEW は 127 文字までにしてください。

PROTOCOL
プリンターへの出力送信時に使用する I/O プロトコルを指定します。

動作環境の情報: メインフレームシステムでは、プロトコルによって、メインフレームと ASCII デバイスを接続するプロトコルコンバータで処理可能な出力形式に出力を変換する方法が記述されます。

制限事項 PROTOCOL は 31 文字までにしてください。

RES
デフォルトプリンター解像度を指定します。

制限事項 RES の値は、MODEL 変数で指定するプリンタープロトタイプで使用可能な解像度値のいずれかである必要があります。

RIGHT
UNITS 変数で指定した単位でデフォルト右余白を指定します。

STYLE
デフォルトのフォントのスタイルを指定します。

制限事項 STYLE の値は、TYPEFACE 変数で指定したタイプフェイスで使用可能なスタイルのいずれかである必要があります。

TOP
UNITS 変数で指定した単位でデフォルト上余白を指定します。

TRANTAB
プリンターへの出力送信時に使用する変換テーブルを指定します。

動作環境の情報: 変換テーブルは、EBCDIC ホストでデータが ASCII デバイスに送信される際に必要です。

制限事項 TRANTAB は 8 文字までにしてください。

TYPEFACE
デフォルトフォントのタイプフェイスを指定します。

制限事項 タイプフェイスは、MODEL 変数で指定したプリンタープロトタイプで使用可能なタイプフェイス名のいずれかである必要があります。

UNITS
余白変数で指定した単位 CM または IN を指定します。

VIEWER
印刷プレビュー中に使用するホストシステムコマンドを指定します。その結果として、PROC PRTDEF によりプレビュープリンターが作成されます。

プレビュープリンターは、印刷前の画面でプリンター出力を表示するために使用する特別プリンターです。

制限事項 VIEWER は 127 文字までにしてください。

ヒント VIEWER の値が指定されていると、PREVIEW、PROTOCOL、DEST、HOSTOPT 変数の値は無視されます。ビューアーコマンドで、通常は入力ファイル名を指定する場所に*%s* を置きます。*%s* は必要に応じて何回でも使用できます。

WEIGHT
デフォルトのフォントの太さを指定します。

制限事項 値は、TYPEFACE 変数で指定したタイプフェイスの有効な太さのいずれかである必要があります。

例: PRTDEF プロシジャ

例 1: 複数プリンター定義の定義

要素: PROC PRTDEF ステートメントオプション
DATA=

詳細

この例では、各種プリンターの設定方法を示します。

プログラム

```
data printers;
input name $ 1-14 model $ 16-42 device $ 46-53 dest $ 57-70;
datalines;
Myprinter      PostScript Level 1 (Color)    PRINTER   printer1
Laserjet       PCL 5 (DeltaRow)              PIPE      lp -dprinter5
Color LaserJet PostScript Level 2 (Color)    PIPE      lp -dprinter2
;

proc prtdef data=printers;
run;
```

プログラムの説明

PRINTERS データセットを作成します。 INPUT ステートメントには、4 つの必須変数の名前が含まれます。各データ行に、1 つのプリンター定義を作成するのに必要な情報が含まれます。

```
data printers;
input name $ 1-14 model $ 16-42 device $ 46-53 dest $ 57-70;
datalines;
Myprinter      PostScript Level 1 (Color)    PRINTER   printer1
Laserjet       PCL 5 (DeltaRow)              PIPE      lp -dprinter5
Color LaserJet PostScript Level 2 (Color)    PIPE      lp -dprinter2
;
```

プリンター属性を含む入力データセットを指定し、プリンター定義を作成します。
PROC PRTDEF で、SAS レジストリに対してプリンター定義が作成され、DATA=オプションで、プリンター属性を含む入力データセットとして PRINTERS が指定されます。

```
proc prtdef data=printers;
run;
```

ログ

例のコード 43.1　プリンター定義後の SAS ログ

```
1   data printers;
2   input name $ 1-14 model $ 16-42 device $ 46-53 dest $ 57-70;
3   datalines;

NOTE: The data set WORK.PRINTERS has 3 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time           0.03 seconds


7   ;
8   proc prtdef data=printers;
9   run;

NOTE: 3 printer definitions added to the registry.
NOTE: 0 printer definitions modified in the registry.
NOTE: 0 printer definitions deleted from the registry.
NOTE: PROCEDURE PRTDEF used (Total process time):
      real time          0.15 seconds
      cpu time           0.01 seconds
```

例 2: SASUSER の PostScript プリンター出力をプレビューするために SASUSER に Ghostview プリンターを作成する

要素:

PROC PRTDEF ステートメントオプション
- DATA=
- LIST
- REPLACE

詳細

この例では、PostScript 出力をプレビューするために、Sasuser ライブラリに Ghostview プリンター定義を作成します。

プログラム

```
data gsview;
  name = "Ghostview";
  desc = "Print Preview with Ghostview";
  model= "PostScript Level 2 (Color)";
  viewer = 'ghostview %s';
  device = "Dummy";
  dest = " ";
run;

proc prtdef data=gsview list replace;
run;
```

プログラムの説明

GSVIEW データセットを作成し、プリンター名、プリンター説明、プリンタープロトタイプ、印刷プレビューに使用するコマンドを指定します。 GSVIEW データセットに含まれる変数の値には、プリンター定義の作成に必要な情報が含まれます。NAME 変数では、プリンター定義データレコードのその他の属性に関連付けられるプリンター名が指定されます。DESC 変数では、プリンターの説明が指定されます。MODEL 変数では、このプリンターの定義時に使用するプリンタープロトタイプが指定されます。VIEWER 変数では、印刷プレビューに使用するホストシステムコマンドが指定されます。GSVIEW をシステムにインストールして、VIEWER の値にそれを見つけるパスを含める必要があります。**%s** なので、値を単一引用符で囲む必要があります。二重引用符を使用した場合、SAS では、**%s** がマクロ変数と見なされます。DEVICE と DEST は必須変数ですが、この例では値は必要ありません。したがって、“dummy”またはブランク値を割り当ててください。

```
data gsview;
  name = "Ghostview";
  desc = "Print Preview with Ghostview";
  model= "PostScript Level 2 (Color)";
  viewer = 'ghostview %s';
  device = "Dummy";
  dest = " ";
run;
```

プリンター属性を含む入力データセットを指定し、プリンター定義を作成し、そのプリンター定義を SAS ログに書き込み、SAS レジストリの印刷定義を置き換えます。 DATA=オプションでは、プリンター属性を含む入力データセットとして GSVIEW が指定されます。PROC PRTDEF では、プリンター定義が作成されます。

LIST オプションでは、作成または置換されるプリンターのリストを SAS ログに書き込むことが指定されます。REPLACE オプションでは、プリンター定義名がすでにレジストリにある名前と一致する場合、そのプリンター定義とレジストリのプリンター定義を置き換えることが指定されます。プリンター定義名が一致しない場合は、新しいプリンター定義がレジストリに追加されます。

```
proc prtdef data=gsview list replace;
run;
```

ログ

例のコード 43.2　GhostView プリンター定義後の SAS ログ

```
10   data gsview;
11   name = "Ghostview";
12   desc = "Print Preview with Ghostview";
13   model= "PostScript Level 2 (Color)";
14   viewer = 'ghostview %s';
15   device = "Dummy";
16   dest = " ";

NOTE: The data set WORK.GSVIEW has 1 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds


17   proc prtdef data=gsview list replace;
18   run;

NOTE: Printer Ghostview created.
NOTE: 1 printer definitions added to the registry.
NOTE: 0 printer definitions modified in the registry.
NOTE: 0 printer definitions deleted from the registry.
NOTE: PROCEDURE PRTDEF used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
```

例 3: すべてのユーザーが使用可能な単一プリンター定義の作成

要素:
PROC PRTDEF ステートメントオプション
DATA=
USESASHELP

制限事項:
USESASHELP オプションを使用するには、Sashelp カタログへの書き込み権限が必要です。

詳細

この例では、次を指定して、Tektronix Phaser 780 プリンターおよび Ghostview 印刷プレビューアーの定義を作成します。

- 下余白を 1 インチに設定
- フォントサイズを 14 ポイントに設定
- 用紙のサイズを A4 に設定

プログラム

```
data tek780;
  name = "Tek780";
  desc = "Test Lab Phaser 780P";
  model = "Tek Phaser 780 Plus";
  device = "PRINTER";
  dest = "testlab3";
  preview = "Ghostview";
  units = "cm";
  bottom = 2.5;
  fontsize = 14;
  papersiz = "ISO A4";
run;

proc prtdef data=tek780 usesashelp;
run;
```

プログラムの説明

TEK780 データセットを作成し、プリンター出力先について適切な情報を指定します。 TEK780 データセットに含まれる変数の値には、プリンター定義の作成に必要な情報が含まれます。この例では、割り当てステートメントを使用してこれらの変数を割り当てます。NAME 変数では、プリンター定義データレコードのその他の属性に関連付けられるプリンター名が指定されます。DESC 変数では、プリンターの説明が指定されます。MODEL 変数では、このプリンターの定義時に使用するプリンタープロトタイプが指定されます。DEVICE 変数では、プリンターへの出力送信時に使用する I/O デバイスの種類が指定されます。DEST 変数では、プリンターの出力先が指定されます。PREVIEW 変数では、印刷プレビューに使用するプリンターが指定されます。UNITS 変数では、余白変数をセンチとインチのどちらで測定するかが指定されます。BOTTOM 変数では、UNITS 変数で指定した単位でデフォルト下余白が指定されます。FONTSIZE 変数では、デフォルトフォントのポイントサイズが指定されます。PAPERSIZ 変数では、デフォルトの用紙のサイズが指定されます。

```
data tek780;
  name = "Tek780";
  desc = "Test Lab Phaser 780P";
```

```
  model = "Tek Phaser 780 Plus";
  device = "PRINTER";
  dest = "testlab3";
  preview = "Ghostview";
  units = "cm";
  bottom = 2.5;
  fontsize = 14;
  papersiz = "ISO A4";
run;
```

TEK780 プリンター定義を作成し、すべてのユーザーが定義を使用できるようにします。 DATA=オプションでは、TEK780 が入力データセットとして指定されます。USESASHELP オプションでは、すべてのユーザーがプリンター定義を使用できるようにすることが指定されます。

```
proc prtdef data=tek780 usesashelp;
run;
```

例 4: プリンター定義の追加、変更、削除

要素: PROC PRTDEF ステートメントオプション
DATA=
LIST

詳細

この例では、次を行います。

- プリンター定義を 2 つ追加
- プリンター定義を変更
- プリンター定義を 2 つ削除

プログラム

```
data printers;
length name   $ 80
  model  $ 80
  device $ 8
  dest   $ 80
  opcode $ 3
  ;
input opcode $& name $& model $& device $& dest $&;
datalines;
add  Color PostScript   PostScript Level 2 (Color)      DISK    sasprt.ps
mod  LaserJet 5         PCL 5 (DeltaRow)                DISK    sasprt.pcl
del  Gray Postscript    PostScript Level 1 (Gray Scale) DISK    sasprt.ps
```

```
del  test            PostScript Level 2 (Color)      DISK    sasprt.ps
add  ColorPS         PostScript Level 2 (Color)      DISK    sasprt.ps
;

proc prtdef data=printers replace list;
run;
```

プログラムの説明

PRINTERS データセットを作成し、プリンター定義に対して実行するアクションを指定します。 PRINTERS データセットに含まれる変数の値には、プリンター定義の作成に必要な情報が含まれます。MODEL 変数では、このプリンターの定義時に使用するプリンタープロトタイプが指定されます。DEVICE 変数では、プリンターへの出力送信時に使用する I/O デバイスの種類が指定されます。DEST 変数では、プリンターの出力先が指定されます。OPCODE 変数では、プリンター定義に対して実行するアクション(追加、削除または変更)が指定されます。最初の Add 操作では、SAS レジストリの Color PostScript に対して新しいプリンター定義が作成されます。2 番目の Add 操作では、SAS レジストリの ColorPS に対して新しいプリンター定義が作成されます。Mod 操作では、レジストリの LaserJet 5 に対する既存プリンター定義が変更されます。Del 操作では、プリンター定義をレジストリから削除します。**&**では、2 つ以上のブランクで文字値を区切ることが指定されます。これにより、name と model の値にブランクを含めることが可能になります。

```
data printers;
length name   $ 80
  model  $ 80
  device $ 8
  dest   $ 80
  opcode $ 3
  ;
input opcode $& name $& model $& device $& dest $&;
datalines;
add  Color PostScript   PostScript Level 2 (Color)      DISK    sasprt.ps
mod  LaserJet 5         PCL 5 (DeltaRow)                DISK    sasprt.pcl
del  Gray Postscript    PostScript Level 1 (Gray Scale) DISK    sasprt.ps
del  test            PostScript Level 2 (Color)      DISK    sasprt.ps
add  ColorPS         PostScript Level 2 (Color)      DISK    sasprt.ps
;
```

複数のプリンター定義を作成し、SAS ログに書き込みます。 DATA=オプションでは、プリンター属性を含む入力データセット PRINTERS が指定されます。PROC PRTDEF では、5 つのプリンター定義が作成され、そのうち 2 つが削除されました。LIST オプションでは、作成または置換されるプリンターのリストをログに書き込むことが指定されます。

```
proc prtdef data=printers replace list;
run;
```

ログ

例のコード 43.3　プリンター変更および削除後の SAS ログ

```
15  data printers;
16  length name   $ 80
17     model  $ 80
18     device $ 8
19     dest   $ 80
20     opcode $ 3
21     ;
22  input opcode $& name $& model $& device $& dest $&;
23  datalines;

NOTE: The data set WORK.PRINTERS has 5 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds


29  ;
30  proc prtdef data=printers list replace;
31  run;

NOTE: Printer Color PostScript modified.
NOTE: Printer LaserJet 5 modified.
NOTE: Printer Gray Postscript deleted.
NOTE: Printer test deleted.
NOTE: Printer ColorPS modified.
NOTE: PROCEDURE PRTDEF used (Total process time):
      real time          0.04 seconds
      cpu time           0.03 seconds
```

例 5: 単一プリンター定義の削除

要素:　PROC PRTDEF ステートメントオプション
　　DELETE

詳細

この例では、レジストリからプリンターを削除する方法を示します。

プログラム

```
data deleteprt;
name='printer1';
run;

proc prtdef data=deleteprt delete list;
```

```
run;
```

プログラムの説明

DELETEPRT データセットを作成します。 NAME 変数には、削除するプリンターの名前が含まれます。

```
data deleteprt;
name='printer1';
run;
```

レジストリからプリンター定義を削除し、削除したプリンターをログに書き込みます。 DATA=オプションで、DELETEPRT が入力データセットとして指定されます。PROC PRTDEF で、SAS レジストリに対してプリンター定義が作成されます。DELETE で、プリンターの削除が指定されます。LIST で、削除したプリンターをログに書き込むことが指定されます。

```
proc prtdef data=deleteprt delete list;
run;
```

ログ

例のコード 43.4　単一プリンター削除後の SAS ログ

```
45   data deleteprt;
46     name='printer1';
47     run;

NOTE: The data set WORK.DELETEPRT has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.00 seconds


48     proc prtdef data=deleteprt delete list;
49     run;

NOTE: Printer printer1 deleted.
NOTE: 0 printer definitions added to the registry.
NOTE: 0 printer definitions modified in the registry.
NOTE: 1 printer definitions deleted from the registry.
NOTE: PROCEDURE PRTDEF used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

44

PRTEXP プロシジャ

概要: PRTEXP プロシジャ

PRTEXP プロシジャの機能

PRTEXP プロシジャを使用すると、個々のユーザーまたはサイトのすべての SAS ユーザーのプリンター定義を SAS レジストリから複製、変更、および作成することができます。次に、PROC PRTEXP で、その属性が SAS ログか SAS データセットに書き込まれます。PROC PRTEXP によってレジストリの SASHELP 部分または SAS レジストリ全体でこれらの属性を検索することを指定できます。

プリンター定義を SAS データセットに書き込むと、後で複製、変更できます。次に、PROC PRTDEF を使用して、入力データセットから、SAS レジストリにプリンター定義を作成できます。PROC PRTDEF ならびにプリンター定義の作成に使用する変数および属性の完全な説明については、“プリンター定義の作成に使用される入力データセット変数” (1540 ページ)を参照してください。

関連項目

43 章, “PRTDEF プロシジャ,” (1537 ページ)

構文: PRTEXP プロシジャ

ヒント: SELECT ステートメントも EXCLUDE ステートメントも使用しなければ、すべてのプリンターが出力に含まれます。

PROC PRTEXP <*options*>;
SELECT*printer(s)*;
EXCLUDE*printer(s)*

ステートメント	タスク	例
“PROC PRTEXP ステートメント”	SAS レジストリからのプリンター属性の取得	Ex. 1, Ex. 2
“EXCLUDE ステートメント”	指定プリンター以外のすべてのプリンターのプリンター属性の取得	
“SELECT ステートメント”	指定プリンターのプリンター属性の取得	Ex. 1, Ex. 2

PROC PRTEXP ステートメント

プリンター定義を複製、変更、作成します。

例: “例 1: SAS ログへの属性の書き込み” (1558 ページ)
“例 2: SAS データセットへの属性の書き込み” (1559 ページ)

構文

PROC PRTEXP <*options*>;

オプション引数

USESASHELP
SAS でレジストリの SASHELP 部分でのみプリンター定義を検索することを指定します。

デフォルト　デフォルトでは、レジストリの SASUSER と SASHELP の両部分でプリンター定義が検索されます。

OUT=*SAS-data-set*
プリンター定義を含む SAS データセットを指定します。

OUT=*SAS-data-set* オプションで指定したデータセットは、各プリンターを定義するために PROC PRTDEF の DATA=*SAS-data-set* オプションで指定したデータセットと同じ種類です。

デフォルト　OUT=*SAS-data-set* を指定しなければ、各プリンターの定義に必要なデータは SAS ログに書き込まれます。

EXCLUDE ステートメント

出力に情報を表示しないプリンターの名前を指定します。

構文

EXCLUDE*printer(s)*;

必須引数

printer(s)
出力に関連情報を含めないプリンターを 1 つ以上指定します。

SELECT ステートメント

出力に情報を含めるプリンターの名前を指定します。

例:
“例 1: SAS ログへの属性の書き込み” (1558 ページ)
“例 2: SAS データセットへの属性の書き込み” (1559 ページ)

構文

SELECT *printer(s)*;

必須引数

printer(s)
出力に関連情報を含めるプリンターを 1 つ以上指定します。

例: PRTEXP プロシジャ

例 1: SAS ログへの属性の書き込み

要素: PROC PRTEXP ステートメントオプション
USESASHELP
SELECT ステートメント

詳細

この例では、プリンターの定義に使用する属性を SAS ログに書き込む方法を示します。

プログラム

関連情報が必要なプリンターを指定し、レジストリの SASHELP 部分のみ検索することを指定して、SAS ログに情報を書き込みます。 SELECT ステートメントで、プリンター PostScript の定義に使用する属性情報を出力に含めることが指定されます。USESASHELP オプションで、SASHELP レジストリでのみ PostScript のプリンター定義を検索することが指定されます。OUT=オプションによる SAS データセットの指定を行わなかったので、各プリンターの定義に必要なデータは SAS ログに書き込まれます。

```
proc prtexp usesashelp;
select postscript;
run;
```

ログ

例のコード 44.1　レジストリの SASHELP 部分からプリンター情報を抽出した後の SAS ログ

```
379  proc prtexp usesashelp;
380  select postscript;
381  run;

NAME:    PostScript
MODEL:   PostScript Level 1 (Color)
DEVICE:  DISK
DEST:    sasprt.ps
HOSTOPT:
PROTOCOL:
TRANTAB:
DESC:    Generic PostScript Level 1 Printer
PREVIEW: Adobe Reader
VIEWER:
PAPERSIZ:
PAPERTYP:
PAPERIN:
PAPEROUT:
RES:     300 DPI
TOP:     0.50
LEFT:    0.50
RIGHT:   0.50
BOTTOM:  0.50
UNITS:   IN
TYPEFACE: <MTmonospace>
WEIGHT:  Normal
STYLE:   Regular
CHARSET: Western
FONTSIZE: 8.00
LRECL:   .
```

例 2: SAS データセットへの属性の書き込み

要素:

PROC PRTEXP ステートメントオプション
　OUT=

SELECT ステートメント

詳細

この例では、PROC PRTDEF がプリンター PCL4、PCL5、PCL5E、PCLC の定義に使用するデータを含む SAS データセットを作成する方法を示します。

プログラム

関連情報が必要なプリンターを指定し、PRDVTER データセットを作成します。
SELECT ステートメントで、プリンター PCL4、PCL5、PCL5E、PCLC が指定されます。OUT=オプションで、SAS データセット PRDVTER が作成されます。これには、PROC PRTDEF がプリンター PCL4、PCL5、PCL5E、PCLC の定義に使用するものと同じ属性が含まれます。USESASHELP を指定しなかったので、SAS では SASUSER と SASHELP の両レジストリが検索されます。

```
proc prtexp out=PRDVTER;
  select pcl4 pcl5 pcl5e pcl5c;
run;

proc print data=prdvter;
run;
```

出力

次のデータセットは 26 の変数および 4 つのオブザベーションを含む Prdter データの表示の一部です。

アウトプット 44.1　Prdvter の出力データセット

The SAS System

Obs	DEST	HOSTOPT	DESC	VIEWER	NAME	MODEL	PREVIEW	TYPEFACE
1	sasprt.pcl		Generic PCL 4 Printer		PCL4	PCL 4	Adobe Reader	
2	sasprt.pcl		Generic PCL 5 Printer		PCL5	PCL 5 (DeltaRow)	Adobe Reader	
3	sasprt.pcl		Generic PCL 5 RGB Color Printer with Alpha Blending		PCL5c	PCL 5rgba (DeltaRow)	Adobe Reader	
4	sasprt.pcl		Generic PCL 5e Printer		PCL5e	PCL 5e (DeltaRow)	Adobe Reader	

45

PWENCODE プロシジャ

概要: PWENCODE プロシジャ

PWENCODE プロシジャの機能

PWENCODE プロシジャを使用すると、パスワードをエンコードできます。エンコーディングはデータを難読化します。暗号化とは異なり、エンコーディングはデータの可逆順列であり、キーは使用しません。

エンコードされたパスワードは、SAS 9 レガシーメタデータサーバーとワークスペースサーバー、SAS Viya プラットフォームの計算サーバーとサービス、CAS サーバーとサービス、およびリレーショナルデータベース管理システム(RDBMS)にアクセスする SAS プログラムで、プレーンテキストパスワードの代わりに使用できます。

概念: PWENCODE プロシジャ

エンコードされたパスワードの SAS プログラムでの使用

PROC PWENCODE でパスワードをエンコードすると、文字列がエンコードされていることを示すタグが出力文字列に挿入されます。タグの例は**{sas001}**です。タグでは、エンコーディング方式が示されます。SAS サーバーと SAS/ACCESS エンジンでタグが認識され、使用前に文字列がデコードされます。パスワードをエンコードすると、プレーンテキストでパスワードを指定しなくても、SAS プログラムを書けます。

注: これには、英数字、スペースおよび特殊文字が含まれます。PROC PWENCODE パスワードには最大で 512 文字まで使用できます。ただし、データセットパスワードは SAS 命名規則に従うものとします。詳細については、*“SAS 名” (SAS プログラマガイド: エッセンシャル)*を参照してください。

エンコードされたパスワードが SAS ログにプレーンテキストで書き込まれることはありません。かわりに、SAS ログではパスワードの各文字が X に置き換えられます。

エンコーディングと暗号化

エンコーディング技術はパスワードを隠しており、このアプローチは悪意のない偶然のパスワード参照を防ぐためのものです。エンコードを行うと、一部のテーブルルックアップ形式によって、ある文字セットが別の文字セットに変換されます。SAS001–SAS005 エンコーディング方式は、エンコーディング方式と見なされます。

それとは対照的に、暗号化では、演算処理および(通常は)“key”値の使用による、ある形式から別の形式へのデータ変換が必要になります。一般に、暗号化はエンコードよりも解読が困難です。

PROC PWENCODE で指定された SAS003、SAS004、および SAS005 方式は、業界標準に準拠した暗号化技術を指定します。これらのオプションは、より長い暗号化キー(たとえば、256 ビット)をサポートします。AES 暗号化アルゴリズムには、より破りにくいパスワードを作成するためのソルティングや複数回の反復が用意されています。

PROC PWENCODE のエンコーディング方式は、“エンコーディング方式 ” (1563 ページ)に示されています。

パスワード保護はセキュリティ戦略の重要な部分ですが、すべてのデータセキュリティニーズに対してパスワード保護だけに頼るべきではありません。破る意思を固めた、知識のある攻撃者がパスワードを破る可能性があります。データは、ファイルシステムのアクセス権、その他のアクセスコントロールメカニズム、休止中や移動中のデータの暗号化など、他のセキュリティ管理によっても保護される必要があります。

エンコーディング方式

次のエンコーディング方式がサポートされています。

表 45.1 サポートされているエンコーディング方式

エンコーディング方式	使用されるデータ暗号化アルゴリズム	エンコードされたパスワード/キー説明
sas001	なし	Base64 を使用してパスワードをエンコードします。
sas002、(**sasenc** でも指定可)	SASProprietary は SAS ソフトウェアに含まれています。	32 ビットの固定キーを使用します。
sas003	AES (Advanced Encryption Standard)	256 ビット固定キーと 16 ビットランダムソルト値を使用します。
sas004	AES (Advanced Encryption Standard)	256 ビット固定キーと 64 ビットランダムソルト値を使用します。
sas005	AES (Advanced Encryption Standard)	256 ビット固定キー、64 ビットランダムソルト値を使用し、追加の反復処理のためにハッシュされています。

SAS003、SAS004、および SAS005 でエンコードされたパスワードは、AES 暗号化と 256 ビットの固定キーを使用します。また、エンコーディング方式にはランダムソルト値が適用されます。そのため、PROC PWENCODE を使用して同じパスワードをエンコードするたびにソルト値は異なり、その結果、エンコードされたパスワードも異なります。ランダムソルト値に加えて、追加の反復のために SAS005 がハッシュされます。SAS005 は、PROC PWENCODE で使用される最高レベルのエンコーディングです。

パスワードを使用した SAS データセット暗号化の SASProprietary は、データの 32 ビットローリングキーエンコーディングの一部として、SAS データセットに保存されているパスワードの一部を使用する暗号です。この暗号化は、中レベルのセキュリティを提供します。今日のコンピューターの速度では、有効なパスワード値の

2,563,160,682,591 の可能な組み合わせに対するブルートフォース攻撃を受ける可能性があり、その多くは同じ 32 ビットキーを生成する必要があります。

SASProprietary データセットパスワード暗号化は、主にレガシー SAS システムの顧客データで使用されます。たとえば、SASProprietary は、SAS Metadata Server、SAS Viya 3.5 以前のリリースのワークスペースサーバー 、または SAS Viya プラットフォームの SAS Compute Server の SAS 9 データで使用できます。

構文: PWENCODE プロシジャ

PROC PWENCODEIN='*password*'< OUT=*fileref*> < METHOD=*encoding-method*>;

ステートメント	タスク	例
“PROC PWENCODE ステートメント”	パスワードのエンコード	Ex. 1, Ex. 2, Ex. 3, Ex. 4

PROC PWENCODE ステートメント

パスワードをエンコードします。

例:

“例 1: パスワードのエンコード” (1566 ページ)
“例 2: エンコードされたパスワードのペーストバッファーへの保存” (1567 ページ)
“例 3: パスワードのエンコーディングに Method=SAS003 を指定” (1568 ページ)
“例 4: パスワードのエンコーディングに Method=SAS005 を指定” (1569 ページ)

構文

PROC PWENCODEIN='*password*'< OUT=*fileref*> < METHOD=*encoding-method*>;

必須引数

IN='*password*'
エンコードするパスワードを指定します。パスワードには最大で 512 文字まで使用できます。これには、英数字、スペースおよび特殊文字が含まれます。

注: データセットパスワードは SAS 命名規則に従うものとします。SAS 命名規則を順守するものであれば、IN=*password* は SAS データセットにも使用できます。詳細については、“SAS 名” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

パスワードに埋め込み単一引用符または二重引用符が含まれている場合は、文字定数引用の標準 SAS ルールを使用します。これらの規則は“定数” (*SAS プログラマガイド: エッセンシャル*)で確認できます。

注: エンコードされたパスワードの各文字は、SAS ログへの書き込み時に、X に置き換えられます。

参照項目: “例 1: パスワードのエンコード” (1566 ページ)

“例 2: エンコードされたパスワードのペーストバッファーへの保存” (1567 ページ)

“例 3: パスワードのエンコーディングに Method=SAS003 を指定” (1568 ページ)

“例 4: パスワードのエンコーディングに Method=SAS005 を指定” (1569 ページ)

オプション引数

OUT=*fileref*
出力文字列の書き込み先のファイル参照名を指定します。OUT=オプションを指定しなければ、出力文字列は SAS ログに書き込まれます。

注: グローバルマクロ変数

_PWENCODE

は、OUT=ファイル参照名に書き込まれた値、または SAS ログに表示された値に設定されます。

参照項目: “例 2: エンコードされたパスワードのペーストバッファーへの保存” (1567 ページ)

METHOD=*encoding-method*
エンコーディング方式を指定します。*encoding-method* でサポートされている値は次のとおりです。

SAS001
Base64 を使用してパスワードをエンコードします。

SAS002
32 ビットの固定キーを使用します。SAS 独自の暗号化を使用します。

別名 SASENC

デフォルト SAS002

SAS003
256 ビット固定キーと 16 ビットランダムソルト値を使用します。AES 暗号化を使用します。

SAS004
256 ビット固定キーと 64 ビットランダムソルト値を使用します。AES 暗号化を使用します。

SAS005
256 ビット固定キー、64 ビットランダムソルト値を使用し、追加の反復処理のためにハッシュされます。AES 暗号化を使用します。

METHOD=オプションを指定しない場合、デフォルトのエンコーディング方式が使用されます。ほとんどの場合、デフォルトの方式は **sas002** です。**SAS002** も、無効なメソッドを指定した場合に使用されるデフォルトのメソッドです。

参照項目:
“例 1: パスワードのエンコード” (1566 ページ)

“例 2: エンコードされたパスワードのペーストバッファーへの保存” (1567 ページ)

“例 3: パスワードのエンコーディングに Method=SAS003 を指定” (1568 ページ)

“例 4: パスワードのエンコーディングに Method=SAS005 を指定” (1569 ページ)

例: PWENCODE プロシジャ

例 1: パスワードのエンコード

要素: IN=引数

詳細

この例では、パスワードをエンコードし、エンコードされたパスワードを SAS ログに書き込む単純なケースを示します。

プログラム

パスワードをエンコードします。

```
proc pwencode in='my password';
run;
```

ログ

SAS ログではパスワードの各文字が X に置き換えられることに注意してください。

```
19   proc pwencode in=XXXXXXXXXXXXX;
20   run;


{SAS002}DBCC571245AD0B31433834F80BD2B99E16B3C969

NOTE: PROCEDURE PWENCODE used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
```

例 2: エンコードされたパスワードのペーストバッファーへの保存

要素: IN=引数
OUT=オプション
CLIPBRD アクセスメソッドを指定した FILENAME ステートメント

DETAILS

この例では、エンコードされたパスワードをペーストバッファーに保存します。その場合は、エンコードされたパスワードを、別の SAS プログラムや、認証ダイアログボックスのパスワードフィールドに貼り付けられます。

プログラム

```
filename clip clipbrd;

proc pwencode in='my password' out=clip;
run;
```

プログラムの説明

CLIPBRD アクセスメソッドを使用してファイル参照名を宣言します。

```
filename clip clipbrd;
```

パスワードをエンコードし、ペーストバッファーに保存します。 OUT=オプションによって、エンコードされたパスワードが、前のステートメントで宣言したファイル参照名に保存されます。

```
proc pwencode in='my password' out=clip;
run;
```

ログ

SAS ログではパスワードの各文字が X に置き換えられることに注意してください。

```
24
25   filename clip clipbrd;
26    proc pwencode in=XXXXXXXXXXXXX out=clip;
27   run;

NOTE: PROCEDURE PWENCODE used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

例 3: パスワードのエンコーディングに Method=SAS003 を指定

要素: IN=引数
METHOD=引数

詳細

この例では、**sas003** エンコーディング方式を使用してパスワードをエンコードし、エンコードされたパスワードを SAS ログに書き込む単純なケースを示します。SAS003 では、パスワードのエンコーディングに 16 ビットソルトを使用します。

プログラム

SAS003 を使用してパスワードをエンコードします。 エンコードされたパスワードは、16 ビットのランダムソルトを含む 256 ビットキーです。

```
proc pwencode in='mypassword' method=sas003;
run;
```

ログ

SAS ログではパスワードの各文字が X に置き換えられることに注意してください。SAS003 エンコーディングは AES 暗号化+16 ビットソルトを使用します。SAS003 はランダムなソルティングを使用するため、次のコードを実行するたびに異なるパスワードが生成されます。

```
8   proc pwencode in=XXXXXXXXXXXXX method=sas003;
29   run;

{SAS003}4837B146585CED2C9FED14A3C946D68E4389

NOTE: PROCEDURE PWENCODE used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds
```

例 4: パスワードのエンコーディングに Method=SAS005 を指定

要素: IN=引数
METHOD=引数

詳細

この例では、**sas005** エンコーディング方式を使用してパスワードをエンコードし、エンコードされたパスワードを SAS ログに書き込む単純なケースを示します。SAS005 は、64 ビットのランダムソルトを使用してパスワードをエンコーディングする 256 ビット固定キーを使用します。

プログラム

SAS005 を使用してパスワードをエンコードします。

```
proc pwencode in='mypassword' method=sas005;
run;
```

ログ

SAS ログではパスワードの各文字が X に置き換えられることに注意してください。SAS005 エンコーディングでは、256 ビットの固定キーと 64 ビットのランダムソルト値で AES 暗号化を使用します。SAS005 は、SHA-256 ハッシュアルゴリズムを使用して保存されたパスワードのセキュリティを強化し、追加反復に対してハッシュ化されます。SAS005 はランダムなソルティングを使用するため、次のコードを実行するたびに異なるパスワードが生成されます。

```
230  proc pwencode in=XXXXXXXXXXXX method=sas005;
231  run;


{SAS005}ADD8AB7108595A7D1A69190D78CDFE6145C1EB849CC7A43D

NOTE: PROCEDURE PWENCODE used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

46

PYTHON プロシジャ

概要: PYTHON プロシジャ

PYTHON プロシジャの機能

Python は、インタープリター型の高水準汎用プログラミング言語です。PYTHON プロシジャを使用すると、SAS コード内の Python プログラミング言語からステートメントを実行できます。外部の Python スクリプトから Python ステートメントをサブミットしたり、Python ステートメントを直接入力したりできます。

PROC PYTHON を使用すると、次のタスクを実行できます。

- SAS セッションで Python コードを実行する
- Python ステートメントでほとんどの SAS 関数を呼び出す
- Python から SAS コードをサブミットする
- SAS データセットまたはビューと Pandas DataFrame の間でデータを移動する
- SAS マクロ変数と Python 変数の間で値を転送する
- PYTHON ステートメント内で PROC FCMP 関数を呼び出す
- SAS Studio の結果に Pyplot グラフィックを表示する

> **注:** グラフィックを表示するには、matplotlib.pyplot モジュールをインストールする必要があります。

Python コードをバッチスクリプトで実行することもできます。詳細については、“Introduction” (*Using the batch Plug-In for the SAS Viya Platform Command-Line Interface*)を参照してください。

概念: PYTHON プロシジャ

PROC PYTHON の仕組み

PROC PYTHON は、SAS Compute Server プロセスからサブプロセスを作成します。このサブプロセスは、ユーザーが選択した Python デプロイメントを使用します。

PROC PYTHON を使用すると、Python コードを SUBMIT ブロック内の 1 つのプログラム全体として、または INTERACTIVE ブロック内で一度に 1 行としてサブミットできます。詳細については、SUBMIT ステートメント (1578 ページ)またはINTERACTIVE ステートメント (1579 ページ)を参照してください。

SUBMIT および ENDSUBMIT ステートメント内で Python コードのブロックをサブミットすると、次の動作が観察されます。

- ステートメントでエラーまたは例外が発生した場合、コードブロックの処理はエラーの時点で停止します。
- 複合ステートメントは、for ループや while ステートメントなどの他のステートメントを含む Python ステートメントです。複合ステートメント内の継続ステートメントは、すべて同じ量(通常は 4 つのスペース)だけインデントされます。複合ステートメントの終わりは、複合ステートメントの最上位ステートメントと同じレベルで新しいステートメントが始まると発生します。複合ステートメントの終わりと次の Python ステートメントの間に空白行は必要ありません。

PROC PYTHON サブプロセスを対話モードで実行すると、Python ステートメントは対話構文を使用して処理されます。これにより、次の動作が発生します。

- ステートメントの結果がエラーまたは例外になった場合、Python インタープリターはすぐに終了するのではなく、後続のステートメントの処理を続行します。
- Python コードの対話型ブロック内の複合ステートメントでは、継続行を終了するために空白行が必要です。したがって、Python コードの対話型ブロックでは、複合ステートメントの後に空白行を追加する必要があります。

PYTHON プロシジャを使用すると、複数のプロシジャ呼び出しで同じサブプロセスを使用できるため、後続の PYTHON プロシジャ呼び出しでオブジェクトまたはデータを使用できます。Python サブプロセスを終了し、必要に応じて、PYTHON プロシジャ呼び出しの開始時または終了時に新しいサブプロセスを作成することを選択できます。詳細については、“例 2: PYTHON プロシジャの再起動と終了の影響を観察する” (1590 ページ)を参照してください。

PYTHON サブプロセスが作成されると、PYTHON プロシジャの一部であるモジュールが自動的にインポートされます。このモジュールは SAS と呼ばれ、SAS と Python インスタンス間の対話を可能にします。SAS モジュールは、SAS Compute Server と Python インスタンス間の対話に使用できるコールバックメソッドを多数提供します。これらのコールバックメソッドにより、Python と SAS の間で変数を共有し(SAS.symget または SAS.symput)、SAS データセットと Pandas DataFrame の間でデータを移動し(SAS.sd2df または SAS.df2sd)、SAS および FCMP 関数を呼び出し(SAS.sasfnc)、Python ステートメント内で SAS コードをサブミットします(SAS.submit)。これらのコールバックメソッドを使用すると、必要に応じて SAS と Python プログラミングを組み合わせたワークフローを作成できます。詳細については、“PROC PYTHON コールバックメソッドの使用” (1582 ページ)を参照してください。

PROC PYTHON の構成

PROC PYTHON がシステム上の Python と対話できるようにするには、いくつかの構成要件があります。ユーザーまたは SAS 管理者は、次の 2 つの環境変数を構成する必要があります。

- PROC_PYPATH は、Python 実行可能ファイルを指しています。Python 実行可能ファイルを含む Python デプロイメントは、通常、SAS Compute Server からアクセス可能なファイルシステムにマウントされます。Python 実行可能ファイルは、PYTHON プロシジャによってトリガーされる PYTHON サブプロセスを作成します。
- PROC_M2PATH は、配置の一部である sas2py.py ファイルの場所を指定します。この環境変数は、最初はこのファイルのデフォルトの場所を指しています。sas2py.py ファイルには、Python デプロイメントによってインポートされる SAS モジュールが含まれています。このファイルは、計算サーバーセッションと対話するコールバックメソッドのインターフェイスを 提供します。

詳細については、“Configure SAS to Run the Python Language” (*SAS Viya Platform: Programming Run-Time Servers*)を参照してください。

SAS Viya 製品は、デフォルトではロックダウン状態で配置されます。その結果として、PROC PYTHON は無効になります。PROC PYTHON を有効にするには、LOCKDOWN ステートメントで PYTHON および SOCKET アクセスメソッドに ENABLE_AMS=オプションを使用します。詳細については、次のトピックを参照してください。

- “Security Features” (*SAS Viya Platform: Programming Run-Time Servers*)の LOCKDOWN システムオプションと LOCKDOWN ステートメント
- “LOCKDOWN” (*SAS プログラマガイド: エッセンシャル*)

SUBMIT と INTERACTIVE Python コードの比較

SUBMIT および ENDSUBMIT ステートメントを使用して Python コードをサブミットすると、Python ステートメントは、プロシジャ呼び出しの最後に到達したときにコードのブロックとしてサブミットされます。すべての Python ステートメントが処理された後、出力が SAS ログに表示されます。SAS コールバックメソッドを含めると、それらの呼び出しからの出力が、Python コードからの出力の前に SAS ログに表示されます。

INTERACTIVE および ENDINTERACTIVE ステートメントを使用して対話式に Python コードをサブミットすると、各 Python ステートメントは入力時に実行されます。Python 処理からの出力には、SAS モジュール(SAS.symget など)を使用する任意のコールバックメソッドからの出力が散在しています。Python コードを対話式に実行すると、ログに Python と SAS の出力が時系列で表示されます。これは、コマンドラインから Python コードを実行するのと似ており、Python コードのデバッグに役立ちます。

SAS Cloud Analytic Services への接続の確立

PROC PYTHON から SAS Cloud Analytic Services(CAS)にアクセスするには、PythonSWAT パッケージを使用する必要があります。その場合、次の 2 つの環境変数を使用する必要があります。

- SAS_SERVICES_TOKEN - SAS Viya へのトークンベースの認証を提供します
- SSLCALISTLOC - これは、Python ランタイムと SAS Viya 間の安全な通信のための SSL 証明書を提供します。

注: このステップが機能するためには、システム上で Python と対話できるように PROC PYTHON が有効になっていることを確認してください。詳細については、“PROC PYTHON の構成” (1573 ページ)を参照してください。

次のコード例では、これらの変数を swat.CAS クラスで使用し、ホスト名、ポート、および認証トークンを渡して接続を作成する方法を示します。

```
cas casauto;
proc python;
submit;
import swat, os
os.environ['CAS_CLIENT_SSL_CA_LIST']=os.environ['SSLCALISTLOC']
sascas = swat.CAS(hostname="sas-cas-server-default-client",port=5570, password=os.environ['SAS_SERVICES_TOKEN'])
r = sascas.caslibinfo()
SAS.show(r['CASLibInfo'])
endsubmit;
run;
cas casauto terminate
```

デフォルトの sas-cas-server-default-client 以外の CAS サーバーまたは CAS ホスト名を使用している場合は、デフォルトではなくそのホスト名を使用してください。詳細については、ホストの ID の確認を参照してください。

この例では、caslibinfo()メソッドを使用して、アクセス権を持つ caslibs にアクセスします。show() メソッドは、次の表のように、出力タブでアクセスできる CAS のライブラリのリストを提供するために使用されます。

Output

Obs	Name	Type	Description	Path	Subdirs
1	AIoTPgMeta	PATH	Stores quality analytic suite postgres tables.	/cas/data/caslibs/aiotPgMeta/	0
2	alm202505	PATH	ALM Data	/cas/data/caslibs/alm202505/	1
3	CASTestTmp	PATH	DDT created caslib	/bigdisk/lax/castest/	1
4	CASUSER(user1)	PATH	Personal File System Caslib	/cas/data/caslibs/casuserlibraries/user1/	1
5	costAndProfitability	PATH	Stores output tables for different cost and profitability management models.	/cas/data/caslibs/costAndProfitability/	0

関連項目

SAS Scripting Wrapper for Analytics Transfer (SWAT)

構文: PYTHON プロシジャ

制限事項:

SAS がロック状態のときは、PYTHON プロシジャは使用できません。サーバー管理者は、このプロシジャがロックダウン状態でも使用できるように、このプロシジャを再有効化できます。詳細については、“Example 3: Enabling PROC PYTHON While in Lockdown Mode” (*SAS Viya Platform: Programming Run-Time Servers*)を参照してください。

PROC PYTHON< INFILE='*file-name*' | fileref> < RESTART> < TERMINATE> < TIMEOUT=*n*>;

<SUBMIT;>

Python statements

<ENDSUBMIT;>

RUN;

PROC PYTHON< INFILE='*file-name*' | fileref> < ECHO> < RESTART> < TERMINATE> < TIMEOUT=*n*>;

<INTERACTIVE;>

Python statements, submitted interactively

<ENDINTERACTIVE;>

RUN;

ステートメント	タスク	例
“PROC PYTHON ステートメント”	SAS コード内で Python ステートメントを実行するか、実行する Python ステートメントを含むファイルを指定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
“SUBMIT ステートメント”	Python ステートメントのブロックの先頭を特定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
“ENDSUBMIT ステートメント”	Python ステートメントのブロックの末尾を特定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5
“INTERACTIVE ステートメント”	対話型 Python ステートメントの開始を識別します。	Ex. 6
“ENDINTERACTIVE ステートメント”	対話型 Python ステートメントの終了を識別します。	Ex. 6

PROC PYTHON ステートメント

SAS セッションで Python ステートメントを実行します。

構文

形式 1: SUBMIT ブロック

PROC PYTHON< INFILE='*file-name*' | fileref> < RESTART> < TERMINATE> < TIMEOUT=*n*>;

形式 2: インタラクティブな Python ブロック

PROC PYTHON< INFILE='*file-name*' | fileref> < ECHO> < RESTART> < TERMINATE> < TIMEOUT=*n*>;

オプション引数

ECHO
対話式にサブミットされた Python コードの行を SAS ログに含めるように指定します。 コード行は、結果の出力の前に表示されます。

制限事項 このオプションは、Python コードを対話式にサブミットする場合にのみ適用されます。

INFILE='*file-name*' | fileref
SAS セッション内で実行する PYTHON ステートメントが含まれているソースファイルまたはファイル参照名を特定します。

要件 ファイル名を指定する場合、その名前を単一引用符か二重引用符で囲みます。

INFILE=オプションを使用する場合は、PYTHON スクリプトへのパスを指定する必要があります。

例 open_data.py で Python コードを使用する:

```
proc python infile='/user/myscripts/open_data.py';
```

Python コードを含むファイルのファイル参照名を定義する:

```
filename script 'my_script.py';

proc python infile=script;
```

RESTART
現在実行中の PYTHON サブプロセスを終了し、新しいサブプロセスを作成します。 その後、PYTHON サブプロセスは、PROC PYTHON への現在および後続の呼び出しに使用されます。

PYTHON コードの新しいブロックの先頭で RESTART を指定します。

例 proc python restart;
submit;
<Python statements>
endsubmit;
run;

TERMINATE
現在の PYTHON プロシジャ呼び出しが完了すると、PYTHON サブプロセスを停止します。次回の PYTHON プロシジャ呼び出しは、PYTHON サブプロセスの新しいインスタンスを開始します。

例 proc python terminate;
interactive;
<Python statements submitted individually>
endinteractive;
run;

TIMEOUT=*n*
Python 環境への接続試行時に PROC PYTHON が終了するまでの秒数を指定します。

デフォルト 60

注 TIMEOUT=のデフォルトは 2023.05 リリースで 60 になりました。

例 proc python timeout=30 <additional options>;
submit;
<Python statements>
endsubmit;
run;

SUBMIT ステートメント

Python コードのブロックの先頭を特定します。SUBMIT ステートメントと ENDSUBMIT ステートメントの間に、PYTHON ステートメントを入力します。

要件 各 SUBMIT ステートメントには、対応する ENDSUBMIT ステートメントが必要です。

構文

SUBMIT;

ENDSUBMIT ステートメント

このステートメントは単独で行に指定され、Python ステートメントのブロックの終わりを識別します。

構文

ENDSUBMIT;

INTERACTIVE ステートメント

対話式にサブミットする Python コードの開始を識別します。

別名: I

要件 各 INTERACTIVE ステートメントには、対応する ENDINTERACTIVE ステートメントが必要です。

構文

INTERACTIVE;

ENDINTERACTIVE ステートメント

このステートメントは単独で行に指定され、対話式にサブミットする Python コードの終わりを識別します。

構文

ENDINTERACTIVE;

使用法: PYTHON プロシジャ

SUBMIT ブロック内での PYTHON ステートメントのサブミット

PROC PYTHON 起動内の、SUBMIT ステートメントと ENDSUBMIT ステートメントの間で PYTHON ステートメントをサブミットできます。次のコードは、1 つの PYTHON print ステートメントを実行します。

```
proc python;
submit;
print("Hello from Python")
endsubmit;
run;
```

次のコードは、文字列をマクロ変数に割り当てます。この例では、マクロ変数 Mymacrovar が SAS.submit コールバックメソッドを使用して定義されます。マクロ変数 Myvar は、SAS.symput コールバックメソッドを使用して定義されています。両方のマクロ変数は、PROC PYTHON への呼び出しの外でアクセスできます。

```
proc python restart;
submit;
SAS.submit("%let mymacrovar=Hi there;")
SAS.symput('myvar',"This variable has global scope")
txt = SAS.symget("mymacrovar")
print(txt)
endsubmit;
run;

%put &=mymacrovar;
%put &=myvar;
```

アウトプット 46.1　PROC PYTHON のマクロ変数を使用した出力

```
89   %let mymacrovar=Hi there;
>>>
Hi there
>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time           1.17 seconds
      cpu time            0.07 seconds

90
91   %put &=mymacrovar;
MYMACROVAR=Hi there
92   %put &=myvar;
MYVAR=This variable has global scope
```

対話型ブロックでの複合 Python ステートメントのサブミット

複合 Python ステートメントは、1 つ以上のステートメントが初期ステートメントによって制御されるステートメントです。例としては、if ステートメント、while ステートメント、for ステートメントなどがあります。PROC PYTHON を対話モードで実行する場合は、最上位の複合ステートメントの末尾とそれに続くステートメントの間に空白行を追加する必要があります。また、最上位の複合ステートメントの継続行内には空白行を埋め込まないでください。これは、複合ステートメントが終了したことを Python パーサーに伝えるためです。現在のインデントレベルに一致する数のスペースのみを 含む行を含めることができます。これらの行は空白行のように見えますが、Python パーサーはそれらを複合ステートメント内の継続行として扱います。

次の例は、対話モードで for ステートメントをサブミットする方法を示しています。for ステートメントは、x の値にわたってループします。この x は、fruits という配列内の連続する値に割り当てられます。

```
proc python;
interactive;
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)

print('first statement after for loop')
endinteractive;
run;
```

for ループとそれに続くステートメントからの出力は次のとおりです。

アウトプット 46.2　For ループ出力

```
>>>
>>>
... ... apple
banana
cherry
>>>
first statement after for loop
>>>
>>>
```

複合ステートメントの後に空白行がない場合、無効な構文のエラーがログに記録されます。

アウトプット 46.3　複合ステートメントの後に空白行がない場合のエラー

```
>>> fruits = ["apple", "banana", "cherry"]
>>> for x in fruits:
...   print
... print('just after for-loop')
  File "<stdin>", line 3
    print('just after for-loop')
          ^
SyntaxError: invalid syntax
>>>
```

PROC PYTHON コールバックメソッドの使用

SAS と対話するコールバックメソッド

PROC PYTHON は、SAS セッションと PYTHON サブプロセス間の通信を可能にする SAS というモジュールを提供します。次のコールバックメソッドを使用することにより、Python コードブロック内で SAS コードをサブミットしたり、SAS 変数を操作したりできます。Python は大文字と小文字を区別する言語であるため、このリストに示されているように大文字を使用してコールバックメソッドを指定します。

SAS.pyplot(*plot*<, filename='matplot.svg'><, filepath=None><, filetype='svg'><, **kwargs>)
matplotlib.pyplot オブジェクトまたは同等の savefig(fname, format=None, **kwargs)メソッドを持つ任意のオブジェクトのプロットをレンダリングします デフォルトでは、画像は Work ディレクトリ内の matplot.svg という一時ファイルに書き込まれます。にあります。グラフィックファイルを保存するには、FILEPATH=引数を指定します。デフォルトのファイル名とは異なる代替名を指定するには、FILEPATH=引数を指定します。ファイル名とファイルパスの値を引用符で囲みます。PNG や GIF など、ODS がサポートしているファイル形式を指定することができます。ファイルの出力形式の詳細なリストについては、“Supported Image Formats for Output Destinations” (*SAS Output Delivery*

System: Procedures Guide)を参照してください。オプションで、SAS が savefig メソッドに渡す**kwargs パラメーターを指定します。詳細については、matplotlib.pyplot.savefig のドキュメントを参照してください。

注: 使用する Python パッケージに savefig メソッドが含まれていない場合は、SAS.renderImage コールバックメソッドを使用して、他の Python メソッドを使ってファイルに書き込んだ画像を表示することができます。

要件 matplotlib.pyplot モジュールをインポートして、プロットオブジェクトを作成および操作します。SAS 管理者に問い合わせて、このモジュールが利用可能かどうかを確認してください。

注 FILETYPE=および**kwargs 引数のサポートは、2023.03 で追加されました。また、このリリースでは、デフォルトの画像ファイルの種類が(PNG ではなく)SVG になりました。

例 参照: “例 6: Python プロットをレンダリングする” (1601 ページ).

SAS.renderImage(filename)
savefig 以外の Python メソッドを使用して作成された画像ファイルをレンダリングします。画像ファイルの完全修飾ファイル名を指定します。サポートされている画像形式は、ODS 出力でサポートされているものと同じです。詳細については、“Supported Image Formats for Output Destinations” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

注 このコールバックメソッドのサポートは、2023.03 で追加されました。

SAS.sasfnc("*function-name*"<, *argument-1*<, *argument-2*...>>)
2023.10 以降、このメソッドは指定された SAS または PROC FCMP 関数を呼び出します。関数に引数が必要な場合は、関数名の後にカンマ区切りのリストで指定します。

関数の結果を Python 変数に割り当てることができます。

例 SAS 日付値を含む変数 **mydate** に対して YEAR 関数を呼び出します。結果を **py_var** に割り当てます。

```
py_var = SAS.sasfnc("YEAR",mydate)
```

SAS.show(*object*,<title>,<count>,<kwargs>)
2025.03 以降、print()の代わりにこのメソッドを使用して、SA ログではなく結果(リスト)に情報を表示します。

object は、Pandas データフレーム、シリーズ、インデックス、ネイティブ Python 型(str、int、float、list、dict を含む)、または matplot プロットです。

title は、出力ページのタイトルになる title2 SAS ステートメントに使用されます。

count は、数値の種類以外の表示するオブジェクトのアイテムの数です。デフォルトは 5 です。

kwargs は、指定されている場合、これらは pyplot に渡されます。これは他の種類のオブジェクトには使用されません。

要件 matplotlib.pyplot モジュールをインポートして、プロットオブジェクトを作成および操作します。SAS 管理者に問い合わせて、このモジュールが利用可能かどうかを確認してください。

SAS.showMLA(*results*, <show_name={True|False}>)
2025.05 以降、このメソッドは SAS Python モデルからの結果を PYTHON プロシジャの結果に表示するために使用されます。

SAS Python モデル適合(model.details_)、または custom_results()メソッド(model.customize_results())からの結果を指定します。customize_results()メソッドの詳細については、SAS Viya: Managing Plots in Python Automatic Graphics を参照してください。

SHOW_NAME=は、出力オブジェクトの名前をタイトルテキストの一部として表示します。デフォルト設定は False です。このオプションを使用して、customize_results()の include_tables、exclude_tables、およびオーダーオプションで使用する出力オブジェクト名を決定します。

SAS.submit("*SAS code*")
Python ステートメントのブロック内でコールバックメソッドが発行されるときに、指定された SAS コードの文字列をサブミットして実行します。

注: サブミットされるコードは完全なプログラムである必要があります。完全なプログラムではない場合は、次のメッセージが表示されます。

警告: submit()で不完全なプログラムを検出しました。そのステップは完了しませんでした。

例 Test という SAS データセットを作成します。
```
SAS.submit("data test; x='python'; y=2; run;")
```

SAS.symget("*SAS-macro-variable*")
指定したマクロ変数の値を返します。ユーザーは、その値を Python 変数に割り当てることができます。

例 マクロ変数 Myvar の値を Python 変数 **py_var** に割り当てます。
```
py_var=SAS.symget("myvar")
```

SAS.symput("*SAS-macro-variable*", *value*)
値をマクロ変数に割り当てます。指定する値は、文字列、数値、または変数名にすることができます。

このコールバックメソッドで作成するマクロ変数には、グローバルスコープがあります。

ユーザーは、SAS.symput 呼び出しのリターンコードを Python 変数に割り当てることができます。

例 Python 変数 **py_var** の値をマクロ変数 Myvar に割り当てます。リターンコードを変数 **rc** に割り当てます。
```
rc=SAS.symput("myvar",py_var)
```

SAS.logMessage("message","message-type")
PROC Python から整形式のメッセージを SAS ログに書き込みます。メッセージの種類に応じて、このメソッドはメモ(青色)、警告(緑色)、またはエラー(赤色)を報告します。このメソッドは 2 つのパラメーターを取ります。

message	SAS ログに書き込まれるメッセージ。これは必須です
messageType	SAS のログメッセージの種類。これはオプションのパラメーターで、デフォルトは NOTE です。必要に応じて、WARNING または ERROR を指定できます。

例 リソースが見つからないという警告をユーザーに送信します

```
proc python;
submit;
<Python code>
SAS.logMessage('Resource not found.', 'warning')
<more Python code>
endsubmit;
run;
```

PYTHON プロシジャ内でデータを転送するコールバックメソッド

次のコールバックメソッドを使用すると、Pandas DataFrame と SAS データセットの間でデータを転送できます。

SAS.df2sd(*data-frame*, '*SAS-data-set*<(*data-set-options*)>'<,
datetimes={*'name':'value' pair(s)*}><, outfmts={*'name':'value' pair(s)*}><,
labels={*'name':'value' pair(s)*}><, char_lengths={*'name':'value' pair(s)*}>)
Pandas DataFrame から SAS データセットにデータを転送します。SAS データセットがすでに存在する場合、データセットはデフォルトで上書きされます。

DATETIMES=は、列名を値'date'または'time'に関連付けます。列の値は、SAS でそれぞれ DATE または TIME 値を生成するために使用されます。DataFrame 日時値の列で'date'または'time'値を指定しない場合、これらの値は DATETIME 値として SAS にロードされます。

OUTFMTS=は、列名を出力 SAS データセット用に定義された SAS フォーマットに関連付けます。

LABELS=は、結果の SAS データセットの列にラベルを割り当てます。

CHAR_LENGTHS=は、SAS データセットにロードされる文字値に長さを割り当てます。これらの長さの値は、DataFrame 列の値に基づいて計算される長さをオーバーライドします。

オプション引数 DATETIMES=、OUTFMTS=、LABELS=、および CHAR_LENGTHS=は、Python ディクショナリを取ります。

例 Pandas DataFrame **my_frame** から SAS データセット Work.Cars にデータを転送します。

```
ds = SAS.df2sd(my_frame, 'work.cars')
```

追加のオプションを使用して、Pandas DataFrame **my_frame** から SAS データセット Work.Cars_exact にデータを転送します。

```
ds = SAS.df2sd(my_frame, 'cars_exact',
      labels={'length':'Length (IN)','mpg_City':'MPG (City)',
           'EngineSize':'Engine Size (L)',
           'MPG_Highway':'MPG (Highway)',
           'Weight':'Weight (LBS)','WheelBase':'Wheelbase (IN)'},
      outfmts={'Msrp':'DOLLAR8.','invoice':'DOLLAR8.'},
      char_lengths={'Model':40,'type': 8})
```

SAS.sd2df('*SAS-data-set*<(*data-set-options*)>')
SAS データセットまたはビューから Pandas DataFrame にデータを転送します。必要に応じて、追加のデータセットオプションを指定して、Pandas DataFrame に渡すデータを制御します。

このコールバックメソッド呼び出しの結果のオブジェクトは、Pandas DataFrame です。

例 Sashelp.Cars から Pandas DataFrame にデータをエクスポートします。

```
df = SAS.sd2df('sashelp.cars')
```

最初の 5 行を Sashelp.Cars から Pandas DataFrame にエクスポートします。

```
df = SAS.sd2df('sashelp.cars(obs=5)')
```

選択した列を Sashelp.Cars から Pandas DataFrame にエクスポートします。

```
df = SAS.sd2df('sashelp.cars(keep=make model)')
```

データ転送時にログ出力を制御するコールバックメソッド

SAS.df2sd および SAS.sd2df コールバックメソッドを使用してデータを転送するときに、これらのコールバックメソッドを使用して、SAS ログに出力を表示する方法または SAS ログに出力を表示するかどうかを変更します。

SAS.hideLOG(True | False)
PROC PYTHON からの Python 出力を、SAS ログではなく Work ディレクトリ内の sas2py.log ファイルにリダイレクトするかどうかを指定します。
SAS.symput など、SAS モジュールを呼び出すコールバックメソッドからの出力は、引き続き SAS ログに表示されます。

このコールバックメソッドは、PROC PRINTTO を使用して出力をリダイレクトします。したがって、PROC PYTHON が完了すると、SAS ログに PROC PRINTTO に関する情報が表示されます。

デフォルト True

例 SAS.hideLOG を呼び出して Python 出力をリダイレクトする

```
proc python;
submit;
#Send Log output to sas2py.log
```

```
SAS.hideLOG(True)
<Python code>
#Send output back to the SAS log
SAS.hideLOG(False)
<Python code>
endsubmit;
run;
```

SAS.printLOG()
SAS.hideLOG コールバックメソッドが呼び出されてから蓄積されたすべての Python 出力を SAS ログに表示します。

このコールバックメソッドを使用すると、SAS.hideLOG の呼び出し以降にエラーが発生したかどうかを確認できます。

例 SAS.hideLOG を呼び出した後の Python 出力を表示する

```
proc python;
submit;
SAS.hideLOG(True)
<Python code>
SAS.printLOG()
<Additional Python code, still redirected to external Work file>
endsubmit;
run;
```

SAS.clearLOG()
このメソッドは、現在キャッシュされている LOG 出力があればすべて削除します。データ転送メソッドからの後続の出力は、HideLOG()の設定に基づいて出力を非表示にするように設定されている場合、その出力をキャッシュし続けます。

例 キャッシュされたログ出力をクリアします。

```
proc python;
submit;
SAS.hideLOG(True)
<Python code>
SAS.printLOG()
<Python code>
SAS.clearLOG()
endsubmit;
run;
```

例: PYTHON プロシジャ

例 1: 外部 PYTHON スクリプトから入力を指定する

要素: FILENAME ステートメント

PROC PYTHON ステートメント、INFILE=オプション

詳細

この例は、**/usr/scripts** ディレクトリにある my_script.py という Python スクリプトを実行する 2 つの方法を示しています。最初のプログラムは、PROC PYTHON ステートメントの INFILE=オプションでファイル名と場所を直接指定します。2 番目のプログラムは、my_script.py に対するファイル参照名を定義し、INFILE=オプションでそのファイル参照名を使用します。

my_script.py の内容は次のとおりです。

```
print("Hello World")
y = "Hi!"
def my_function():
   print("Inside the my_script.py script")
```

プログラム: INFILE=オプションを使用して Python スクリプトの場所を指定する

PROC PYTHON ステートメントを実行し、PYTHON スクリプトの場所と名前を指定します。 ファイルを SAS Studio 内の場所にアップロードします。INFILE=オプションを使用して、Python スクリプトの名前と場所を指定します。INFILE=オプションを呼び出すと、SAS は my_script.py で Python ステートメントを実行します。スクリプトを実行して作成された変数と関数は、PROC PYTHON の SUBMIT および ENDSUBMIT ブロック内で使用できます。

```
proc python infile='/tmp/my_script.py';
submit;
my_function()
print("y = " + y)
endsubmit;
run;
```

プログラム: Python スクリプトをファイル参照名として指定する

```
filename script 'Users/<myuser>/My Folder/my_script.py';

proc python infile=script;
submit;
my_function()
print("y = " + y)
endsubmit;
run;
```

プログラムの説明

Python スクリプトとその場所を指定するファイル参照名を定義します。 Python スクリプトを SAS Studio にアップロードします。FILENAME ステートメントを使用して、Python スクリプト用に script というファイル参照名を作成します。

```
filename script 'Users/<myuser>/My Folder/my_script.py';
```

PROC PYTHON ステートメントを実行し、Python スクリプトを識別するファイル参照名を指定します。 PROC PYTHON ステートメントで INFILE=オプションを指定すると、スクリプトが呼び出されます。スクリプトを実行して作成された変数と関数は、PROC PYTHON の SUBMIT および ENDSUBMIT ブロック内で使用できます。

```
proc python infile=script;
submit;
my_function()
print("y = " + y)
endsubmit;
run;
```

例のコード 46.1　外部の Python スクリプトを呼び出した後の SAS ログ

```
80   filename script '/tmp/my_script.py';
81
82   proc python infile=script;
83   submit
NOTE: Python initialized.
Python 3.6.5 (default, Jun 24 2020, 17:52:46)
[GCC 8.3.1 20191121 (Red Hat 8.3.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
83 !      ;
84   my_function()
85   print("y = " + y)
86   endsubmit;
87   run;
88   %let SAS_workpath = %sysfunc(pathname(work));
>>>
Hello World
>>>
>>>
>>>
Inside the my_script.py script
y = Hi!
>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time           1.68 seconds
      cpu time            0.05 seconds
```

例 2: PYTHON プロシジャの再起動と終了の影響を観察する

要素: PROC PYTHON ステートメント、RESTART オプション
PROC PYTHON ステートメント、TERMINATE オプション

詳細

この例は、複数のプロシジャ呼び出しにわたって PYTHON プロシジャのステータスを維持する効果を示しています。RESTART または TERMINATE オプションを使用すると、PYTHON プロシジャのステータスがリセットされることがわかります。

プログラム

```
proc python;
submit;
y = "Hi!"
def my_function():
   print("Inside the function")
endsubmit;
run;

proc python;
submit;
my_function()
print("y = " + y)
endsubmit;
run;

proc python restart;
submit;
x = "New variable x is created"
print("x = " + x)
#References to function, variable from previous PROC PYTHON call
my_function()
print("y = " + y)
endsubmit;
run;

proc python terminate;
submit;
#Reference to variable defined in previous PROC PYTHON call
print("x = " + x)
x = "Updating the value for x"
print("x = " + x)
```

```
#Redefining a function called my_function
def my_function():
   print("Inside the proc step")
endsubmit;
run;

proc python;
submit;
print("x = " + x)
my_function()
endsubmit;
run;
```

プログラムの説明

PROC PYTHON を呼び出し、変数 y と my_function という関数を定義します。 文字列値を変数 y に割り当てます。別の文字列を出力する my_function という関数を定義します。

```
proc python;
submit;
y = "Hi!"
def my_function():
   print("Inside the function")
endsubmit;
run;
```

後続の PYTHON プロシジャ呼び出しで、以前に定義された関数と変数を参照します。 関数 my_function を呼び出し、変数 y の値を出力します。PYTHON プロシジャのステータスは後続のプロシジャ呼び出し間で維持されるため、これらの項目の定義は維持されます。

```
proc python;
submit;
my_function()
print("y = " + y)
endsubmit;
run;
```

RESTART オプションを使用して、PYTHON プロシジャのステータスをリセットします。 PYTHON プロシジャのステータスは、このプロシジャ呼び出しの開始時にリセットされます。x という新しい変数を定義します。以前に定義された変数または関数は使用できなくなります。これらを参照すると、SAS ログにエラーが記録され、プロシジャ呼び出しが終了します。

```
proc python restart;
submit;
x = "New variable x is created"
print("x = " + x)
#References to function, variable from previous PROC PYTHON call
my_function()
print("y = " + y)
endsubmit;
run;
```

TERMINATE オプションを使用して、PROC PYTHON のステータスの維持を停止します。 TERMINATE オプションを使用すると、PYTHON プロシジャのステータスはプロシジャ呼び出しの終了時にリセットされます。変数 x は、PROC PYTHON への前回の呼び出しからまだ定義されています。関数 my_function の新しい定義を指定します。この関数の以前の定義は、PROC PYTHON の前回の呼び出しで RESTART オプションを指定したときに失われています。

```
proc python terminate;
submit;
#Reference to variable defined in previous PROC PYTHON call
print("x = " + x)
x = "Updating the value for x"
print("x = " + x)
#Redefining a function called my_function
def my_function():
   print("Inside the proc step")
endsubmit;
run;
```

PROC PYTHON の前回の呼び出しの終了時に PROC PYTHON のステータスがリセットされたことを確認します。 PROC PYTHON の前回の呼び出しで TERMINATE オプションを指定したため、そのプロシジャ呼び出しが終了した後、変数または関数定義は維持されませんでした。x を参照すると、SAS ログにエラーが記録され、プロシジャ呼び出しが終了します。

```
proc python;
submit;
print("x = " + x)
my_function()
endsubmit;
run;
```

アウトプット 46.4 RESTART および TERMINATE オプションの効果を示す出力

```
80   proc python;
81   submit
NOTE: Python initialized.
Python 3.6.5 (default, Jun 24 2020, 17:52:46)
[GCC 8.3.1 20191121 (Red Hat 8.3.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
81 !     ;
82   y = "Hi!"
83   def my_function():
84       print("Inside the function")
85   endsubmit;
86   run;
87   %let SAS_workpath = %sysfunc(pathname(work));
>>>
>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time          3.88 seconds
      cpu time           0.06 seconds

88
89   proc python;
90   submit
NOTE: Resuming Python state from previous PROC PYTHON invocation.
90 !     ;
91   my_function()
92   print("y = " + y)
93   endsubmit;
94   run;
>>>
Inside the function
y = Hi!

>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds

95
96   proc python restart;
97   submit
NOTE: Python terminated.
NOTE: Previous Python state destroyed.
NOTE: Python initialized.
Python 3.6.5 (default, Jun 24 2020, 17:52:46)
[GCC 8.3.1 20191121 (Red Hat 8.3.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
```

```
97 !     ;
98   x = "New variable x is created"
99   print("x = " + x)
100  #References to function, variable from previous PROC PYTHON call
101  my_function()
102  print("y = " + y)
103  endsubmit;
104  run;
105  %let SAS_workpath = %sysfunc(pathname(work));
>>>
x = New variable x is created

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 4, in <module>
NameError: name 'my_function' is not defined

>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
     real time          1.36 seconds
     cpu time           0.08 seconds
106
107  proc python terminate;
108  submit
NOTE: Resuming Python state from previous PROC PYTHON invocation.
108!     ;
109  #Reference to variable defined in previous PROC PYTHON call
110  print("x = " + x)
111  x = "Updating the value for x"
112  print("x = " + x)
113  #Redefining a function called my_function
114  def my_function():
115      print("Inside the proc step")
116  endsubmit;
117  run;
>>>
x = New variable x is created
x = Updating the value for x

>>>
>>>
NOTE: Python terminated.
NOTE: PROCEDURE PYTHON used (Total process time):
     real time          0.00 seconds
     cpu time           0.00 seconds
118
119  proc python;
120  submit
NOTE: Python initialized.
Python 3.6.5 (default, Jun 24 2020, 17:52:46)
[GCC 8.3.1 20191121 (Red Hat 8.3.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
120!     ;
121  print("x = " + x)
122  my_function()
123  endsubmit;
124  run;
125  %let SAS_workpath = %sysfunc(pathname(work));
>>>
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 1, in <module>
NameError: name 'x' is not defined

>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
     real time          1.14 seconds
     cpu time           0.04 seconds
```

例 3: SAS コードをサブミットし、PROC PYTHON 内で SAS 関数を呼び出す

要素: SAS.submit
SAS.sasfnc

詳細

この例では、SAS.submit コールバックメソッドを使用して Python コードのブロック内で SAS コードをサブミットする方法を示します。この例では、SAS.submit コールバックメソッドを使用して、Test というデータセットを作成します。Test データセットは、PROC PYTHON への呼び出しの外部で使用できます。この例では、SAS.sasfnc コールバックメソッドを使用して SAS 関数 SHA256HEX を呼び出し、値'abc'に対して取得された SHA256 ダイジェストを表示する方法も示しています。

プログラム

```
proc python;
submit;
var1 = "'python'"
var2 = 2

SAS.submit("data work.test; x={}; y={}; run;".format(var1,var2))

var3 = SAS.sasfnc("sha256hex","abc")
print("var3 = " + var3)

endsubmit;
run;

proc print data=test;
run;
```

プログラムの説明

PROC PYTHON への呼び出しを開始し、SUBMIT ステートメントを使用して Python コードのブロックを開始します。 値を 2 つの Python 変数 var1 と var2 に割り当てます。文字列値は、値'python'が DATA ステップに送信されるように、追加の引用符のセットを取ります。

```
proc python;
submit;
```

```
var1 = "'python'"
var2 = 2
```

SAS.submit コールバックメソッドを呼び出して、SAS データセット Work.Test を作成します。 Python format 関数を使用して、値を X と Y に割り当てます。

```
SAS.submit("data work.test; x={}; y={}; run;".format(var1,var2))
```

SAS.sasfnc コールバックメソッドを使用して SAS 関数を呼び出します。 文字列 "abc"に対して SHA256HEX 関数を呼び出し、戻り値を Python 変数 var3 に割り当てます。Python print 関数を呼び出して、var3 の値を示す文字列を表示します。

```
var3 = SAS.sasfnc("sha256hex","abc")
print("var3 = " + var3)
```

Python コードのブロックを終了し、生成されたデータセット Work.Test を出力します。 ENDSUBMIT ステートメントを呼び出して、Python コードブロックを終了します。PROC PYTHON の呼び出しに続いて、Work.Test データセットを参照する PROC PRINT 呼び出しを行います。データセットは PROC PYTHON 呼び出しの外部で使用できることに注意してください。

```
endsubmit;
run;

proc print data=test;
run;
```

アウトプット 46.5　PROC PYTHON での SAS.submit および SAS.sasfnc 呼び出しからの出力

Obs	x	y
1	python	2

例のコード 46.2 PROC PYTHON での SAS.submit および SAS.sasfnc 呼び出しからのログ

```
80   proc python;
81   submit
NOTE: Python initialized.
Python 3.6.5 (default, Jun 24 2020, 17:52:46)
[GCC 8.3.1 20191121 (Red Hat 8.3.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
81 !     ;
82   var1 = "'python'"
83   var2 = 2
84   SAS.submit("data work.test; x={}; y={}; run;".format(var1,var2))
85   var3 = SAS.sasfnc("sha256hex","abc")
86   print("var3 = " + var3)
87   endsubmit;
88   run;
89   %let SAS_workpath = %sysfunc(pathname(work));
90   data work.test; x='python'; y=2; run;
NOTE: The data set WORK.TEST has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time          0.00 seconds
      cpu time           0.00 seconds

>>>
var3 = BA7816BF8F01CFEA414140DE5DAE2223B00361A396177A9CB410FF61F20015AD
>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time          3.29 seconds
      cpu time           0.04 seconds

91
92   proc print data=test;
93   run;
NOTE: There were 1 observations read from the data set WORK.TEST.
NOTE: The PROCEDURE PRINT printed page 1.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.02 seconds
      cpu time           0.02 seconds
```

例 4: PROC PYTHON で SAS マクロ変数を作成する

要素:

SAS マクロ変数の定義
SAS.symget コールバックメソッド
SAS.symput コールバックメソッド

詳細

この例では、SAS マクロ変数を作成し、それに Python 変数の値を代入する方法を示します。マクロ変数にはグローバルスコープがあり、PROC PYTHON 呼び出しの外部で参照できます。

プログラム

```
%let myvar = Jane Doe;

proc python;
submit;
x = SAS.symget('myvar')
print('macro variable myvar = ' + x)

py_var = 'Inside python'
SAS.symput('macrovar', py_var)
endsubmit;
run;

%put &=macrovar;
%put &=myvar;
```

プログラムの説明

PROC PYTHON への呼び出しの外部で、マクロ変数 Myvar を割り当てます。

```
%let myvar = Jane Doe;
```

PROC PYTHON を呼び出し、Myvar の値を Python 変数に割り当てます。
SAS.symget コールバックメソッドを使用して、SAS マクロ変数 Myvar の値を Python 変数 x に割り当てます。Python print 関数を使用して、x の値を表示します。

```
proc python;
submit;
x = SAS.symget('myvar')
print('macro variable myvar = ' + x)
```

Python 変数を定義し、文字列値を割り当てます。 Python 変数 py_var を定義し、それに文字列値を割り当てます。次に、SAS.symput を呼び出して、py_var の値を Macrovar 変数の値に置き換えます。

```
py_var = 'Inside python'
SAS.symput('macrovar', py_var)
endsubmit;
run;
```

PROC PYTHON の外部にあるグローバルマクロ変数 Macrovar を参照します。
SAS.symput で定義するマクロ変数は、グローバル変数です。したがって、PROC

PYTHON への呼び出しの外部でそれらを参照できます。マクロ変数 Myvar も引き続き使用できます。

```
%put &=macrovar;
%put &=myvar;
```

例のコード 46.3　マクロ変数からの SAS ログの例

```
80   %let myvar = Jane Doe;
81
82   proc python;
83   submit
NOTE: Python initialized.
Python 3.6.5 (default, Jun 24 2020, 17:52:46)
[GCC 8.3.1 20191121 (Red Hat 8.3.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
83 !     ;
84   x = SAS.symget('myvar')
85   print('macro variable myvar = ' + x)
86   py_var = 'Inside python'
87   SAS.symput('macrovar', py_var)
88   endsubmit;
89   run;
90   %let SAS_workpath = %sysfunc(pathname(work));
>>>
macro variable myvar = Jane Doe
>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time           2.64 seconds
      cpu time            0.05 seconds

91
92   %put &=macrovar;
MACROVAR=Inside python
93   %put &=myvar;
MYVAR=Jane Doe
```

例 5: PROC PYTHON で FCMP 関数を呼び出す

要素:

SAS.sasfnc コールバックメソッド

PROC FCMP

詳細

SAS.sasfnc 関数を使用すると、FCMP プロシジャを使用して作成した関数を呼び出すことができます。この例は、FCMP 関数 fnc1 を呼び出して、2 つの数値を合計します。

プログラム

```
proc fcmp outlib=work.funcs.trial;
  function fnc1(num1, num2);
      return(num1+num2);
  endsub;
quit;

options cmplib=work.funcs;

proc python;
submit;
x = SAS.sasfnc("fnc1",1,2)
print("x = ", int(x))
endsubmit;
run;
```

プログラムの説明

2 つの数値引数の合計を返す fnc1 という関数を定義します。 FCMP プロシジャを使用して、fnc1 関数を定義します。この場合、fnc1 は渡された 2 つの引数の合計を返します。

```
proc fcmp outlib=work.funcs.trial;
  function fnc1(num1, num2);
      return(num1+num2);
  endsub;
quit;
```

CMPLIB システムオプションを指定します。 このシステムオプションは、プログラムコンパイラ時に含めるコンパイラサブルーチンを含む SAS データセットを指定します。

```
options cmplib=work.funcs;
```

Python コマンドのブロックから fnc1 関数を呼び出します。 Python コマンドのブロックからの出力は SAS ログに出力されます。

```
proc python;
submit;
x = SAS.sasfnc("fnc1",1,2)
print("x = ", int(x))
endsubmit;
```

```
run;
```

例のコード 46.4 PROC FCMP からの SAS ログの例

```
78   proc fcmp outlib=work.funcs.trial;
79     function fnc1(num1, num2);
80           return(num1+num2);
81     endsub;
82   quit;
NOTE: Function fnc1 saved to work.funcs.trial.
NOTE: PROCEDURE FCMP used (Total process time):
      real time          0.02 seconds
      cpu time           0.05 seconds

83
84   options cmplib=work.funcs;
85
86   proc python;
87   submit;
NOTE: Python initialized.
Python 3.6.5 (default, Jun 24 2020, 17:52:46)
[GCC 8.3.1 20191121 (Red Hat 8.3.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
90   x = SAS.sasfnc("fnc1",1,2)
91   print("x = ", int(x))
92   endsubmit;
93   run;
94   %let SAS_workpath = %sysfunc(pathname(work));
>>>
x =  3
>>>
>>>
NOTE: PROCEDURE PYTHON used (Total process time):
      real time          2.29 seconds
      cpu time           0.11 seconds
```

例 6: Python プロットをレンダリングする

要素: matplotlib.pyplot モジュールのインポート
SAS.pyplot コールバックメソッド

詳細

SAS.pyplot コールバックメソッドを使用して、matplotlib.pyplot Python モジュールの関数で作成および変更するプロットオブジェクトをレンダリングします。この例では、データは NumPy ライブラリの配列関数を使用して作成されます。

プログラム

```
proc python echo;
interactive;
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery-nogrid')

# make data:
X, Y = np.meshgrid([1, 2, 3, 4], [1, 2, 3, 4])
angle = np.pi / 180 * np.array([[15., 30, 35, 45],
                                [25., 40, 55, 60],
                                [35., 50, 65, 75],
                                [45., 60, 75, 90]])
amplitude = np.array([[5, 10, 25, 50],
                      [10, 15, 30, 60],
                      [15, 26, 50, 70],
                      [20, 45, 80, 100]])
U = amplitude * np.sin(angle)
V = amplitude * np.cos(angle)

# plot:
fig, ax = plt.subplots()

ax.barbs(X, Y, U, V, barbcolor='C0', flagcolor='C0', length=7, linewidth=1.5)

ax.set(xlim=(0, 4.5), ylim=(0, 4.5))

SAS.pyplot(plt)

endinteractive;
run;
```

プログラムの説明

PROC PYTHON を呼び出し、必要なライブラリとモジュールをインポートします。 ECHO オプションを使用すると、対話式にサブミットされた Python コードの行が SAS ログに含まれます。INTERACTIVE ステートメントは、対話式にサブミットされた Python コマンドのブロックを開始します。Python import コマンドは matplotlib.pyplot モジュールと NumPy ライブラリをロードします。matplotlib.pyplot モジュールを使用すると、プロット関数にアクセスできます。NumPy ライブラリを使用すると、配列を操作できます。別名 plt を使用して pyplot モジュール内の関数にアクセスし、別名 np を使用して NumPy 関数にアクセスします。

```
proc python echo;
interactive;
import matplotlib.pyplot as plt
import numpy as np
```

プロットスタイルを指定します。 プロットのレンダリング方法に関する設定を指定します。pyplot モジュールの詳細については、matplotlib のドキュメントを参照してください。

```
plt.style.use('_mpl-gallery-nogrid')
```

プロット用のデータを生成します。 NumPy 関数を使用して、プロット用のデータを生成します。NumPy 関数の詳細については、NumPy のドキュメントを参照してください。

```
# make data:
X, Y = np.meshgrid([1, 2, 3, 4], [1, 2, 3, 4])
angle = np.pi / 180 * np.array([[15., 30, 35, 45],
                                [25., 40, 55, 60],
                                [35., 50, 65, 75],
                                [45., 60, 75, 90]])
amplitude = np.array([[5, 10, 25, 50],
                      [10, 15, 30, 60],
                      [15, 26, 50, 70],
                      [20, 45, 80, 100]])
U = amplitude * np.sin(angle)
V = amplitude * np.cos(angle)
```

プロットの内容を指定します。 上記で生成したデータを使用して追加のプロットオプションを指定することにより、pyplot 関数を呼び出してプロットを設計します。

```
# plot:
fig, ax = plt.subplots()

ax.barbs(X, Y, U, V, barbcolor='C0', flagcolor='C0', length=7, linewidth=1.5)

ax.set(xlim=(0, 4.5), ylim=(0, 4.5))
```

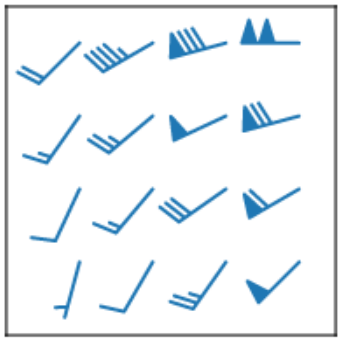
SAS.pyplot コールバックメソッドを使用してプロットをレンダリングします。 plt というプロットオブジェクトの SAS.pyplot コールバックメソッドを呼び出します。結果のプロットを PNG ファイルに保存するには、オプションの filename=および filepath=引数を指定します。たとえば、プロットをデータディレクトリ内の MyPlot.png というファイルに保存するには、SAS.pyplot(plt,filename="MyPlot",filepath="/myuser/data")を指定できます。

```
SAS.pyplot(plt)
```

PROC PYTHON への呼び出しを終了します。 ENDINTERACTIVE ステートメントを呼び出して、対話式にサブミットされた Python コードのブロックを終了します。

```
endinteractive;
run;
```

アウトプット 46.6 SAS.pyplot コールバックメソッドからの出力



アウトプット 46.7　SAS.pyplot の例からのログ出力

```
80   proc python;
81   interactive
NOTE: Python initialized.
Python 3.8.13 (default, Feb  8 2023, 16:30:40)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-16)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
81 !            ;
82   %let SAS_workpath = %sysfunc(pathname(work));
>>>
>>>
>>>
>>>
>>>
>>>
>>>
...
...
...
>>>
...
...
...
>>>
>>>
>>>
>>>
>>>
>>>
>>>
<matplotlib.quiver.Barbs object at 0x7f2d28cd8700>
>>>
>>>
[(0.0, 4.5), (0.0, 4.5)]
>>>
>>>
83   data _null_; declare odsout obj();
83 ! obj.image(file:"/opt/sas/viya/config/var/tmp/compsrv/default/4b1a2912-
e74b-4412-9ca5-baf5707be926/SAS_work7DE0000000B5_sas-comp
83 ! ute-server-f9ea2fb2-ce2b-44cd-b582-d64bc2c05d4b-jpnwc/matplot.svg");run;
NOTE: DATA statement used (Total process time):
      real time           0.07 seconds
      cpu time            0.02 seconds

84   data _null_; rc=filename('_delplot',
84 ! '/opt/sas/viya/config/var/tmp/compsrv/default/4b1a2912-e74b-4412-9ca5-
baf5707be926/SAS_work7DE0000000B5_sas-compute-server-f9ea
84 ! 2fb2-ce2b-44cd-b582-d64bc2c05d4b-jpnwc/matplot.svg'); if rc = 0 and
fexist('_delplot') then do;    rc = fdelete('_delplot');
84 !  end;run;
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

>>>
>>>
>>>
85   run;
NOTE: PROCEDURE PYTHON used (Total process time):
      real time           8.81 seconds
      cpu time            2.27 seconds
```

47

QDEVICE プロシジャ

概要: QDEVICE プロシジャ

QDEVICE プロシジャの機能

QDEVICE プロシジャでは、グラフィックデバイスおよびユニバーサルプリンターについてのレポートが作成されます。このレポートの情報を使用すると、特定のアプリケーション向けの使用に最適のデバイスやプリンターを決定できます。

異なる 6 つのレポートが使用可能です。これらのレポートでは、カラーサポート、デフォルトの出力サイズ、余白量、解像度、サポートされているフォント、ハードウェアシンボル、ハードウェアの塗りつぶしの種類、ハードウェアの線スタイル、デバイスオプションなどの情報が要約されます。

このプロシジャの出力は、SAS ログまたは出力 SAS データセットに送ることができます。

構文: QDEVICE プロシジャ

デフォルト: 作成するレポート、プリンターまたはデバイスを指定しなかった場合、プロシジャでは、ウィンドウ環境を使用して SAS を実行するとデフォルトディスプレイデバイス用に、その他のモードではデフォルトユニバーサルプリンター用に、GENERAL レポートが生成されます。

注: DEVICE、PRINTER または VAR ステートメントは複数指定できます。ステートメントは指定された順番で処理されます。

PROC QDEVICE
< REPORT=GENERAL | FONT | DEVOPTION | LINESTYLE | RECTANGLE | SYMBOL>
< OUT=*SAS-data-set*>
< CATALOG=*catalog-name*>
< DEVLOC=GDEVICE*n* | SASHELP | _ALL_ | *libref*>
< REGISTRY=SASHELP | SASUSER>
< SUPPORT=YES | NO | ALL>
< UNITS=IN | CM;>

DEVICE<*device-name(s)*> < _ALL_> < _HTML_> < _LISTING_> < _RTF_>;

PRINTER <*printer-name(s)*> < _ALL_> < _PCL_> < _PDF_> < _PRINTER_> < _PS_>;

VAR *variable-1*<*variable-2* ...>;

ステートメント	タスク	例
“PROC QDEVICE ステートメント”	(任意)出力データセット、生成するレポート、サポート機能または未サポート機能のリストの有無、サイズ情報の指定	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7
“DEVICE ステートメント”	レポート生成対象の SAS/GRAPH デバイスの指定	Ex. 2, Ex. 7
“PRINTER ステートメント”	レポート生成対象のユニバーサルプリンターの指定	Ex. 6, Ex. 3
“VAR ステートメント”	生成レポートに含める情報(変数)の指定	Ex. 5

PROC QDEVICE ステートメント

検証された入力データ、レポートの内容、作成する出力の種類を制御します。

構文

PROC QDEVICE
< REPORT=GENERAL | FONT | DEVOPTION | LINESTYLE | RECTANGLE | SYMBOL>
< OUT=*SAS-data-set*>
< CATALOG=*catalog-name*>
< DEVLOC=GDEVICE*n* | SASHELP | _ALL_ | *libref*>
< REGISTRY=SASHELP | SASUSER>
< SUPPORT=YES | NO | ALL>
< UNITS=IN | CM>;

オプション引数の要約

CATALOG=*catalog-name*
: SAS デバイスカタログの名前を指定してデバイスを検索します。

DEVLOC=GDEVICE*n* | SASHELP | _ALL_ | *libref*
: デバイスカタログが位置する単数または複数のライブラリを指定します。

OUT=*SAS-data-set*
: レポートの出力 SAS データセットを指定します。

REGISTRY=SASHELP | SASUSER
: ユニバーサルプリンターのクエリ時に SAS レジストリのどの部分を検索するかを指定します。

REPORT=DEVOPTION | FONT | GENERAL | LINESTYLE | RECTANGLE | SYMBOL

生成するレポートの種類を指定します。

SUPPORT=YES | NO | ALL
サポート機能のみ、未サポート機能のみ、または全機能のいずれをレポートするのかを指定します。

UNITS=IN | CM
GENERAL レポートにおいて特定の変数の値の単位をインチ(IN)単位でレポートするか、センチメートル(CM)単位でレポートするかを指定します。

オプション引数

CATALOG=*catalog-name*
SAS デバイスカタログの名前を指定してデバイスを検索します。

別名 C=

CAT=

デフォルト DEVICES

操作 CATALOG=オプションは DEVLOC=オプションとともに機能します。CATALOG=オプションを指定すると、DEVLOC=オプションにより指定されたライブラリ(sashelp.mycatalog など)で SAS が検索します。

例 “例 7: レポートに対するユーザーライブラリおよびカタログの指定” (1649 ページ)

DEVLOC=GDEVICE*n* | SASHELP | _ALL_ | *libref*
デバイスカタログが位置する単数または複数のライブラリを指定します。 指定できるライブラリは次のとおりです。

GDEVICE*n*
デバイスの SAS/GRAPH デバイスライブラリの 1 つを検索するように指定します。*n* は 0～9 です。

SASHELP
Sashelp ライブラリでデバイスを検索するよう指定します。

ALL
ライブラリ Gdevice0～Gdevice9、次に Sashelp ライブラリという順序でデバイスを検索するよう指定します。これらライブラリからのデバイス検索はすべてレポートされます。

libref
検索する有効な SAS ライブラリを指定します。

デフォルト DEVLOC=オプションを指定しなかった場合、ライブラリは次の順序で検索されます。

1. Gdevice0–Gdevice9
2. Sashelp

DEVLOC=_ALL_を指定しなければ、指定したデバイスの検索の初回がレポートされます。

操作 DEVLOC=オプションは CATALOG=オプションとともに機能します。CATALOG=オプションを指定すると、DEVLOC=オプションにより指定されたライブラリ(sashelp.mycatalog など)で SAS が検索します。

例 “例 7: レポートに対するユーザーライブラリおよびカタログの指定” (1649 ページ)

OUT=*SAS-data-set*
レポートの出力 SAS データセットを指定します。

デフォルト SAS ログ

REGISTRY=SASHELP | SASUSER
ユニバーサルプリンターのクエリ時に SAS レジストリのどの部分を検索するかを指定します。

SASHELP
レジストリの SASHELP セクションを検索します。

SASUSER
レジストリの SASUSER セクションを検索します。

別名 REG=

デフォルト SASHELP と SASUSER

REPORT=DEVOPTION | FONT | GENERAL | LINESTYLE | RECTANGLE | SYMBOL
生成するレポートの種類を指定します。 要求できるレポートの種類は 1 つだけです。各レポートに含まれる変数の説明については、“すべてのレポートに有効な変数” (1613 ページ)を参照してください。

DEVOPTION
指定デバイスでサポートされるハードウェアデバイスオプションのレポートを作成します。

制限事項 このレポートは、ユニバーサルプリンターでは使用できません。

FONT
指定したデバイスまたはプリンターでサポートされるすべてのシステムフォントとデバイス常駐フォントのレポートを作成します。

参照項目: フォントカテゴリの説明については、“SAS/GRAPH, System, and Device-Resident Fonts” (*SAS/GRAPH: Reference*)を参照してください。

GENERAL
指定したデバイスまたはプリンターに関する一般情報のレポートを作成します。このレポートには、出力先、余白サイズ、デフォルトフォント情報、解像度、色情報、ピクセル単位サイズなどの情報が含まれます。これはデフォルトレポートです。

LINESTYLE
指定デバイスでサポートされるハードウェアの線スタイルのレポートを作成します。

制限事項　このレポートは、ユニバーサルプリンターでは使用できません。

RECTANGLE
指定デバイスでサポートされるハードウェアの塗りつぶしの種類のレポートを作成します。

制限事項　このレポートは、ユニバーサルプリンターでは使用できません。

SYMBOL
指定デバイスでサポートされるハードウェアシンボルのレポートを作成します。

制限事項　このレポートは、ユニバーサルプリンターでは使用できません。

デフォルト　GENERAL

参照項目:　各レポートに含まれる変数の説明については、“すべてのレポートに有効な変数” (1613 ページ)を参照してください。

SUPPORT=YES | NO | ALL
サポート機能のみ、未サポート機能のみ、または全機能のいずれをレポートするのかを指定します。

YES
サポートされているハードウェア機能とオプションのみをレポートします。

NO
サポートされていないハードウェア機能とオプションのみをレポートします。

ALL
ハードウェア機能とオプションについて、サポートされているものとされていないものの両方をレポートします。

デフォルト　YES

制限事項　このオプションは、DEVOPTION、LINESTYLE、RECTANGLE、SYMBOL レポートのいずれかの作成時にのみデバイスに適用されます。

UNITS=IN | CM
GENERAL レポートにおいて特定の変数の値の単位をインチ(IN)単位でレポートするか、センチメートル(CM)単位でレポートするかを指定します。

HEIGHT、WIDTH、LEFT、LMIN、RIGHT、RMIN、BOTTOM、BMIN、TOP、TMIN、HRES、VRES 変数の値を、インチまたはセンチのいずれかでレポートされます。HRES と VRES の値は、ピクセルパーインチかピクセルパーセンチでレポートされます。

デフォルト　IN

制限事項　このオプションは、GENERAL レポートの作成時にのみ適用されます。

DEVICE ステートメント

どの SAS/GRAPH デバイス用にレポートを作成するかを指定します。

要件 デバイス名_ALL_、_HTML_、_LISTING_、_RTF_のうちから少なくとも 1 つ指定する必要があります。

構文

DEVICE<*device-name(s)*> < _ALL_ > < _HTML_ > < _LISTING_ > < _RTF_>;

オプション引数

device-name(s)
レポート生成対象のデバイスを指定します。デバイス名はブランクスペースで区切ります。スペースを 含むデバイス名を引用符で囲みます。ワイルドカード文字*および?を使用して、類似した名前のすべてのデバイスを 報告できます。

*
*は、対象デバイス名のこの位置における任意の数の文字を表わします。

要件 ワイルドカード文字を含むデバイス名は、必ず引用符で囲んでください。

注 同一のデバイス名に含まれる*と?は指定できます。

例 'svg*'

例 “例 3: SAS/GRAPH デバイスドライバーとユニバーサルプリンターのレポートの生成” (1639 ページ)

?
?は対象デバイス名の中の 1 文字を表します。対象デバイス名の中の複数の連続する?文字を指定できます。

要件 ワイルドカード文字を含むデバイス名は、必ず引用符で囲んでください。

注 同一のデバイス名に含まれる*と?は指定できます。

例 'tiff?300'

ALL
全デバイスのレポートを生成します。

HTML
ODS HTML 出力先で使用するデフォルトデバイスを決定し、そのデバイスに対するレポートを生成します。デバイスは、SAS レジストリの ODS キーで割り当てられるデフォルト HTML バージョンに基づいています。

LISTING
ODS リスト出力先で使用するデフォルトデバイスを決定し、そのデバイスに対するレポートを生成します。デフォルト値はホスト固有のディスプレイデバイスです。

RTF
ODS RTF 出力先で使用するデフォルトデバイスを決定し、そのデバイスに対するレポートを生成します。

PRINTER ステートメント

どのユニバーサルプリンター用にレポートを生成するかを指定します。

要件 プリンター名_ALL_、_PCL_、_PDF_、_PRINTER_、_PS_のうちから少なくとも 1 つ指定する必要があります。

構文

PRINTER<*printer-name(s)*> < _ALL_ > < _PCL_ > < _PDF_ > < _PRINTER_ > < _PS_ >;

オプション引数

printer-name(s)
レポート生成対象のユニバーサルプリンターを指定します。プリンター名にスペースが含まれる場合は、そのプリンター名を引用符で囲みます。プリンター名はブランクスペースで 区切ります。ワイルドカード文字*および?を使用して、類似した名前のすべてのプリンターを 報告できます。

*****
対象プリンター名の*の位置にある任意の数の文字と一致するすべてのプリンターをレポートすることを示します。

要件 ワイルドカード文字を含むプリンター名は、必ず引用符で囲んでください。

注 同一のプリンター名に含まれる*と?は指定できます。

例 'pcl*'はプリンター pcl4、pcl5、pcl5c、pcl5e をレポートします。

?
対象プリント名に適合する全プリンターのレポートを意味します。?位置の文字は任意です。対象プリンター名の同一位置に同じ数の文字を表示するには、*printer-name* で複数の?文字を使用できます。

要件 ワイルドカード文字を含むプリンター名は、必ず引用符で囲んでください。

注 同一のプリンター名に含まれる*と?は指定できます。

例 'tiff?'は、プリンター tiffa と tiffk をレポートします。プリンター tiff は 4 文字だけなのでレポートされません。

例 “例 3: SAS/GRAPH デバイスドライバーとユニバーサルプリンターのレポートの生成” (1639 ページ)

ALL
全ユニバーサルプリンターのレポートを生成します。

PCL
ODS PCL 出力先で使用するデフォルトプリンターを決定し、そのプリンターに対するレポートを生成します。

PDF
ODS PDF 出力先で使用するデフォルトプリンターを決定し、そのプリンターに対するレポートを生成します。

PRINTER
ODS PRINTER 出力先で使用するデフォルトプリンターを決定し、そのプリンターに対するレポートを生成します。

_PS_
ODS PS 出力先で使用するデフォルトプリンターを決定し、そのプリンターに対するレポートを生成します。

VAR ステートメント

レポートにどの変数を含めるかを指定します。レポートでの変数の順序は、VAR ステートメントで指定した順序によって決定されます。

デフォルト: VAR ステートメントを指定しない場合は、レポートに対する変数はすべて、デフォルトの順序でレポートに含められます。

ヒント: VAR ステートメントを指定する場合は、変数を少なくとも 1 つ指定する必要があります。そうでなければ、ステートメントは無視されます。

構文

VAR *variable-1*<*variable-2* ...>;

すべてのレポートに有効な変数

DESC
デバイスまたはプリンターのデフォルト説明を表示します。

LOCATION
検出されたデバイスエントリについて、Gdevice0-Gdevice9 ライブラリ、または Sashelp ライブラリ、または DEVLOC=オプションにより指定されているライブラリの物理的位置を表示します。ユニバーサルプリンターの場合、この変数で

は、プリンターが見つかった SAS レジストリ(SASHELP または SASUSER)が表示されます。

NAME
デバイスまたはプリンターの名前を表示します。

TYPE
デバイスまたはプリンターの種類を表示します。 デバイスとプリンターの種類は次のとおりです。

- グラフデバイス
- プリンターインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル
- ユニバーサルプレビューアー
- ユニバーサルプリンター

参照項目: “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

DEVOPTION レポート変数

詳細については、“DEVOPTION レポートの作成” (1631 ページ)を参照してください。

BIT
対応するデバイスオプションに対する DEVOPTS 文字列のビット位置を表示します。

参照項目: “DEVOPTS= Device Parameter” (*SAS/GRAPH: Reference*)

BITSTRING
対応するデバイスオプションのビットパターンを表示します。

参照項目: “DEVOPTS= Device Parameter” (*SAS/GRAPH: Reference*)

DESC
デバイスまたはプリンターのデフォルト説明を表示します。

LOCATION
検出されたデバイスエントリについて、Gdevice0-Gdevice9 ライブラリ、または Sashelp ライブラリ、または DEVLOC=オプションにより指定されているライブラリの物理的位置を表示します。

NAME
デバイスまたはプリンターの名前を表示します。

ODESC
デバイスに対する有効なハードウェアオプションの説明を表示します。

OPTION
デバイスに対する有効なハードウェアオプションの名前を表示します。

SUPPORT
デバイスオプションを表示します。

操作 SUPPORT 変数の値は、PROC QDEVICE ステートメントの SUPPORT=オプションに影響されます。SUPPORT=YES の場合、レポートにはサポートされているデバイスオプションが表示されます。SUPPORT=NO の場合、レポートにはサポートされていないデバイスオプションが表示されます。SUPPORT=ALL の場合、レポートには、サポートされているデバイスオプションもサポートされていないデバイスオプションもすべて 表示されます。

TYPE
デバイスまたはプリンターの種類を表示します。

デバイスとプリンターの種類は次のとおりです。

- グラフデバイス
- プリンターインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル

参照項目: “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

FONT レポート変数

詳細については、“FONT レポートの作成” (1629 ページ)を参照してください。

ALIAS
プロシジャにより登録されているフォントの別名をレポートします。

DESC
デバイスまたはプリンターのデフォルト説明を表示します。

FONT
デフォルトフォントの名前を表示します。

参照項目: “Default Fonts” (*SAS/GRAPH: Reference*)

“FONT レポートの変数ラベル” (1630 ページ)

FSTYLE
出力データセットに、各フォントのフォントスタイル(Roman や Italic など)およびフォントの太さを表示します。

制限事項 レポート出力が SAS ログに送られた場合、FONT レポートには、フォントファミリ名(Courier、Helvetica、Times など)のみ表示されます。特定のフォントスタイルはレポートされません。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、スタイルは使用できません。詳細については“CHARREC= Device Parameter” (*SAS/GRAPH: Reference*)を参照してください。

参照項目: “FONT レポートの変数ラベル” (1630 ページ)

FTYPE
フォントの種類(Printer Resident、System、Software など)を表示します。

注 出力データセットの FTYPE 変数の値は、Printer Resident、System または Software です。値 Software は、ハードウェアフォントサポートが無効な SAS/GRAPH デバイスの FONT レポートにのみ表示されます。

FWEIGHT
出力データセットに、各フォントのフォントの太さ(Normal や Bold など)およびフォントスタイルを表示します。

制限事項 レポート出力が SAS ログに送られた場合、FONT レポートには、フォントファミリ名(Courier、Helvetica、Times など)のみ表示されます。特定のフォントの太さはレポートされません。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、太さは使用できません。詳細については“CHARREC= Device Parameter” (*SAS/GRAPH: Reference*)を参照してください。

FVERSION
フォントのバージョンを指定します。

制限事項 レポート出力が SAS ログに送られた場合、FONT レポートには、フォントファミリ名(Courier、Helvetica、Times など)のみ表示されます。特定のフォントのバージョンはレポートされません。

LOCATION
検出されたデバイスエントリについて、Gdevice0-Gdevice9 ライブラリ、または Sashelp ライブラリ、または DEVLOC=オプションにより指定されているライブラリの物理的位置を表示します。ユニバーサルプリンターの場合、この変数では、プリンターが見つかった SAS レジストリ(SASHELP または SASUSER)が表示されます。

NAME
デバイスまたはプリンターの名前を表示します。

TYPE
デバイスまたはプリンターの種類を表示します。 デバイスとプリンターの種類は次のとおりです。

- グラフデバイス
- プリンターインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル
- ユニバーサルプレビューアー
- ユニバーサルプリンター

参照項目: “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

GENERAL レポート変数

詳細については、“GENERAL レポートの作成” (1625 ページ)を参照してください。

ALIAS
プロシジャにより登録されているフォントの別名をレポートします。

ANIMATION
アニメーションのアクティブ化、有効化、無効化、ユニバーサルプリンターへの非対応のいずれかを指定します。

Active	Active ODS HTML 出力のグラフィックがアニメーションで一緒にグループ化されていることを示します。Animate=Start と Animate=Stop は無視されます。
有効	アニメーションがサポートされていることを示します。アニメーションを開始するには Animate=Start を、停止するには Animate=Stop を指定してください。
Disabled	アニメーションはサポートされていますが、デバイスまたはプリンターには無効となっていることを示します。
Unsupported	アニメーションがデバイスまたはプリンターにサポートされないことを示します。

BMIN
下余白の最小サイズを表示します。

BOTTOM
下余白の現在のサイズを表示します。

CLRSPACE
カラーサポート(カラースペース)の種類(RGB、RGBA、CMYK、HLS など)を表示します。

参照項目: “Color-Naming Schemes” (*SAS/GRAPH: Reference*)

COLS
出力に横の列の数を表示します。

参照項目: “Cells” (*SAS/GRAPH: Reference*)

COMPRESSION
圧縮が使用される状況を示します。圧縮は常に有効にするか、圧縮オプションで指定された場合にのみ有効にするか、あるいはデバイスまたはプリンターで使用されないようにすることができます。

レポートの圧縮値は次のとおりです。

Always	圧縮が常に有効であることを示します。
オプション	圧縮が UPRINTCOMPRESSION システムオプションまたは ODS PRINTER ステートメントの COMPRESS=オプションにより指定されていることを示します。

Never　圧縮がデバイスまたはプリンターには絶対に使用されないよう指定します。

COMPMETHOD
デバイスまたはプリンターによって圧縮がサポートされる場合、使用する圧縮方法を示します。

注　Compression が Never でどの圧縮も使用できない場合は、SAS ログに Compression Method 値は表示されません。

DESC
デバイスまたはプリンターのデフォルト説明を表示します。

DEST
デバイスまたはプリンターが出力をプリンターまたはディスプレイデバイスに直接送信しない場合、デバイスまたはユニバーサルプリンターのデフォルト出力先を表示します。デバイスが出力をプリンターまたはディスプレイデバイスに直接送信する場合、DEST の値はブランクになります。

出力がモニターに送られる場合、または、Windows で、出力がプリンターに送られる場合、出力先がブランク値になることがあります。

EMBEDDING
フォントの埋め込みがサポートされるかどうかを示します。

ALWAYS　デバイスまたはプリンターのフォントの埋め込みが常に有効です。

OPTION　FONTEMBEDDING システムオプションで、フォントの埋め込みをサポートするかどうかが制御されます。

NEVER　フォントの埋め込みはサポートされません。

FHEIGHT
デフォルトフォントの高さを各単位で表示します。

注　フォント名をデバイスエントリの CHARREC リストから取得する場合、高さは使用できません。詳細については、“CHARREC= Device Parameter” (*SAS/GRAPH: Reference*)を参照してください。

FONT
デフォルトフォントの名前を表示します。

参照項目:　“Default Fonts” (*SAS/GRAPH: Reference*)

FORMAT
出力フォーマットタイプ(EMF、EMF Plus、EMF Dual、PostScript、GIF、Host Display など)を表示します。

参照項目:　“Commonly Used Devices” (*SAS/GRAPH: Reference*)

FSTYLE
デフォルトのフォントスタイル(Roman、Regular など)を表示します。

操作　出力を SAS ログに送る GENERAL レポートで FSTYLE 変数を指定した結果は、FONTS レポートに対して FSTYLE 変数を指定したときに得られる結果とは異なります。GENERAL レポートでは、フォントスタイルが SAS ログに

レポートされます。FONT レポートでは、SAS ログレポートにフォントファミリ名のみ表示されます。特定のフォントスタイルはレポートされません。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、スタイルは使用できません。詳細については"CHARREC= Device Parameter" (*SAS/GRAPH: Reference*)を参照してください。

FWEIGHT
デフォルトのフォントの太さ(Normal、Medium など)を表示します。

操作 出力を SAS ログに送る GENERAL レポートで FWEIGHT 変数を指定した結果は、FONTS レポートに対して FWEIGHT 変数を指定したときに得られる結果とは異なります。GENERAL レポートでは、フォントの太さが SAS ログにレポートされます。FONT レポートでは、SAS ログレポートにフォントファミリ名のみ表示されます。特定のフォントの太さはレポートされません。

注 フォント名をデバイスエントリの CHARREC リストから取得する場合、太さは使用できません。詳細については、"CHARREC= Device Parameter" (*SAS/GRAPH: Reference*)を参照してください。

FVERSION
フォントのバージョンを指定します。

HEIGHT
デバイスまたはプリンターに送信される出力のデフォルトの縦の高さ(UNITS 単位)を表示します。

HRES
デバイスまたはプリンターに送信される出力の水平解像度(pixels per UNIT)を表示します。水平解像度は式 HRES=XPIXELS/WIDTH によって計算されます。

操作 VAR ステートメントで HRES 変数か VRES 変数のどちらかを指定した場合、SAS ログでは水平解像度と垂直解像度がラベル XxY Resolution を使用して一緒に表示されます。出力データセットでは、HRES と VRES は別々にレポートされます。

参照項目: "Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices" (*SAS/GRAPH: Reference*)

IOTYPE
デバイスまたはプリンターにより使用される入力/出力のタイプ(DISK、PRINTER、PIPE、GTERM など)を表示します。

参照項目: "FILENAME ステートメント" (*SAS グローバルステートメント: リファレンス*) および"DEVTYPE= Device Parameter" (*SAS/GRAPH: Reference*)

LEFT
出力の左余白のサイズを表示します。

LMIN
最小左余白を表示します。

LOCATION
検出されたデバイスエントリについて、Gdevice0-Gdevice9 ライブラリ、または Sashelp ライブラリ、または DEVLOC=オプションにより指定されているライブラリの物理的位置を表示します。ユニバーサルプリンターの場合、この変数では、プリンターが見つかった SAS レジストリ(SASHELP または SASUSER)が表示されます。

MAXCOLORS
デバイスまたはプリンターでサポートされる色の最大数を表示します。

参照項目: "Maximum Number of Colors Displayed on a Device" (*SAS/GRAPH: Reference*)

MODULE
デバイスドライバーモジュールの名前を指定します。

NAME
デバイスまたはプリンターの名前を表示します。

PROTOTYPE
ユニバーサルプリンターの定義に使用するプロトタイプ(モデル)を表示します。

参照項目: "ユニバーサル印刷" (*SAS プログラマガイド: エッセンシャル*)

RIGHT
右余白のサイズを表示します。

RMIN
右余白の最小サイズを表示します。

ROWS
出力の縦の行の数を表示します。

参照項目: "Cells" (*SAS/GRAPH: Reference*)

TMIN
出力の最小上余白を表示します。

TOP
上余白のサイズを表示します。

TYPE
デバイスまたはプリンターの種類を表示します。 デバイスとプリンターの種類は次のとおりです。

- グラフデバイス
- プリンターインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル
- ユニバーサルプレビューアー
- ユニバーサルプリンター

参照項目: “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

UNITS
サイズを表示する際の単位(インチを示す IN かセンチを示す CM)を表示します。SAS ログでは、UNITS の値はそれぞれ inches か centimeters で表示されます。出力データセットでは、UNITS の値は IN か CM で表示されます。

操作 VAR ステートメントで、サイズ、余白または解像度の変数を指定しない場合、SAS ログでは、サイズ、余白または解像度の測定に使用する単位が示されます。次に、例を示します。

```
Name: EMF
Units: inches
```

VAR ステートメントで、サイズ、余白または解像度の変数を指定した場合、SAS ログでは値とともに単位が表示されます。次に、例を示します。

```
XxY Resolution: 96x96 pixels per inch
```

VISUAL
表示色の種類(Indexed Color、Direct Color、True Color、Monochrome、Gray Scale など)を表示します。

VRES
デバイスまたはプリンターに送信される出力の垂直解像度(pixels per UNIT)を表示します。

垂直解像度は式 VRES=YPIXELS/HEIGHT で計算されます。

操作 VAR ステートメントで HRES 変数か VRES 変数のどちらかを指定した場合、SAS ログでは水平解像度と垂直解像度がラベル XxY Resolution を使用して一緒に表示されます。出力データセットでは、HRES と VRES は別々にレポートされます。

参照項目: “Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” (*SAS/GRAPH: Reference*)

WIDTH
デバイスまたはプリンターに送信される出力の幅(UNITS 単位)を表示します。

XPIXELS
出力の幅をピクセル単位で表示します。

参照項目: “XPIXELS= Device Parameter and Graphics Option” (*SAS/GRAPH: Reference*)と“Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” (*SAS/GRAPH: Reference*)

YPIXELS
出力の高さをピクセル単位で表示します。

参照項目: “YPIXELS= Device Parameter and Graphics Option” (*SAS/GRAPH: Reference*)と“Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX=

Graphics Options to Set the Resolution for the Traditional Devices” (*SAS/GRAPH: Reference*)

LINESTYLE レポート変数

詳細については、“LINESTYLE レポートの作成” (1633 ページ)を参照してください。

DESC
デバイスまたはプリンターのデフォルト説明を表示します。

LINE
デバイスまたはプリンターでサポートされる線のスタイルを表示します。

操作 SAS ログ LINESTYLE レポートでは、LINE 変数と SUPPORT 変数が一緒にレポートされます。VAR ステートメントで LINE 変数か SUPPORT 変数のどちらかを指定すると、線のスタイルは、Supported Line Styles または Unsupported Line Styles の変数ラベルを使用してレポートされます。

参照項目: “Line Types” (*SAS/GRAPH: Reference*)

LOCATION
Gdevice0-Device9 物理的位置、またはデバイスエントリが検出された Devices カタログを含む Sashelp ライブラリの物理的位置、または DEVLOC=オプションにより指定されるライブラリの物理的位置を表示します。

NAME
デバイスの名前を表示します。

SUPPORT
デバイスラインのスタイルを表示します。

操作 SUPPORT 変数の値は、PROC QDEVICE ステートメントの SUPPORT=オプションに影響されます。SUPPORT=YES の場合、レポートにはサポートされている線のスタイルが表示されます。SUPPORT=NO の場合、レポートにはサポートされていない線のスタイルが表示されます。SUPPORT=ALL の場合、レポートには、サポートされている線のスタイルもサポートされていない線のスタイルもすべて表示されます。

TYPE
デバイスまたはプリンターの種類を表示します。

- グラフデバイス
- プリンターインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル

参照項目: “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

RECTANGLE レポート変数

詳細については、“RECTANGLE レポートの作成” (1634 ページ)を参照してください。

DESC
デバイスまたはプリンターのデフォルト説明を表示します。

FILL
デバイスでサポートされているハードウェアの塗りつぶしの種類を表示します。

操作 SAS ログ RECTANGLE レポートでは、FILL 変数と SUPPORT 変数が一緒にレポートされます。VAR ステートメントで FILL 変数または SUPPORT 変数を指定した場合、塗りつぶし名は、ラベル Supported Hardware Fills またはラベル Unsupported Hardware Fills のどちらかを使用してレポートされます。

参照項目: “PATTERN Statement” (*SAS/GRAPH: Reference*)

LOCATION
Gdevice0-Device9 物理的位置、またはデバイスエントリが検出された Devices カタログを含む Sashelp ライブラリの物理的位置、または DEVLOC=オプションにより指定されるライブラリの物理的位置を表示します。

NAME
デバイスの名前を表示します。

SUPPORT
ハードウェア塗りつぶしを表示します。

操作 SUPPORT 変数の値は、PROC QDEVICE ステートメントの SUPPORT=オプションに影響されます。SUPPORT=YES の場合、レポートにはサポートされているハードウェア塗りつぶしが表示されます。SUPPORT=NO の場合、レポートにはサポートされていないハードウェア塗りつぶしが表示されます。SUPPORT=ALL の場合、レポートには、サポートされているハードウェア塗りつぶしもサポートされていないハードウェア塗りつぶしもすべて表示されます。

TYPE
デバイスまたはプリンターの種類を表示します。

次に種類のリストを示します。

- グラフデバイス
- プリンターインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル

参照項目: “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

SYMBOL レポート変数

詳細については、“SYMBOL レポートの作成” (1636 ページ)を参照してください。

DESC
デバイスまたはプリンターのデフォルト説明を表示します。

LOCATION
Gdevice0-Device9 物理的位置、またはデバイスエントリが検出された Devices カタログを含む Sashelp ライブラリの物理的位置、または DEVLOC=オプションにより指定されるライブラリの物理的位置を表示します。

NAME
デバイスの名前を表示します。

SUPPORT
デバイスまたはプリンターのシンボルを表示します。

操作 SUPPORT 変数の値は、PROC QDEVICE ステートメントの SUPPORT=オプションに影響されます。SUPPORT=YES の場合、レポートにはサポートされているシンボルのスタイルが表示されます。SUPPORT=NO の場合、レポートにはサポートされていないシンボルが表示されます。SUPPORT=ALL の場合、レポートには、サポートされているシンボルもサポートされていないシンボルもすべて表示されます。

SYMBOL
ハードウェアシンボルの名前を指定します。

操作 SAS ログ SYMBOL レポートでは、VAR ステートメントで SYMBOL 変数か SUPPORT 変数のどちらかを指定した場合、シンボル名は、Supported Hardware Symbols ラベルまたは Unsupported Hardware Symbols ラベルを使用してレポートされます。

参照項目: “SYMBOL, NOSYMBOL Graphics Options” (*SAS/GRAPH: Reference*)と “SYMBOLS= Device Parameter” (*SAS/GRAPH: Reference*)

TYPE
デバイスまたはプリンターの種類を表示します。

次に種類のリストを示します。

- グラフデバイス
- プリンターインターフェイスデバイス
- ショートカットデバイス
- システムディスプレイ
- システムメタファイル

参照項目: “Device Categories and Modifying Default Output Attributes” (*SAS/GRAPH: Reference*)

使用法: QDEVICE プロシジャ

全レポート共通の変数

次の変数は、どのレポートでも使用できます。

- NAME
- DESC
- TYPE
- LOCATION

変数の説明については、“すべてのレポートに有効な変数” (1613 ページ)を参照してください。

GENERAL レポートの作成

GENERAL レポートでは、指定したデバイスまたはプリンターに関する一般情報のレポートが作成されます。この情報には、余白サイズ、デフォルトフォント情報、解像度、色情報が含まれます。

GENERAL レポート変数について

変数の説明については、“GENERAL レポート変数” (1617 ページ)を参照してください。

次のテーブルに、GENERAL レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこのテーブルの表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	説明	DESCRIPTION OF DEVICE OR PRINTER

変数	SAS ログラベル	出力データセットラベル
MODULE	Module	DRIVER MODULE
TYPE	タイプ	TYPE OF DEVICE OR PRINTER
LOCATION	デバイスには"Device Catalog" プリンターには"Registry"	LOCATION OF DEVICE OR PRINTER DEFINITION
PROTOTYPE	Prototype	PRINTER PROTOTYPE
FONT	Default Typeface	FONT TYPEFACE DEFAULT
ALIAS	Typeface Alias	FONT TYPEFACE ALIAS
FSTYLE	Font Style	FONT STYLE DEFAULT
FWEIGHT	Font Weight	FONT WEIGHT DEFAULT
FHEIGHT	Font Height	FONT HEIGHT DEFAULT
FVERSION	Font Version	FONT VERSION
MAXCOLORS	Maximum Colors	MAXIMUM NUMBER OF SUPPORTED COLORS
VISUAL	Visual Color	TYPE OF VISUAL COLOR
CLRSPACE	Color Support	TYPE OF COLOR SUPPORT
DEST	Destination	OUTPUT DESTINATION DEFAULT
IOTYPE	I/O Type	TYPE OF I/O DEFAULT
FORMAT	Data Format	OUTPUT DATA FORMAT
HEIGHT	Height	HEIGHT OF OUTPUT
WIDTH	幅	WIDTH OF OUTPUT
UNITS	Units[1]	UNITS FOR SIZE, MARGINS AND RESOLUTION
YPIXELS	Ypixels	VERTICAL PIXELS

変数	SAS ログラベル	出力データセットラベル
XPIXELS	Xpixels	HORIZONTAL PIXELS
ROWS	Rows (vpos)	ROWS
COLS	Columns (hpos)	COLUMNS
LEFT	Left Margin	LEFT MARGIN
LMIN	Minimum Left Margin	MINIMUM LEFT MARGIN
RIGHT	Right Margin	RIGHT MARGIN
RMIN	Minimum Right Margin	MINIMUM RIGHT MARGIN
BOTTOM	Bottom Margin	BOTTOM MARGIN
BMIN	Minimum Bottom Margin	MINIMUM BOTTOM MARGIN
TOP	Top Margin	TOP MARGIN
TMIN	Minimum Top Margin	MINIMUM TOP MARGIN
HRES	XxY Resolution	HORIZONTAL PIXELS PER UNIT
VRES	XxY Resolution	VERTICAL PIXELS PER UNIT
COMPRESSION	Compression Enabled	COMPRESSION ENABLED
COMPMETHOD	Compression Method	COMPRESSION METHOD
EMBEDDING	Font Embedding	FONT EMBEDDING SUPPORT
ANIMATION	アニメーション	ANIMATION SUPPORT

1 単位の種類が変数の値とともに表示される場合(左余白の 0 inches など)、SAS ログへの出力には Units ラベルが表示されません。

サイズ変数の値に影響を及ぼすシステムオプション

ユニバーサルプリンターの場合、HEIGHT、WIDTH、LEFT、RIGHT、BOTTOM、TOP、LMIN、RMIN、BMIN、TMIN 変数の値は、PAPERSIZE、LEFTMARGIN、RIGHTMARGIN、BOTTOMMARGIN、TOPMARGIN SAS システムオプションの設定に影響を受けます。デフォルトの用紙のサイズは、SAS ロケール(ユニバーサルプリンターのデフォルトサイズに影響を及ぼす)によって決定されます。SAS/GRAPH デバイスの場合、これらの変数値はシステムオプション設定の影響を受けません。

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

例: GENERAL レポート

次の QDEVICE プロシジャでは、SVG ユニバーサルプリンター用の GENERAL レポートが作成されます。

```
proc qdevice;
  printer svg;
run;
```

SAS ログの GENERAL レポートを次に示します。

```
78   proc qdevice; printer svg; run;
                Name: SVG
         Description: Scalable Vector Graphics 1.1
              Module: SASPDSVG
                Type: Universal Printer
            Registry: SASHELP
           Prototype: SVG 1.1
     Default Typeface: Cumberland AMT
      Typeface Alias: Courier
          Font Style: Regular
         Font Weight: Normal
         Font Height: 8 points
        Font Version: Version 1.03
      Maximum Colors: 16777216
        Visual Color: Direct Color
       Color Support: RGBA
         Destination: sasprt.svg
           I/O Type: DISK
         Data Format: SVG
              Height: 6.25 inches
               Width: 8.33 inches
             Ypixels: 600
             Xpixels: 800
          Rows(vpos): 50
       Columns(hpos): 114
         Left Margin: 0 inches
 Minimum Left Margin: 0 inches
        Right Margin: 0 inches
 Minimum Right Margin: 0 inches
       Bottom Margin: 0 inches
Minimum Bottom Margin: 0 inches
          Top Margin: 0 inches
   Minimum Top Margin: 0 inches
      XxY Resolution: 96x96 pixels per inch
 Compression Enabled: Never
  Compression Method: Deflate
     Font Embedding: Option
          Animation: Enabled
```

FONT レポートの作成

FONT レポートでは、指定したデバイスまたはプリンターでサポートされるすべてのシステムフォントおよびデバイス 常駐フォントのレポートが作成されます。

FONT レポート変数について

変数の説明については、“FONT レポート変数” (1615 ページ)を参照してください。

次のテーブルに、FONT レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこのテーブルの表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	説明	DESCRIPTION OF DEVICE OR PRINTER
TYPE	タイプ	TYPE OF DEVICE OR PRINTER
LOCATION	デバイスには"Device Catalog" プリンターには"Registry"	LOCATION OF DEVICE OR PRINTER DEFINITION
FONT	フォントまたは種類に依存	FONT TYPEFACE
ALIAS	フォント名の右に(alias: *alias*)	FONT TYPEFACE ALIAS
FTYPE	フォントの種類に依存	FONT TYPE
FSTYLE	なし	FONT STYLE
FWEIGHT	なし	FONT WEIGHT
FVERSION	なし	FONT VERSION

FONT レポートの変数ラベル

FONT レポートで FONT、FTYPE、FSTYLE、FWEIGHT 変数のいずれかを指定した場合、SAS ログに表示される変数ラベルは変化します。出力データセットの変数ラベルは常に同じで、それぞれ FONT TYPEFACE DEFAULT、FONT TYPE、FONT STYLE DEFAULT、FONT WEIGHT DEFAULT です。

VAR ステートメントにより変数 FONT、FTYPE、FSTYLE、FWEIGHT、FVERSION のうち 1 つ以上が指定されている場合、SAS ログではフォントタイプラベルとフォントファミリ名のみがレポートされます。フォントのスタイル、太さ、バージョンは SAS ログにはレポートされません。表示されるフォントタイプラベルはフォントタイプに依存します。ラベルの例としては、Supported Font Typefaces、Supported Resident Typefaces、Supported TrueType Typefaces、Supported Type1 Typefaces などがあります。

例: FONT レポート

次の QDEVICE プロシジャでは、PDF グラフィックデバイス用の FONT レポートが作成されます。

```
proc qdevice report=font;
```

```
  device activex;
run;
```

SAS ログの FONT レポートの一部を次に示します。

```
                Name: PDF
         Description: Portable Document Format Version 1.5
                Type: Shortcut Device
      Device Catalog: your-SAS-path/sashelp
Supported Resident Typefaces: Adobe Caslon
                              Adobe Caslon Oldstyle
                              Adobe Caslon Small Caps
                              Adobe Caslon Small Caps Oldstyle
                              Adobe Garamond
                              Adobe Garamond Oldstyle
                              Adobe Garamond Small Caps Oldstyle
                              Adobe Garamond Titling
                              Adobe Wood Type Ornaments One
                              Adobe Wood Type Ornaments Two
                              Albertus MT
                              Americana
                              Amigo
                              Antique Olive
                              Antique Olive Compact
                              Antique Olive Condensed
                              AvantGarde-Book
                              AvantGarde-BookOblique
                              AvantGarde-Demi
                              AvantGarde-DemiOblique
                              Barmeno
                              Bernhard Modern
                              Birch
                              Blackoak
                              Bodoni
                              Bodoni Poster
                              Bodoni Poster Compressed
                              Bookman-Demi
                              Bookman-DemiItalic
                              Bookman-Light
                              Bookman-LightItalic
                              Carta
                              Clarendon
```

DEVOPTION レポートの作成

DEVOPTION レポートでは、指定したデバイスでサポートされるハードウェアデバイスオプションのレポートが作成されます。このレポートは、ユニバーサルプリンターでは使用できません。

DEVOPTION レポート変数について

変数の説明については、“DEVOPTION レポート変数” (1614 ページ)を参照してください。

次のテーブルに、DEVOPTION レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこのテーブルの表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	説明	DESCRIPTION OF DEVICE OR PRINTER
TYPE	タイプ	TYPE OF DEVICE OR PRINTER
LOCATION	Device Catalog	LOCATION OF DEVICE OR PRINTER DEFINTION
BIT	Bit Position	DEVICE OPTION BIT POSITON
BITSTRING	Bit Pattern	DEVICE OPTION BIT PATTERN
OPTION	Device Option	DEVICE OPTION NAME
ODESC	Option Description	DEVICE OPTION DESCRIPTION
SUPPORT	Support	DEVICE OPTION SUPPORT

例: DEVOPTION レポート

次の QDEVICE プロシジャでは、SASEMF グラフィックデバイス用の DEVOPTION レポートが作成されます。SUPPORT のデフォルトが YES なので、サポートされているオプションもレポートされます。

```
proc qdevice report=devoption;
  device sasemf;
run;
```

SAS ログの l レポートリストの一部を次に示します。

```
          Name: SASEMF
   Description: Version 9.3 Enhanced Metafile Format
          Type: Shortcut Device
Device Catalog: your-SAS-path/sashelp
    Bit Position: 0
     Bit Pattern: 8000000000000000
   Device Option: GDCIRCLEARC
Option Description: Hardware is capable of drawing circles
         Support: Yes
    Bit Position: 1
     Bit Pattern: 4000000000000000
   Device Option: GDPIEFILL
Option Description: Device has hardware pie-fill capability
         Support: Yes
    Bit Position: 3
     Bit Pattern: 1000000000000000
   Device Option: GDCRT
Option Description: Hardware is a CRT or the device acts like a CRT
         Support: Yes
    Bit Position: 5
     Bit Pattern: 0400000000000000
   Device Option: GDPOLYGONFILL
Option Description: Device has polygonfill capability
         Support: Yes
    Bit Position: 7
     Bit Pattern: 0100000000000000
   Device Option: GDRGB
Option Description: Hardware is capable of defining colors in one or more color spaces
         Support: Yes
    Bit Position: 8
     Bit Pattern: 0080000000000000
   Device Option: GDMBPOLY
Option Description: Hardware can draw polygons with multiple boundaries
         Support: Yes
```

LINESTYLE レポートの作成

LINESTYLE レポートでは、指定したデバイスでサポートされるハードウェアの(点)線のスタイルのレポートが作成されます。

LINESTYLE レポート変数について

変数の説明については、“LINESTYLE レポート変数” (1622 ページ)を参照してください。

次のテーブルに、LINESTYLE レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこのテーブルの表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER

変数	SAS ログラベル	出力データセットラベル
DESC	説明	DESCRIPTION OF DEVICE OR PRINTER
TYPE	タイプ	TYPE OF DEVICE OR PRINTER
LOCATION	Device Catalog	LOCATION OF DEVICE OR PRINTER DEFINTION
LINE	Supported Line Styles Unsupported Line Styles	HARDWARE DASHED LINE NUMBER
SUPPORT	Supported Line Styles Unsupported Line Styles	HARDWARE DASHED LINE SUPPORT

例: LINESTYLE レポート

次の QDEVICE プロシジャでは、LJ5PS デバイス用の LINESTYLE レポートが作成されます。SUPPORT のデフォルトが YES なので、サポートされている線のスタイルもレポートされます。

```
proc qdevice report=linestyle;
  device svg;
run;
```

SAS ログの LINESTYLE レポートを次に示します。

```
          Name: SVG
   Description: Scalable Vector Graphics 1.1
          Type: Shortcut Device
Device Catalog: your-SAS-path/sashelp
Supported Line Styles: 1-46
```

RECTANGLE レポートの作成

RECTANGLE レポートでは、指定したデバイスでサポートされるハードウェアの塗りつぶしの種類のレポートが作成されます。

RECTANGLE レポート変数について

変数の説明については、“RECTANGLE レポート変数” (1623 ページ)を参照してください。

次のテーブルに、RECTANGLE レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこのテーブルの表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	説明	DESCRIPTION OF DEVICE or PRINTER
TYPE	タイプ	TYPE OF DEVICE OR PRINTER
LOCATION	Device Catalog	LOCATION OF DEVICE OR PRINTER DEFINTION
FILL	Supported Hardware Fills	HARDWARE RECTANGLE FILL NAME
SUPPORT	Supported Hardware Fills Unsupported Hardware Fills	HARDWARE RECTANGLE FILL SUPPORT

例: RECTANGLE レポート

次の QDEVICE プロシジャでは、SASPRTG ユニバーサルプリンター用の RECTANGLE レポートが作成されます。SUPPORT のデフォルトが YES なので、サポートされているハードウェア塗りつぶしもレポートされます。

```
proc qdevice report=rectangle;
  device sasprtg;
run;
```

SAS ログの RECTANGLE レポートを次に示します。

```
            Name: SASPRTG
     Description: POSTSCRIPT LEVEL 1
            Type: Printer Interface Device
  Device Catalog: your-sas-path\sashelp
Supported Hardware Fills: Empty,Solid
```

SASPRTG はプリンターインターフェイスデバイスです。ユニバーサル印刷がアクティブなので、SASPRTG はデフォルトユニバーサルプリンターとインターフェイスでつながっています。したがって、レポートにはユニバーサルプリンターとして PostScript Level 1 プリンターの情報が表示されます。

SYMBOL レポートの作成

SYMBOL レポートでは、指定したデバイスでサポートされるハードウェアシンボルのレポートが作成されます。

SYMBOL レポート変数について

変数の説明については、“SYMBOL レポート変数” (1624 ページ)を参照してください。

次のテーブルに、SYMBOL レポートで使用可能な変数に加えて、SAS ログか出力データセットのどちらかで使用する変数のラベルを示します。VAR ステートメントを指定しなかった場合、変数はこのテーブルの表示順序で表示されます。

変数	SAS ログラベル	出力データセットラベル
NAME	Name	NAME OF DEVICE OR PRINTER
DESC	説明	DESCRIPTION OF DEVICE OR PRINTER
TYPE	タイプ	TYPE OF DEVICE OR PRINTER
LOCATION	Device Catalog	LOCATION OF DEVICE OR PRINTER DEFINTION
SYMBOL	Supported Hardware Symbols Unsupported Hardware Symbols	HARDWARE SYMBOL NAME
SUPPORT	Support	DEVICE OPTION SUPPORT

例: SYMBOL レポート

次の QDEVICE プロシジャでは、CGM デバイス用の SYMBOL レポートが作成されます。SUPPORT のデフォルトが YES なので、サポートされているハードウェアシンボルもレポートされます。

```
proc qdevice report=symbol;
  device cgm;
run;
```

SAS ログの SYMBOL レポートを次に示します。

```
            Name: CGM
     Description: CGM generator--binary output
            Type: Graph Device
  Device Catalog: your-sas-path\sashelp
Supported Hardware Symbols: Plus,X,Star
```

例: QDEVICE プロシジャ

例 1: デフォルトのディスプレイデバイスのレポートの生成

要素: PROC QDEVICE

詳細

次の例では、デフォルトディスプレイデバイス用の一般レポートを作成します。

プログラム

```
proc qdevice;
run;
```

ログ

OUT=オプションを指定しなかった場合、QDEVICE プロシジャでは、SAS ログに出力が送信されます。

例のコード 47.1　PROC QDEVICE 実行後の SAS ログ

```
NOTE: No DEVICE or PRINTER statement was specified.The default "PDF" printer will be used.
              Name: PDF
       Description: Portable Document Format Version 1.5
            Module: SASPDPDF
              Type: Universal Printer
          Registry: SASHELP
         Prototype: PDF Version 1.5
   Default Typeface: Cumberland AMT
     Typeface Alias: Courier
        Font Style: Regular
       Font Weight: Normal
       Font Height: 8 points
      Font Version: Version 1.03
    Maximum Colors: 16777216
      Visual Color: Direct Color
     Color Support: RGBA
       Destination: sasprt.pdf
         I/O Type: DISK
       Data Format: PDF
            Height: 10.5 inches
             Width: 8 inches
           Ypixels: 3150
           Xpixels: 2400
       Rows(vpos): 85
    Columns(hpos): 120
       Left Margin: 0.25 inches
Minimum Left Margin: 0 inches
      Right Margin: 0.25 inches
Minimum Right Margin: 0 inches
     Bottom Margin: 0.25 inches
Minimum Bottom Margin: 0 inches
        Top Margin: 0.25 inches
Minimum Top Margin: 0 inches
     XxY Resolution: 300x300 pixels per inch
Compression Enabled: Option
Compression Method: Deflate
    Font Embedding: Option
         Animation: Unsupported
```

例 2: 全デバイスの GENERAL レポートの生成

要素:　PROC QDEVICE ステートメントオプション: OUT=
DEVICE ステートメント

詳細

次の例では、全デバイスの一般レポートを作成し、その結果を WORK.ALLDEVICES に書き込みます。

全デバイスのレポートを生成するには、_ALL_キーワードを使用します。

プログラム

```
proc qdevice out=allDevices;
  device _all_;
run;
```

出力

次の図は、**Viewtable** ウィンドウに表示されるレポートの一部です。

アウトプット 47.1　全デバイスの出力データセットレポート

	NAME	DESC	MOD...	TYPE
1	BMP	BMP File Fmt--256 colors	SASGDIMG	Graph Device
2	CGM	CGM generator--binary output	SASGDC...	Graph Device
3	CGMC	CGM generator w/colors--binary output	SASGDC...	Graph Device
4	CGMOF93L	CGM for MS landscape w/o hardware li...	SASGDC...	Graph Device
5	CGMOF93P	CGM for MS portrait w/o hardware lines	SASGDC...	Graph Device
6	CGMOF97L	CGM for Office 97 landscape mode	SASGDC...	Graph Device
7	CGMOF97P	CGM for Microsoft Office 97 portrait	SASGDC...	Graph Device
8	CGMOFML	CGM for MS Office w/ Metric Scaling land	SASGDC...	Graph Device
9	CGMOFMP	CGM for MS Office w/ Metric Scaling port	SASGDC...	Graph Device
10	CLJ	HP Color LaserJet Printer--Letter Size	SASGDHPL	Graph Device
11	CLJPS	HP Color LaserJet--PostScript--Letter	SASGDPSL	Graph Device
12	CLJPSA4	HP Color LaserJet--PostScript--A4 size	SASGDPSL	Graph Device
13	DJT2C	DesignJet 2000 Series 24" color HPGL/2	SASGDHPL	Graph Device

例 3: SAS/GRAPH デバイスドライバーとユニバーサルプリンターのレポートの生成

要素:
PROC QDEVICE ステートメントオプション: OUT=
DEVICE ステートメント
PRINTER ステートメント

詳細

次の例では、EMF に落ち着くすべてのデバイス、PDF および SVG?ユニバーサルプリンター向けの一般レポートを作成します。結果は WORK.MYREPORT データセットに書き込まれます。REPORT=オプションを指定しなかった場合、QDEVICE プロシジャでは、一般レポートが生成されます。

プログラム

```
proc qdevice out=myreport;
  device '*emf';
  printer pdf 'svg?';
run;
```

出力

次の図は、**Viewtable** ウィンドウに表示されるレポートの一部です。

アウトプット 47.2　EMF デバイス、PDF プリンター、SVG プリンターの出力データセットレポート

Obs	NAME	DESC	MODULE	TYPE
1	EMF	Enhanced Metafile Format Plus Extensions	SASGDDMX	Shortcut Device
2	SASEMF	Version 9.3 Enhanced Metafile Format	SASGDDMX	Shortcut Device
3	PDF	Portable Document Format Version 1.5	SASPDPDF	Universal Printer
4	SVGT	Transparent SVG	SASPDSVG	Universal Printer
5	SVGZ	Compressed Scalable Vector Graphics 1.1	SASPDSVG	Universal Printer

例 4: デフォルトプリンターのレポートの生成

要素: PROC QDEVICE ステートメント
PRINTER ステートメント

詳細

デフォルトでは、Windows 環境の SAS での印刷は、ユニバーサル印刷ではなく、デフォルトの Windows プリンターによって実行されます。したがって、QDEVICE プロシジャで printer _PRINTER_ステートメントを使用した場合は、異なる結果が出ます。Windows 環境では、NOUPRINT システムオプションがデフォルトで、レポートはデフォルトの Windows プリンターとつながるプリンターインターフェイスデバイスに基づきます。UNIX 環境では、UPRINT システムオプションが設定されているため、レポートはデフォルトの SAS ユニバーサルプリンターに基づきます。

REPORT=オプションを指定しないので、QDEVICE プロシジャでは、一般レポートが生成されます。OUT=オプションを指定しなければ、結果は SAS ログに書き込まれます。_PRINTER_キーワードによって、レポート対象のデフォルトプリンターが決定され、そのプリンターのレポートが生成されます。

詳細については、“Setting Up and Managing Universal Printing” (*SAS Viya Platform Universal Printing*)を参照してください

プログラム: Windows

```
proc qdevice;
  printer _PRINTER_;
run;
```

ログ: デフォルトプリンターレポート

例のコード 47.2　デフォルトプリンターの SAS ログ出力レポート

```
          Name: PostScript Level 1
   Description: Generic PostScript Level 1 Printer
        Module: SASPDPSL
          Type: Universal Printer
      Registry: SASHELP
     Prototype: PostScript Level 1 (Color)
Default Typeface: Cumberland AMT
 Typeface Alias: Courier
    Font Style: Regular
   Font Weight: Normal
   Font Height: 8 points
  Font Version: Version 1.03
 Maximum Colors: 16777216
  Visual Color: Direct Color
 Color Support: CMYK
   Destination: sasprt.ps
     I/O Type: DISK
   Data Format: PostScript
        Height: 10 inches
         Width: 7.5 inches
       Ypixels: 3000
       Xpixels: 2250
    Rows(vpos): 81
 Columns(hpos): 112
   Left Margin: 0.5 inches
Minimum Left Margin: 0 inches
  Right Margin: 0.5 inches
Minimum Right Margin: 0 inches
  Bottom Margin: 0.5 inches
Minimum Bottom Margin: 0 inches
    Top Margin: 0.5 inches
Minimum Top Margin: 0 inches
  XxY Resolution: 300x300 pixels per inch
Compression Enabled: Never
  Font Embedding: Option
     Animation: Unsupported
```

プログラム: UNIX

```
proc qdevice;
  printer _PRINTER_;
run;
```

ログ :UNIX 環境のデフォルトユニバーサルプリンターレポート

例のコード 47.3　UNIX 環境でのデフォルトユニバーサルプリンターの SAS ログ出力レポート

```
NOTE: The "PostScript Level 1" printer will be used by default with the ODS PRINTER destination.
                Name: PostScript Level 1
         Description: Generic PostScript Level 1 Printer
              Module: SASPDPSL
                Type: Universal Printer
            Registry: SASHELP
           Prototype: PostScript Level 1 (Color)
     Default Typeface: Cumberland AMT
      Typeface Alias: Courier
          Font Style: Regular
         Font Weight: Normal
         Font Height: 8 points
        Font Version: Version 1.03
      Maximum Colors: 16777216
        Visual Color: Direct Color
       Color Support: CMYK
         Destination: sasprt.ps
             I/O Type: DISK
         Data Format: PostScript
              Height: 10 inches
               Width: 7.5 inches
             Ypixels: 3000
             Xpixels: 2250
         Rows(vpos): 81
       Columns(hpos): 112
         Left Margin: 0.5 inches
  Minimum Left Margin: 0 inches
        Right Margin: 0.5 inches
  Minimum Right Margin: 0 inches
       Bottom Margin: 0.5 inches
Minimum Bottom Margin: 0 inches
          Top Margin: 0.5 inches
   Minimum Top Margin: 0 inches
      XxY Resolution: 300x300 pixels per inch
 Compression Enabled: Never
       Font Embedding: Option
            Animation: Unsupported
```

例 5: FONT レポートの生成

要素:

PROC QDEVICE ステートメントオプション
- REPORT=
- OUT=

PRINTER ステートメント

詳細

最初の例では、プリンターで使用可能なすべてのプリンター常駐フォントおよびシステムフォントのレポートが生成されます。結果は Work.Myfonts データセットに書き込まれます。

2 番目の例では、SAS プログラムで、マクロ、DATA ステップおよび PRINT プロシジャを使用して、デバイスのフォントリストが作成されます。

プログラム

```
proc qdevice report=font out=myfonts;
  printer 'PDF';
run;
```

出力

次の出力は、レポートの最初の行を示しています。

アウトプット 47.3　フォントレポートの出力データセット

Obs	NAME	DESC	TYPE	LOCATION	FONT	ALIAS	FTYPE	FSTY
1	PDF	Portable Document Format Version 1.5	Universal Printer	SASHELP	Adobe Caslon		Printer Resident	Italic

プログラム

```
/* Macro FONTLIST - Report fonts supported by a device */

%macro fontlist(type, name);

proc qdevice report=font out=fonts;
  &type &name;
  var font ftype fstyle fweight;
run;

data;
  set fonts;
  drop ftype;
  length type $16;
  if ftype = "System"
   then do;
     if substr(font,2,3) = "ttf" then type = "TrueType";
       else if substr(font,2,3) = "at1" then type = "Adobe Type1";
       else if substr(font,2,3) = "cff" then type = "Adobe CFF/Type2";
```

```
        else if substr(font,2,3) = "pfr" then type = "Bitstream PFR";
        else type = "System";
      if type ^= "System" then font = substr(font,7,length(font)-6);
        else if substr(font,1,1) = "@"
            then font = substr(font, 2,length(font)-1);
      end;
      else type = "Printer Resident";
run;

proc sort;
  by font;
run;

title "Fonts Supported by the %upcase(&name) &type";

proc print label;
  label fstyle="Style" fweight="Weight" font="Font" type="Type";
run;


%mend fontlist;



%fontlist(device, pdf)
```

プログラムの説明

マクロフォントリストを作成します。 %macro ステートメントで、マクロが開始されます。マクロに対する入力は、(デバイスでもプリンターでも)種類と、デバイスまたはプリンターの名前です。

```
/* Macro FONTLIST - Report fonts supported by a device */

%macro fontlist(type, name);
```

デバイスに対するデータセット fonts を作成します。 マクロ入力変数 type と name を使用して、QDEVICE プロシジャでフォントレポートを作成します。出力はデータセット fonts に書き込まれます。

```
proc qdevice report=font out=fonts;
  &type &name;
  var font ftype fstyle fweight;
run;
```

フォントの種類を分類します。 フォントは、System、TrueType、Adobe Type1、Adobe CFF/Type2、Bitstream PFR、または Printer Resident のいずれかの種類になります。

```
data;
  set fonts;
  drop ftype;
  length type $16;
  if ftype = "System"
   then do;
     if substr(font,2,3) = "ttf" then type = "TrueType";
       else if substr(font,2,3) = "at1" then type = "Adobe Type1";
```

```
      else if substr(font,2,3) = "cff" then type = "Adobe CFF/Type2";
      else if substr(font,2,3) = "pfr" then type = "Bitstream PFR";
      else type = "System";
     if type ^= "System" then font = substr(font,7,length(font)-6);
      else if substr(font,1,1) = "@"
          then font = substr(font, 2,length(font)-1);
    end;
    else type = "Printer Resident";
run;
```

フォントデータセットをフォント名順に並べ替えます。

```
proc sort;
  by font;
run;
```

デバイスまたはプリンターのフォントを印刷します。

```
title "Fonts Supported by the %upcase(&name) &type";

proc print label;
  label fstyle="Style" fweight="Weight" font="Font" type="Type";
run;
```

マクロを終了します。

```
%mend fontlist;
```

マクロ&fontlist を使用して、PCL5c デバイスのデータセットを作成、出力します。

```
%fontlist(device, pdf)
```

出力

アウトプット 47.4　PDF デバイスでサポートされるフォントの表示の一部

Fonts Supported by the PDF device

Obs	Font	Style	Weight	Type
1	Adobe Caslon	Italic	Bold	Printer Resident
2	Adobe Caslon	Italic	Normal	Printer Resident
3	Adobe Caslon	Italic	Semi Bold	Printer Resident
4	Adobe Caslon	Roman	Bold	Printer Resident
5	Adobe Caslon	Roman	Normal	Printer Resident
6	Adobe Caslon	Roman	Semi Bold	Printer Resident
7	Adobe Caslon Oldstyle	Italic	Bold	Printer Resident
8	Adobe Caslon Oldstyle	Italic	Normal	Printer Resident
9	Adobe Caslon Oldstyle	Italic	Semi Bold	Printer Resident
10	Adobe Caslon Oldstyle	Roman	Bold	Printer Resident
11	Adobe Caslon Oldstyle	Roman	Normal	Printer Resident
12	Adobe Caslon Oldstyle	Roman	Semi Bold	Printer Resident
13	Adobe Caslon Small Caps	Roman	Semi Bold	Printer Resident
14	Adobe Caslon Small Caps Oldstyle	Roman	Normal	Printer Resident
15	Adobe Garamond	Italic	Bold	Printer Resident
16	Adobe Garamond	Italic	Normal	Printer Resident
17	Adobe Garamond	Italic	Semi Bold	Printer Resident
18	Adobe Garamond	Roman	Bold	Printer Resident
19	Adobe Garamond	Roman	Normal	Printer Resident
20	Adobe Garamond	Roman	Semi Bold	Printer Resident

例 6: デバイスオプションレポートの生成

要素:

PROC QDEVICE ステートメントオプション
- REPORT=
- SUPPORT=
- OUT=

DEVICE ステートメント

詳細

次の例では、PNG デバイス向けデバイスオプション(DEVOPTIONS)レポートを作成します。レポートは出力データセットに書き込まれます。

プログラム

```
proc qdevice report=devoption support=all out=devop;
  device pdf;
run;

proc print data=devop;
   var bit bitstring option odesc;
   where Support="Yes";
   title "Supported PDF Device Options";
run;
```

プログラムの説明

PDF デバイスのオプションをレポートします。 REPORT=DEVOPTION オプションは、デバイスオプションレポートの作成を指定するものです。SUPPORT=ALL は、すべてのデバイス 機能をレポートするよう指定するものです。オプション OUT=DEVOP は、データセット WORK.DEVOP を作成するものです。DEVICE PDF ステートメントは、PDF デバイスに関してレポートするよう指定するものです。

```
proc qdevice report=devoption support=all out=devop;
  device pdf;
run;
```

デバイスオプションレポートを印刷します。 WORK.DEVOP データセットを印刷すると、印刷されたレポートには、Support="Yes"であるデバイスオプションの BIT 変数、BITSTRING 変数、OPTION 変数、ODESC 変数が表示されます。

```
proc print data=devop;
   var bit bitstring option odesc;
   where Support="Yes";
   title "Supported PDF Device Options";
run;
```

出力

アウトプット 47.5　PDF デバイス向け出力データセット

Supported PDF Device Options

Obs	BIT	BITSTRING	OPTION	ODESC
1	0	8000000000000000	GDCIRCLEARC	Hardware is capable of drawing circles
2	1	4000000000000000	GDPIEFILL	Device has hardware pie-fill capability
4	3	1000000000000000	GDCRT	Hardware is a CRT or the device acts like a CRT
5	4	0800000000000000	GDTRANSPARENCY	Device supports transparency
6	5	0400000000000000	GDPOLYGONFILL	Device has polygonfill capability
8	7	0100000000000000	GDRGB	Hardware is capable of defining colors in one or more color spaces
9	8	0080000000000000	GDMBPOLY	Hardware can draw polygons with multiple boundaries
10	9	0040000000000000	GDOPACITY	Hardware is capable of supporting opacity
13	14	0002000000000000	GDHRDCHR	Hardware characters are supported by the device
14	15	0001000000000000	GDXLIMIT	There is no limit on max value allowed for x coordinate
15	16	0000800000000000	GDYLIMIT	There is no limit on max value allowed for y coordinate
17	18	0000200000000000	GDTXJUSTIFY	Hardware is capable of justifying proportional text
20	24	0000008000000000	GDUNICODE	Device supports the use of the Unicode font attribute
21	25	0000004000000000	GDPOLYLINE	Hardware is capable of supporting polylines
24	28	0000000800000000	GDTRUETYPE	Device supports the use of TrueType fonts
32	36	0000000008000000	GDIMAGE	Device is capable of drawing images
35	39	0000000001000000	GDIMGROTATE	Device is incapable of doing image rotation
36	40	0000000000800000	GDTRUECOLOR	Hardware is a 24-bit true color device
37	41	0000000000400000	GDFONTATTR	Device supports setting font attributes
38	42	0000000000200000	GDSCANLNFNT	Device will use scan line font rendering
40	44	0000000000080000	GDTEXTCLIP	Hardware will clip text at the device limits
46	50	0000000000002000	GDAUTOSIZE	Autosize text to fit rows and columns
50	54	0000000000000200	GDPOLYOUTLINE	Device draws polygon outlines
51	55	0000000000000100	GDPRINTERPATH	Device temporarily sets printerpath to that of the device name
52	56	0000000000000080	GDOPTPASSTHRU	PAPERSIZE option sets default value of PAPERSIZE goption
53	57	0000000000000040	GDPIEOUTLINE	Driver draws pie slice outlines (empty pies)
54	60	0000000000000008	GDNOROTATE	Force the graphics sublib not to rotate the graph

例 7: レポートに対するユーザーライブラリおよびカタログの指定

要素:　PROC QDEVICE ステートメントオプション
- CATALOG=
- DEVLOC=

PRINTER ステートメント

詳細

この例では、PROC QDEVICE ステートメントの DEVLOC=オプションと CATALOG=オプションを使用して、ユーザー指定のライブラリおよびカタログにおける GIF プリンター用の一般レポートを作成します。結果は SAS ログに書き込まれます。

プログラム

デバイスのライブラリおよびカタログの割り当て

```
libname devlib 'file-path/em';
proc qdevice report=general devloc=devlib cat=mydevices;
  device gif;
run;
```

ログ

例のコード 47.4 特定のデバイスライブラリおよびカタログに対する SAS ログレポート

```
144  libname devlib 'file-path/em';
NOTE: Libref DEVLIB was successfully assigned as follows:
      Engine:       V9
      Physical Name: file-path/em
145  proc qdevice report=general devloc=devlib cat=mydevices;
146     device gif;
147  run;

                Name: GIF
         Description: Graphics Interchange Format RGB Color/Alpha Blending
              Module: SASGDDMX
                Type: Shortcut Device
      Device Catalog: file-path
           Prototype: GIF
    Default Typeface: Cumberland AMT
      Typeface Alias: Courier
          Font Style: Regular
         Font Weight: Normal
         Font Height: 8 points
        Font Version: Version 1.03
      Maximum Colors: 16777216
       Visual Color: True Color
      Color Support: RGBA
        Destination: sasprt.gif
          I/O Type: DISK
        Data Format: GIF
             Height: 6.25 inches
              Width: 8.33 inches
            Ypixels: 600
            Xpixels: 800
         Rows(vpos): 50
      Columns(hpos): 114
        Left Margin: 0 inches
 Minimum Left Margin: 0 inches
       Right Margin: 0 inches
Minimum Right Margin: 0 inches
      Bottom Margin: 0 inches
Minimum Bottom Margin: 0 inches
         Top Margin: 0 inches
  Minimum Top Margin: 0 inches
     XxY Resolution: 96x96 pixels per inch
 Compression Enabled: Always
  Compression Method: LZW
     Font Embedding: Never
          Animation: Enabled
```

48

R プロシジャ

概要: R プロシジャ

R プロシジャの機能

R は、主に統計計算とデータ分析のために設計されたインタープリター型の高レベルプログラミング言語です。R プロシジャを使用すると、SAS コード内の R プログラミング言語からステートメントを実行できます。外部の R スクリプトから R ステートメントをサブミットしたり、R ステートメントを直接入力したりできます。

PROC R を使用すると、次のタスクを実行できます。

- SAS セッションで R コードを実行する
- R ステートメントでほとんどの SAS 関数を呼び出す
- R から SAS コードをサブミットする
- SAS データセットまたはビューとデータフレームの間でデータを移動する
- SAS マクロ変数と R 変数の間で値を転送する
- R ステートメントでほとんどの SAS 関数を呼び出す
- SAS Studio の結果に R (Base R または ggplot2)からのグラフィックを表示する

R コードをバッチスクリプトで実行することもできます。詳細については、"Introduction" (*Using the batch Plug-In for the SAS Viya Platform Command-Line Interface*)を参照してください。

注: すべての機能を使用するには、次の R パッケージをインストールする必要があります。

- R6
- arrow
- haven
- plotly
- svglite

概念: R プロシジャ

PROC R の仕組み

PROC R は、SAS Compute Server プロセスからサブプロセスを作成します。このサブプロセスは、ユーザーが選択した R 配置を使用します。PROC R では、R コードを SUBMIT ブロックの 1 つのプログラム全体としてサブミットできます。詳細については、SUBMIT ステートメント (1659 ページ)を参照してください。

SUBMIT および ENDSUBMIT ステートメント内で R コードのブロックをサブミットすると、次のビヘイビアーが発生します。

- ステートメントでエラーまたは例外が発生した場合、コードブロックの処理はエラーの時点で停止します。
- R の複合ステートメントは、for ループ、ループ、if/else ブロック、関数定義などの他のステートメントを含む式であり、中かっこ{}で囲まれます。R では構文的にインデントは必要ありませんが、複合ステートメント内の継続行は通常、読みやすくするために一貫してインデントされます。複合ステートメントは閉じかっこ}で終了します。これはブロックの終了を表します。複合ステートメントの終わりと次の R ステートメントの間に空白行は必要ありません。

R プロシジャを使用すると、複数のプロシジャ呼び出しで同じサブプロセスを使用できるため、後続の R プロシジャ呼び出しでオブジェクトまたはデータを使用できます。R 終了し、必要に応じて、R 開始時または終了時に、新しいサブプロセスを作成することを選択できます。詳細については、“例 2: R プロシジャの再起動と終了の影響を観察する” (1666 ページ)を参照してください。

R サブプロセス作成されると、R プロシジャの一部であるモジュールが自動的にインポートされます。このモジュールは SAS と呼ばれ、SAS と R インスタンス間の対話を可能にします。SAS モジュールは、SAS Compute Server と R インスタンス間の対話に使用できるコールバックメソッドを多数提供します。これらのコールバックメソッドにより、R と SAS の間で変数を共有し(symget または symput)、SAS データセットと R データフレームの間でデータを移動し(sd2df または df2sd)、SAS 関数を呼び出し(sasfnc)、R ステートメント内で SAS コードをサブミットします(submit)。これらのコールバックメソッドを使用すると、必要に応じて SAS と R プログラミングを組み合わせたワークフローを作成できます。詳細については、“PROC R コールバックメソッドの使用” (1660 ページ)を参照してください。

PROC R の構成

PROC R がシステム上の R と対話できるようにするには、いくつかの構成要件があります。ユーザーまたは SAS 管理者は、次の 2 つの環境変数を構成する必要があります。

- PROC_PYPATH は、R 実行可能ファイルを指しています。R 実行可能ファイルを含む R 配置は、通常、SAS Compute Server からアクセス可能なファイルシステムにマウントされます。R 実行可能ファイルは、R プロシジャによってトリガーされる R サブプロセスを作成します。
- PROC_M2PATH は、配置の一部である SAS.R ファイルの場所を指定します。この環境変数は、最初はこのファイルのデフォルトの場所を指しています。SAS.R ファイルには、R 配置によってインポートされる SAS モジュールが含まれています。このファイルは、計算サーバーセッションと対話するコールバックメソッドのインターフェイスを提供します。

詳細については、"Configure SAS to Run External Languages Using SAS Configurator for Open Source" (*SAS Viya Platform: Programming Run-Time Servers*)を参照してください。

SAS Viya 製品は、デフォルトではロックダウン状態で配置されます。その結果として、PROC R は無効になります。PROC R を有効にするには、LOCKDOWN ステートメントで R および SOCKET アクセスメソッドに ENABLE_AMS=オプションを使用します。詳細は下記を参照してください。

- "Security Features" (*SAS Viya Platform: Programming Run-Time Servers*)の LOCKDOWN システムオプションと LOCKDOWN ステートメント
- "LOCKDOWN" (*SAS プログラマガイド: エッセンシャル*)

SUBMIT および ENDSUBMIT ステートメントを使用して R コードをサブミットする

SUBMIT および ENDSUBMIT ステートメントを使用して R コードをサブミットすると、R ステートメントは、プロシジャ呼び出しの最後に到達したときにコードのブロックとしてサブミットされます。すべての R ステートメントが処理された後、出力が SAS ログに表示されます。SAS コールバックメソッドを含めると、それらの呼び出しからの出力が、R コードからの出力の前に SAS ログに表示されます。

関連項目

SAS Scripting Wrapper for Analytics Transfer (SWAT)

構文: R プロシジャ

PROC R< INFILE='*file-name*' | fileref> < RESTART> < TERMINATE> < TIMEOUT=*n*>;
<SUBMIT;>
R statements
<ENDSUBMIT;>
RUN;

ステートメント	タスク	例
“PROC R ステートメント”	SAS コード内で R ステートメントを実行するか、実行する R ステートメントを含むファイルを指定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6
“SUBMIT ステートメント”	R ステートメントのブロックの先頭を特定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6
“ENDSUBMIT ステートメント”	R ステートメントのブロックの末尾を特定します。	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6

PROC R ステートメント

SAS セッションで R ステートメントを実行します。

制限事項: SAS がロック状態のときは、R プロシジャは使用できません。サーバー管理者は、このプロシジャがロックダウン状態でも使用できるように、このプロシジャを再有効化できます。詳細については、“Enable or Disable Access Methods” (*SAS Viya Platform: Programming Run-Time Servers*)を参照してください。RLANG オプションを VIYA_LOCKDOWN_USER_METHODS リストに追加する必要があります。

構文

形式 1: SUBMIT ブロック

PROC R< INFILE='*file-name*' | fileref> < RESTART> < TERMINATE> < TIMEOUT=*n*>;

オプション引数

INFILE='*file-name*' | fileref
SAS セッション内で実行する R ステートメントが含まれているソースファイルまたはファイル参照名を特定します。

要件 ファイル名を指定する場合、その名前を単一引用符か二重引用符で囲みます。

INFILE=オプションを使用する場合は、R スクリプトへのパスを指定する必要があります。

例 open_data.r で R コードを使用する:

```
proc r infile='/user/myscripts/open_data.r';
```

R コードを含むファイルのファイル参照名を定義する。

```
filename script 'my_script.r';

proc r infile=script;
```

RESTART
現在実行中の R サブプロセスを終了し、新しいサブプロセスを作成します。 その後、R サブプロセスは、PROC R の現在および後続の呼び出しに使用されます。

R の新しいブロックの先頭で RESTART を指定します。

例

```
proc r restart;
submit;
<R statements>
endsubmit;
run;
```

TERMINATE
現在の R プロシジャ呼び出しが完了すると、R サブプロセスを停止します。次回の R プロシジャ呼び出しは、R サブプロセスの新しいインスタンスを開始します。

例

```
proc r terminate;
submit;
<R statements>
endsubmit;
run;
```

TIMEOUT=*n*
R への接続試行時に PROC R が終了するまでの秒数を指定します。

デフォルト 60

例

```
proc r timeout=30 <additional options>;
submit;
<R statements>
endsubmit;
run;
```

SUBMIT ステートメント

R コードのブロックの先頭を特定します。SUBMIT ステートメントと ENDSUBMIT ステートメントの間に、R ステートメントを入力します。

要件 各 SUBMIT ステートメントには、対応する ENDSUBMIT ステートメントが必要です。

構文

SUBMIT;

ENDSUBMIT ステートメント

このステートメントは単独で行に指定され、R ステートメントのブロックの終わりを識別します。

構文

ENDSUBMIT;

使用法: R プロシジャ

SUBMIT ブロック内での R コードのサブミット

PROC R 起動内の、SUBMIT ステートメントと ENDSUBMIT ステートメントの間で R ステートメントをサブミットできます。次のコードは、1 つの R print ステートメントを実行します。

```
proc r;
submit;
print("Hello from R")
endsubmit;
run;
```

アウトプット 48.1　出力

```
[1] "Hello from R"
NOTE: PROCEDURE R used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

次のコードは、文字列をマクロ変数に割り当てます。この例では、マクロ変数 Mymacrovar が submit コールバックメソッドを使用して定義されます。マクロ変数 Myvar は、symput コールバックメソッドを使用して定義されています。両方のマクロ変数は、PROC R の呼び出しの外でアクセスできます。

```
proc r restart;
submit;
submit("%let mymacrovar=Hi there;")
symput('myvar', "This variable has global scope")
txt <- symget("mymacrovar")
print(txt)
endsubmit;
run;

%put &=mymacrovar;
%put &=myvar;
```

アウトプット 48.2　PROC R のマクロ変数を使用した出力

```
82   submit("%let mymacrovar=Hi there;")
83   symput('myvar', "This variable has global scope")
84   txt <- symget("mymacrovar")
85   print(txt)
86   endsubmit;
87   run;
88   %let mymacrovar=Hi there;
[1] "Hi there"
NOTE: PROCEDURE R used (Total process time):
      real time           1.01 seconds
      cpu time            0.18 seconds

89
90   %put &=mymacrovar;
MYMACROVAR=Hi there
91   %put &=myvar;
MYVAR=This variable has global scope
```

PROC R コールバックメソッドの使用

SAS と対話するコールバックメソッド

PROC R は、SAS セッションと R サブプロセス間の通信を可能にする SAS というモジュールを提供します。次のコールバックメソッドを使用して、R コードブロック

内で SAS コードをサブミットしたり、SAS 変数を操作したりできます。R は大文字と小文字を区別する言語であるため、このリストに示されているように大文字を使用してコールバックメソッドを指定します。

rplot(*r-plot-object*, <*filename = "plot-name.extension"*>)
結果ウィンドウのグラフを印刷します。

例 rplot(p)

```
rplot(quote(plot(mtcars$mpg, mtcars$wt,
          main = "MPG vs Weight",
          xlab = "Weight (1000 lbs)",
          ylab = "Miles per Gallon",
          col = "blue",
          pch = 19,
          cex = 1.5)),
    filename = "cars_scatter.png")
```

renderImage(*full-path-to-image-file*)
画像ファイルを結果ウィンドウにレンダリングします。有効なファイルの種類は、'png'、'jpg'、'jpeg'、'svg'、'gif'、'bmp'、'tiff'です。

例
```
full_path <- paste0(sas$workpath, "myplot.png")
renderImage(full_path)
```

sasfnc("*function-name*" <, *argument-1* <, *argument-2...*>>)
SAS 関数を呼び出します(使用している%sysfunc 部分を削除します)。関数に引数が必要な場合は、関数名の後にカンマ区切りのリストで指定します。

例 WORK ライブラリ参照名の存在を確認します
```
sasfnc("exist", "work", "LIBREF")
```

SAS の TODAY 関数を呼び出します。
```
sasfnc("today")
```

注: PROC R は現在、PROC FCMP で作成された関数の呼び出しをサポートしていません。

show(*r-object*, <*title="title"*>, <*count=number*>)
結果ウィンドウに R オブジェクトを表示します。

例
```
show(df, title = "Cars", count = 10)
```

submit("*sas-code*")
R セッション内で SAS コードをサブミットします。

例
```
submit("%let year=2025;")
```

symget("*sas-macro-variable*")
SAS マクロ変数の値を取得します。

例 SYSUSERID マクロ変数の値を取得します。
```
symget("SYSUSERID")
```

symput("*sas-macro-variable*", "*R-variable*")
SAS マクロ変数の値を設定します。

例 year の値を 2025 に設定します。

```
symput("year", "2025")
```

R プロシジャ内でデータを転送するコールバックメソッド

次のコールバックメソッドを使用すると、データフレームと SAS データセットの間でデータを転送できます。

sd2df("*sas_dataset*")
SAS データセットをローカル R データフレームにエクスポートします。

例 sashelp.cars を R データフレームにエクスポートする

```
df <- sd2df(table = "cars", libref = "sashelp")

df <- sd2df("sashelp.cars")
```

df2sd(*df*, "*sas_dataset*")
R データフレームを SAS データセットに転送します。

例 df データフレームを work.cars_r のデータセットに変換します (work はデフォルトのライブラリです)

```
df2sd(df, "cars_r")

df2sd(df, "cars_r", "work")
```

PROC R 関数のデバッグメッセージの表示

PROC R は、PROC R 関数の実行時に追加の診断メッセージを表示するために使用できるオプションのデバッグ機能を提供します。これは、SAS セッションで R コードを実行するときの内部処理の確認と問題のトラブルシューティングを支援することを目的としています。

デバッグメッセージを有効します

PROC R 関数のデバッグメッセージを有効にするには、実行前または実行中にデバッグフラグを設定します。デバッグモードを有効にすると、PROC R 関数はロジックを進める際に追加のメッセージを表示します。

デバッグ出力を有効にするには、次の 2 つの方法がサポートされています。

- PROC R セッション内で R オプションを設定すること

- SAS Compute サービス構成で環境変数を定義する

注: デバッグモードを有効にすると、すべてのデバッグメッセージが標準 PROC R ログ出力に書き込まれます。デバッグメッセージは、ファイルに書き込まれず、SAS Compute サーバーポッド内に保存されません。

R オプションの使用

サブミットされた R コード内で R.DEBUG オプションを TRUE に設定することで、現在の PROC R セッションのデバッグメッセージを有効にできます。

例のコード 48.1 例

```
%let SYM=1;
proc r; submit;
   options(R.DEBUG = TRUE)
   sym <- symget("SYM")
   endsubmit;
run;
```

環境変数の使用

SAS Compute サービスの環境変数を定義して、デバッグメッセージを有効にすることもできます。このアプローチでは、コンテナーレベルでのデバッグ出力が可能です。この構成は、Kubernetes 環境で SAS Compute ジョブ PodTemplate を変更することで適用されます。

```
template.spec.containers.env
- name: RLANG_DEBUG
  value: "TRUE"
```

トラブルシューティング: 環境からすべての 変数の削除

PROC R で R コードを実行する場合、R セッションには、PROC R が SAS Compute セッションとの通信に使用するいくつかの内部変数が含まれます。これらの変数を削除すると、計算セッションが応答しなくなる可能性があります。

次の R コードコードを使用すると、環境からすべての 変数を削除できます。

```
rm(list=ls())
```

PROC R では、このコマンドによって次の変数が削除されます。

- ctor
- obj
- PortVal
- sas

これらの変数が削除されると、計算セッションがハングする可能性があります。次の解決策のいずれかで、この問題を回避できます。

- PROC R 内では、rm(list = ls())(または同等の「すべてを削除する」パターン) を使用しないでください。
- ctor、obj、portVal、または sas を削除しません。
- PROC R 内部変数を削除リストから除外します。次の例を参照してください。

例のコード 48.2　例: PROC R 内部変数を除くすべてのオブジェクトを削除する

```
proc r;
submit;
  x  <- 10
  y  <- 11
  ch <- "abc"

  # Remove all objects except PROC R internal variables
  rm(list = setdiff(ls(), c("ctor", "obj", "portVal", "sas")))
endsubmit;
run;
```

例: R プロシジャ

例 1: 外部 R スクリプトから入力を指定する

要素:　FILENAME ステートメント
PROC R ステートメント、INFILE=オプション

詳細

この例では、/usr/scripts ディレクトリにある my_script.r という R スクリプトを実行する 2 つの方法を示します。最初のプログラムは、PROC R ステートメントの INFILE=オプションでファイル名と場所を直接指定します。2 番目のプログラムは、my_script.r に対するファイル参照名を定義し、INFILE=オプションでそのファイル参照名を使用します。

my_script.r のコンテンツは次のとおりです。

```
print("Hello World")
y <- "Hi!"
my_function <- function() {
  print("Inside the my_script.R script")
```

```
}
```

プログラム: INFILE=オプションを使用して R スクリプトの場所を指定する

PROC R ステートメントを実行し、R スクリプトの場所と名前を指定します。ファイルを SAS Studio 内の場所にアップロードします。 INFILE=オプションを使用して、R スクリプトの名前と場所を指定します。INFILE=オプションを呼び出すと、SAS は my_script.r で R ステートメントを実行します。スクリプトを実行して作成された変数と関数は、PROC R で SUBMIT および ENDSUBMIT ブロック内で使用できます。

```
proc r infile='tmp/my_script.r';
submit;
my_function()
print(paste("y =", y))
endsubmit;
run;
```

プログラム: R スクリプトをファイル参照名として指定する

```
filename script 'Users/<myuser>/My Folder/my_script.r';

proc r infile=script;
submit;
my_function()
print(paste("y =", y))
endsubmit;
run;
```

プログラムの説明

R スクリプトとその場所を指定するファイル参照名を定義します。 R スクリプトを SAS Studio にアップロードします。FILENAME ステートメントを使用して、R スクリプト用の script という名前のファイル参照名を作成します。

```
filename script 'Users/<myuser>/My Folder/my_script.r';
```

PROC R ステートメントを実行し、R スクリプトを識別するファイル参照名を指定します。 PROC R ステートメントで INFILE=オプションを指定すると、スクリプトが呼び出されます。スクリプトを実行して作成された変数と関数は、PROC R で SUBMIT および ENDSUBMIT ブロック内で使用できます。

```
proc r infile=script;
submit;
my_function()
print(paste("y =", y))
endsubmit;
```

```
run;
```

例のコード 48.1　外部の R スクリプトを呼び出した後の SAS ログ

```
80   filename script 'Users/myuser/myfolder/my_script.R';
81   proc r infile=script;
82   submit
NOTE: Resuming R state from previous PROC R invocation.
82 !      ;
83   my_function()
84   print(paste("y =", y))
85   endsubmit;
86   run;
[1] "Hello World"
[1] "Inside the my_script.R script"
[1] "y = Hi!"
NOTE: PROCEDURE R used (Total process time):
      real time          0.04 seconds
      cpu time           0.04 seconds
```

例 2: R プロシジャの再起動と終了の影響を観察する

要素:　PROC R ステートメント、RESTART オプション
PROC R ステートメント、TERMINATE オプション

詳細

この例は、複数のプロシジャ呼び出しにわたって R プロシジャのステータスを維持する効果を示しています。RESTART または TERMINATE オプションを使用すると、R プロシジャのステータスがリセットされることがわかります。

プログラム

```
proc r;
submit;
y <- "Hi!"
my_function <- function() {
  print("Inside the function")
}
endsubmit;
run;
```

```
proc r;
submit;
my_function()
print(paste("y =", y))
endsubmit;
run;

proc r restart;
submit;
x <- "New variable x is created"
print(paste("x =", x))
#References to function, variable from previous PROC R call
my_function()
print(paste("y =", y))
endsubmit;
run;

proc r terminate;
submit;
#Reference to variable defined in previous PROC R call
print(paste("x =", x))
x <- "Updating the value for x"
print(paste("x =", x))
# Redefining a function called my_function
my_function <- function() {
  print("Inside the proc step")
}
endsubmit;
run;

proc r;
submit;
print(paste("x =", x))
my_function()
endsubmit;
run;
```

プログラムの説明

PROC R を呼び出し、変数 y と my_function という関数を定義します。 文字列値を変数 y に割り当てます。別の文字列を出力する my_function という関数を定義します。

```
proc r;
submit;
y <- "Hi!"
my_function <- function() {
  print("Inside the function")
}
endsubmit;
run;
```

後続の R プロシジャ呼び出しで、以前に定義された関数と変数を参照します。 関数 my_function を呼び出し、変数 y の値を出力します。R プロシジャのステータスは後続のプロシジャ呼び出し間で維持されるため、これらの項目の定義は維持されます。

```
proc r;
submit;
my_function()
print(paste("y =", y))
endsubmit;
run;
```

RESTART オプションを使用して、R プロシジャのステータスをリセットします。 R プロシジャのステータスは、このプロシジャ呼び出しの開始時にリセットされます。x という新しい変数を定義します。以前に定義された変数または関数は使用できなくなります。これらを参照すると、SAS ログにエラーが記録され、プロシジャ呼び出しが終了します。

```
proc r restart;
submit;
x <- "New variable x is created"
print(paste("x =", x))
#References to function, variable from previous PROC R call
my_function()
print(paste("y =", y))
endsubmit;
run;
```

TERMINATE オプションを使用して、PROC R ステータスの維持を停止します。 TERMINATE オプションを使用すると、R プロシジャのステータスはプロシジャ呼び出しの終了時にリセットされます。変数 x は、PROC R への前回の呼び出しからまだ定義されています。 my_function 関数に新しい定義を指定してください。この関数の以前の定義は、PROC R 前回の呼び出しで RESTART オプションを指定したときに失われています。

```
proc r terminate;
submit;
#Reference to variable defined in previous PROC R call
print(paste("x =", x))
x <- "Updating the value for x"
print(paste("x =", x))
# Redefining a function called my_function
my_function <- function() {
  print("Inside the proc step")
}
endsubmit;
run;
```

PROC R の前回の呼び出しの終了時に PROC R ステータスがリセットされたことを確認します。 PROC R の前回の呼び出しで TERMINATE オプションを指定したため、そのプロシジャ呼び出しが終了した後、変数または関数定義は維持されませんでした。x を参照すると、SAS ログにエラーが記録され、プロシジャ呼び出しが終了します。

```
proc r;
submit;
print(paste("x =", x))
my_function()
endsubmit;
run;
```

アウトプット 48.3　RESTART および TERMINATE オプションの効果を示す出力

```
80   proc r;
81   submit
NOTE: Resuming R state from previous PROC R invocation.
81 !       ;
82   y <- "Hi!"
83   my_function <- function() {
84     print("Inside the function")
85   }
86   endsubmit;
87   run;
NOTE: PROCEDURE R used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
89   proc r;
90   submit
NOTE: Resuming R state from previous PROC R invocation.
90 !       ;
91   my_function()
92   print(paste("y =", y))
93   endsubmit;
94   run;
[1] "Inside the function"
[1] "y = Hi!"
NOTE: PROCEDURE R used (Total process time):
      real time           0.04 seconds
      cpu time            0.00 seconds
```

```
98   x <- "New variable x is created"
99   print(paste("x =", x))
100  #References to function, variable from previous PROC R call
101  my_function()
102  print(paste("y =", y))
103  endsubmit;
104  run;
ERROR: Unhandled R exception.
[1] "x = New variable x is created"
ERROR: [R] Error: could not find function "my_function"
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE R used (Total process time):
      real time           0.89 seconds
      cpu time            0.18 seconds
```

```
106  proc r terminate;
107  submit
NOTE: Resuming R state from previous PROC R invocation.
107!       ;
108  #Reference to variable defined in previous PROC R call
109  print(paste("x =", x))
110  x <- "Updating the value for x"
111  print(paste("x =", x))
112  # Redefining a function called my_function
113  my_function <- function() {
114    print("Inside the proc step")
115  }
116  endsubmit;
117  run;
[1] "x = New variable x is created"
[1] "x = Updating the value for x"
NOTE: R terminated.
NOTE: PROCEDURE R used (Total process time):
      real time           0.04 seconds
      cpu time            0.02 seconds
```

```
119  proc r;
120  submit
NOTE: R initialized.
WARNING: ignoring environment value of R_HOME
120!       ;
121  print(paste("x =", x))
122  my_function()
123  endsubmit;
124  run;
ERROR: Unhandled R exception.
ERROR: [R] Error: object 'x' not found
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE R used (Total process time):
      real time           0.92 seconds
      cpu time            0.17 seconds
```

例 3: SAS コードをサブミットし、PROC R 内で SAS 関数を呼び出す

要素: submit コールバックメソッド
sasfnc コールバックメソッド

詳細

この例では、submit コールバックメソッドを使用して R コードのブロック内で SAS コードをサブミットする方法を示します。この例では、submit コールバックメソッドを使用して、Test というデータセットを作成します。Test データセットは、PROC R の呼び出しの外部で使用できます。この例では、sasfnc コールバックメソッドを

使用して SAS 関数 SHA256HEX を呼び出し、値'abc'に対して取得された SHA256 ダイジェストを表示する方法も示します。

プログラム

```
proc r;
submit;
var1 <- "'r'"
var2 <- 2

submit(
  paste(
    "data work.test; x=", var1, "; y=", var2, "; run;",
    sep = ""
  )
)

var3 <- sasfnc("sha256hex","abc")
print(paste("var3 = ", var3))

endsubmit;
run;


proc print data=test;
run;
```

プログラムの説明

PROC R への呼び出しを開始し、SUBMIT ステートメントを使用して R コードのブロックを開始します。 値を 2 つの R 変数 var1 と var2 に割り当てます。文字列値は、値'r'が DATA ステップに送信されるように、追加の引用符のセットを取ります。

```
proc r;
submit;
var1 <- "'r'"
var2 <- 2
```

submit コールバックメソッドを呼び出して、SAS データセット Work.Test を作成します。 R format 関数を使用して、値を X と Y に割り当てます。

```
submit(
  paste(
    "data work.test; x=", var1, "; y=", var2, "; run;",
    sep = ""
  )
)
```

SAS.sasfnc コールバックメソッドを使用して SAS 関数を呼び出します。 文字列"abc"に対して SHA256HEX 関数を呼び出し、戻り値を R 変数 var3 に割り当てます。R print 関数を呼び出して、var3 の値を示す文字列を表示します。

```
var3 <- sasfnc("sha256hex","abc")
print(paste("var3 = ", var3))
```

R コードのブロックを終了し、生成されたデータセット Work.Test を出力します。
ENDSUBMIT ステートメントを呼び出して、R コードブロックを終了します。PROC R の呼び出しに続いて、Work.Test データセットを参照する PROC PRINT 呼び出しを行います。データセットは PROC R 呼び出しの外部で使用できることに注意してください。

```
endsubmit;
run;

proc print data=test;
run;
```

アウトプット 48.4　PROC R でのサブミットおよび sasfnc 呼び出しからの出力

Obs	x	y
1	r	2

例のコード 48.2　PROC R でのサブミットおよび sasfnc 呼び出しのログ

```
NOTE: The data set WORK.TEST has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds

[1] "var3 =  BA7816BF8F01CFEA414140DE5DAE2223B00361A396177A9CB410FF61F20015AD"
NOTE: PROCEDURE R used (Total process time):
      real time          0.25 seconds
      cpu time           0.16 seconds

96
97   proc print data=test;
98   run;
NOTE: There were 1 observations read from the data set WORK.TEST.
NOTE: The PROCEDURE PRINT printed page 1.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.06 seconds
      cpu time           0.05 seconds
```

例 4: PROC R で SAS マクロ変数を作成する

要素:

SAS マクロ変数の定義
symget コールバックメソッド
pyplot コールバックメソッド

詳細

この例では、SAS マクロ変数を作成し、それに R 値を代入する方法を示します。マクロ変数にはグローバルスコープがあり、PROC R 呼び出しの外部で参照できます。

プログラム

```
%let myvar = Jane Doe;

proc r;
submit;
x <- symget('myvar')
print(paste('macro variable myvar = ', x))

r_var <- 'Inside r'
symput('macrovar', r_var)
endsubmit;
run;

%put &=macrovar;
%put &=myvar;
```

プログラムの説明

PROC R の呼び出しの外部で、マクロ変数 Myvar を割り当てます。

```
%let myvar = Jane Doe;
```

PROC R を呼び出し、Myvar の値を R 変数に割り当てます。 symget コールバックメソッドを使用して、SAS マクロ変数 Myvar の値を R 変数 x に割り当てます。R print 関数を使用して、x の値を表示します。

```
proc r;
submit;
x <- symget('myvar')
print(paste('macro variable myvar = ', x))
```

R 変数を定義し、文字列値を割り当てます。 R 変数 py_var を定義し、それに文字列値を割り当てます。次に、symput を呼び出して、r_var の値を Macrovar 変数の値に置き換えます。

```
r_var <- 'Inside r'
symput('macrovar', r_var)
endsubmit;
run;
```

PROC R 外部にあるグローバルマクロ変数 Macrovar を参照します。 symput で定義するマクロ変数は、グローバル変数です。したがって、PROC R への呼び出しの外部でこれらを参照できます。マクロ変数 Myvar も使用できます。

```
%put &=macrovar;
```

```
%put &=myvar;
```

例のコード 48.3　マクロ変数からの SAS ログの例

```
84   x <- symget('myvar')
85   print(paste('macro variable myvar = ', x))
86
87   r_var <- 'Inside r'
88   symput('macrovar', r_var)
89   endsubmit;
90   run;
[1] "macro variable myvar =  Jane Doe"
NOTE: PROCEDURE R used (Total process time):
      real time          0.15 seconds
      cpu time           0.05 seconds

91
92   %put &=macrovar;
MACROVAR=Inside r
93   %put &=myvar;
MYVAR=Jane Doe
```

例 5: R プロットのレンダリング

要素:

rplot コールバックメソッド
renderImage コールバックメソッド

詳細

次のサンプルプログラムは、rplot()を使用して Base R および ggplot2 で作成されたグラフを印刷する方法と、renderImage()を使用して結果ウィンドウに既存の画像をレンダリングする方法を示しています。

プログラム: Base R 付きの R プロットをレンダリングする

```
proc r;
submit;

x <- 1:100
y <- x^2 + rnorm(100, 0, 100)

rplot({

plot(x, y,
```

```
    main = "Base R Scatter Plot",
    xlab = "X Values",
    ylab = "Y Values",
    col = "blue",
    pch = 19)

  abline(lm(y ~ x), col = "red", lwd = 2)
})

endsubmit;
run;
```

プログラムの説明

PROC R を呼び出します。 SUBMIT ステートメントの識別は、R コードのブロックの先頭を特定します。

```
proc r;
submit;
```

プロット用のデータを生成します。 プロットの入力値を作成します。x シーケンスを定義し、単純な式とランダム変動から対応する y 値を計算します。

```
x <- 1:100
y <- x^2 + rnorm(100, 0, 100)
```

プロットを SAS 出力にレンダリングします。 プロットコマンドを rplot()で囲み、グラフィックがキャプチャされて SAS 結果出力に表示されるようにします。

```
rplot({
```

プロットのコンテンツと基本表示を指定します。 x ベクトルと y ベクトルから散布図を描画します。タイトル、軸ラベル、およびいくつかの表示オプション(ポイントの色とマーカーの種類)を設定します。

```
plot(x, y,
    main = "Base R Scatter Plot",
    xlab = "X Values",
    ylab = "Y Values",
    col = "blue",
    pch = 19)
```

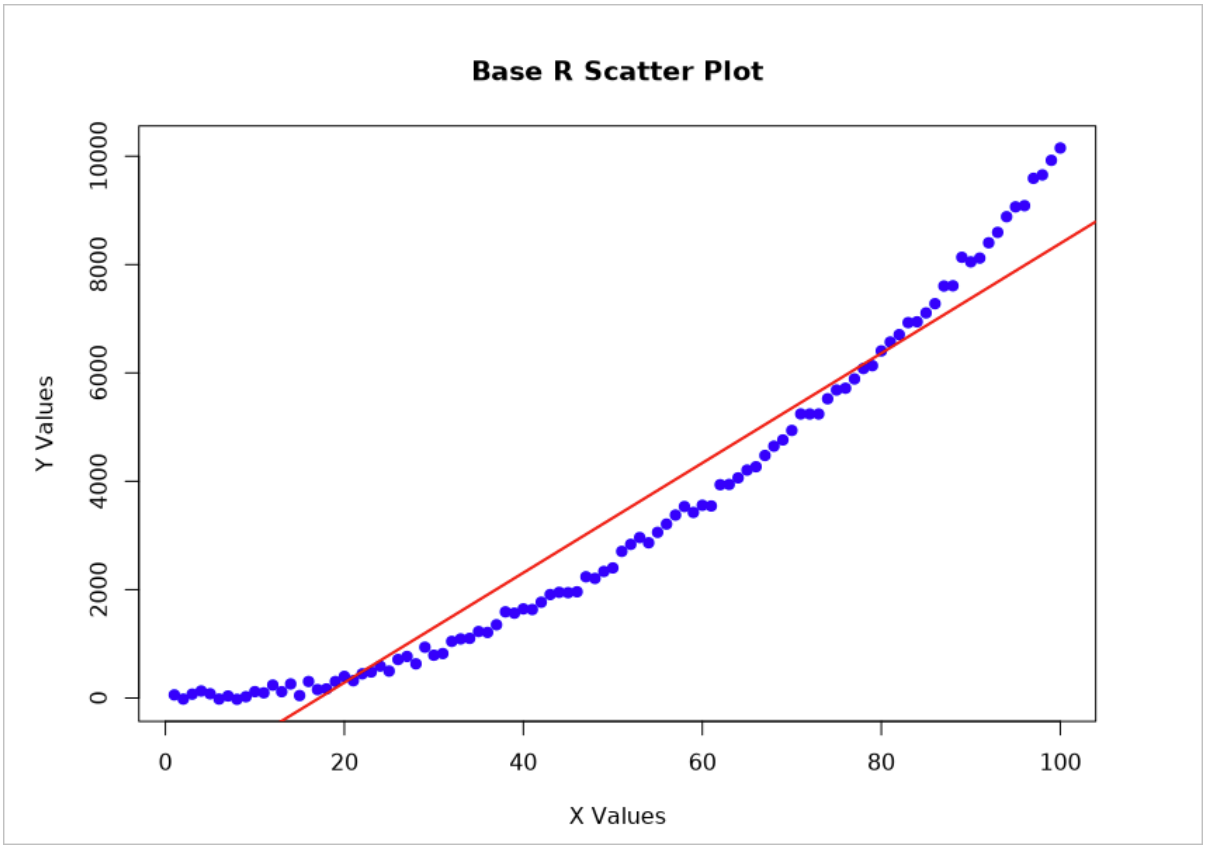
当てはめ線をプロットに追加します。 単純モデルを当てはめ、結果の線を既存のプロットに重ね合わせます。スタイルオプションを使用して、線を視覚的に区別します。

```
  abline(lm(y ~ x), col = "red", lwd = 2)
})
```

R ブロックを終了し、PROC R を終了します。 サブミットされた R コードを ENDSUBMIT で閉じます。RUN でプロシジャを実行して、ステップを完了し、出力を終了します。

```
endsubmit;
run;
```

アウトプット 48.5　Rplot コールバックメソッドからの出力



プログラム: ggplot2 付きの R プロットをレンダリングする

```
proc r;
submit;
library(ggplot2)

df <- data.frame(
  category = rep(c("A", "B", "C", "D"), each = 25),
  value = c(rnorm(25, 10, 2),
            rnorm(25, 15, 3),
            rnorm(25, 12, 2.5),
            rnorm(25, 18, 4))
)

p <- ggplot(df, aes(x = category, y = value, fill = category)) +
  geom_boxplot(alpha = 0.7) +
  theme_minimal() +
  labs(title = "Distribution by Category",
       x = "Category",
       y = "Value") +
  scale_fill_brewer(palette = "Set2")

rplot(p, filename = "boxplot_demo.png")

endsubmit;
```

```
run;
```

プログラムの説明

PROC R を呼び出し、必要な R パッケージをロードします。 PROC R を開始し、SUBMIT ブロックを開き、次のステートメントが R コードとして実行されるようにします。ggplot2 をロードして、gramter-of-graphics アプローチを使用してプロットを作成できるようにします。

```
proc r;
submit;
library(ggplot2)
```

プロット用のサンプルデータを生成します。 カテゴリ列と数値列を含むデータフレームを作成します。これらの値は rnorm()を使用して生成され、カテゴリごとに異なる分布の例が作成されます。

```
df <- data.frame(
  category = rep(c("A", "B", "C", "D"), each = 25),
  value = c(rnorm(25, 10, 2),
            rnorm(25, 15, 3),
            rnorm(25, 12, 2.5),
            rnorm(25, 18, 4))
)
```

ggplot オブジェクトを作成し、その外観を定義します。 カテゴリを X 軸に、値を Y 軸にマッピングし、塗りつぶしを使用してグループを色分けするプロットオブジェクトを作成します。箱ひげ図レイヤーを追加し、テーマ、ラベル、カラーパレットなどの一般的なプレゼンテーションオプションを適用します。

```
p <- ggplot(df, aes(x = category, y = value, fill = category)) +
  geom_boxplot(alpha = 0.7) +
  theme_minimal() +
  labs(title = "Distribution by Category",
       x = "Category",
       y = "Value") +
  scale_fill_brewer(palette = "Set2")
```

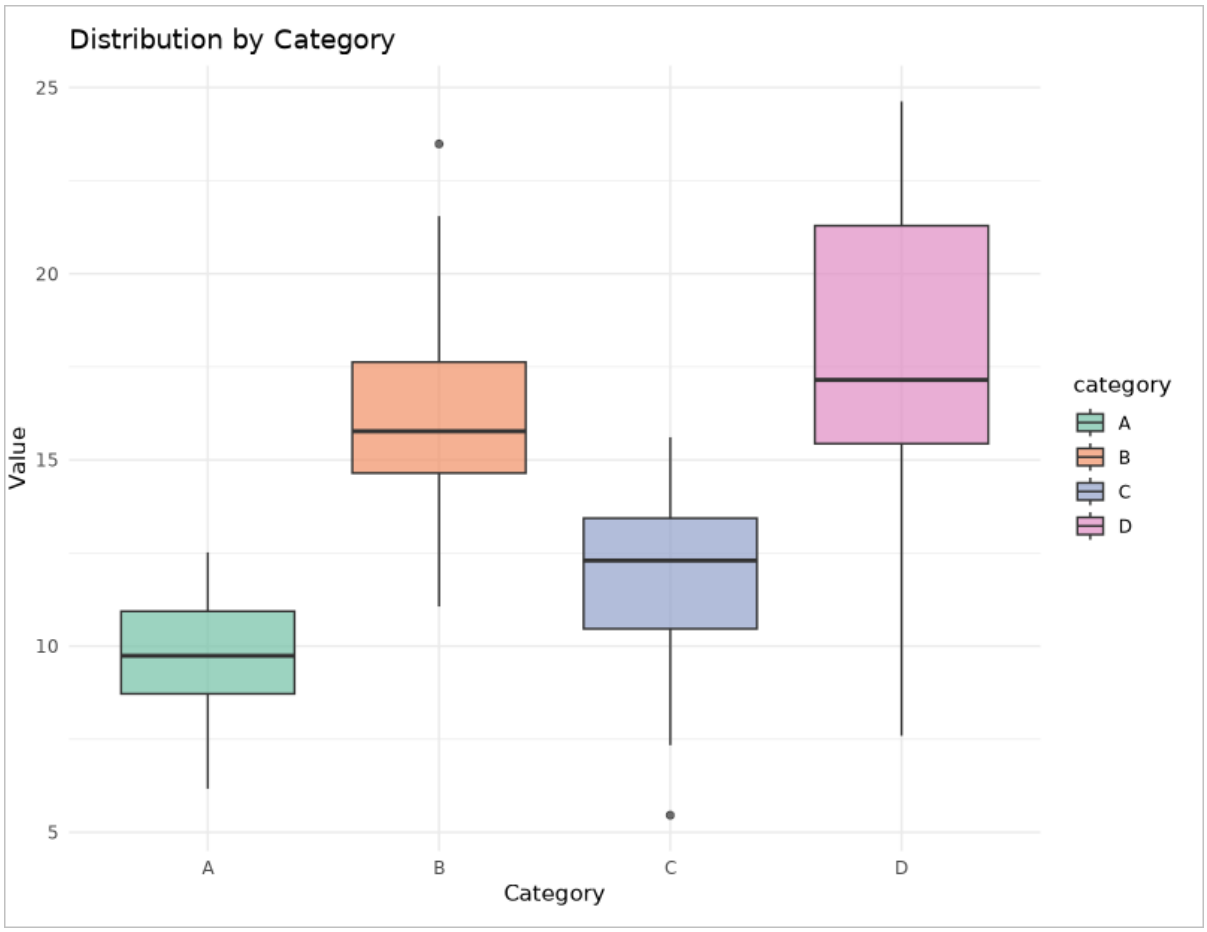
SAS 出力でプロットをレンダリングし、ファイルに書き込みます。 SAS 結果出力に表示されるように、ggplot オブジェクトをレンダリングします。filename=を指定して、レンダリングされたイメージを PNG ファイルに保存します。

```
rplot(p, filename = "boxplot_demo.png")
```

R ブロックを終了し、PROC R を終了します。 サブミットされた R コードを ENDSUBMIT で閉じます。RUN でプロシジャを実行して、ステップを完了し、出力を終了します。

```
endsubmit;
run;
```

アウトプット 48.6　Rplot コールバックメソッドからの出力



プログラム: 複数のプロットを順番にレンダリング

```
proc r;
submit;

library(ggplot2)

cars_df <- sd2df("sashelp.cars")

p1 <- ggplot(cars_df, aes(x = Weight, y = MPG_City)) +
  geom_point(aes(color = Type), alpha = 0.6, size = 3) +
  geom_smooth(method = "lm", se = TRUE, color = "red") +
  theme_minimal() +
  labs(title = "Fuel Efficiency vs Weight",
       x = "Weight (lbs)",
       y = "City MPG")

rplot(p1, filename = "scatter_mpg.png")

type_summary <- aggregate(MSRP ~ Type, data = cars_df, FUN = mean)

p2 <- ggplot(type_summary, aes(x = reorder(Type, MSRP), y = MSRP)) +
  geom_col(fill = "steelblue", alpha = 0.8) +
  coord_flip() +
  theme_minimal() +
```

```
  labs(title = "Average MSRP by Vehicle Type",
      x = "Vehicle Type",
      y = "Average MSRP ($)")

rplot(p2, filename = "bar_msrp.png")

p3 <- ggplot(cars_df, aes(x = Horsepower)) +
  geom_histogram(binwidth = 20, fill = "darkgreen",
                 alpha = 0.7, color = "black") +
  theme_minimal() +
  labs(title = "Distribution of Horsepower",
      x = "Horsepower",
      y = "Count")

rplot(p3, filename = "hist_hp.png")

endsubmit;
run;
```

プログラムの説明

PROC R を呼び出して、R サブミットブロックを開始します。 PROC R を開始し、SUBMIT ブロックを開き、次のステートメントが R コードとして実行されるようにします。

```
proc r;
submit;
```

必要な R パッケージをロードします。 ggplot2 をロードして、その関数とレイヤー構文を使用してプロットを作成できるようにします。

```
library(ggplot2)
```

SAS データセットを R データフレームに読み込みます。 SAS データ セット sashelp.cars を cars_df という名前のデータ フレームとして R にインポートします。これにより、3 つのプロットすべてで使用されるソースデータが提供されます。

```
cars_df <- sd2df("sashelp.cars")
```

プロット 1(傾向線付き散布図)を指定します。 都市の燃費と自動車の重みを比較する散布図を、自動車の種類ごとに点が色分けされています。一致する行を追加して全体のリレーションシップを要約し、ラベルと単純なテーマを適用します。

```
p1 <- ggplot(cars_df, aes(x = Weight, y = MPG_City)) +
  geom_point(aes(color = Type), alpha = 0.6, size = 3) +
  geom_smooth(method = "lm", se = TRUE, color = "red") +
  theme_minimal() +
  labs(title = "Fuel Efficiency vs Weight",
      x = "Weight (lbs)",
      y = "City MPG")
```

SAS 出力にプロット 1 をレンダリングします。 SAS 結果出力に表示されるように、ggplot オブジェクトをレンダリングします。filename=を指定して、画像を PNG ファイルに保存します。

```
rplot(p1, filename = "scatter_mpg.png")
```

プロット 2 の要約データを作成します(種類別平均 MSRP)。 自動車の種類ごとの平均 MSRP を計算するグループ化された要約を計算します。

```
type_summary <- aggregate(MSRP ~ Type, data = cars_df, FUN = mean)
```

プロット 2(棒グラフ)を指定します。 自動車の種類ごとの平均 MSRP を値別に並べた棒グラフを作成します。座標を反転すると、カテゴリラベルが読みやすく、基本ラベルとスタイル指定がしやすくなります。

```
p2 <- ggplot(type_summary, aes(x = reorder(Type, MSRP), y = MSRP)) +
  geom_col(fill = "steelblue", alpha = 0.8) +
  coord_flip() +
  theme_minimal() +
  labs(title = "Average MSRP by Vehicle Type",
       x = "Vehicle Type",
       y = "Average MSRP ($)")
```

SAS 出力にプロット 2 をレンダリングします。 2 番目の ggplot オブジェクトを SAS 結果出力にレンダリングします。指定したファイル名を使用して、棒グラフを PNG ファイルに保存します。

```
rplot(p2, filename = "bar_msrp.png")
```

プロット 3(ヒストグラム)を指定します。 データの馬力値の分布を表示するヒストグラムを作成します。ビンの幅を制御し、ラベルとテーマを追加します。

```
p3 <- ggplot(cars_df, aes(x = Horsepower)) +
  geom_histogram(binwidth = 20, fill = "darkgreen",
                 alpha = 0.7, color = "black") +
  theme_minimal() +
  labs(title = "Distribution of Horsepower",
       x = "Horsepower",
       y = "Count")
```

SAS 出力にプロット 3 をレンダリングします。 ヒストグラムを SAS 結果出力にレンダリングします。filename=を指定して、レンダリングされたイメージを PNG ファイルに保存します。

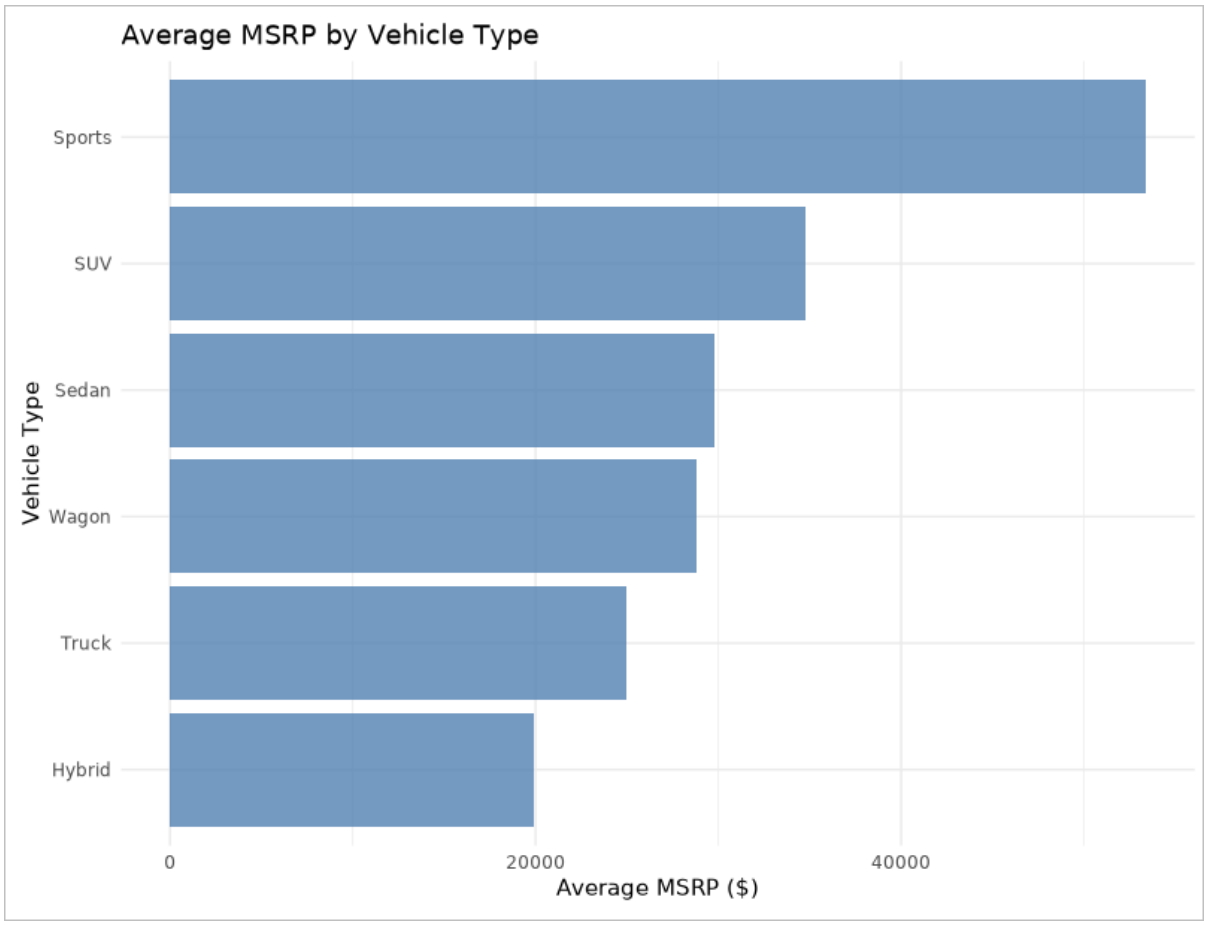
```
rplot(p3, filename = "hist_hp.png")
```

R ブロックを終了し、PROC R を終了します。 サブミットされた R コードを ENDSUBMIT で閉じます。RUN でプロシジャを実行して、処理を完了し、出力を終了します。

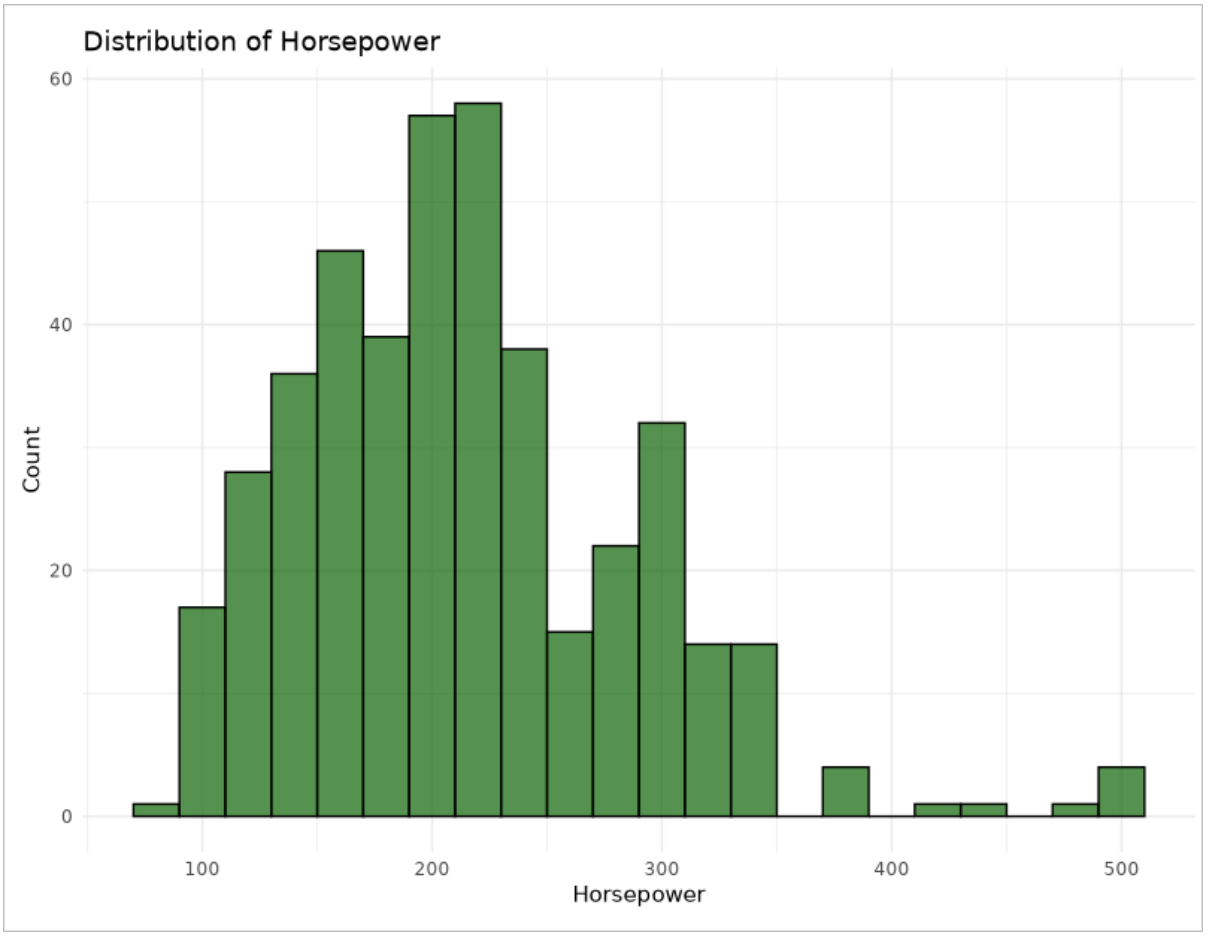
```
endsubmit;
run;
```

アウトプット 48.7　Rplot コールバックメソッドからの出力散布図

アウトプット 48.8　Rplot コールバックメソッドからの出力



アウトプット 48.9 Rplot コールバックメソッドからのヒストグラムの出力



プログラム: 既存の画像の表示

```
proc r;
submit;

library(ggplot2)

data <- data.frame(
  x = seq(0, 10, 0.1),
  y = sin(seq(0, 10, 0.1))
)

p <- ggplot(data, aes(x = x, y = y)) +
  geom_line(color = "purple", size = 1.5) +
  geom_point(color = "orange", size = 2) +
  theme_classic() +
  labs(title = "Sine Wave Visualization",
       x = "X", y = "sin(X)")

plot_path <- file.path(sas$workpath, "sine_wave.png")
ggsave(plot_path, p, width = 8, height = 5, dpi = 150)

renderImage(plot_path)

endsubmit;
```

```
run;
```

プログラムの説明

PROC R を呼び出して、R サブミットブロックを開始します。 SAS 内で PROC R を開始し、SUBMIT ブロックを開き、次のステートメントが R コードとして実行されるようにしますは。

```
proc r;
submit;
```

プロットパッケージをロードします。 ggplot2 をロードして、その関数とレイヤー構文を使用してプロットを作成します。

```
library(ggplot2)
```

プロット用のデータを生成します。 x のシーケンスと、sin()で計算される対応する y 値を含む小さなデータフレームを作成します。

```
data <- data.frame(
  x = seq(0, 10, 0.1),
  y = sin(seq(0, 10, 0.1))
)
```

ggplot オブジェクトを作成し、その外観を設定します。 線レイヤーを使用して正弦関数を描画し、曲線に沿った点を強調表示するプロットオブジェクトを作成します。テーマとラベルを適用します。

```
p <- ggplot(data, aes(x = x, y = y)) +
  geom_line(color = "purple", size = 1.5) +
  geom_point(color = "orange", size = 2) +
  theme_classic() +
  labs(title = "Sine Wave Visualization",
       x = "X", y = "sin(X)")
```

プロットを画像ファイルに保存します(まだ表示せずに)。 SAS Work ディレクトリでファイルパスを作成し、それを画像の出力先として使用します。後でレンダリングできるように、ggsave()を使用して ggplot オブジェクトを PNG ファイルに保存します。

```
plot_path <- file.path(sas$workpath, "sine_wave.png")
ggsave(plot_path, p, width = 8, height = 5, dpi = 150)
```

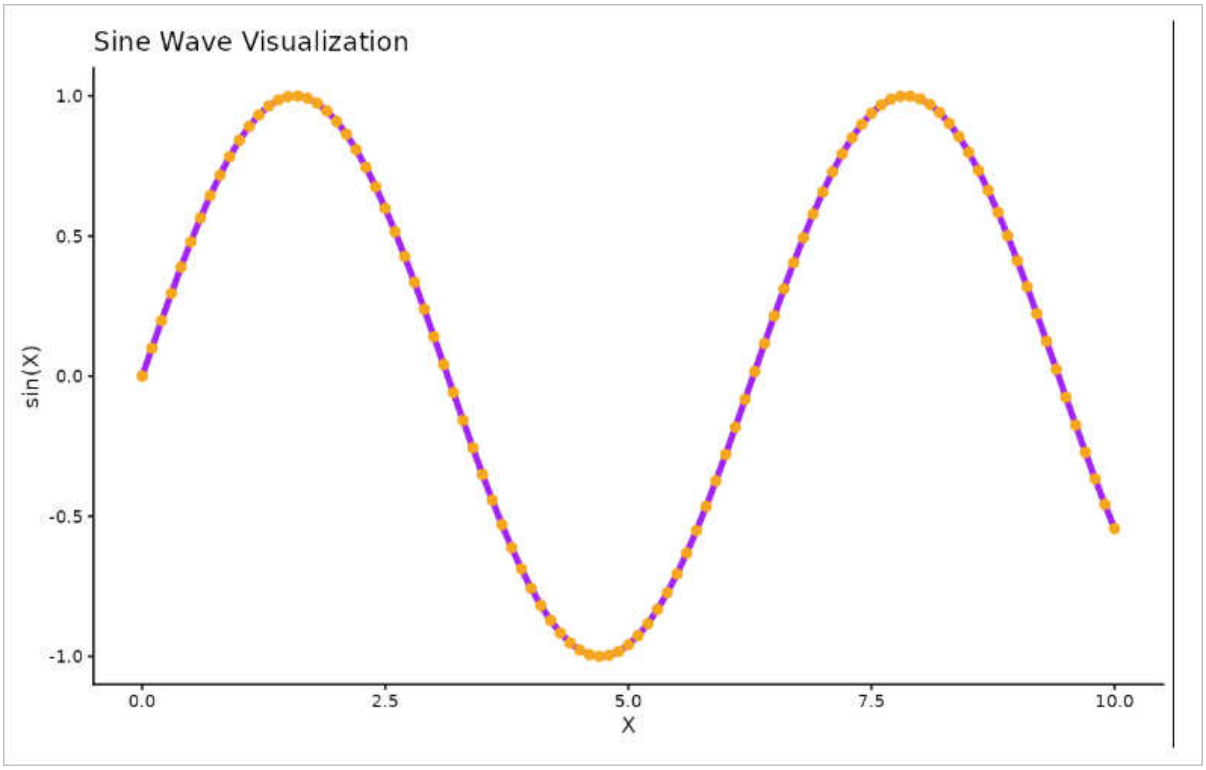
保存した画像を結果ウィンドウにレンダリングします。 plot_path で参照される画像ファイルが、結果ウィンドウに表示されるようにレンダリングします。

```
renderImage(plot_path)
```

R ブロックを終了し、PROC R を終了します。 サブミットされた R コードを ENDSUBMIT で閉じます。RUN でプロシジャを実行して、処理を完了し、出力を終了します。

```
endsubmit;
run;
```

アウトプット 48.10　RenderImage コールバックメソッドからの出力イメージ



例 6: データフレームの表示

要素:

show バックメソッド
sd2df コールバックメソッド
df2sd コールバックメソッド

詳細

次のプログラム例では、SAS データセットと R データフレーム間で転送し、SAS 結果ウィンドウに出力を表示する sd2df、df2sd、および関数を示します。

プログラム: SAS データセットを作成し、データフレームとして R にインポートする

```
data work.mydata;
  length a 8;
  length b_value 8;
  length var_32 $32;
```

```
 var_32 = "x";
 a = 1;
 b_value = 2;
 output;
run;

proc r;
submit;

df <- sd2df("mydata", "work");

show(df)

endsubmit;
run;
```

プログラムの説明

入力として使用する SAS データセットを作成します。 この DATA ステップは、3 つの変数(a、b_value、および文字 var_32)を使用して work.mydata を作成し、1 つのオブザベーションが WORK ライブラリに書き込まれます。

```
data work.mydata;
 length a 8;
 length b_value 8;
 length var_32 $32;
 var_32 = "x";
 a = 1;
 b_value = 2;
 output;
run;
```

PROC R を開始して、R サブミットブロックを開きます。 R プロシジャを開始し、SUBMIT ブロックを開始して、次のステートメントが SAS セッション内で R コードとして解釈および実行されるようにします。

```
proc r;
submit;
```

SAS データセットを R データフレームに読み込みます。 SAS データセット work.mydata を df という名前のデータフレームとして R にインポートします。

```
df <- sd2df("mydata", "work");
```

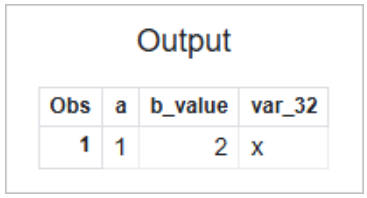
SAS 結果ウィンドウに R データフレームを表示します。 R データフレーム出力を SAS にレンダリングして、結果ウィンドウに表示できるようにします。

```
show(df)
```

R ブロックを終了し、PROC R を終了します。 R サブミットブロックを閉じ、PROC R を実行して処理を終了し、表示出力を生成します。

```
endsubmit;
run;
```

アウトプット 48.11　2 番目の Show コールバックメソッドからの出力データフレーム



Output

Obs	a	b_value	var_32
1	1	2	x

プログラム: SAS データセットを R にインポートし、選択した行と要約テーブルを表示する

```
proc r;
submit;

cars_df <- sd2df("sashelp.cars")

show(cars_df, title = "SASHELP.CARS (First 10 Rows)", count = 10)

summary_stats <- data.frame(
  Statistic = c("Mean MPG", "Max Horsepower", "Min Weight",
                "Avg MSRP", "Total Cars"),
  Value = c(
    mean(cars_df$MPG_City, na.rm = TRUE),
    max(cars_df$Horsepower, na.rm = TRUE),
    min(cars_df$Weight, na.rm = TRUE),
    mean(cars_df$MSRP, na.rm = TRUE),
    nrow(cars_df)
  )
)

show(summary_stats, title = "Cars Dataset Summary Statistics")

endsubmit;
run;
```

プログラムの説明

PROC R を呼び出して、R サブミットブロックを開始します。 SAS 内で PROC R を開始し、SUBMIT ブロックを開き、次のステートメントが R コードとして実行されるようにしますは。

```
proc r;
submit;
```

SAS データセットを R データフレームに読み込みます。 sashelp.cars データセットを cars_df という名前のデータフレームとして R にインポートします。

```
cars_df <- sd2df("sashelp.cars")
```

結果ウィンドウにデータフレームのサブセットを表示します。 show()を使用して、SAS 結果ウィンドウにデータフレームをレンダリングします。title=オプションおよび count=オプションは、ラベルを提供し、出力を最初の行に制限します。

```
show(cars_df, title = "SASHELP.CARS (First 10 Rows)", count = 10)
```

主要な統計量の要約テーブルを作成します。 インポートされたデータから基本メジャーを計算し、単純な 2 列のデータフレームに保存します。

```
summary_stats <- data.frame(
  Statistic = c("Mean MPG", "Max Horsepower", "Min Weight",
                "Avg MSRP", "Total Cars"),
  Value = c(
    mean(cars_df$MPG_City, na.rm = TRUE),
    max(cars_df$Horsepower, na.rm = TRUE),
    min(cars_df$Weight, na.rm = TRUE),
    mean(cars_df$MSRP, na.rm = TRUE),
    nrow(cars_df)
  )
)
```

結果ウィンドウに要約テーブルを表示します。 show()を再度使用して、結果ウィンドウに要約データフレームをレンダリングします。

```
show(summary_stats, title = "Cars Dataset Summary Statistics")
```

R ブロックを終了し、PROC R を終了します。 サブミットされた R コードを ENDSUBMIT で閉じます。RUN でプロシジャを実行して、処理を完了し、出力を終了します。

```
endsubmit;
run;
```

アウトプット 48.12　最初の Show コールバックメソッドからの出力データフレーム

SASHELP.CARS (First 10 Rows)

Obs	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase	Length
1	Acura	MDX	SUV	Asia	All	36945	33337	3.5	6	265	17	23	4451	106	189
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	23820	21761	2.0	4	200	24	31	2778	101	172
3	Acura	TSX 4dr	Sedan	Asia	Front	26990	24647	2.4	4	200	22	29	3230	105	183
4	Acura	TL 4dr	Sedan	Asia	Front	33195	30299	3.2	6	270	20	28	3575	108	186
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	43755	39014	3.5	6	225	18	24	3880	115	197
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	46100	41100	3.5	6	225	18	24	3893	115	197
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	89765	79978	3.2	6	290	17	24	3153	100	174
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	25940	23508	1.8	4	170	22	31	3252	104	179
9	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	35940	32506	1.8	4	170	23	30	3638	105	180
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	31840	28846	3.0	6	220	20	28	3462	104	179

アウトプット 48.13　2 番目の SecondShow コールバックメソッドからの出力データフレーム

Cars Dataset Summary Statistics

Obs	Statistic	Value
1	Mean MPG	20.06
2	Max Horsepower	500.00
3	Min Weight	1850.00
4	Avg MSRP	32774.86
5	Total Cars	428.00

プログラム: R データフレームを作成し、SAS データセットに転送する

```
proc r;
submit;

df <- data.frame(
  id      = 1:10,
  value   = c(1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10.1),
  charcol = c("a","b","c","a","b","c","a","b","c","a")
)

df2sd(df, "mydata","work")

endsubmit;
run;

proc print data = mydata;
run;
```

プログラムの説明

PROC R を開始して、R サブミットブロックを開きます。 R プロシジャを開始し、SUBMIT ブロックを開始して、次のステートメントが SAS 内で R コードとして実行されるようにします。

```
proc r;
submit;
```

メモリに R データフレームを作成します。 df という名前の R データフレームを、3 つの列で作成します: 整数 ID シーケンス(1:10)、数値列(value)、文字列(charcol)。

```
df <- data.frame(
  id      = 1:10,
  value   = c(1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10.1),
  charcol = c("a","b","c","a","b","c","a","b","c","a")
)
```

R データフレームを SAS データセットに転送します。 R データフレーム df を、work.mydata という名前の SAS データセットに変換し、後続の SAS ステップで使用できるようにします。

```
df2sd(df, "mydata","work")
```

R ブロックを終了し、PROC R を終了します。 サブミットされた R コードを閉じ、PROC R を実行して、SAS への転送を完了します。

```
endsubmit;
run;
```

新しく作成された SAS データセットを表示します。 R データフレームから生成された行と列の値を確認できるように、mydata のコンテンツを(デフォルトで WORK ライブラリから)出力します。

```
proc print data = mydata;
run;
```

アウトプット 48.14　PRINT プロシジャからのログ出力

```
NOTE: There were 10 observations read from the data set WORK.MYDATA.
NOTE: The PROCEDURE PRINT printed page 1.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.13 seconds
      cpu time            0.10 seconds
```

アウトプット 48.15　PRINT プロシジャからの出力 SAS データセット

Obs	id	value	charcol
1	1	1.1	a
2	2	2.2	b
3	3	3.3	c
4	4	4.4	a
5	5	5.5	b
6	6	6.6	c
7	7	7.7	a
8	8	8.8	b
9	9	9.9	c
10	10	10.1	a

49

RANK プロシジャ

概要: RANK プロシジャ

RANK プロシジャの機能

RANK プロシジャでは、SAS データセットの全オブザベーションにわたって 1 つ以上の数値変数のランクが計算され、そのランクが新しい SAS データセットに出力されます。PROC RANK 単独では、印刷出力は作成されません。

データのランク付け

次の出力は、単純な PROC RANK ステップによる 1 変数の値のランク付け結果を示しています。この例では、新しいランク付け変数が、4 日間の試合での 5 人のゴルファーの最終順位を示しています。最も打数の少ない選手が 1 位になります。次のステートメントでは、出力が生成されます。

```
proc rank data=golf out=rankings;
  var strokes;
  ranks Finish;
run;

proc print data=rankings;
run;
```

アウトプット 49.1　最小ランク値の最小変数値への割り当て

```
                 The SAS System                                1

         Obs     Player     Strokes     Finish

          1      Jack         279          2
          2      Jerry        283          3
          3      Mike         274          1
          4      Randy        296          4
          5      Tito         302          5
```

次の出力では、市議会議員候補者が地区別に選挙の得票数によってランク付けされています。候補者は在職期間によってもランク付けされています。

この例では、PROC RANK で次のタスクを実行できることが示されます。

- ランキングの順序を逆にして、最大値のランクを 1 に、その次に大きな値のランクを 2 に、というようにランク付けします。

- オブザベーションを、複数の変数値順に、別々にランク付けします。
- BY グループ内でオブザベーションをランク付けします。
- タイ値を処理します。

このレポートを作成するプログラムの説明については、“例 2: BY グループ内での値のランク付け” (1715 ページ)を参照してください。

アウトプット 49.2　各 BY グループ内での最小ランク値の最大変数値への割り当て

Results of City Council Election 1

District=1

Obs	Candidate	Vote	Years	Vote Rank	Years Rank
1	Cardella	1689	8	1	1
2	Latham	1005	2	3	2
3	Smith	1406	0	2	3
4	Walker	846	0	4	3

N = 4

District=2

Obs	Candidate	Vote	Years	Vote Rank	Years Rank
5	Hinkley	912	0	3	3
6	Kreitemeyer	1198	0	2	3
7	Lundell	2447	6	1	1
8	Thrash	912	2	3	2

N = 4

概念: RANK プロシジャ

コンピューターリソース

PROC RANK では、ランク付けするすべての変数について、すべてのオブザベーションの該当変数の値がメモリに格納されます。

統計アプリケーション

ランクは変数値の分布の調査に役立ちます。ランクを *n* または *n*+1 で除算して 0 から 1 までの範囲の値を得て、その値によって累積分布関数を推定します。これらの分数ランクに逆累積分布関数を適用すると、確率四分位スコアを得られます。これらのスコアを元の値と比較すると、分布に対する適合度を判断できます。たとえば、データのセットが正規分布している場合、正規スコアは元の値の線形関数になり、元の値に対するスコアのプロットは直線になります。

多くのノンパラメトリックな手法は、変数のランクの分析に基づいています。

- ランクに適用される 2 標本の *t* 検定は、その有意水準の *t* 近似を使用した Wilcoxon のランク和検定に相当します。ランクではなく正規スコアに *t* 検定を適用した場合、その検定は Van der Waerden 検定に相当します。*t* 検定を中央値スコア(GROUPS=2)に適用した場合、その検定は中央値検定に相当します。
- ランクに適用される一元配置分散分析は、Kruskal-Wallis の *k* 標本検定に相当します。また、多くの場合、パラメトリックなプロシジャで生成される F 検定をランクに適用した結果は、Kruskal-Wallis で使用されるX^2近似より優れています。この検定は、範囲を広げて他のランクスコアに使用できます(Quade 1966)。
- ブロック計画に対する Friedman の二元分析は、BY グループ内のランク付けを行い、次に、そのランクについて分散の主効果分析を実行することによって得られます(Conover 1998)。
- Iman と Conover (1979)の提唱した手法でランクの変換を行うと、回帰関係を調べられます。

タイ値の処理

PROC RANK で値をランク付けする際、BY グループ内で値の等しい分析変数が 2 つ以上ある場合は、データのタイ値が存在します。値の区別ができず、また、通常はランクの適切な基準となり得る明白な情報がこれ以上ないため、PROC RANK では、その値に異なるランクは割り当てられません。タイ値には、任意に異なるランクを割り当てられます。ただし、ランクを使用したノンパラメトリックな統計検定などの統計アプリケーションでは、通常はタイ値と同じランクを割り当てます。

これらの統計検定では通例、データ元は連続分布と見なされ、タイの確率は理論的にはゼロです。実際には、測定の不正確さや、デジタルコンピューター内の表現の正確性の限界などの理由により、しばしばタイ値が発生します。また、これらの統計検定では、通常、タイ値のグループに平均ランクを割り当てます。ランク合計が保持され、したがって、累積分布関数の推定値が歪められないため、平均ランクの割り当てをお勧めします。

統計量内外の適用のために、RANK プロシジャでは、タイ値の処理を制御する TIES= オプションが提供されます。このオプションのデフォルト値は、指定したランク付け方法またはスコアリング方法によって決定されるもので、PROC RANK ステートメントのオプションで指定できます。ランク付け方法とスコアリング方法について

は、TIES=LOW、TIES=HIGH または TIES=MEAN の場合、最初はタイ値が区別可能な場合と同様に処理されます。これらの方法はすべて、BY グループ内の分析変数値の並べ替えから始まり、次に、数列内の位置を示す序数が各非欠損値に割り当てられます。

続いて、非スコアリング方法の場合、PROC RANK では、TIES=LOW により最小値を選択するか、TIES=HIGH により最大値を選択するか、または TIES=MEAN によりタイ値グループの序数の平均値を計算することによって、タイ値が解決されます。この後、PROC RANK では、さらに 1 つ以上の変換(スケーリング、変換、切り捨てなど)を加えることにより、この値からランクを得られます。

スコアリング方法には、正規スコアリングと Savage スコアリングが含まれます。非スコアリング方法には、順序尺度ランク付け、デフォルト、および FRACTION、NPLUS1、GROUPS=、PERCENT オプションによって要求される方法が含まれます。スコアリング方法 NORMAL=および SAVAGE では、PROC RANK によって、タイ値がデータ内に存在しない場合と同じように、適切な式によって確率分位点スコアが得られます。次に、PROC RANK では、タイ値を解決するために、タイグループ内の全スコアの最小値が選択されるか、最大値が選択されるか、または平均が計算されます。

TIES=DENSE の場合、すべてのランク付けおよびスコアリング方法において、タイ値は区別できないものとして処理され、タイグループ内の各値に同じ序数が割り当てられます。他の TIES=解決方法と同様に、すべてのランク付けおよびスコアリング方法は、分析変数値の並べ替えから始まり、次に、序数が割り当てられます。ただし、タイ値のグループは単一値として処理されます。グループに割り当てる序数は、そのグループの前の値に序数が割り当てられている場合、その序数と+1 だけ異なります。グループの直後の値に序数が割り当てられている場合は、その序数と-1 だけ異なります。したがって、BY グループ内の最小序数は 1 で、最大序数は BY グループの一意の非欠損値数です。

序数を割り当てた後、PROC RANK は、スケーリングのために、非欠損値数のかわりに一意の非欠損値数を使用してランクとスコアを計算します。累積分布関数推定値がひずむ傾向があるため、一般に、密度の高いランク付けでは、ノンパラメトリックな統計検定での使用は受け入れられません。

PROC RANK が､分析変数の内部数値に関する計算の基礎を成すことに注意してください。プロシジャでは、分析前にこれらの値のフォーマットや丸めが行われることはありません。値の内部表現が異なる場合は、たとえわずかな違いであっても、PROC RANK でタイ値として処理されることはありません。これが各自のデータに関係する場合は、PROC RANK を呼び出す前に、適切な数量で分析変数を丸めます。ROUND 関数の詳細については、*“ROUND 関数” (SAS 関数と CALL ルーチン: リファレンス)*を参照してください。

PROC RANK の In-Database 処理

In-Database 処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。セキュリティの強化は、機密データを DBMS から抽出する必要がないため可能です。

より迅速な処理が可能になる理由は次のとおりです。

- データが、比較的低速なネットワーク接続を介して移送されるかわりに、高速の二次記憶装置を使用して、DBMS 上でローカル操作されます。
- DBMS では、より多くのリソースが用意されている可能性があります。
- DBMS には、実行するクエリを高度にパラレルで、スケーラブルな方法で最適化できる能力が備えられている可能性があります。

PROC RANK の In-Database 処理では、次のデータベース管理システムがサポートされています。

- Amazon Redshift
- DB2
- Google BigQuery

  注: PROC RANK の In-database 処理には、FLOAT64 型ではない VAR 変数が必要です。

- Greenplum
- Hadoop
- Impala
- Microsoft SQL Server
- MySQL (2021.1.5 現在)
- Netezza
- Oracle
- PostgreSQL
- SAP HANA
- Snowflake
- Spark
- Teradata
- Vertica
- Yellowbrick

表統計量の存在は、RANK プロシジャの In-Database 処理のパフォーマンスに影響を及ぼします。DBMS が表統計量を自動生成するように構成されていない場合は、受け入れ可能な In-Database パフォーマンスを達成するために表統計量の手動生成が必要になる可能性があります。

注: DB2 の場合、最小の入力テーブル以外についてはすべて 、表統計量の生成(自動か手動のどちか)を強くお勧めします。

RANK プロシジャの入力データセットが、サポート DBMS との SAS/ACCESS インターフェイスによる 通常の行の取得元であるデータベース内に存在するテーブルまたはビューの場合、PROC RANK では、DBMS 内の大部分またはすべての 作業を実行できます。そのような In-Database 処理が実行可能かどうかを決定する因子は、他にもいくつかあります。In-Database 処理は、次の環境では発生しません。

- 入力データセットで RENAME=データセットオプションを指定した場合。
- WHERE ステートメントが RANK プロシジャのコンテキストで表示されているか、入力データセットで WHERE=データセットオプションが指定されており、WHERE ステートメントまたはオプションに、DBMS に相当するものがない SAS 関数、または DBMS 内で SAS による使用のためのインストールが未実行の出力形式への参照が含まれる場合。
- BY ステートメントで指定した変数のいずれかに、関連する出力形式がある場合。PROC RANK では、フォーマットされた BY 変数の In-Database 処理はサポートされていません。
- FORMAT ステートメントがプロシジャコンテキスト内に出現し、BY ステートメントで指定した変数に適用された場合、In-Database 処理は実行できません。RANK では、フォーマットされた BY 変数の In-Database 処理はサポートされていません。DBMS では、FORMAT または ATTRIB ステートメントがプロシジャコンテキスト内に出現する場合のみ、出力形式を変数に関連付けられます。
- TIES=CONDENSE オプションでは、Oracle DBMS における RANK プロシジャの In-Database 処理がサポートされていません。このオプションを使用した場合、SQL 生成および In-Database 処理実行が行われません。

PROC RANK による DBMS 内のデータ処理が可能な場合、SQL クエリが生成されます。PROC RANK の In-Database 呼び出し時に生成される SQL クエリの構造は、次のようないくつかの因子に依存しています。

- ターゲット DBMS
- 使用するランク付け方法
- ランク付けする変数の数
- BY ステートメントと WHERE ステートメントの包含
- 使用する PROC RANK オプション(TIES=や DESCENDING など)

SQL クエリでは、必要な計算が表現され、DBMS にサブミットされます。このクエリの結果は、RANK プロシジャの出力先がテーブルの場合は DBMS 内に新しいテーブルとして残され、そうでなければ、SAS に返されます。MSGLEVEL オプションと SQLGENERATION オプションの設定によって、In-Database 処理が実行されたかを示すメッセージを SAS ログに出力するかどうかが決定されます。生成された SQL を検証するには、SQL_IP_TRACE オプションまたは SASTRACE=オプションを設定します。SQL_IP_TRACE によって、PROC RANK で生成された SQL が示されます。詳細については、*SAS/ACCESS for Relational Databases: リファレンス*の SASTRACE=オプション、または *SAS(R) Analytics Accelerator 1.3 for Teradata: Guide* の SQL_IP_TRACE オプションを参照してください。

SAS プロシジャの In-Database 実行に影響するシステムオプション、ライブラリオプション、データセットオプション、ステートメントオプションの設定の詳細については、*SAS/ACCESS for Relational Databases: リファレンス*の SQLGENERATION=LIBNAME オプションおよび SQLGENERATION=オプションを参照してください。

構文: RANK プロシジャ

ヒント:

ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントは、RANK プロシジャと併用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (23 ページ)を参照してください。

グローバルステートメントを使用することもできます。リストについては、“グローバルステートメント” (23 ページ)および“SAS グローバルステートメントのディクショナリ” (*SAS グローバルステートメント: リファレンス*)を参照してください。

発生する In-Database 処理に対し、データは、SAS In-Database 処理用として適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、“PROC RANK の In-Database 処理” (1695 ページ)を参照してください。

PROC RANK<*options*>;

ATTRIB *variable-list(s) attribute-list(s)*;

BY<DESCENDING>*variable-1*
<<DESCENDING> *variable-2* ...>
<NOTSORTED>;

FORMAT *variable-1*<...*variable-n*> <*format*> <DEFAULT=*default-format*>;

LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;

VAR *data-set-variables(s)*;

RANKS *new-variables(s)*;

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク	例
“PROC RANK ステートメント”	SAS データセットの 1 つ以上の数値変数のランク計算、および新しい SAS データセットへのランク出力	Ex. 1, Ex. 2, Ex. 3
“BY ステートメント”	BY グループごとの別々のランクセットの計算	Ex. 2, Ex. 3
“RANKS ステートメント”	ランク割り当て先の変数の識別	Ex. 1, Ex. 2
“VAR ステートメント”	ランク付けする変数の指定	Ex. 1, Ex. 2, Ex. 3

PROC RANK ステートメント

1 つ以上の数値変数のランクを計算します。

制限事項: 単一の PROC RANK ステップでは、ランク付けの方法を 1 つしか指定できません。

例:
“例 1: 複数の変数の値のランク付け” (1712 ページ)
“例 2: BY グループ内での値のランク付け” (1715 ページ)
“例 3: ランクに基づいたオブザベーションのグループ分け” (1717 ページ)

構文

PROC RANK <*options*>;

オプション引数の要約

Compute fractional ranks

NPLUS1
: 各ランクを分母 *n*+1 で除算して、分数ランクを計算します。

Create an output data set

OUT=*SAS-data-set*
: 出力データセットを指定します。

Preserve values

PRESERVERAWBYVALUES
: すべての BY 変数の生の値を保持します。

Reverse the order of the rankings

DESCENDING
: ランクを逆順にします。

Specify how to rank tied values

TIES=HIGH | LOW | MEAN | DENSE
: タイデータ値の正規スコアやランクの計算方法を指定します。

Specify the input data set

DATA=*SAS-data-set*
: 入力 SAS データセットを指定します。

Specify the ranking method

FRACTION
: 各ランクを、ランク付け変数が非欠損値のオブザベーション数で除算し、分数ランクを計算します。

GROUPS=*number-of-groups*
0 から *number-of-groups* - 1 までの範囲のグループ値を割り当てます。

NORMAL=BLOM | TUKEY | VW
ランクから正規スコアを計算します。

PERCENT
ランクにおける非欠損値を含むオブザベーションの割合を計算します。

SAVAGE
ランクから Savage(または指数)スコアを計算します。

オプション引数

DATA=*SAS-data-set*
入力 SAS データセットを指定します。

制限事項 別のユーザーが同時にデータセットを更新している場合、同時アクセスをサポートするエンジンでは PROC RANK を使用できません。

In-Database 処理を実行するためには、データセットの指定によって、サポートされている DBMS に存在するテーブルを参照する必要があります。

参照項目: “入力データセット” (24 ページ)

DESCENDING
ランクを逆順にします。 DESCENDING を指定すると、最大値にランク 1、その次に大きな値にランク 2、というようにランク付けされます。指定しなければ、昇順にランク付けされます。

参照項目: “例 1: 複数の変数の値のランク付け” (1712 ページ)

“例 2: BY グループ内での値のランク付け” (1715 ページ)

FRACTION
各ランクを、ランク付け変数が非欠損値のオブザベーション数で除算し、分数ランクを計算します。

別名 F

操作 TIES=HIGH が、FRACTION オプション指定時のデフォルトです。TIES=HIGH を指定した場合、分数ランクは右連続な累積経験分布関数の値と見なされます。

参照項目: NPLUS1 オプション

GROUPS=*number-of-groups*
0 から *number-of-groups* - 1 までの範囲のグループ値を割り当てます。 通常の指定は、パーセント点を示す GROUPS=100、十分位数を示す GROUPS=10、および四分位数を示す GROUPS=4 です。たとえば、GROUPS=4 により、元の値は 4 つのグループに分かれます。たとえば、GROUPS=4 を指定すると、元の値が 4 グループに分割され、最小値にはデフォルトで四分位数値 0、最大値には四分位数値 3 が割り当てられます。

グループ値の計算式は次の通りです。

$$\mathrm{FLOOR}(rank * k/(n+1))$$

FLOOR は FLOOR 関数、*rank* は値の順位、*k* は GROUPS=の値、*n* は TIES=LOW、TIES=MEAN および TIES=HIGH でランク付け変数が非欠損値のオブザベーション数です。TIES=DENSE の場合、*n* は一意の非欠損値を含むオブザベーション数です。

オブザベーション数がグループ数で均等に割り切れる場合、グループの境界にタイ値がなければ、各グループのオブザベーション数は同じになります。多くのタイ値を含む変数によってオブザベーションをグループ化すると、結果的にグループのバランスが悪くなる可能性があります。これは、PROC RANK では常に、同じ値のオブザベーションは同じグループに割り当てられるためです。

ヒント　グループ値を逆順に並べ替えるには、DESCENDING を使用します。

参照項目: “例 3: ランクに基づいたオブザベーションのグループ分け” (1717 ページ)

NORMAL=BLOM | TUKEY | VW
ランクから正規スコアを計算します。 結果変数の表示は正規分布します。*n* は、TIES=LOW、TIES=MEAN、および TIES=HIGH でランク付け変数が非欠損値のオブザベーション数です。TIES=DENSE の場合、*n* は一意の非欠損値を含むオブザベーション数です。式は次のとおりです。

BLOM
$y_i=\Phi^{-1}((r_i-3/8)/(n+1/4))$

TUKEY
$y_i=\Phi^{-1}((r_i-1/3)/(n+1/3))$

VW
$y_i=\Phi^{-1}((r_i)/(n+1))$

これらの式では、Φ^{-1} は正規累積の逆(PROBIT)関数、r_i は *i* 番目のオブザベーションのランク、*n* はランク付け変数の非欠損オブザベーション数です。

VW は van der Waerden を表します。 NORMAL=VW,を指定すると、スコアをノンパラメトリックな位置の検定に使用できます。3 つの正規スコアはすべて、正規分布の正確な期待順序統計の近似値です(*正規スコア*とも呼ばれます)。BLOM 版の表示は、他の方法よりわずかに適合度が高くなります(Blom 1958; Tukey 1962)。

制限事項　NORMAL=オプションを使用すると、In-Database 処理は行われません。

操作　TIES=オプションを指定すると、PROC RANK では、非タイ値に基づいてランクから正規スコアが計算され、TIES=の指定が結果のスコアに適用されます。

NPLUS1
各ランクを分母 *n*+1 で除算して、分数ランクを計算します。このとき、*n* は TIES=LOW、TIES=MEAN および TIES=HIGH でランク付け変数が非欠損値のオブザベーション数です。TIES=DENSE の場合、*n* は一意の非欠損値を含むオブザベーション数です。

別名 FN1

N1

操作 TIES=HIGH が、NPLUS1 オプション指定時のデフォルトです。

参照項目: FRACTION オプション

OUT=*SAS-data-set*
出力データセットを指定します。 *SAS-data-set* が存在しない場合は、PROC RANK で作成されます。OUT=を省略した場合、DATA*n* 命名規則を使用してデータセット名が指定されます。

操作 In-Database 処理が実行され、OUT=もサポートされた DBMS テーブルを参照している場合、IN=と OUT=の両方で同じライブラリを参照すると、すべての処理が DBMS で実行可能になり、結果的に出力テーブルが直接作成されます。この場合、SAS には結果が返されません。

PRESERVERAWBYVALUES
すべての BY 変数の生の値を保持します。 (その変数の出力データセットへのプロパゲート時)。PRESERVERAWBYVALUES オプションが指定されておらず、かつ 1 つの BY 変数が指定されている場合、各 BY グループの代表値が出力データセットに書き込まれます。複数の BY 変数を指定した場合は、各 BY グループの代表値セットが出力データセットに書き込まれます。

PERCENT
変数の非欠損値を持つオブザベーションの数で各ランクを除算し、結果に 100 を掛けてパーセンテージを取得します。 *n* は、TIES=LOW、TIES=MEAN、および TIES=HIGH のランキング変数の非欠損値を持つオブザベーションの数です。TIES=DENSE の場合、*n* は一意の非欠損値を含むオブザベーション数です。

別名 P

操作 TIES=HIGH が、PERCENT オプション指定時のデフォルトです。

ヒント 累積百分率を計算するには PERCENT を使用しますが、パーセント点を計算するには GROUPS=100 を使用します。

SAVAGE
次の式(Lehman 1998)によってランクから Savage(または指数)スコアを計算します。

$$y_i = \left[\sum_{j=n-r_i+1} \left(\frac{1}{j}\right) \right] - 1$$

操作 TIES=オプションを指定すると、PROC RANK では、非タイ値に基づいてランクから Savage スコアが計算され、TIES=の指定が結果のスコアに適用されます。

TIES=HIGH | LOW | MEAN | DENSE
タイデータ値の正規スコアやランクの計算方法を指定します。

HIGH
対応するランクのうちの最大値を割り当てます(NORMAL=指定時は正規スコアの最大値)。

LOW
対応するランクのうちの最小値を割り当てます(NORMAL=指定時は正規スコアの最小値)。

MEAN
対応するランクの平均値を割り当てます(NORMAL=指定時は正規スコアの平均値)。

DENSE
タイ値を単一の順序統計量として扱って、スコアとランクを計算します。デフォルトの方法では、ランクは、1 の数から始まり、ランク付けされている変数の一意な非欠損値数で終わる連続した整数です。タイ値には同じランクが割り当てられます。

注: CONDENSE は、DENSE の別名です。

デフォルト　MEAN (FRACTION オプションも PERCENT オプションも無効の場合)。

操作　NORMAL=オプションを指定すると、TIES=の指定が、正規スコアの計算に使用するランクではなく、正規スコアに適用されます。

参照項目:
“タイ値の処理” (1694 ページ)

“例 1: 複数の変数の値のランク付け” (1712 ページ)

“例 2: BY グループ内での値のランク付け” (1715 ページ)

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント　SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント　この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント　この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

BY ステートメント

BY グループごとに別々のランクセットを作成します。

操作:　BY ステートメントで NOTSORTED オプションを指定すると、In-Database 処理を実行できません。
FORMAT ステートメントなどを使用して、入力データセットの BY 変数のいずれかにフォーマットすると、In-Database 処理が実行されません。

参照項目:　“例 3: ランクに基づいたオブザベーションのグループ分け” (1717 ページ)

例:　“例 2: BY グループ内での値のランク付け” (1715 ページ)
“例 3: ランクに基づいたオブザベーションのグループ分け” (1717 ページ)

構文

BY <DESCENDING> *variable-1*
<<DESCENDING> *variable-2* ...>
<NOTSORTED>;

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定した場合、プロシジャはインデックスを 使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ別の BY グループとして扱います。

SAS/ACCESS エンジンを使用している場合は、BY ステートメントを指定すると、データが常に並べ替え後の順序で返されます。NOTSORTED オプションを指定すると、オプションは無視され、In-Database 処理が実行されます。

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。*“出力形式の取り消し” (SAS DATA ステップステートメント: リファレンス)*を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、*“PUT ステートメント: リスト” (SAS DATA ステップステートメント: リファレンス)*を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=’ ‘>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項 ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: *"文字定数と文字変数" (SAS プログラマガイド: エッセンシャル)*.

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
```

```
run;
```

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD

MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

RANKS ステートメント

ランク値に対する新しい変数を作成します。

デフォルト: RANKS ステートメントを省略した場合、出力データセットではランク値が元の変数値と置き換えられます。

要件 RANKS ステートメントを使用する場合は、VAR ステートメントも使用する必要があります。

例: “例 1: 複数の変数の値のランク付け” (1712 ページ)
“例 2: BY グループ内での値のランク付け” (1715 ページ)

構文

RANKS *new-variables(s)*;

必須引数

new-variable(s)
VAR ステートメントでリストされる変数のランクを含める新しい変数を 1 つ以上指定します。RANKS ステートメントで最初にリストされた変数には、VAR ステートメントで最初にリストされた変数のランクが含まれます。RANKS ステートメントで 2 番目にリストされた変数には、VAR ステートメントで 2 番目にリストされた変数のランクが含まれ、以下も同様に処理されます。

VAR ステートメント

入力変数を指定します。

別名: VARIABLES

デフォルト: VAR ステートメントを省略した場合、PROC RANK では、入力データセットの全数値変数のランクが計算されます。

例:
“例 1: 複数の変数の値のランク付け” (1712 ページ)
“例 2: BY グループ内での値のランク付け” (1715 ページ)
“例 3: ランクに基づいたオブザベーションのグループ分け” (1717 ページ)

構文

VAR *data-set-variables(s)*;

必須引数

data-set-variable(s)
ランク計算対象の変数を 1 つ以上指定します。

詳細

RANKS ステートメントでの VAR ステートメントの使用

RANKS ステートメントの使用時には、VAR ステートメントが必須です。これらのステートメントをあわせて使用すると、RANK ステートメントで名前を指定したランク付け変数が作成されます。この変数は、VAR ステートメントで指定した入力変数に対応しています。RANKS ステートメントを省略した場合、出力データセットではランク値が元の値と置き換えられます。

結果: RANK プロシジャ

欠損値

ランクまたはランクスコアが出力データセットで元の値と置き換えられる際、欠損値はランク付けされず、欠損したままです。

出力データセット

RANK プロシジャでは、ランクまたはランクスコアを含む SAS データセットが作成されますが、印刷出力は作成されません。出力データセットを印刷するには、PROC PRINT、PROC REPORT または別の SAS レポートツールを使用します。

出力データセットには、入力データセットの全変数に加えて、RANK ステートメントで指定した変数が含まれます。RANKS ステートメントを省略した場合、出力データセットではランク値が元の変数値と置き換えられます。

数値精度

In-Database 処理については、SQL で RANK プロシジャによって 表現される演算処理とその実行順序は、SAS 内で実行される処理と基本的に同じです。ただし、In-Database 処理の結果は、SAS での直接の結果と比較して、数値の差が小さくなる場合があります。

例: RANK プロシジャ

例 1: 複数の変数の値のランク付け

要素: PROC RANK ステートメントオプション

DESCENDING
TIES=
RANKS ステートメント
VAR ステートメント
PRINT プロシジャ

詳細

この例では、次のアクションを実行します。

- ランクを逆順にして、最大値にランク 1 を指定
- 可能な限り最高のランクをタイ値に割り当て
- ランク付け変数を作成し、元の変数とともに印刷

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data cake;
  input Name $ 1-10 Present 12-13 Taste 15-16;
  datalines;
Davis      77 84
Orlando    93 80
Ramey      68 72
Roe        68 75
Sanders    56 79
Simms      68 77
Strickland 82 79
;

proc rank data=cake out=order descending ties=low;

  var present taste;
  ranks PresentRank TasteRank;
run;

proc print data=order;
  title "Rankings of Participants' Scores";
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、SAS ジョブが開始する日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

CAKE データセットを作成します。 このデータセットには、ケーキ作りコンテストにおける、各参加者の姓、出来ばえのスコア、および味のスコアが含まれます。

```
data cake;
  input Name $ 1-10 Present 12-13 Taste 15-16;
  datalines;
Davis      77 84
Orlando    93 80
Ramey      68 72
Roe        68 75
Sanders    56 79
Simms      68 77
Strickland 82 79
;
```

降順で数値変数のランクを生成し、出力データセット ORDER を作成します。 DESCENDING で、ランクが逆順になり、最高スコアにランク 1 が指定されます。TIES=LOW で、可能な限り最高のランクがタイ値に指定されます。OUT=で、出力データセット Order が作成されます。

```
proc rank data=cake out=order descending ties=low;
```

ランクを含む新しい変数を 2 つ作成します。 VAR ステートメントで、ランク付けする変数が指定されます。RANKS ステートメントで、2 つの新しい変数 PresentRank と TasteRank が作成され、それぞれに変数 Present と Taste のランクが含められます。

```
  var present taste;
  ranks PresentRank TasteRank;
run;
```

データセットを出力します。 PROC PRINT で、ORDER データセットが印刷されます。TITLE ステートメントでタイトルを指定します。

```
proc print data=order;
  title "Rankings of Participants' Scores";
run;
```

出力: リスト

アウトプット 49.3　参加者のスコアのランク付け

```
                Rankings of Participants' Scores                    1

                                                 Present    Taste
      Obs    Name          Present    Taste       Rank       Rank

       1     Davis            77        84          3          1
       2     Orlando          93        80          1          2
       3     Ramey            68        72          4          7
       4     Roe              68        75          4          6
       5     Sanders          56        79          7          3
       6     Simms            68        77          4          5
       7     Strickland       82        79          2          3
```

例 2: BY グループ内での値のランク付け

要素:

PROC RANK ステートメントオプション
- DESCENDING
- TIES=

BY ステートメント

RANKS ステートメント

VAR ステートメント

PRINT プロシジャ

詳細

この例では、次のアクションを実行します。

- BY グループ内のオブザベーションを別々にランク付け
- ランクを逆順にして、最大値にランク 1 を指定
- 可能な限り最高のランクをタイ値に割り当て
- ランク付け変数を作成し、元の変数とともに印刷

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data elect;
  input Candidate $ 1-11 District 13 Vote 15-18 Years 20;
  datalines;
Cardella    1 1689 8
Latham      1 1005 2
Smith       1 1406 0
Walker      1  846 0
Hinkley     2  912 0
Kreitemeyer 2 1198 0
Lundell     2 2447 6
Thrash      2  912 2
;

proc rank data=elect out=results ties=low descending;

  by district;

  var vote years;
  ranks VoteRank YearsRank;
run;

proc print data=results n;
```

```
  by district;
  title 'Results of City Council Election';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、SAS ジョブが開始する日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

ELECT データセットを作成します。 このデータセットには、各候補の姓、選挙区番号、得票総数、市議会の経験年数が含まれます。

```
data elect;
  input Candidate $ 1-11 District 13 Vote 15-18 Years 20;
  datalines;
Cardella    1 1689 8
Latham      1 1005 2
Smith       1 1406 0
Walker      1  846 0
Hinkley     2  912 0
Kreitemeyer 2 1198 0
Lundell     2 2447 6
Thrash      2  912 2
;
```

降順で数値変数のランクを生成し、出力データセット RESULTS を作成します。 DESCENDING で、ランクが逆順になり、最高得票総数にランク 1 が指定されます。TIES=LOW で、可能な限り最高のランクがタイ値に指定されます。OUT=で、出力データセット Results が作成されます。

```
proc rank data=elect out=results ties=low descending;
```

BY グループごとに別々のランクセットを作成します。 BY ステートメントは、ランク付けを District の値別に分割します。

```
  by district;
```

ランクを含む新しい変数を 2 つ作成します。 VAR ステートメントで、ランク付けする変数が指定されます。RANKS ステートメントで、新しい変数 VoteRank と YearsRank が作成され、それぞれに変数 Vote と Years のランクが含められます。

```
  var vote years;
  ranks VoteRank YearsRank;
run;
```

データセットを出力します。 PROC PRINT で、Results データセットが印刷されます。N オプションでは、各 BY グループのオブザベーション数が印刷されます。TITLE ステートメントでタイトルを指定します。

```
proc print data=results n;
  by district;
  title 'Results of City Council Election';
run;
```

出力: リスト

次の出力では、第 2 区で Hinkley と Thrash が 912 票でタイになっています。TIES=LOW なので、双方ともランクが 3 になります。

アウトプット 49.4　市議選の結果

```
                      Results of City Council Election                      1

---------------------------------- District=1 ----------------------------------

                                                 Vote    Years
               Obs    Candidate    Vote    Years  Rank     Rank

                1     Cardella     1689      8      1        1
                2     Latham       1005      2      3        2
                3     Smith        1406      0      2        3
                4     Walker        846      0      4        3

                                N = 4


---------------------------------- District=2 ----------------------------------

                                                  Vote    Years
              Obs    Candidate      Vote    Years  Rank     Rank

               5     Hinkley         912      0      3        3
               6     Kreitemeyer    1198      0      2        3
               7     Lundell        2447      6      1        1
               8     Thrash          912      2      3        2

                                N = 4
```

例 3: ランクに基づいたオブザベーションのグループ分け

要素:

PROC RANK ステートメントオプション
　GROUPS=
BY ステートメント
VAR ステートメント
PRINT プロシジャ
SORT プロシジャ

詳細

この例では、次のアクションを実行します。

- 2 つの入力変数の値に基づいてオブザベーションをグループ分け
- BY グループ内のオブザベーションを別々にグループ分け
- 元の変数値をグループ値に置換

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data swim;
  input Name $ 1-7 Gender $ 9 Back 11-14 Free 16-19;
  datalines;
Andrea  F 28.6 30.3
Carole  F 32.9 24.0
Clayton M 27.0 21.9
Curtis  M 29.0 22.6
Doug    M 27.3 22.4
Ellen   F 27.8 27.0
Jan     F 31.3 31.2
Jimmy   M 26.3 22.5
Karin   F 34.6 26.2
Mick    M 29.0 25.4
Richard M 29.7 30.2
Sam     M 27.2 24.1
Susan   F 35.1 36.1
;

proc sort data=swim out=pairs;
  by gender;
run;

proc rank data=pairs out=rankpair groups=3;

  by gender;

  var back free;
run;

proc print data=rankpair n;
  by gender;
  title 'Pairings of Swimmers for Backstroke and Freestyle';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションで、SAS ジョブの開始日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Swim データセットを作成します。 このデータセットには、水泳選手の名前、および背泳と自由型のタイム(秒)が含まれます。この例では、男子クラス内と女子クラス内で両泳法のタイムに基づいて水泳選手のペアを作り、全員が各泳法でタイムの似た相手とペアになれるようにします。

```
data swim;
  input Name $ 1-7 Gender $ 9 Back 11-14 Free 16-19;
  datalines;
Andrea  F 28.6 30.3
Carole  F 32.9 24.0
Clayton M 27.0 21.9
Curtis  M 29.0 22.6
Doug    M 27.3 22.4
Ellen   F 27.8 27.0
Jan     F 31.3 31.2
Jimmy   M 26.3 22.5
Karin   F 34.6 26.2
Mick    M 29.0 25.4
Richard M 29.7 30.2
Sam     M 27.2 24.1
Susan   F 35.1 36.1
;
```

Swim データセットを並べ替えて、出力データセット Pairs を作成します。 PROC SORT で、データセットが Gender 順に並べ替えられます。グループごとに別々のランクセットを取得するには、このアクションが必要です。OUT=で、出力データセット Pairs が作成されます。

```
proc sort data=swim out=pairs;
  by gender;
run;
```

3 つにグループ分けされたランクを生成し、出力データセットを作成します。 GROUPS=3 で、使用可能なグループ値(0、1、2)のいずれかが各泳法の各水泳選手に割り当てられます。OUT=で、出力データセット Rankpair が作成されます。

```
proc rank data=pairs out=rankpair groups=3;
```

BY グループごとに別々のランクセットを作成します。 BY ステートメントは、ランク付けを Gender 別に分割します。

```
  by gender;
```

変数の元の値をランク値に置き換えます。 VAR ステートメントで、Back と Free がランク付け対象の変数であることが指定されます。PROC RANK で RANKS ステートメントを指定しなかった場合、出力データセットでは元の変数値がグループ値に置き換えられます。

```
  var back free;
run;
```

データセットを出力します。 PROC PRINT で、Rankpair データセットが印刷されます。N オプションでは、各 BY グループのオブザベーション数が印刷されます。TITLE ステートメントでタイトルを指定します。

```
proc print data=rankpair n;
  by gender;
  title 'Pairings of Swimmers for Backstroke and Freestyle';
run;
```

出力: リスト

次の出力では、グループ値によって、各泳法でタイムの似た水泳選手にペアを組ませます。たとえば、Andrea と Ellen が背泳で一緒に組むのは、2 人のタイムが女子クラスで最速なためです。男子水泳選手のグループのバランスが取れていないのは、男子水泳選手が 7 人いるためです。泳法ごとに、水泳選手が 3 人いるグループが 1 つできます。

アウトプット 49.5　背泳と自由型の水泳選手のペアリング

```
              Pairings of Swimmers for Backstroke and Freestyle              1

---------------------------------- Gender=F ----------------------------------

                        Obs    Name      Back    Free

                          1    Andrea      0       1
                          2    Carole      1       0
                          3    Ellen       0       1
                          4    Jan         1       2
                          5    Karin       2       0
                          6    Susan       2       2

                               N = 6


---------------------------------- Gender=M ----------------------------------

                        Obs    Name       Back    Free

                          7    Clayton      0       0
                          8    Curtis       2       1
                          9    Doug         1       0
                         10    Jimmy        0       1
                         11    Mick         2       2
                         12    Richard      2       2
                         13    Sam          1       1

                               N = 7
```

参考文献

Blom, G. 1958. *Statistical Estimates and Transformed Beta Variables*. New York, New York: John Wiley & Sons, Inc.

Conover, W.J. 1998. *Practical Nonparametric Statistics, Third Edition*. New York, New York: John Wiley & Sons, Inc.

Conover, W.J. and R.L.Iman. 1976. “On Some Alternative Procedures Using Ranks for the Analysis of Experimental Designs.” *Communications in Statistics* A5 (14): 1348-1368.

Conover, W.J. and R.L.Iman. 1981. “Rank Transformations as a Bridge between Parametric and Nonparametric Statistics.” *The American Statistician* 35: 124-129.

Iman, R.L. and W.J.Conover. 1979. “The Use of the Rank Transform in Regression.” *Technometrics* 21: 499-509.

Lehman, E.L. 1998. *Nonparametrics: Statistical Methods Based on Ranks*. Upper Saddle River, New Jersey: Prentice Hall.

Quade, D. 1966. “On Analysis of Variance for the K-Sample Problem.” *Annals of Mathematical Statistics* 37: 1747-1758.

Tukey, John W. 1962. “The Future of Data Analysis.” *Annals of Mathematical Statistics* 33: 22.

50

REGISTRY プロシジャ

概要: REGISTRY プロシジャ

REGISTRY プロシジャの機能

REGISTRY プロシジャでは、SAS レジストリが管理されます。レジストリには 2 つのパーツがあります。1 つのパーツは Sashelp ライブラリに保存され、もう一つのパーツは Sasuser ライブラリに保存されます。

REGISTRY プロシジャでは、次を実行できます。

- レジストリファイルをインポートして Sashelp レジストリおよび Sasuser レジストリを作成します。
- レジストリの全部または一部を別のファイルにエクスポートします。
- SAS ログにレジストリの内容をリストします。
- レジストリとファイルの内容を比較します。
- レジストリファイルをアンインストールします。
- キーまたは値が上書きまたはアンインストールされた場合に、詳細ステータス情報を配信します。
- Sasuser レジストリのエントリをクリアします。
- レジストリが存在するかを検証します。
- 診断情報をリストします。

詳細については、SAS レジストリを参照してください。

構文: REGISTRY プロシジャ

PROC REGISTRY <*options*>;

ステートメント	タスク	例
“PROC REGISTRY ステートメント”	レジストリファイルの管理	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6

PROC REGISTRY ステートメント

SAS レジストリを管理します。

例:

“例 1: SAS レジストリへのファイルのインポート” (1733 ページ)
“例 3: レジストリと外部ファイルの比較” (1736 ページ)
“例 2: レジストリファイルのリスト表示とエクスポート” (1734 ページ)
“例 4: レジストリファイルの比較” (1737 ページ)
“例 5: STARTAT=オプションを使用したキーシーケンス全体の指定” (1739 ページ)
“例 6: フォントリストの表示” (1739 ページ)

構文

PROC REGISTRY <*options*>;

オプション引数の要約

CLEARSASUSER
: ユーザーが追加したキーを Sasuser レジストリから消去します。

COMPAREREG1='*libname.registry-name-1*'
: 2 つのレジストリファイルを比較します。

COMPAREREG2='*libname.registry-name-2*'
: 2 つのレジストリファイルを比較します。

COMPARETO=*file-specification*
: レジストリとファイルの内容を比較します。

DEBUGOFF
: レジストリのデバッグを無効化します。

DEBUGON
: レジストリのデバッグを有効化します。

EXPORT=*file-specification*
: レジストリの内容を指定ファイルに書き込みます。

FOLLOWLINKS
: LIST コマンドの処理時に検出されたリンクを含めます。

FULLSTATUS
: IMPORT=および UNINSTALL=オプションの結果について SAS ログで追加情報を提供します。

IMPORT=*file-specification*
: 指定ファイルをレジストリにインポートします。

KEYSONLY
: LIST、LISTUSER、LISTHELP、LISTREG の各オプションの出力をキーのみ表示するように制限します。

LEVELS=*n*
: LIST、LISTUSER、LISTHELP、LISTREG の各オプションの表示レベル数を制限します。

LIST
: SAS ログにレジストリの内容を書き込みます。特定のキーをリストするには、このオプションを STARTAT=オプションと一緒に使用します。

LISTHELP
: レジストリの Sashelp 部分の内容を SAS ログに書き込みます。

LISTREG='*libname.registry-name*'
: レジストリの内容をログに送信します。

LISTUSER
: レジストリの Sasuser 部分の内容を SAS ログに書き込みます。

STARTAT='*key-name*'
: 指定キーでレジストリの内容のエクスポートや書き込みや比較を開始します。

UNINSTALL=*file-specification*
: 指定ファイル内のすべてのキーと値を指定レジストリから削除します。

UPCASE
: すべての内向きのキー名に大文字を使用します。

UPCASEALL
: ファイルのインポート時に、すべてのキー、名前および項目値に大文字を使用します。

USESASHELP
: SAS レジストリの Sashelp 部分に対する指定操作を実行します。

オプション引数

CLEARSASUSER
: ユーザーが追加したキーを SAS レジストリの Sasuser 部分から消去します。

COMPAREREG1='*libname.registry-name-1*'
: 比較対象の 2 つのレジストリのうち 1 つを指定します。結果は SAS ログに表示されます。

libname
: レジストリファイルが存在するライブラリの名前です。

registry-name-1
: 最初のレジストリの名前です。

要件 COMPAREREG1 は COMPAREREG2 と一緒に使用する必要があります。

操作 単一キーとそのサブキーすべてを 指定するには、STARTAT=オプションを指定します。

例 “例 4: レジストリファイルの比較” (1737 ページ)

COMPAREREG2='*libname.registry-name-2*'
: 比較対象の 2 つのレジストリの 2 番目を指定します。結果は SAS ログに表示されます。

libname
: レジストリファイルが存在するライブラリの名前です。

registry-name-2
: 2 番目のレジストリの名前です。

要件 COMPAREREG2 は COMPAREREG1 と一緒に使用する必要があります。

例 “例 4: レジストリファイルの比較” (1737 ページ)

COMPARETO=*file-specification*
: レジストリ情報を含むファイルとレジストリの内容を比較します。キーと値がファイルにあってレジストリにはない場合、その情報が返されます。次の項目が差異としてレポートされます。

- 外部ファイルで定義されているが、レジストリでは定義されていないキー
- 外部ファイルにはあるがレジストリにはない指定キーの値名
- 似ている名前のキーの似ている名前の値の内容の差異

COMPARETO=では、レジストリにあるがファイルにないキーと値は差異としてレポートされません。レジストリはさまざまなファイルの断片から成ることが多いためです。

File-specification は次の値のうちのいずれかになります。

'***external-file***'
レジストリ情報を含む外部ファイルのパスと名前です。

fileref
外部ファイルに割り当てられたファイル参照名です。

要件 FILENAME ステートメント、FILENAME 関数、**エクスプローラー**ウィンドウ、適切な動作環境のコマンドのいずれかで、ファイル参照名と外部ファイルを事前に関連付けておく必要があります。

操作 デフォルトでは、PROC REGISTRY で *file-specification* とレジストリの Sasuser 部分が比較されます。*file-specification* とレジストリの Sashelp 部分を比較するには、オプション USESASHELP を指定します。

参照項目: 使用法: REGISTRY プロシジャ (1730 ページ)(レジストリ情報を含むファイルの構築方法の詳細)

例 “例 3: レジストリと外部ファイルの比較” (1736 ページ)

DEBUGON
詳細なログエントリを提供して、レジストリのデバッグを有効化します。

DEBUGOFF
レジストリのデバッグを無効化します。

EXPORT=*file-specification*
レジストリの内容を指定ファイルに書き込みます。 このとき、*file-specification* は次の値のいずれかになります。

'***external-file***'
レジストリ情報を含む外部ファイルの名前です。

fileref
外部ファイルに割り当てられたファイル参照名です。

要件 FILENAME ステートメント、FILENAME 関数、**エクスプローラー**ウィンドウ、適切な動作環境のコマンドのいずれかで、ファイル参照名と外部ファイルを事前に関連付けておく必要があります。

file-specification がすでに存在する場合は、PROC REGISTRY で上書きされます。その他の場合は、PROC REGISTRY でファイルが作成されます。

操作 デフォルトでは、EXPORT=でレジストリの Sasuser 部分が指定ファイルに書き込まれます。レジストリの Sashelp 部分を書き込むには、USESASHELP オプションを指定します。USESASHELP を使用するには、Sashelp ライブラリへの書き込み権限が必要です。

単一キーとそのサブキーすべてをエクスポートするには、STARTAT=オプションを指定します。

例 “例 2: レジストリファイルのリスト表示とエクスポート” (1734 ページ)

FOLLOWLINKS
LIST オプションの処理時に検出されたリンクを含めます。

通常、LIST オプションでは、リンク項目の値が表示されます。FOLLOWLINKS オプションを使用した場合、リンクはキーとして扱われ、リンクに含まれる項目が表示されます。

FULLSTATUS
IMPORT=および UNINSTALL オプションの実行結果として、追加または削除されたキー、サブキーおよび値をリストします。

IMPORT=*file-specification*
SAS レジストリにインポートするファイルを指定します。PROC REGISTRY では、既存のレジストリは上書きされません。そのかわりに、既存のレジストリを指定ファイルの内容で更新します。

注: ファイル拡張子.sasxreg は必要ありません。

File-specification は次の値のうちのいずれかになります。

'***external-file***'
レジストリ情報を含む外部ファイルのパスと名前です。

fileref
外部ファイルに割り当てられたファイル参照名です。

要件 FILENAME ステートメント、FILENAME 関数、**エクスプローラー**ウィンドウ、適切な動作環境のコマンドのいずれかで、ファイル参照名と外部ファイルを事前に関連付けておく必要があります。

操作 デフォルトでは、IMPORT=で SAS レジストリの Sasuser 部分にファイルがインポートされます。レジストリの Sashelp 部分にファイルをインポートするには、USESASHELP オプションを指定します。USESASHELP を使用するには、Sashelp への書き込み権限が必要です。

ファイルのインポート時に SAS ログで追加情報を取得するには、FULLSTATUS を使用します。

参照項目: 使用法: REGISTRY プロシジャ (1730 ページ)(レジストリ情報を含むファイルの構築方法の詳細)

例 “例 1: SAS レジストリへのファイルのインポート” (1733 ページ)

KEYSONLY
LIST、LISTUSER、LISTHELP、LISTREG の各オプションの出力をキーのみ表示するように制限します。

LEVELS=*n*
LIST、LISTUSER、LISTHELP、LISTREG の各オプションの表示レベル数を制限します。

要件 LEVEL ≥ 1。LEVELS=0 は、LEVELS を指定しない場合と同様の動作になります。

LIST
SAS ログに SAS レジストリ全体の内容を書き込みます。

操作 単一キーとそのサブキーすべてを 書き込むには、STARTAT=オプションを使用します。

LISTHELP
レジストリの Sashelp 部分の内容を SAS ログに書き込みます。

操作 単一キーとそのサブキーすべてを 書き込むには、STARTAT=オプションを使用します。

LISTREG='*libname.registry-name*'
ログに指定レジストリの内容をリストします。

libname
レジストリファイルが存在するライブラリの名前です。

registry-name
レジストリの名前です。

次に例を示します。

```
proc registry listreg='sashelp.regstry';
run;
```

操作 単一キーとそのサブキーすべてをリストするには、STARTAT=オプションを使用します。

LISTUSER
レジストリの Sasuser 部分の内容を SAS ログに書き込みます。

操作 単一キーとそのサブキーすべてを 書き込むには、STARTAT=オプションを使用します。

例 “例 2: レジストリファイルのリスト表示とエクスポート” (1734 ページ)

STARTAT='*key-name*'
単一キーとそのサブキーすべての 内容をエクスポートするか書き込みます。

ルートキーの下の任意のサブキーでリスト表示を開始する場合、キーシーケンス全体を指定する必要があります。

操作 STARTAT=は、EXPORT=、LIST、LISTHELP、LISTUSER、COMPAREREG1=、COMPAREREG2=、LISTREG オプションと一緒に使用します。

例 “例 4: レジストリファイルの比較” (1737 ページ)

UNINSTALL=*file-specification*
指定ファイル内のすべてのキーと 値を指定レジストリから削除します。

File-specification は次の値のうちのいずれかになります。

'***external-file***'
削除するキーと値を含む外部ファイルの名前です。

fileref
外部ファイルに割り当てられたファイル参照名です。ファイル参照名を割り当てるには、次の操作を実行します。

- エクスプローラーウィンドウを使用
- "FILENAME ステートメント" (*SAS グローバルステートメント: リファレンス*)を使用

操作 デフォルトでは、UNINSTALL で SAS レジストリの Sasuser 部分からキーと値が削除されます。レジストリの Sashelp 部分からキーと値を削除するには、USESASHELP オプションを指定します。このオプションを使用するには、Sashelp への書き込み権限が必要です。

レジストリのアンインストール時に SAS ログで追加情報を取得するには、FULLSTATUS を使用します。

参照項目: 使用法: REGISTRY プロシジャ (1730 ページ)(レジストリ情報を含むファイルの構築方法の詳細)

UPCASE
すべての内向きのキー名に大文字を使用します。

UPCASEALL
ファイルのインポート時に、すべてのキー、名前および項目値に大文字を使用します。

USESASHELP
SAS レジストリの Sashelp 部分に対する指定操作を実行します。

操作 USESASHELP は、IMPORT=、EXPORT=、COMPARETO、UNINSTALL のいずれかのオプションと一緒に使用します。USESASHELP を IMPORT=または UNINSTALL と一緒に使用するには、Sashelp への書き込み権限が必要です。

使用法: REGISTRY プロシジャ

レジストリファイルの構造

SAS レジストリエディターまたはテキストエディターを 使用してレジストリファイルを作成できます。

レジストリファイルには特定の構造が必要です。レジストリファイルの各エントリは、キー名と、それに続く次行の 1 つ以上の値から成ります。キー名では、定義するキーまたはサブキーが識別されます。その後の値ではどれでも、キーに関連付ける名前またはデータが指定されます。

キー名の指定

キー名は、角かっこ([と])の間に 1 行で入力します。サブキーを指定するには、ルートキーで始まる複数のキー名をかっこの間に入力します。複数のキー名を並べる場合は、バックスラッシュ(\)で名前を区切ります。単一キー名またはキー名のシーケンスの長さは 255 文字(角かっことバックスラッシュを含む)を超えないようにしてください。キー名には、バックスラッシュ以外の任意の文字を含められます。

有効なキー名シーケンスの例は次のとおりです。これらのシーケンスは典型的な SAS レジストリです。

- [CORE\EXPLORER\MENUS\ENTRIES\CLASS]
- [CORE\EXPLORER\NEWMEMBER\CATALOG]
- [CORE\EXPLORER\NEWENTRY\CLASS]
- [CORE\EXPLORER\ICONS\ENTRIES\LOG]

キーの値の指定

キー名に続く行に、関連付ける各値を入力します。各キーに複数の値を指定できますが、各値は別々の行に入力します。

値の一般的な形式は次のとおりです。

value-name=value-content

value-name には、デフォルト値の名前を示すアットマーク(@)かまたは二重引用符で囲まれた任意のテキスト文字列を指定できます。テキスト文字列にアンパサンド(&)が含まれている場合、アンパサンドに続く文字(大文字か小文字のどちらか)は値名のショートカットです。詳細については、“サンプルレジストリエントリ” (1732 ページ)を参照してください。

文字列全体が 255 文字(引用符とアンパサンドを含む)を超えないようにしてください。これにはバックスラッシュ(\)以外の任意の文字を含められます。

Value-content は次のいずれかになります。

- 後に数値が続いている文字列 **double:**.
- 文字列。かっこ内に任意の文字を入れられます。何も入れず("")とすることも可能です。

  注: 引用符で囲まれている文字列にバックスラッシュを入力するには、隣接する 2 つのバックスラッシュを使用します。二重引用符を含めるには、隣接する 2 つの二重引用符を使用します。

- 文字列 **hex:**に続けて 255 字までの任意の 16 進文字の数字を指定し、カンマで区切ります。16 進文字が 1 行を超える場合は、行末をバックスラッシュにして、

データが次行に続くことを示します。レジストリエディターでは、16 進値は"バイナリ値"とも呼ばれます。

- 文字列 **dword:**に続けて、符号なしの長い 16 進値を指定します。
- 文字列 **int:**に続けて、符号付きの長い整数値を指定します。
- 文字列 **uint:**に続けて、符号なしの長い整数値を指定します。

次のリストには、有効なレジストリ値のサンプルが含まれます。

- 倍精度値=double:2.4E-44
- 文字列="my data"
- バイナリデータ=16 進数: 01,00,76,63,62,6B
- 倍長語=dword:00010203
- 符号付き整数値=int:-123
- 符号なし整数値(小数)=dword:0001E240

サンプルレジストリエントリ

レジストリエントリの内容と表示は、目的に応じて変化する場合があります。

実際のレジストリテキストファイルがどのように表示されるかを確認するには、PROC REGISTRY で、LISTUSER および STARTAT=オプションを使用して、レジストリキーの内容を SAS ログに書き込みます。

次の例は、SASUSER レジストリエントリをログに送信する構文を示しています。

```
proc registry
   listuser
   startat='sasuser-registry-key-name';
run;
```

次の例は、STARTAT=オプションの値を示しています。

```
proc registry
   listuser
   startat='HKEY_SYSTEM_ROOT\CORE\PRINTING\PRINTERS\PostScript\DEFAULT
SETTINGS';
run;
```

次の例では、サブキーのリストは CORE\PRINTING\PRINTERS\PostScript\DEFAULT SETTINGS キーから始まります。

例のコード 50.1 PostScript プリンターのレジストリエントリ

```
NOTE: Contents of SASUSER REGISTRY starting at subkey [CORE\
PRINTING\PRINTERS\PostScript\DEFAULT SETTINGS key]

 Font Character Set="Western"
 Font Size=double:12
 Font Style="Regular"
 Font Typeface="Courier"
 Font Weight="Normal"
 Margin Bottom=double:0.5
 Margin Left=double:0.5
 Margin Right=double:0.5
 Margin Top=double:0.5
 Margin Units="IN"
 Paper Destination=""
 Paper Size="Letter"
 Paper Source=""
 Paper Type=""
 Resolution="300 DPI"
```

例: REGISTRY プロシジャ

例 1: SAS レジストリへのファイルのインポート

要素: IMPORT=
FILENAME ステートメント

詳細

この例では、SAS レジストリの Sasuser 部分にファイルがインポートされます。次のソースファイルには、レジストリファイルの有効なキー名シーケンスの例が含まれています。

```
[HKEY_USER_ROOT\AllGoodPeopleComeToTheAidOfTheirCountry]
 @="This is a string value"
  "Value2"=""
  "Value3"="C:\\This\\Is\\Another\\String\\Value"
```

プログラム

```
filename source 'external-file';

proc registry import=source;
run;
```

プログラムの説明

レジストリの有効テキストを含むファイルにファイル参照名を割り当てます。 FILENAME ステートメントで、ファイル参照名 SOURCE が、レジストリに読み込むテキストを含む外部ファイルに割り当てられます。

```
filename source 'external-file';
```

PROC REGISTRY を呼び出して、レジストリの入力を含むファイルをインポートします。 PROC REGISTRY では、ファイル参照名 SOURCE によって識別される入力ファイルが読み取られます。IMPORT=では、デフォルトで、SAS レジストリの Sasuser 部分に書き込まれます。

```
proc registry import=source;
run;
```

ログ

例のコード 50.2　SAS レジストリへファイルをインポートした結果

```
Parsing REG file and loading the registry please wait....
Registry IMPORT is now complete.
```

例 2: レジストリファイルのリスト表示とエクスポート

要素: EXPORT=
LISTUSER

詳細

通常、レジストリファイルは非常に大きいです。レジストリの一部をエクスポートするには、STARTAT=オプションを使用します。

この例では、SAS レジストリの Sasuser 部分がリスト表示され、外部ファイルにエクスポートされます。

プログラム

```
proc registry
  listuser
  export='external-file';
run;
```

プログラムの説明

レジストリの Sasuser 部分の内容を SAS ログに書き込みます。 LISTUSER オプションを使用すると、PROC REGISTRY では、レジストリの Sasuser 部分全体がログに書き込まれます。

```
proc registry
  listuser
```

レジストリを指定ファイルにエクスポートします。 EXPORT=オプションでは、SAS レジストリの Sasuser 部分のコピーが外部ファイルに書き込まれます。

```
  export='external-file';
run;
```

ログ

例のコード 50.3　SAS レジストリファイルをリスト表示しエクスポートした結果

```
Starting to write out the registry file, please wait...
The export to file external-file is now complete.
Contents of SASUSER REGISTRY.
[ HKEY_USER_ROOT]
[   CORE]
[    EXPLORER]
[     CONFIGURATION]
      Initialized= "True"
[     FOLDERS]
[       UNXHOST1]
        Closed= "658"
        Icon= "658"
        Name= "Home Directory"
        Open= "658"
        Path= "~"
```

例 3: レジストリと外部ファイルの比較

要素: COMPARETO=オプション
FILENAME ステートメント

詳細

この例では、SAS レジストリの Sasuser 部分を外部ファイルと比較します。.txt ファイル拡張子で保存されたバックアップファイルと現在のレジストリファイルの差異を確認する場合、このような比較が有用です。

レジストリの Sashelp 部分と外部ファイルを比較するには、USESASHELP オプションを指定します。

この SAS ログには、レジストリの Sasuser 部分と指定外部ファイルの差異が 2 つ示されます。"Initialized"の値はレジストリでは"True"ですが、外部ファイルでは"False"です。“Icon”の値はレジストリでは“658”ですが、外部ファイルでは“343”です。

プログラム

```
filename testreg 'external-file';

proc registry
  compareto=testreg;
run;
```

プログラムの説明

レジストリと比較するテキストを含む外部ファイルにファイル参照名を割り当てます。 FILENAME ステートメントで、ファイル参照名 TESTREG が外部ファイルに割り当てられます。

```
filename testreg 'external-file';
```

指定ファイルと SAS レジストリの Sasuser 部分を比較します。 COMPARETO オプションでは、ファイルとレジストリの内容が比較されます。キーと値がファイルにあってレジストリにはない場合、その情報が返されます。

```
proc registry
  compareto=testreg;
```

```
run;
```

ログ

例のコード 50.4　レジストリと外部ファイルを比較した結果

```
Parsing REG file and comparing the registry please wait....
COMPARE DIFF: Value "Initialized" in
[HKEY_USER_ROOT\CORE\EXPLORER\CONFIGURATION]: REGISTRY TYPE=STRING, CURRENT
VALUE="True"
COMPARE DIFF: Value "Initialized" in
[HKEY_USER_ROOT\CORE\EXPLORER\CONFIGURATION]: FILE TYPE=STRING, FILE
VALUE="False"
COMPARE DIFF: Value "Icon" in
[HKEY_USER_ROOT\CORE\EXPLORER\FOLDERS\UNXHOST1]: REGISTRY TYPE=STRING,
CURRENT VALUE="658"
COMPARE DIFF: Value "Icon" in
[HKEY_USER_ROOT\CORE\EXPLORER\FOLDERS\UNXHOST1]: FILE TYPE=STRING, FILE
VALUE="343"
Registry COMPARE is now complete.
COMPARE: There were differences between the registry and the file.
```

例 4: レジストリファイルの比較

要素: COMPAREREG1=および COMPAREREG2=オプション
STARTAT=オプション

詳細

この例では、REGISTRY プロシジャオプション COMPAREREG1=および COMPAREREG2=を使用して、比較する 2 つのレジストリファイルを指定します。

プログラム

```
libname proclib 'SAS-library';
proc registry comparereg1='sasuser.regstry'
  startat='CORE\EXPLORER'
  comparereg2='proclib.regstry';
run;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリにはレジストリファイルが含まれます。

```
libname proclib 'SAS-library';
```

PROC REGISTRY を開始し、比較に使用する最初のレジストリファイルを指定します。

```
proc registry comparereg1='sasuser.regstry'
```

指定レジストリキー以降のレジストリキーとの比較に制限します。 STARTAT=オプションでは、比較スコープが、CORE キーの下の EXPLORER サブキーに制限されます。デフォルトでは、両レジストリの内容全体が比較対象になります。

```
  startat='CORE\EXPLORER'
```

比較に使用する 2 番目のレジストリファイルを指定します。

```
  comparereg2='proclib.regstry';
run;
```

ログ

例のコード 50.5　2 つのレジストリファイルを比較した結果

```
NOTE: Comparing registry SASUSER.REGSTRY to registry PROCLIB.REGSTRY
NOTE: Diff in Key (CORE\EXPLORER\MENUS\FILES\SAS) Item (1;&Open)
SASUSER.REGSTRY Type: String len  17 data PGM;INCLUDE '%s';
PROCLIB.REGSTRY Type: String len  15 data WHOSTEDIT '%s';

NOTE: Diff in Key (CORE\EXPLORER\MENUS\FILES\SAS) Item (3;&Submit)
SASUSER.REGSTRY Type: String len  23 data PGM;INCLUDE '%s';SUBMIT
PROCLIB.REGSTRY Type: String len  21 data WHOSTEDIT '%s';SUBMIT

NOTE: Diff in Key (CORE\EXPLORER\MENUS\FILES\SAS) Item (4;&Remote Submit)
SASUSER.REGSTRY Type: String len  35 data SIGNCHECK;PGM;INCLUDE '%s';RSUBMIT;
PROCLIB.REGSTRY Type: String len  33 data SIGNCHECK;WHOSTEDIT '%s';RSUBMIT;

NOTE: Diff in Key (CORE\EXPLORER\MENUS\FILES\SAS) Item (@)
SASUSER.REGSTRY Type: String len  17 data PGM;INCLUDE '%s';
PROCLIB.REGSTRY Type: String len  15 data WHOSTEDIT '%s';

NOTE: Item (2;Open with &Program Editor) in key
    (CORE\EXPLORER\MENUS\FILES\TXT) not found in registry PROCLIB.REGSTRY
NOTE: Diff in Key (CORE\EXPLORER\MENUS\FILES\TXT) Item (4;&Submit)
SASUSER.REGSTRY Type: String len  24 data PGM;INCLUDE '%s';SUBMIT;
PROCLIB.REGSTRY Type: String len  22 data WHOSTEDIT '%s';SUBMIT;

NOTE: Diff in Key (CORE\EXPLORER\MENUS\FILES\TXT) Item (5;&Remote Submit)
SASUSER.REGSTRY Type: String len  35 data SIGNCHECK;PGM;INCLUDE '%s';RSUBMIT;
PROCLIB.REGSTRY Type: String len  33 data SIGNCHECK;WHOSTEDIT '%s';RSUBMIT;
```

例 5: STARTAT=オプションを使用したキーシーケンス全体の指定

要素: EXPORT オプション
STARTAT=オプション

詳細

次の例では、STARTAT=オプションの使用方法を示します。ルートキーの下の任意のサブキーでリスト表示を開始する場合、キーシーケンス全体を指定する必要があります。ルートキーはオプションです。

プログラム

```
proc registry export = my-fileref
startat='core\explorer\icons';
run;
```

例 6: フォントリストの表示

要素: LISTHELP オプション
STARTAT オプション

詳細

次の例では、ODS フォントのリストが SAS ログに書き込まれます。

プログラム

```
proc registry clearsasuser; run;
proc registry listhelp startat='ods\fonts'; run;
proc registry clearsasuser; run;
```

ログ

例のコード 50.6　SAS レジストリのフォントリストを表示した結果

```
NOTE: Contents of SASHELP REGISTRY starting at subkey [ods\fonts]
[ ods\fonts]
 dings="Wingdings"
 monospace="Courier New"
 MTdings="Monotype Sorts"
 MTmonospace="Cumberland AMT"
 MTsans-serif="Albany AMT"
 MTsans-serif-unicode="Monotype Sans WT J"
 MTserif="Thorndale AMT"
 MTserif-unicode="Thorndale Duospace WT J"
 MTsymbol="Symbol MT"
 sans-serif="Arial"
 serif="Times New Roman"
 symbol="Symbol"
[  ja_JP]
  dings="Wingdings"
  monospace="MS Gothic"
  MTdings="Wingdings"
  MTmonospace="MS Gothic"
  MTsans-serif="MS PGothic"
  MTsans-serif-unicode="MS PGothic"
  MTserif="MS PMincho"
  MTserif-unicode="MS PMincho"
  MTsymbol="Symbol"
  sans-serif="MS PGothic"
  serif="MS PMincho"
  symbol="Symbol"
[  ko_KR]
  dings="Wingdings"
  monospace="GulimChe"
  MTdings="Wingdings"
  MTmonospace="GulimChe"
  MTsans-serif="Batang"
  MTsans-serif-unicode="Batang"
  MTserif="Gulim"
  MTserif-unicode="Gulim"
  MTsymbol="Symbol"
  sans-serif="Batang"
  serif="Gulim"
  symbol="Symbol"
[  th_TH]
  dings="Wingdings"
  monospace="Thorndale Duospace WT J"
  MTdings="Monotype Sorts"
  MTmonospace="Cumberland AMT"
  MTsans-serif="Monotype Sans WT J"
  MTsans-serif-unicode="Thorndale Duospace WT J"
  MTserif="Thorndale Duospace WT J"
  MTserif-unicode="Monotype Sans WT J"
  MTsymbol="Symbol MT"
  sans-serif="Angsana New"
  serif="Thorndale Duospace WT J"
  symbol="Symbol"
```

51

REPORT プロシジャ

概要: REPORT プロシジャ

REPORT プロシジャの機能

REPORT プロシジャは、PRINT プロシジャ、MEANS プロシジャ、TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートの作成が可能な 1 つのレポート作成ツールに組み合わせます。PROC REPORT は、次のように使用します。

- 他の SAS プロシジャと同じように、PROC REPORT ステートメントで一連のステートメントをサブミットします。これらのステートメントは、プログラムエディターから PROC REPORT ステートメントでサブミットします。
- アクセス可能な出力テーブルを作成します。ACCESSIBLETABLE システムオプションが指定されている場合、一部のテーブルのレイアウトを変更し、アクセス可能にします。ACCESSIBLECHECK システムオプションを使用して、テーブルがアクセス可能かどうかを確認できます。ACCESSIBLETABLE システムオプションが使用され、CAPTION=オプションが PROC REPORT で使用される場合、キャプションが出力に表示されます。

  PROC REPORT を使用してアクセス可能なテーブルを作成する方法の詳細については、*ODS および ODS Graphics を使用したアクセシブルな SAS Viya プラットフォーム出力の作成* を参照してください。

PROC REPORT で作成可能なレポートの種類について

詳細レポートには、レポートに選択されたすべてのオブザベーションに対して 1 行が含まれます。これらの行はそれぞれレポート行、**詳細レポート行**です。**要約レポート** では、1 行で複数のオブザベーションを表すようにデータがまとめられます。これらの行はそれぞれ詳細行、**要約レポート行**とも呼ばれます。

詳細レポートと要約レポートには、レポート行のほかに**要約レポート行**(ブレーク行)を含めることができます。要約行では、一連の詳細行またはすべての詳細行の数値データが要約されます。PROC REPORT は、デフォルト要約とカスタマイズされた要約を提供します“ブレーク行の使用” (1760 ページ)を参照してください。

この概要では、PROC REPORT で作成可能なレポートの種類を説明します。これらのレポートで使用されるデータセットと出力形式を作成するステートメントについては、“例 1: 変数の選択とレポートの要約行の作成” (1849 ページ)を参照してください。出力形式は、永久 SAS ライブラリに保存されます。追加レポートとそれを作成するステートメントについては、REPORT プロシジャの例を参照してください。

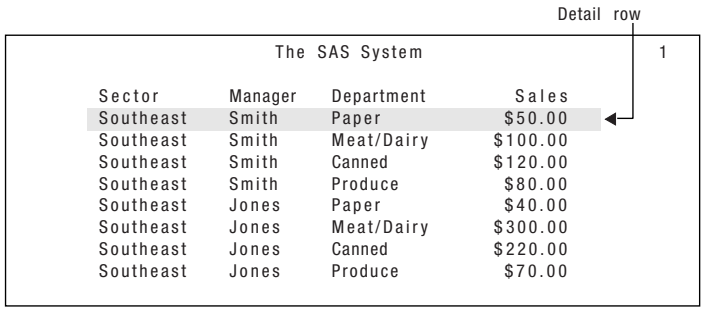
各種レポートの表示形式について

これらのレポートで使用されるデータセットには、食料品店チェーンの 8 店舗の 1 日の売上高が含まれます。

単一 PROC REPORT ステップにより、単一の PROC PRINT ステップにより作成されるレポートと似たレポートが作成されます。図 51.33 (1744 ページ)は、PROC REPORT で作成可能な最も単純な種類のレポートを示しています。レポートを作成するステートメントがその後に続きます。プログラムで使用するデータセットと出力形式は、“例 1: 変数の選択とレポートの要約行の作成” (1849 ページ)で作成されます。WHERE ステートメントと FORMAT ステートメントは重要ではありませんが、ここでは出力量を制限し、値をわかりやすくします。

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=64 pagesize=60
      fmtsearch=(proclib);


proc report data=grocery;
  where sector='se';
  format sector $sctrfmt. manager $mgrfmt.
       dept $deptfmt. sales dollar10.2;
run;
```

図 51.1　各オブザベーションの詳細行を含む単純な詳細レポート

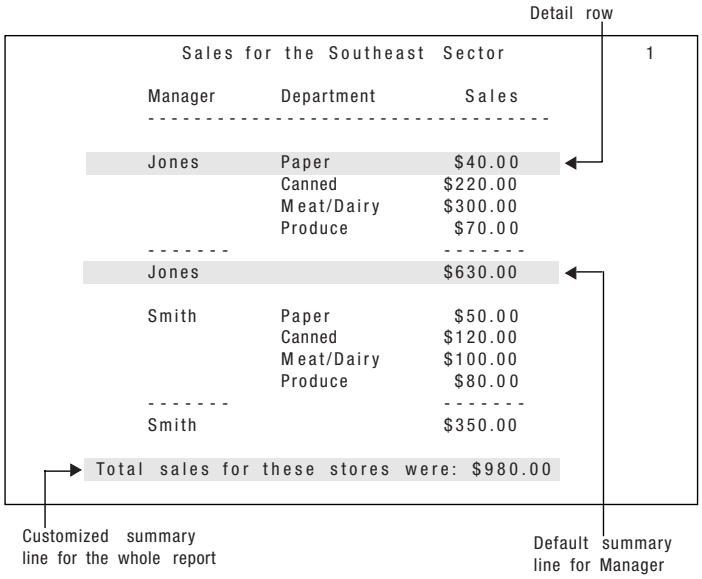


次の図のレポートでは、上図と同じオブザベーションを使用しています。ただし、このレポートを作成するステートメントは、次を実行します。

- マネージャーと部門の値によって行を並べ替える。
- マネージャーの値ごとにデフォルト要約行を作成する。
- レポート全体のカスタマイズされた要約行を作成する。カスタマイズされた要約によって要約情報のコンテンツと表示形式を制御できますが、追加の PROC REPORT ステートメントを記述して作成する必要があります。

このレポートを作成するプログラムの説明については、“例 2: レポートでの行の並べ替え” (1853 ページ)を参照してください。

図 51.2　デフォルト要約とカスタマイズされた要約を含む並べ替えられた詳細レポート

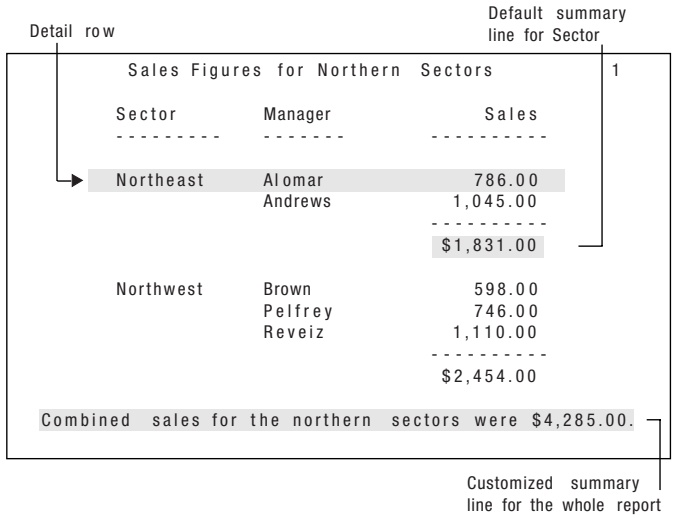


次の図の要約レポートには、北セクタの店舗ごとに 1 行が含まれています。各詳細行は、入力データセットにおける 4 つのオブザベーション、部門ごとに 1 つのオブザベーションを表しています。個別の部門に関する情報は、このレポートには表示されません。代わりに、各詳細行の売上値は、すべての 4 つの部門の売上値の合計です。複数のオブザベーションをレポートの 1 行にまとめるだけでなく、このレポートを作成するステートメントは、次を実行します。

- 列ヘッダーのテキストをカスタマイズする
- 市の各セクタに対する売上を総計するデフォルト要約行を作成する
- 両方のセクタに対する売上を総計するカスタマイズされた要約行を作成する

このレポートを作成するプログラムの説明については、“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)を参照してください。

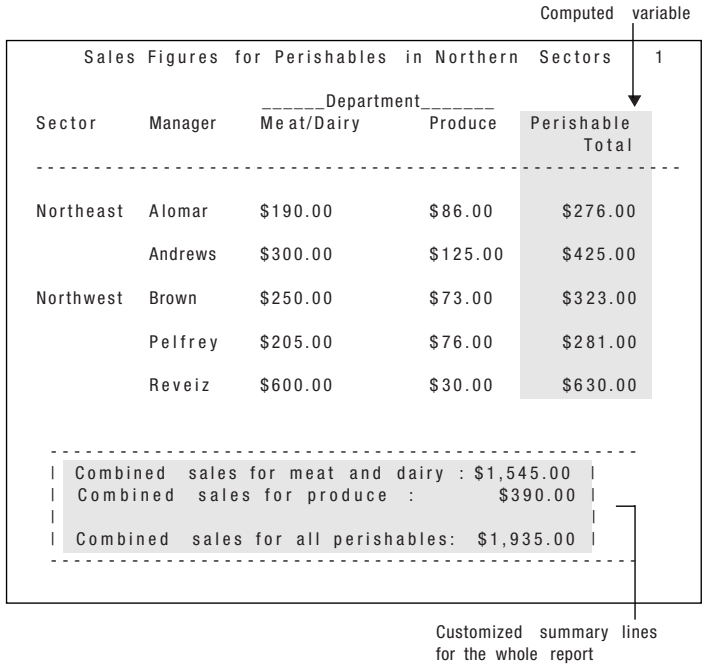
図 51.3　デフォルト要約とカスタマイズされた要約を含む要約レポート



次の図の要約レポートは、上図に似ています。大きな違いは、個別の部門に関する情報も含まれている点です。部門の選択値がそれぞれレポートの列となります。また、このレポートを作成するステートメントでは、入力データセットにない変数を計算し、表示します。

このレポートを作成するプログラムの説明については、“例 6: 変数の値ごとに列を作成する” (1866 ページ)を参照してください。

図 51.4　変数の各値の列を含む要約レポート

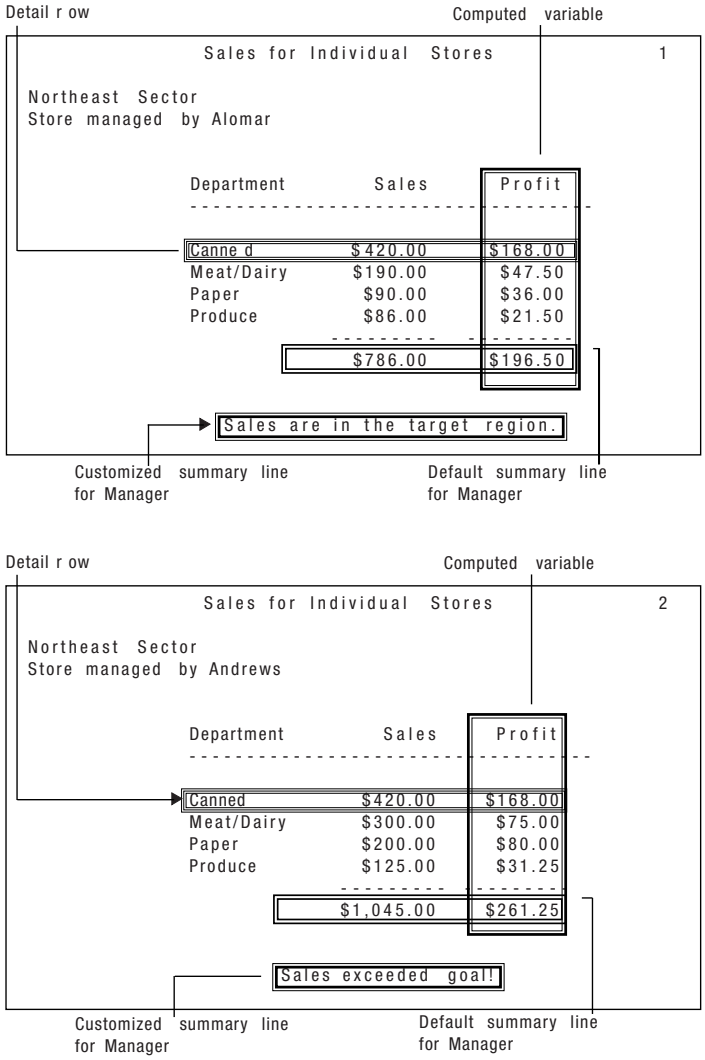


次の図のカスタマイズされたレポートには、各マネージャーの店舗を個別のページに表示します。ここでは最初の 2 ページだけを表示しています。このレポートを作成するステートメントは、次を作成します。

- レポートのページごとにカスタマイズされたヘッダー
- 入力データセットにない計算済み変数(利益)
- マネージャーの店舗の売上合計に依存するテキストを含むカスタマイズされた要約

このレポートを作成するプログラムの説明については、“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)を参照してください。

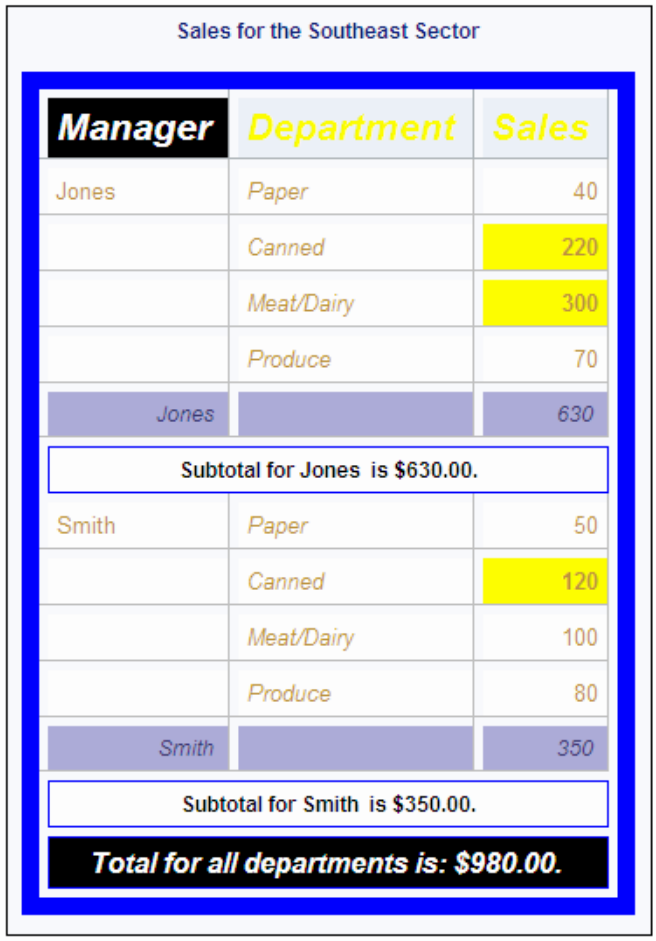
図 51.5 カスタマイズされた要約レポート



次の図のレポートでは、フォントの種類、フォントサイズ、位置合わせ、およびテーブルの枠の幅、セル間のスペースの 幅などを制御するためのカスタマイズされたスタイル要素を使用しています。このレポートは、Output Delivery System (ODS) の HTML 出力先と STYLE=オプションをプロシジャの複数のステートメントで使用して作成されました。

このレポートを作成するプログラムの説明については、“例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1892 ページ)を参照してください。ODS の詳細については、 “Output Delivery System” (71 ページ)を参照してください。

図 51.6　HTML 出力



概念: REPORT プロシジャ

レポートのレイアウト

レイアウトの計画

レポート作成は、レポートの表示形式を明確に理解して取り組むと簡単です。決定すべき重要なことは、レポートのレイアウトです。レイアウトを設計するには、次の点について考えます。

- レポートの各列に表示する項目
- 各列を表示する順序
- 特定の変数の値ごとの列の表示の要否
- レポートで、オブザベーションごとに 1 行を使用する必要性、あるいは複数のオブザベーションの情報を 1 行にまとめる必要性の要否
- 各行を表示する順序

レポートのレイアウトを考えたら、PROC REPORT で COLUMN ステートメントと DEFINE ステートメントを使用して、レイアウトを構成します。

COLUMN ステートメントは、レポートの列に表示する項目をリストし、列の調整を記述し、複数の列にわたるヘッダーを定義します。レポート項目には、次が可能です。

- データセット変数
- プロシジャによって 計算された統計量
- レポートのその他の項目から計算する変数

入力データセットのすべての変数をデータセットと同じ順序で含める場合、COLUMN ステートメントを省略します。

DEFINE ステートメントで、レポートの項目の特性を定義します。これらの特性には、レポートの項目、列ヘッダーのテキスト、値の表示に使用する出力形式を PROC REPORT がどのように使用するかが含まれます。

レポートでの変数の使い方

レポートのレイアウトの多くは、DEFINE ステートメントで変数に指定する使い方によって決定されます。データセット変数の場合、次のような使い方があります。

DISPLAY ORDER ACROSS GROUP ANALYSIS

レポートには、入力データセットにない変数を含めることができます。これらの変数の使い方が COMPUTED である必要があります。

表示変数

1 つ以上の表示変数を含むレポートには、入力データセットのオブザベーションごとに 1 行が含まれます。表示変数は、レポートの行の順序に影響しません。順序変数が表示変数の左側に表示されない場合、レポートの行の順序は、データセットのオブザベーションの順序を反映します。デフォルトで、PROC REPORT はすべての文字変数を表示変数として扱います。例については、“例 1: 変数の選択とレポートの要約行の作成” (1849 ページ)を参照してください。

順序変数

1 つ以上の順序変数を含むレポートには、入力データセットのオブザベーションごとに 1 行が含まれます。表示変数が順序変数の左側に表示されない場合、PROC REPORT は順序変数のフォーマットした値の昇順で詳細行を並べ替えます。DEFINE ステートメントで ORDER=および DESCENDING を使用して、デフォルトの順序を変更できます。

レポートに複数の順序変数が含まれている場合、PROC REPORT は、これらの変数をレポートの左から右へ並べ替えることにより詳細行の順序を作成します。PROC REPORT は、その左側の順序変数が値を変更しない限り、値が変わらない場合は、順序変数の値を行から行に繰り返しません。例については、“例 2: レポートでの行の並べ替え” (1853 ページ)を参照してください。

オブザベーションの順序は、DBMS テーブルに本来定義されていません。DBMS テーブルの入力データに ORDER=DATA オプションを指定した場合、PROCREPORT からデータベーステーブルに書き込まれる行の順序は保持されない可能性があります。

グループ変数

レポートに 1 つ以上のグループ変数が含まれている場合、PROC REPORT はすべてのグループ変数に対しフォーマットされた値の一意の組み合わせを持つデータセットからのすべてのオブザベーションを 1 行にまとめようとします。

PROC REPORT はグループを作成するとき、グループ変数のフォーマットされた値の昇順で詳細行を並べ替えます。DEFINE ステートメントで ORDER=および DESCENDING を使用して、デフォルトの順序を変更できます。

レポートに複数のグループ変数が含まれている場合、REPORT プロシジャは、これらの変数をレポートの左から右へ並べ替えることにより詳細行の順序を作成します。PROC REPORT は、その左側のグループ変数が値を変更しない限り、値が変わらない場合は、グループ変数の値を行から行に繰り返しません。

クラス変数を使用するプロシジャを熟知していると、グループ変数が PROC TABULATE の行ディメンションで使用されるクラス変数であることがわかります。

注: グループを常に作成できるわけではありません。レポートに 1 つ以上の統計量が関連付けられていない順序変数または表示変数が含まれている場合、PROC REPORT はオブザベーションをグループにまとめることはできません。COLUMN ステートメント (1793 ページ)を参照してください。SAS ログのメッセージは、グループを作成できなかった理由を説明しています。代わりに、PROC REPORT は順序変数と同じようにグループ変数を表示する詳細レポートを作成します。PROC REPORT が詳細レポートを作成する場合でも、グループ変数として定義する変数にはその使い方が定義に保持されます。

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)を参照してください。

分析変数

分析変数は、レポートのセルによって表されるすべてのオブザベーションに対する統計量の計算に使用される数値変数です。(列変数をグループ変数または順序変数と組み合わせ、セルが表すオブザベーションを決定します)。変数の定義または COLUMN ステートメントで統計量と分析変数を関連付けます。デフォルトで PROC REPORT は、数値変数を合計統計量の計算に使用される分析変数として使用します。

分析変数の値は、レポートで表示される場所によって異なります。

- 詳細レポートでは、詳細行の分析変数の値は、単一のオブザベーションに対して計算された変数と関連付けられた統計量の値です。単一のオブザベーションに対する統計量の計算は、実用的ではありません。ただし、変数を分析変数として使用することにより、オブザベーションセットまたはすべてのオブザベーションに対し要約行を作成できます。
- 要約レポートで、分析変数に対して表示されている値は、レポートのセルによって表されるオブザベーションセットに対して計算されるように指定する統計量の値です。
- レポートの要約行で、分析変数の値は、要約行のセルによって表されるすべてのオブザベーションに対して計算されるように指定する統計量の値です。

詳細については、“BREAK ステートメント” (1781 ページ)と“RBREAK ステートメント” (1815 ページ)ステートメントを参照してください。

例として、“例 2: レポートでの行の並べ替え” (1853 ページ)、“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)、“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)、と“例 6: 変数の値ごとに列を作成する” (1866 ページ).を参照してください。

注: 要約行を含むレポートで SAS 日付を使用する際は注意が必要です。SAS 日付は数値変数です。日付は他種の変数(ORDER、GROUP、または DISPLAY)として明示的に定義しない限り、PROC REPORT によって要約されます。

列変数

PROC REPORT は、列変数の値ごとに列を作成します。PROC REPORT は、列を列変数のフォーマットした値の昇順で並べ替えます。DEFINE ステートメントで ORDER=および DESCENDING を使用して、デフォルトの順序を変更できます。その他の変数が列の定義に使用できない場合、PROC REPORT は N 統計量(レポートのセルに属している入力データセットのオブザベーション数)を表示します。COLUMN ステートメント (1793 ページ)を参照。

注: 表示変数と列変数が列を共有している場合、レポートには同じ列にはない別の変数も含まれている必要があります。列変数によって作成された列を参照する場合は、_cn_構文を使用する必要があります。

クラス変数を使用するプロシジャを熟知していると、列変数が PROC TABULATE で列ディメンションで使用されるクラス変数に似ていることがわかります。通常、列変数は、順序変数またはグループ変数を組み合わせて使用します。例については、“例 6: 変数の値ごとに列を作成する” (1866 ページ)を参照してください。

計算変数

計算変数は、レポートに対して定義する変数です。入力データセットにはなく、PROC REPORT によって入力データセットに追加されません。ただし、計算変数は作成されると出力データセットに含まれます。

計算変数は、次の方法で追加します。

- 計算された変数を COLUMN ステートメントに含める
- 変数の使用法を DEFINE ステートメントで COMPUTED として定義する
- 変数に関連付けられた計算ブロックで変数の値を計算する

例については、“例 6: 変数の値ごとに列を作成する” (1866 ページ)、“例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)、および“例 10: 出力データセットの作成と計算変数の保存” (1882 ページ)を参照してください。

位置と使い方の相互作用

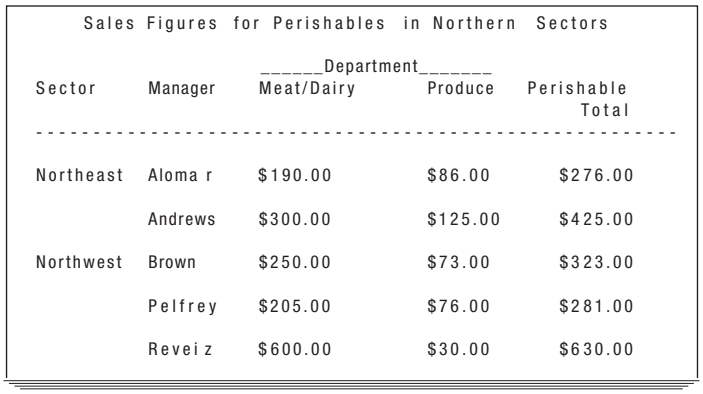
レポートの各変数の位置と使い方によって、レポートの構造とコンテンツが決まります。PROC REPORT は、順序変数とグループ変数のフォーマットされた値に従ってレポートの行を、COLUMN ステートメントで指定されているように左から右に並べ替えます。同様に、PROC REPORT は、列変数の列を変数の値に従って左から右に並べ替えます。

複数の項目は、レポートの列のコンテンツを一括で定義できます。たとえば次の図では、3 番目と 4 番目の列に表示される値は、売上(分析変数)および部門(列変数)によって一括で決定されます。この調整は、レポートの積み重ね項目と呼ばれます。各項目がヘッダーを生成し、ヘッダーが相互に積み重ねられるためです。

```
options nodate pageno=1 fmtsearch=(proclib);
proc report data=grocery split='*';
  column sector manager department,sales perish;
  define sector / group format=$sctrfmt. 'Sector' '';
  define manager / group format=$mgrfmt. 'Manager* ';
  define department/ across format=$deptfmt. '_Department_';
  define sales / analysis sum format=dollar11.2 ' ';
  define perish / computed format=dollar11.2 'Perishable Total';
  break after manager / skip;
  compute perish;
    perish=_c3_+_c4_;
```

```
  endcomp;
  title "Sales Figures for Perishables in Northern Sectors";
  where sector contains 'n' and (department='p1' or department='p2');
run;
title;
```

図 51.7 部門と売上の積み重ね



Sales Figures for Perishables in Northern Sectors

Sector	Manager	Department Meat/Dairy	Department Produce	Perishable Total
Northeast	Aloma r	$190.00	$86.00	$276.00
	Andrews	$300.00	$125.00	$425.00
Northwest	Brown	$250.00	$73.00	$323.00
	Pelfrey	$205.00	$76.00	$281.00
	Reveiz	$600.00	$30.00	$630.00

複数の項目を使用して列のコンテンツを定義する場合、多くても次のうち 1 つの項目を 1 つの列に含めることができます。

- 表示変数(それ以上/以下の統計量を含む/含まない)
- 分析変数(それ以上/以下の統計量を含む/含まない)
- 順序変数
- グループ変数
- 計算変数

これらのうち複数の項目が 1 つの列にあると、表示する値に関して RROC REPORT で競合が生じます。

次のテーブルに、列を共有できるレポート項目を示します。

注: 順序変数をその他のレポート項目と積み重ねることはできません。

表 51.1 列を共有できるレポート項目

	表示 (Display)	分析 (Analysis)	順序 (Order)	グループ (Group)	計算 (Computed)	列(Across)	統計量
表示 (Display)						X*	X
分析 (Analysis)						X	X
順序 (Order)							

	表示 (Display)	分析 (Analysis)	順序 (Order)	グループ (Group)	計算 (Computed)	列(Across)	統計量
グループ (Group)						X	
計算変数						X	
列(Across)	X*	X			X	X	X
統計量	X	X				X	

*表示変数と列変数が列を共有する場合、レポートには同じ列にはない別の変数も含まれている必要があります。

列が積み重ねレポート項目によって定義されている場合、PROC REPORT は ACROSS の使い方を持たない積み重ねの最下位レポート項目に指定される出力形式を使用することによって、列の値にフォーマットを適用します。

次の項目が列を専有できます。

- 表示変数
- 分析変数
- 順序変数
- グループ変数
- 計算変数
- 列変数
- N 統計量

注: 列変数によって専有される列の値は、度数カウントです。

計算ブロックの使用

計算ブロックについて

計算ブロックは、COMPUTE ステートメントと ENDCOMP ステートメントの間に表示される 1 つ以上のプログラミングステートメントです。これらの 2 つのステートメントの間で、出力をカスタマイズする他の SAS ステートメント(割り当て、CALL DEFINE、および LINE ステートメント)を使用できます。PROC REPORT は、COLUMN ステートメントの記述順序で左から右に変数を読み取ることによって、デ

ータセットを処理します。このプロシジャは一度に 1 列および 1 行ずつレポートを作成し、COMPUTE ステートメントはレポートの作成時に実行されます。

計算ブロックは、COMPUTE ステートメントで作成できます。COMPUTE ステートメントには、次の種類の引数を複数含められます。

report-item
レポート項目には、データセット変数、統計量、または計算変数を指定できます。

location
プロセスのどのポイントで *target* の値に関連して計算ブロックを実行するのかを指定します。この場所には、レポートの上下やオブザベーションセットの前後を指定できます。

target
いつ計算ブロックが実行するかを制御します。COMPUTE ステートメントに対して場所(BEFORE または AFTER)を指定した場合、*target* を指定できます。対象には、グループまたは順序変数あるいは_PAGE_変数を指定できます。

注: COMPUTE ステートメントを使用する場合、対応する BREAK または RBREAK ステートメントを使用する必要はありません。“ブレーク行の使用” (1760 ページ)を参照してください。“例 2: レポートでの行の並べ替え” (1853 ページ)を参照してください。ここでは、COMPUTE AFTER を使用しますが、RBREAK ステートメントは使用しません。これらのステートメントは、1 つ以上の BREAK ステートメントまたは RBREAK ステートメントオプションを実装するときにのみ使用します。“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)を参照してください。ここでは、COMPUTE AFTER MANAGER と BREAK AFTER MANAGER の両方が使用されます。

計算ブロックの使用法の詳細については、The REPORT Procedure: A Primer for the Compute Block を参照してください。

計算ブロックの目的

レポート項目と関連付けられている計算ブロックは、次のタスクを実行できます。

- レポートの列に表示され、入力データセットには表示されない変数を定義する。
- レポート項目の表示属性を定義する“CALL DEFINE ステートメント” (1790 ページ)を参照してください。
- 要約行の“合計”の表示など、レポート項目の値を定義または変更する。

場所と関連付けられている計算ブロックは、カスタマイズされた要約を記述できます。

また、すべての計算ブロックで、計算実行のために大部分の SAS 言語要素を使用できます。“計算ブロックのコンテンツ” (1756 ページ)を参照してください。PROC REPORT ステップには複数の計算ブロックを含めることができますが、それらをネストすることはできません。

計算ブロックのコンテンツ

計算ブロックは、プログラム編集ステートメントから始まり、ENDCOMP ステートメントで終わります。計算ブロック内では、次の SAS 言語要素を使用できます。

- %INCLUDE ステートメント
- DATA ステップステートメントは、次のとおりです。

  ARRAY
  配列参照
  割り当て
  CALL
  CONTINUE
  DO(すべてのフォーム)END
  END
  IF-THEN/ELSE
  LENGTH
  RETURN
  合計

- コメント
- Null ステートメント
- マクロ変数とマクロ呼び出し
- すべての DATA ステップ関数

また、計算ブロック内で、次のような PROC REPORT 機能を使用することもできます。

- カスタマイズされた要約の計算ブロックには、1 つ以上の LINE ステートメントを含めることができます。これによりカスタマイズされたテキストとフォーマットされた値が要約に挿入されます。“LINE ステートメント” (1814 ページ)を参照してください。
- レポート項目の計算ブロックには、1 つ以上の CALL DEFINE ステートメントを含めることができます。これにより、項目の値がレポートに挿入されるたびに色や出力形式などの属性が設定されます。“CALL DEFINE ステートメント” (1790 ページ)を参照してください。
- 任意の計算ブロックで自動変数_BREAK_を参照できます。“自動変数_BREAK_” (1761 ページ)を参照してください。

計算ブロックのレポート項目を参照するための 4 つの方法

計算ブロックは、レポートの列を形成するレポート項目を参照できます(列が表示されるかどうか)。計算ブロックのレポート項目を次の 4 つの方法のうちいずれかで参照します。

- 名前。
- 変数と、変数を使用して計算する統計量の名前の両方を識別する複合名。複合名は次のような形になります。

variable-name.statistic

- COLUMN ステートメントまたは **DEFINITION** ウィンドウを使用して作成する別名。
- 次の形式の列番号。

'_C*n*_'

ここで、*n* は、レポートの列の番号(左から右方向)です。

注: 列番号が必要なのは、COMPUTED 変数が列を ACROSS 変数と共有している場合だけです。

注: NOPRINT と NOZERO を使用して定義する列がレポートに表示されない場合でも、列を番号別に参照する場合はそれらの列をカウントする必要があります。“NOPRINT” (1806 ページ)と “NOZERO” (1807 ページ)の説明を参照してください。

注: 欠損値を含む変数を参照すると、欠損値が発生します。計算ブロックが欠損値を含む変数を参照する場合、PROC REPORT はその変数をブランク(文字変数の場合)またはピリオド(数値変数の場合)として表示します。

次のテーブルに、計算ブロックにおける各種類の参照の使用方法を示します。

変数の種類	参照元	例
グループ	名前*	部門
順序	名前*	部門
計算	名前*	部門
表示	名前*	部門
統計量と列を共有している表示	複合名*	Sales.sum
分析	複合名*	Sales.mean
列変数と列を共有する種類	列番号**	'_c3_'

*変数に別名がある場合、その変数を別名で参照する必要があります。

**変数に別名がある場合でも、変数を列番号別に参照する必要があります。

“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)(分析変数を別名別に参照)、“例 6: 変数の値ごとに列を作成する” (1866 ページ) (変数を列番号別に

参照)、および"例 8: レポートに計算されたパーセント列を表示する" (1874 ページ)(グループ変数と計算変数を名前別に参照)を参照してください。

計算ブロック処理

一般に、計算ブロックは、次の順序で実行されます。

```
1 COMPUTE report-item;
2 COMPUTE BEFORE;
3 COMPUTE BEFORE target;
4 COMPUTE BEFORE _PAGE_;
5 COMPUTE AFTER;
6 COMPUTE AFTER target;
7 COMPUTE AFTER _PAGE_;
```

COMPUTE ステートメントの動作は、含まれた引数に依存しています。次の構文と動作で、COMPUTE ステートメントの特定の引数に基づく計算ブロックのアクションを説明します。

注: PROC REPORT は値をレポート行の列に、左から右に割り当てます。その結果、レポートの右側に表示される変数に基づいて計算変数の計算を行うことはできません。PROC REPORT でレポートを作成する(一般的な)方法については、結果: REPORT プロシジャ (1838 ページ)を参照してください。

COMPUTE *report-item*;

report-item 引数を含めた場合、その特定の列の処理時に、すべてのオブザベーションに対して計算ブロックが実行されます。一般に、このステートメントを使用するのは、特定の *report-item* 列に対して、値の計算、値の変更、出力形式の変更、スタイル属性の適用、一時変数の作成のいずれかを行うためです。

COMPUTE BEFORE;

この構文では、計算ブロックは、最初の詳細行の前に実行されます。ブロックは一度だけ実行されます。分析変数の全体の要約値は、計算ブロックで使用可能です。また、このブロックで作成された一時変数の値は、他の計算ブロックで使用可能です。グループ変数または順序変数の値は、このブロックでは使用できません。LINE ステートメントによって生成されたテキストは、レポートのヘッダーと最初の詳細行の間に表示されます。CALL DEFINE ステートメントでは、RBREAK BEFORE ステートメントによって作成される行の属性が設定されます。

COMPUTE BEFORE *target*;

この構文を使用すると、ターゲット変数の値が変更されたときに計算ブロックが実行されます。この構文では:

- *target* は、グループ変数または順序変数のいずれかを指定します。

- BEFORE は *location* の値です。この値では、*target* の特定の値に対してセクションの上部(または最初)で計算ブロックを実行することが指定されます。
- レポートのこの特定のセクションのターゲット変数の値は、計算ブロックで使用できます。
- このブロックで作成された一時変数の値は、他の計算ブロックで使用可能です。
- この特定の *target* 値の要約値は、計算ブロックに使用可能です。
- CALL DEFINE ステートメントでは、BREAK BEFORE *target* ステートメントによって作成される行の属性が設定されます。

COMPUTE BEFORE _PAGE_;

計算ブロックは、ページごとに 1 回実行されます。改ページは、出力の送信先または次の BREAK ステートメント構文を使用して生成できます。

BREAK <location> <break-var> /PAGE;

通常、このブロックは、LINE ステートメントを使用してテキストを出力します。テキストは、ページ上部のヘッダーの上に表示されます。ただし、それでもなおテーブルの一部です。LINE ステートメントに記述された変数はいずれもレポートの最初の詳細行からその値を取得します。

COMPUTE AFTER;

計算ブロックは最後の詳細行の後に実行され、(各レポートごとに)一度だけ実行されます。分析変数の全体の要約値も計算ブロックで使用可能です。グループ変数または順序変数の値は、このブロックでは使用できません。LINE ステートメントによって生成されたテキストは、レポートの最後の詳細行の後に表示されます。PROC REPORT コードに BY ステートメントが含まれている場合、このブロックは各 BY 値の後に実行されます。CALL DEFINE ステートメントでは、RBREAK AFTER ステートメントによって作成される行の属性が設定されます。

COMPUTE AFTER *target*;

この構文を使用した場合、対象(ブレーク)変数の値が変更されると計算ブロックが実行されます。この構文では:

- *target* では、グループ変数か順序変数のどちらかとして定義された変数を指定します。
- AFTER は *location* の値です。この値では、*target* の特定の値に対してセクションの下部(または最後)で計算ブロックを実行することが指定されます。
- レポートのこの特定のセクションのターゲット変数の値は、計算ブロックで使用できます。
- この特定の *target* 値の要約値は、計算ブロックに使用可能です。
- CALL DEFINE ステートメントでは、BREAK AFTER *target* ステートメントによって作成される行の属性が設定されます。

COMPUTE AFTER _PAGE_;

計算ブロックは、ページごとに 1 回実行されます。改ページは、出力の送信先または次の BREAK ステートメント構文を使用して生成できます。

BREAK <location> <break-var> /PAGE;

LINE ステートメントに記述された変数はいずれもレポートの最後の詳細行からその値を取得します。テキストはテーブルの最後に配置されますが、それでもなおテーブルの一部です。

注: テキストの正確な場所は、各ページに出力される情報の量に基づいて、ページごとに変更される可能性があります。複数のページにまたがる場合、テキストはテーブルの最後のページに配置されます。

ブレーク行の使用

ブレーク行について

ブレーク行はテキストの行(ブランクを含む)で、レポートの**ブレーク**という特定の場所に表示されます。レポートには、複数のブレークを含めることができます。通常、ブレーク行はレポートの一部を視覚的に区切る、情報を要約する、またはその両方を行うために使用されます。ブレーク行は次の場所に示されます。

- レポートの始めと最後
- 各ページの上部または下部
- オブザベーションセット間(グループ変数または順序変数の値が変わるたび)

ブレーク行には次を含めることができます。

- text
- 行セットまたはレポート全体に対して計算された値

ブレーク行の作成

ブレーク行の作成には、2 つの方法があります。最初の方法はより簡単です。デフォルトの要約を作成します。2 つ目の方法はより複雑です。カスタマイズされた要約を作成し、デフォルト要約を多少変更する方法を提供します。デフォルト要約とカスタマイズされた要約は、レポートの同じ場所で表示可能です。

デフォルト要約は BREAK ステートメント、RBREAK ステートメントを使用して作成します。デフォルト要約を使用して、レポートの一部を視覚的に区切る、数値変数に関する情報を要約する、またはその両方を行うことができます。オプションによりブレーク行の表示形式をある程度制御できますが、数値変数の要約を選択する場合、要約情報のコンテンツと配置を制御できません。(情報を要約するブレーク行は、要約行です)。

カスタマイズされた要約は、計算ブロックで作成されます。カスタマイズされた要約の表示形式とコンテンツを制御できますが、それを行うためのコードを記述する必要があります。

ブレーク行の順序

カスタマイズされた要約の行の順序を制御します。ただし、PROC REPORT はデフォルト要約の行の順序、デフォルト要約に関連したカスタマイズされた要約の配置を制御します。デフォルト要約に複数のブレーク行が含まれている場合、ブレーク行が表示される順序は次のとおりです。

1 要約行

2 改ページ

自動変数_BREAK_

PROC REPORT は、_BREAK_という変数を自動的に作成します。この変数には、次が含まれます。

- ブランク(現在行がブレークの一部でない場合)
- ブレーク変数の値 (現在行がオブザベーションセット間のブレークの一部である場合)
- 値_**RBREAK**_(現在行がレポートの始めまたは最後のブレークの一部である場合)
- 値_**PAGE**_(現在行がページの始めまたは最後のブレークの一部である場合)

複合名の使用

レポートの統計量を使用する場合、通常、Sales.sum などの複合名別に計算ブロックで参照します。ただし、レポートの部分によっては、同じ名前でもその意味が異なります。次の出力のレポートについて考えます。出力を作成するステートメントが後に続きます。使用されるユーザー定義の出力形式が PROC FORMAT ステップ (1851 ページ)によって作成されます。

```
libname proclib
'SAS-library';

options nodate pageno=1 fmtsearch=(proclib);
proc report data=grocery;
  column sector manager sales;
  define sector / group format=$sctrfmt.;
  define sales  / analysis sum
                  format=dollar9.2;
  define manager / group format=$mgrfmt.;
  break after sector / summarize;
  rbreak after / summarize;
  compute after;
    sector='Total:';
  endcomp;
```

```
run;
```

例のコード 51.1　Sales.sum の 3 つの異なる意味

```
                The SAS System
1

          Sector    Manager     Sales
          Northeast Alomar    $786.00    1
                    Andrews   $1,045.00
          ---------           ---------
          Northeast           $1,831.00    2

          Northwest Brown     $598.00
                    Pelfrey   $746.00
                    Reveiz    $1,110.00
          ---------           ---------
          Northwest           $2,454.00

          Southeast Jones     $630.00
                    Smith     $350.00
          ---------           ---------
          Southeast           $980.00

          Southwest Adams     $695.00
                    Taylor    $353.00
          ---------           ---------
          Southwest           $1,048.00

          ==========          ==========
          Total:              $6,313.00    3
          ==========          ==========
```

ここで、Sales.sum には 3 つの異なる意味があります。

1　詳細行では、値は市のあるセクタの 1 人のマネージャーの店舗の売上です。たとえば、レポートの最初の詳細行は、Alomar が経営する店舗の売上が$786.00 であったことを表しています。

2　グループ要約行では、値は 1 つのセクタのすべての店舗の売上です。たとえば、最初のグループ要約行は、北東セクタの売上が$1,831.00 であったことを表しています。

3　レポート要約行では、値$6,313.00 はその市のすべての店舗の売上です。

注: 計算ブロックの別名を持つ統計量を参照する場合、複合名は使用しないでください。通常、別名を使用する必要があります。ただし、統計量と列変数が列を共有する場合は、列番号で参照する必要があります。“計算ブロックのレポート項目を参照するための 4 つの方法” (1756 ページ)を参照してください。

PROC REPORT によってサポートされている ODS 出力先

PROC REPORT ではすべての ODS 出力先をサポートしています。詳細については、“ODS 出力先について” (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

PROC REPORT STYLE=オプションは OUTPUT を除くすべての ODS 出力先をサポートしています。詳細については、“PROC HTTP で ODS スタイルを使用する” (1823 ページ)を参照してください。

PROC REPORT は、ODS DOCUMENT プロシジャをサポートします。DOCUMENT 出力先により、分析を再実行したり、データベースクエリを繰り返すことなくデータを異なる方法で異なる出力先に再構築、ナビゲート、再生することができます。DOCUMENT 出力先により出力ストリーム全体が"生"形式で利用可能になり、アクセスしてカスタマイズできるようになります。出力は、データコンポーネントおよびテーブル定義として元の内部表現で保持されます。詳細については、“DOCUMENT Procedure” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

PROC REPORT は、SAS 出力データセットの OUTPUT 出力先をサポートします。ODS ではデータの論理構造とネイティブフォームを認識しているため、プロシジャが内部的に使用した同じ結果のデータセットを表す SAS データセットを出力できます。詳細については、“ODS OUTPUT ステートメント” (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

入力データセットのスレッド処理

THREAD オプションにより、入力データセットの並列処理の有効化または無効化が可能になります。スレッド処理により、処理操作における一定の並列処理が達成されます。この並列処理の目的は、所定の操作を完了するための処理時間を削減し、それによって追加 CPU リソースのコストを抑えることです。

SAS システムオプション CPUCOUNT=の値はスレッド化された並べ替えのパフォーマンスに影響します。CPUCOUNT=は、スレッド化されたプロシジャで使用できるシステム CPU の数を示します。

詳細については、“THREADS システムオプション” (*SAS システムオプション: リファレンス*) および、“CPUCOUNT= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

計算された統計は、オブザベーションが処理される順序に応じてわずかに異なる場合があります。このような変動は、浮動小数点演算によって導入される数値エラーによるものであり、その結果は概算であり、正確ではないと見なす必要があります。オブザベーション処理の順序は、マルチスレッド処理または並列処理の非決定論的影響を受ける可能性があります。処理の順序は、ACCESS エンジンを介してクエリ結果を提供する DBMS などのデータソースによって 生成されるオブザベーションの一貫性のない、または非決定論的な並べ替えによっても 影響を受ける可能性があり

ます。詳細については、“Base SAS のスレッド処理” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

構文: REPORT プロシジャ

制限事項: PROC REPORT は、VARCHAR データ型をサポートしていません。

アクセシビリティの注: ACCESSIBLECHECK および ACCESSIBLETABLE システムオプションを使用できます。ACCESSIBLETABLE は、一部のテーブルのレイアウトを変更してアクセス可能にし、テーブルに視覚的なキャプションを追加します。ACCESSIBLECHECK は、テーブルがアクセス可能かどうかを確認し、SAS ログに情報を出力します。詳細については、“Creating Accessible Tables” (*Creating Accessible SAS Viya Platform Output Using ODS and ODS Graphics*)を参照してください。

ACCESSIBLETABLE システムオプションを CAPTION=オプションおよび CONTENTS=オプションとともに使用して、テーブルの表示可能なキャプションと、By ディレクティブを使用した目次の表示可能な変更を表示できます。

アクセシビリティの注: ACCESSIBLETABLE システムオプションを CAPTION=オプションおよび CONTENTS=オプションとともに使用して、テーブルの表示可能なキャプションと、By ディレクティブを使用した目次の表示可能な変更を表示できます。

ヒント: Output Delivery System をサポートします。詳細については、“Output Delivery System: 基本概念” (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

ステートメント ATTRIB、FORMAT、LABEL、WHERE を PROC SORT プロシジャと使用できます。

発生する In-Database 処理に対し、データは SAS In-Database 処理用に適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、“PROC REPORT の In-database 処理” (1834 ページ)を参照してください。

```
PROC REPORT<options>;
    ATTRIB variable-list(s) attribute-list(s);
    BREAK location break-variable< / options>
    BY< DESCENDING>variable-1<< DESCENDING>variable-2...> < NOTSORTED>;
    COLUMNcolumn-specification(s);
        COMPUTE location<target>
          < / STYLE=<style-override(s)> >;
            LINE specification(s);
            . . . select SAS language elements . . . . .
            ENDCOMP;
        COMPUTE report-item </ type-specification>;
            CALL DEFINE (column-id ', < ' attribute-name', value >
                | _ROW_, < 'attribute-name', value >);
            . . . select SAS language elements . . .
```

ENDCOMP;

DEFINE *report-item* / <*options*>;

FORMAT *variable-1*<...*variable-n*> <*format*> < DEFAULT=*default-format*>;

FREQ *variable*;

LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;

RBREAK *location*< / *options*>;

WEIGHT *variable*;

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク	例
“PROC REPORT ステートメント”	要約レポートまたは詳細レポートを作成します	Ex. 1, Ex. 2, Ex. 6, Ex. 4, Ex. 9, Ex. 10, Ex. 13
“BREAK ステートメント”	グループ変数または順序変数の値の変更時にデフォルト要約を作成します	Ex. 2, Ex. 5, Ex. 6, Ex. 7
“BY ステートメント”	各 BY グループの個別のレポートを作成します	
“CALL DEFINE ステートメント”	現在行の特定の列に属性値を設定します	Ex. 5, Ex. 13
“COLUMN ステートメント”	すべての列の調整と複数の列にわたるヘッダーの調整を記述します	Ex. 1, Ex. 3, Ex. 6, Ex. 4, Ex. 8, Ex. 9
“COMPUTE ステートメント”	PROC REPORT がレポートの作成時に実行する 1 つ以上のプログラミングステートメントを指定します	Ex. 2, Ex. 3, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 10, Ex. 13
“ENDCOMP ステートメント”	PROC REPORT がレポートの作成時に実行する 1 つ以上のプログラミングステートメントを指定します	Ex. 2
“DEFINE ステートメント”	レポート項目の使用方法と表示方法を記述します	Ex. 2, Ex. 3, Ex. 5, Ex. 6, Ex. 4, Ex. 7, Ex. 8, Ex. 10, Ex. 10, Ex. 11, Ex. 13
“FREQ ステートメント”	オブザベーションを入力データセットで複数回表示されるように扱います。	
“LINE ステートメント”	カスタマイズされた要約を記述するための PUT ステートメントの機能のサブセットを提供します	Ex. 2, Ex. 3, Ex. 5, Ex. 6, Ex. 7

ステートメント	タスク	例
“RBREAK ステートメント”	レポートの始めまたは最後、あるいは各 BY グループの始めと最後にデフォルト要約を作成します	Ex. 1, Ex. 8
“WEIGHT ステートメント”	統計量計算の分析変数の重みを指定します。	

PROC REPORT ステートメント

PRINT プロシジャ、MEANS プロシジャ、TABULATE プロシジャの機能と DATA ステップの機能を、さまざまなレポートの作成が可能な 1 つのレポート作成ツールに組み合わせます。

サポート: スレッド処理

例:
- “例 1: 変数の選択とレポートの要約行の作成” (1849 ページ)
- “例 2: レポートでの行の並べ替え” (1853 ページ)
- “例 6: 変数の値ごとに列を作成する” (1866 ページ)
- “例 4: 1 つの変数に複数の統計量を表示する” (1860 ページ)
- “例 9: PROC REPORT での欠損値の処理法” (1878 ページ)
- “例 10: 出力データセットの作成と計算変数の保存” (1882 ページ)
- “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1892 ページ)
- “例 14: セル幅の=スタイル属性を PROC REPORT とともに使用する” (1900 ページ)

構文

PROC REPORT<*options*>;

オプション引数の要約

CAPTION=*text* < **#BYLINE** > < **#BYVAL** > < **#BYVAR** >

DATA=*SAS-data-set*
: 入力データセットを指定します。

NOALIAS
: 計算ブロックに別名が必要になる以前に作成されたレポートを使用します。

OUT=*SAS-data-set*
: 出力データセットを指定します。

THREADS

NOTHREADS

SAS システムオプション THREADS | NOTHREADS を無効にします。

Control classification levels

COMPLETECOLS

NOCOMPLETECOLS
列変数値のすべての可能な組み合わせを作成します。

COMPLETEROWS

NOCOMPLETEROWS
グループ変数値のすべての可能な組み合わせを作成します。

Control ODS output

CONTENTS='*link-text*' < **#BYLINE** > < **#BYVAL** > < **#BYVAR** >
出力に対し目次エントリのテキストを指定します。

SPANROWS
単一のセルが値が同じすべての行の列を占めるように指定します。

STYLE<(*location(s)*)>=*style-override(s)*
レポートの異なる部分に使用する 1 つ以上のスタイルの上書きを指定します。

Control the layout of the report

BYPAGENO=*number*
BY グループ間のページ番号をリセットします。

CENTER

NOCENTER
レポートと要約テキストを中央揃えまたは左揃えにするかどうかを指定します。

MISSING
欠損値をグループ変数、順序変数、または列変数の有効値とみなします。

SHOWALL
列の表示を非表示にする DEFINE ステートメントのオプションを無効にします。

Control the statistical analysis

EXCLNPWGT
非正の重み値がついたオブザベーションを分析から除外します。

QMARKERS=*number*
P^2 分位点推定方法に使用するサンプルサイズを指定します。

QMETHOD=OS | P2
分位点推定方法を指定します。

QNTLDEF=1 | 2 | 3 | 4 | 5
分位点を計算する算術定義を指定します。

VARDEF=*divisor*
分散の計算に使用する除数を指定します。

Customize column headings

NAMED
name=を、name=が値の列ヘッダーであるレポートの各値の前に書き込みます。

NOHEADER
: 列ヘッダーを非表示にします。

SPLIT='*character*'
: 区切り文字を指定します。

Store and retrieve report definitions, PROC REPORT statements, and your report profile

LIST
: 現在のレポートを作成する PROC REPORT コードを SAS ログに書き込みます。

NOEXEC
: レポートの作成を抑制します。

OUTREPT=*libref.catalog.entry*
: サブミットする PROC REPORT ステップによって定義されるレポート定義を指定したカタログに保存します。

REPORT=*libref.catalog.entry*
: 使用するレポート定義を指定します。

オプション引数

BYPAGENO=*number*
: BY ステートメントが存在する場合、各 BY グループの開始時にページ番号を指定します。

  範囲 0 より大きい正の整数。

  制限事項 BY ステートメントが存在しない場合、このオプションは影響しません。

  操作 BYPAGENO=オプションも ODS ESCAPECHAR THISPAGE 関数に影響します。

CAPTION=*text* < **#BYLINE** > < **#BYVAL** > < **#BYVAR** >
: 各テーブルの前に追加するキャプションテキスト文字列を指定します。文字列が指定されていない場合、キャプションはデフォルトで CONTENTS=オプションで指定されたテキストになります。

  このオプションは、ACCESSIBLETABLE システムオプションが指定されている場合、テーブルのキャプションを視覚的かつアクセス可能にします。

  NOACCESSIBLETABLE システムオプションが指定されている場合、キャプションは表示されませんが、キャプションテキストは引き続きアクセス可能です。

  BY ステートメントが有効な場合、次のオプションを使用できます。

  #BYLINE
  : テキスト文字列に指定した#BYLINE を先頭と末尾にブランクを含まない BY 行全体で置き換えます。BY 行は、 *variable-name*=*value* の出力形式を使用します。

  #BYVAL*n*
  #BYVAL(*BY-variable-name*)
  : テキスト文字列に指定した#BYVAL を、指定した BY 変数の現在の値に置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAL2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAL(YEAR)では、BY 変数 YEAR を指定します。*variable-name* では、大文字と小文字は区別されません。

#BYVAR*n*
#BYVAR(*BY-variable-name*)
テキスト文字列に指定した#BYVAR を、BY-変数の名前またはこの変数に関連付けられているラベル(BY 行が通常表示するもの)で置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAR2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAR(SITES)では、BY 変数 SITES を指定します。*variable-name* では、大文字と小文字は区別されません。

制限事項 各キャプションを有効にするには、ACCESSIBLETABLE システムオプションを指定する必要があります。

ヒント PROC DOCUMENT OBBNOTE オプションを使用して、キャプションを表示または編集できます。

参照項目: "Creating Accessible Tables" (*Creating Accessible SAS Viya Platform Output Using ODS and ODS Graphics*)

CENTER | NOCENTER
レポートと要約テキスト(カスタマイズされたブレーク行)を中央揃えまたは左揃えにするかどうかを指定します。

PROC REPORT は、検出したこれらの中央揃え指定のうち最初のものを有効化します。

- PROC REPORT ステートメントの CENTER オプションまたは NOCENTER オプション、または **ROPTIONS** ウィンドウの CENTER トグル
- PROC REPORT ステートメントの REPORT=を使用してロードされるレポート定義に保存される CENTER オプションまたは NOCENTER オプション
- SAS システムオプション CENTER または NOCENTER

操作 CENTER が有効な場合、PROC REPORT はレポートの最左の変数の前のスペースを無視します。

COMPLETECOLS | NOCOMPLETECOLS
1 つ以上の組み合わせが入力データセット内で発生しない場合でも、列変数の値に可能なすべての組み合わせを作成します。結果として、列ヘッダーは単一 BY グループ内のレポートのすべての論理ページに対して同一です。

デフォルト　COMPLETECOLS

操作　DEFINE ステートメントの PRELOADFMT オプションにより、度数がゼロの場合でも PROC REPORT が列変数の組み合わせに対しすべてのユーザー定義のフォーマット範囲を使用するようになります。

COMPLETEROWS | NOCOMPLETEROWS
1 つ以上の組み合わせが入力データセット内で発生しない場合でも、グループ変数の値のすべての可能な組み合わせを表示します。結果として、行ヘッダーは単一 BY グループ内のレポートのすべての論理ページに対して同一です。

デフォルト　NOCOMPLETEROWS

操作　DEFINE ステートメントの PRELOADFMT オプションにより、度数がゼロの場合でも PROC REPORT がグループ変数の組み合わせに対しすべてのユーザー定義のフォーマット範囲を使用するようになります。

CONTENTS='*link-text*' < **#BYLINE** > < **#BYVAL** > < **#BYVAR** >
デフォルトで、または STYLE=オプションをサポートする ODS 出力先のオプション設定によって作成された目次のエントリのテキストを指定します。

ACCESSIBLETABLE システムオプションが指定されている場合、By ディレクティブを使用して、目次に視覚的な出力を作成できます。

BY ステートメントが有効な場合、次のオプションを使用できます。

#BYLINE
テキスト文字列に指定した#BYLINE を先頭と末尾にブランクを含まない BY 行全体で置き換えます。BY 行は、 *variable-name*=*value* の出力形式を使用します。

#BYVAL*n*
#BYVAL(*BY-variable-name*)
テキスト文字列に指定した#BYVAL を、指定した BY 変数の現在の値に置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAL2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAL(YEAR)では、BY 変数 YEAR を指定します。*variable-name* では、大文字と小文字は区別されません。

#BYVAR*n*
#BYVAR(*BY-variable-name*)
テキスト文字列に指定した#BYVAR を、BY-変数の名前またはこの変数に関連付けられているラベル(BY 行が通常表示するもの)で置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAR2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAR(SITES)では、BY 変数 SITES を指定します。*variable-name* では、大文字と小文字は区別されません。

制限事項 By ディレクティブを使用して目次に表示される出力テキストを作成するには、ACCESSIBLETABLE システムオプションを指定する必要があります。ACCESSSIBLETABLE オプションなしで CONTENTS='*link-text*'を使用できます。

ヒント OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

参照項目: REPORT プロシジャを使用したアクセス可能なテーブルの作成(*ODS および ODS Graphics を使用したアクセシブルな SAS Viya プラットフォーム出力の作成*)。

DATA=*SAS-data-set*
入力データセットを指定します。

参照項目: “入力データセット” (24 ページ)

EXCLNPWGT
非正(ゼロまたは負)の重みがついたオブザベーションを分析から除外します。デフォルトでは、PROC REPORT は非正の重みが付いたオブザベーションを重みがゼロのオブザベーションのように扱い、オブザベーションの合計数にカウントします。

別名 EXCLNPWGTS

要件 WEIGHT ステートメントを使用する必要があります。

参照項目: “WEIGHT ステートメント” (1818 ページ)

LIST
現在のレポートを作成する PROC REPORT コードを SAS ログに書き込みます。

この LIST は、次の点でサブミットするステートメントと異なっていることがあります。

- 指定していない可能性のある一部のデフォルト値が表示される。
- REPORT プロシジャ固有でない一部のステートメントが、PROC REPORT ステップでサブミットするか、以前にサブミットしたかどうかに関係なく省略される。これらのステートメントには、次が含まれます:

BY FOOTNOTE FREQ TITLE WEIGHT WHERE

- これらの PROC REPORT ステートメントオプションは省略されています:

LIST
OUT=
OUTREPT=

REPORT=

- これらの style(<*location*>)=オプションが含まれています:

CENTER　RIGHT
HEADER　USAGE
LEFT

- SAS システムオプションを省略します。
- 自動マクロ変数を解決します。

制限事項　DEFINE ステートメントで style(column)=を指定した場合、LIST オプションは列ごとのスタイルをサポートしません。

MISSING
欠損値をグループ変数、順序変数、または列変数の有効値とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ異なる値とみなされます。各欠損値のグループがレポートに表示されます。MISSING オプションを省略すると、PROC REPORT はグループ変数、順序変数、列変数に対する欠損値を含むオブザベーションをレポートに含めません。

参照項目:　"欠損値" (*SAS プログラマガイド: エッセンシャル*)。

例　"例 9: PROC REPORT での欠損値の処理法" (1878 ページ)

NAMED
name=を、*name* が値の列ヘッダーであるレポートの各値の前に書き込みます。

操作　NAMED オプションの使用時、PROC REPORT は自動的に NOHEADER オプションを使用します。

ヒント　NAMED を WRAP オプションと組み合わせて使用し、個別のページに広いレポートの列を置くのではなく、レポートの単一行に対しすべての列を連続行にラップするレポートを作成します。

NOALIAS
計算ブロックに別名が必要になる以前に作成されたレポートを使用できます。NOALIAS を使用すると、計算ブロックの別名を使用できません。

NOEXEC
レポートの作成を抑制します。NOEXEC を OUTREPT=と使用して、レポート定義をカテゴリエントリに保存します。NOEXEC を LIST、REPORT=と使用して、指定したレポート定義のリストを表示します。

別名　NOEXECUTE

NOHEADER
複数の列にわたるヘッダーを含む列ヘッダーを非表示にします。

OUT=*SAS-data-set*
出力データセットを指定します。このデータセットが存在しない場合、PROC REPORT によって作成されます。データセットには、レポート行ごとに 1 つのオブザベーション、一意の要約行ごとに 1 つのオブザベーションが含まれます。カスタマイズされた要約とデフォルト要約をレポートの同じ場所で使用すると、出

力データセットにはオブザベーションが 1 つだけ含まれます。これは、2 つの要約のデータ表示方法のみ異なるためです。カスタマイズに関する情報(下線、色、テキストなど)はデータではなく、出力データセットには保存されません。

出力データセットには、レポートの列ごとに変数が 1 つ含まれます。PROC REPORT は、出力データセットの対応する変数の名前としてレポート項目の名前を使用しようとします。ただし、データセット変数が列変数に満たない、または列変数を超える場合、またはデータセット変数が COLUMN ステートメントに別名なしで複数回表示される場合、この代替を実行することはできません。この場合、変数の名前は列番号(_C1_、_C2_など)に基づきます。

入力データセット変数から派生する出力データセット変数は、対応する出力形式を入力データセットに保持します。列を定義する唯一の項目が列変数でない限り、PROC REPORT はレポートの対応する列ヘッダーからこれらの変数に対しラベルを派生します。この場合、変数にラベルはありません。複数の項目が 1 つの列に積み重ねられると、対応する出力データセット変数のラベルが列の分析変数から取得されます。

出力データセットには、_BREAK_という文字変数も含まれます。出力データセットのオブザベーションがレポートの詳細行から派生する場合、_BREAK_の値は、欠損かブランクになります。オブザベーションが要約行から派生する場合、_BREAK_の値は要約行または**_RBREAK_**と関連付けられているブレーク変数の名前です。オブザベーションが COMPUTE BEFORE _PAGE_ステートメントまたは COMPUTE AFTER _PAGE_ステートメントから派生する場合、_BREAK_の値は**_PAGE_**となります。ただし、COMPUTE BEFORE _PAGE_と COMPUTE AFTER _PAGE_の場合、**_PAGE_**値は出力データセットにのみ書き込まれます。この値はプロシジャの実行中は自動変数_BREAK_の値として利用できません。

ヒント 出力データセットは、ODS OUTPUT ステートメントを使用して作成できます。ODS OUTPUT によって作成されるデータセットは、OUT=オプションによって作成されるものと同じです。“ODS OUTPUT ステートメント” (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

例 “例 10: 出力データセットの作成と計算変数の保存” (1882 ページ)

“例 10: 出力データセットの作成と計算変数の保存” (1882 ページ)

OUTREPT=*libref.catalog.entry*
サブミットする PROC REPORT ステップによって定義されるレポート定義を指定したカタログエントリに保存します。PROC REPORT はエントリに種類 REPT を割り当てます。

保存されたレポート定義は、サブミットするステートメントと次の点で異なります。

- REPORT プロシジャ固有でない一部のステートメントが、PROC REPORT ステップでサブミットするか、ステップのサブミット時にすでに有効かどうかに関係なく省略される。これらのステートメントには、次が含まれます:

BY	TITLE
FOOTNOTE	WEIGHT
FREQ	WHERE

- 次の PROC REPORT ステートメントオプションが含まれる。その他のオプションは省略される。

CENTER	PANELS
COLWIDTH	PCTLDEF
COMPLETECOLS	QMARKERS
COMPLETEROWS	QMETHOD
MISSING	SHOWALL
NAMED	SPLIT
NOHEADER	VARDEF
PAGESIZE	

- SAS システムオプションを省略します。
- 自動マクロ変数を解決します。

QMARKERS=*number*
P² 推定方法に使用するマーカーのデフォルト数を指定します。マーカー数によって固定メモリ空間のサイズを制御します。

デフォルト　デフォルト値は、要求する分位点によって異なります。中央値(P50)の場合、*number* は 7 です。分位点(P25 と P75)の場合、*number* は 25 です。分位点 P1、P5、P10、P90、P95 または P99 の場合、*number* は 105 です。複数の分位点を要求する場合、PROC REPORT は、*number* の最も大きいデフォルト値を使用します。

範囲　3 より大きい奇数の整数

ヒント　デフォルト設定を超えるマーカー数を増やし、推定の正確性を向上させます。マーカー数を減らしてコンピューティングリソースを節約することができます。

QMETHOD=OS | P2
PROC REPORT が分位点の計算時に入力データの処理に使用する方法を指定します。オブザベーション数が QMARKERS=オプションの値以下で、QNTLDEF=オプションの値が 5 の場合、いずれの方法でも結果は同じになります。

OS
順序統計量を使用します。PROC UNIVARIATE は、この方法を使用します。

注: この方法は、非常にメモリを消費する可能性があります。

P2
P² 方法を使用して、分位点を概算します。

デフォルト　OS

制限事項　QMETHOD=P2 の場合、PROC REPORT は MODE と重み付き分位点を計算しません。

ヒント　QMETHOD=P2 の場合、一部の分位点(P1、P5、P95、P99)の信頼性のある推定は、裾の重い分布または歪度が高い分布を含むデータセットなど、一部のデータセットには可能でないことがあります。

QNTLDEF=1 | 2 | 3 | 4 | 5
QMETHOD=オプションの値が OS の場合にプロシジャが分位点の計算に使用する算術定義を指定します。QMETHOD=P2 の場合、QNTLDEF=5 を使用する必要があります。

別名 PCTLDEF=

デフォルト 5

参照項目: “分位数と関連統計量” (2415 ページ)

REPORT=*libref.catalog.entry*
使用するレポート定義を指定します。PROC REPORT は、すべてのレポート定義を種類 REPT のエントリとして SAS カタログに保存します。

操作 REPORT=を使用する場合、COLUMN ステートメントを使用できません。

参照項目: “OUTREPT=*libref.catalog.entry*” (1773 ページ)

SHOWALL
列の表示を非表示にする DEFINE ステートメントのオプションを無効にします。

参照項目: “DEFINE ステートメント” (1799 ページ)の NOPRINT と NOZERO

SPANROWS
GROUP 列または ORDER 列の値が複数の行で同じ場合、値が同じすべての行の列を占める単一のセルに値が表示されるように指定します。ボックスが列のその部分に対して作成され、そのボックスに行は表示されません。

SPANROWS オプションにより、値がページにわたって分割するときに GROUP 変数値と ORDER 変数値を繰り返すこともできます。PDF、PS、TAGSETS.RTF 出力先だけが、機能のこの部分をサポートします。

注 SPANROWS オプションは、データセットまたは OUTPUT 出力先に影響しません。

ページの下部に LINE ステートメントが表示される場合、値が RTF および TAGSETS.RTF ページにわたって分割するときに GROUP 変数値と ORDER 変数値は繰り返されません。

ヒント 要約行は、その他の場合は単一セルにまたがる一連の行の真ん中に表示される場合、列に独自のセルを挿入します。このアクションにより、要約行の後に来る GROUP 変数または ORDER 変数の値が変更されない場合でもまたがったセルが 2 つのセルに分割されます。

SPLIT='*character*'
区切り文字を指定します。PROC REPORT は、その文字に達すると列テキストを分割し、ヘッダーを次の行に続けます。区切り文字が発生するたびにラベルの最大 256 文字に考慮されますが、区切り文字自体は列ヘッダーまたはテキスト値の一部ではありません。

デフォルト slash (/)

制限事項 このオプションは、の ODS 出力先の列ヘッダーでのみ機能します。

STYLE<(*location(s)*)>=*style-override(s)*
レポートの異なる部分に使用する 1 つ以上のスタイルの上書きを指定します。

location(s)
STYLE=オプションが影響するレポートの部分を識別します。次のテーブルに、*location* の値によって影響を受けるレポート内の各部分を示します。

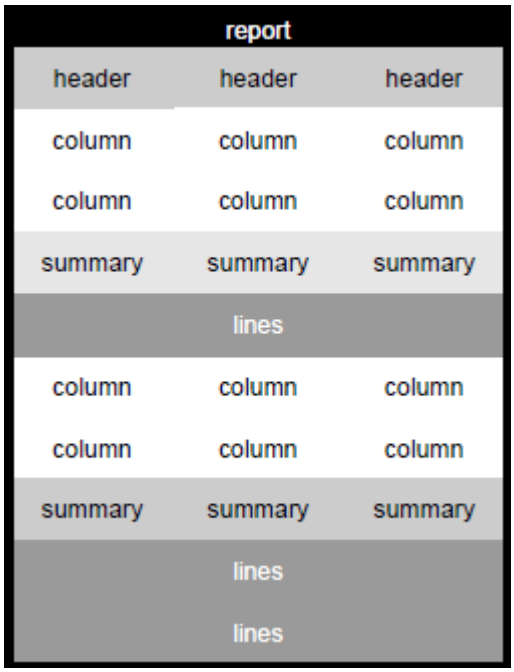
location の有効な値とデフォルト値は、STYLE=オプションが記述されるステートメントによって異なります。次のテーブルでは、*location* の有効な値とデフォルト値をステートメントごとに示します。同じ STYLE=オプションで複数の *location* を指定するには、各値をスペースで区切ります。

表 51.2 PROC REPORT の各ステートメントの場所とデフォルトのスタイル要素

ステートメント	有効な location 値	デフォルトの location 値	レポート内で影響を受ける部分	デフォルトのスタイル要素
PROC REPORT	COLUMN HEADER | HDR SUMMARY REPORT LINES CALLDEF	REPORT	レポート全体	Table
BREAK	SUMMARY LINES	SUMMARY	要約行	DataEmphasis
CALL DEFINE	CALLDEF	CALLDEF	CALL DEFINE ステートメントによって識別されるセル	Data
COMPUTE	LINES	LINES	LINE ステートメントによって生成される行	LineContent
DEFINE	COLUMN HEADER| HDR	COLUMN HEADER	列のセル 列ヘッダー	COLUMN: Data HEADER: Header
RBREAK	SUMMARY LINES	SUMMARY	要約行 LINE ステートメントによって生成される行	DataEmphasis

次のテーブルに表示されるすべての名前は、PROC ステートメントの *location(s)*の代わりに使用できます。DEFINE ステートメントには、column と header を受け入れます。BREAK および BRBREAK ステートメントは summary と lines を受け入れます。

図 51.8 PROC REPORT の領域および対応するステートメント



PROC REPORT の領域以外の PROC REPORT ステートメントでの指定は、PROC REPORT ステートメントの同じ指定を無効にします。ただし、PROC REPORT ステートメントで指定し、別の PROC REPORT ステートメントで無効にしないスタイル属性は継承されます。たとえば、PROC REPORT ステートメントですべての列ヘッダーに対し青の背景と白い前景の指定し、PROC REPORT DEFINE ステートメントで変数の列ヘッダーに対し灰色の背景を指定すると、特定の列ヘッダーの背景は灰色になり、前景は(PROC REPORT ステートメントでの指定に従って)白になります。

style-override

レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。スタイルのオーバーライドは、次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

  注: これらのスタイルは、PROC ステートメントで指定されているスタイルよりも優先されます。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*

<*style-attribute-name-2=style-attribute-value-2 ...*>]

注: 角かっこ([と])の代わりに中かっこ({と})を使用できます。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。SAS はいくつかの スタイルテンプレートを提供しています。ユーザーは TEMPLATE プロシジャを使用して、独自のスタイルテンプレートを作成できます。*SAS Output Delivery System: プロシジャガイド*を参照してください。

参照項目: PROC REPORT でのスタイルの使用の詳細については、“PROC HTTP で ODS スタイルを使用する” (1823 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)を参照してください。

style-attribute-name
変更する属性を指定します。PROC PRINT、PROC TABULATE および PROC REPORT の STYLE=オプションで設定できる一般的に使用されているスタイル属性のリストについては、表 51.98 (1828 ページ)を参照してください。

参照項目: PROC REPORT でのスタイルの使用の詳細については、“PROC HTTP で ODS スタイルを使用する” (1823 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)を参照してください。

style-attribute-value
属性の値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目: PROC REPORT でのスタイルの使用の詳細については、“PROC HTTP で ODS スタイルを使用する” (1823 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用したスタイル属性値の割り当て” (1833 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)を参照してください。

制限事項 OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目: 詳細については、“テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)を参照してください。

例 “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1892 ページ)

THREADS | NOTHREADS
入力データセットの並列処理を有効化または無効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします。詳細については、“Base SAS のスレッド処理” (*SAS プログラマガイド: エッセンシャル*) と“THREADS システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

デフォルト SAS システムオプション THREADS | NOTHREADS の値。

制限事項 サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

操作 PROC REPORT は、BY ステートメントが指定されるとき、または SAS システムオプション CPUCOUNT が 2 未満のとき以外は SAS システムオプション THREADS の値を使用します。PROC REPORT ステートメントで THREADS オプションを指定して、PROC REPORT がこうした状況で並列処理を使用するように強制することができます。

注 並列処理でもあるスレッド処理が有効な場合、オブザベーションは予想外の順序で返されることがあります。ただし、BY ステートメントが指定されている場合、オブザベーションは正しく並べ替えられます。

VARDEF=*divisor*
分散と標準偏差の計算に使用する除数を指定します。

DF
自由度

除数の式は次のとおりです。

$n - 1$

N
オブザベーション数

除数の式は次のとおりです。

n

WDF
重みの合計 - 1

除数の式は次のとおりです。

$(\Sigma_i w_i) - 1$

WEIGHT
重みの合計

除数の式は次のとおりです。

$\Sigma_i w_i$

別名 WGT

プロシジャは分散を*CSS*/*divisor*として計算します。*CSS*は修正平方和で、$\Sigma (x_i - \bar{x})^2$と等しくなります。分析変数に重み付けをする場合、*CSS*は$\Sigma w_i(x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト DF

要件 平均値の標準誤差とスチューデントの *t* 検定を比較するには、VARDEF=のデフォルト値を使用します。

ヒント WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。*i* 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$、およびw_iは *i* 番目のオブザベーションの重みです。これにより、オブザベーションの分散の推定が単位重みとともに生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな *n* に対し)$\sigma^2/\bar{w}$の推定になります。$\bar{w}$は平均重みです。これにより、オブザベーションの分散の漸近的推定が平均重みとともに生成されます。

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

BREAK ステートメント

ブレーク(グループ変数または順序変数の値の変更)時にデフォルト要約を作成します。要約の情報は一連のオブザベーションに適用されます。オブザベーションは、レポートのブレーク変数、そのブレーク変数の左側のその他すべてのグループ変数または順序変数に対する値の一意の組み合わせを共有します。

例:
“例 2: レポートでの行の並べ替え” (1853 ページ)
“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)
“例 6: 変数の値ごとに列を作成する” (1866 ページ)
“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)

構文

BREAK *location break-variable*< / *options*>;

オプション引数の要約

CONTENTS='*link-text*'
: 目次で使用されるリンクテキストを指定します。

PAGE
: 最終ブレーク行の後に新しいページを開始します。

STYLE<*location(s)*>=<*style-override*>
: デフォルト要約行、カスタマイズされた要約行、またはその両方に使用するスタイルの上書きを指定します。

SUMMARIZE
: ブレーク行の各グループの要約行を書き込みます。

SUPPRESS
: 要約行のブレーク変数の値の印刷、ブレーク変数を含む列のブレーク行の下線または上線の印刷を行いません。

必須引数

location
ブレーク行の配置を制御します。次の値のうちいずれかになります。

AFTER
: ブレーク変数の値が同じ各行セットの最終行の直後にブレーク行を置きます。

BEFORE
: ブレーク変数の値が同じ各行セットの開始行の直前にブレーク行を置きます。

break-variable
グループ変数または順序変数です。REPORT プロシジャはこの変数の値が変わるたびにブレーク行を書き込みます。

オプション引数

CONTENTS='*link-text*'
デフォルトで、または STYLE=オプションをサポートする ODS 出力先のオプション設定によって作成された目次のエントリのテキストを指定します。PAGE=オプションと *link-text* の CONTENTS=オプションが指定されると、PROC REPORT は *link-text* の値を目次で作成されたテーブルのリンクとして使用します。

デフォルト
: BREAK AFTER ステートメントで CONTENTS=オプションが指定されず、PAGE オプションが指定されている場合、目次のデフォルトのリンクテキストは“Table N”となります。N は整数です。

制限事項
: CONTENTS=が指定され、PAGE オプションが指定されていない場合、PROC REPORT は SAS ログファイルに警告メッセージを生成します。

操作 DEFINE ステートメントにページオプションがあり、BREAK BEFORE ステートメントで PAGE オプションが指定され、指定されている CONTENTS=オプションの値が空の引用符以外である場合、PROC REPORT はディレクトリを目次に追加し、リンクをそのディレクトリのテーブルに追加します。この相互作用の詳細については、DEFINE ステートメント (1799 ページ)の CONTENTS=オプションを参照してください。

BREAK BEFORE ステートメントが指定され CONTENTS=' 'オプションと PAGE=オプションが指定されている場合、PROC REPORT は目次でディレクトリを作成しません。かわりに、PROC REPORT は DEFINE ステートメントの CONTENTS=値を使用して、目次へのリンクを作成します。DEFINE ステートメントに CONTENTS=オプションがない場合、PROC REPORT は DEFINE ステートメントで記述されているデフォルトのテキストを使用してリンクを作成します。デフォルトのテキスト情報の説明については、DEFINE ステートメント (1799 ページ) の CONTENTS=オプションを参照してください。

RTF 出力の場合、ODS RTF ステートメントで CONTENTS=YES オプションをオンにしない限り、CONTENTS=オプションは RTF Body ファイルに影響しません。その場合、**目次**ページは RTF 出力ファイルの前に挿入されます。次に、PROC REPORT の CONTENTS=オプションテキストがこの別の**目次**ページに表示されます。

注 BREAK BEFORE ステートメントが存在し、PAGE オプションが指定され、CONTENTS=オプションが指定されていない場合、デフォルトのリンクテキストは場所変数と場所変数の値です。場所変数は、BREAK 変数と関連付けられています。値は BREAK 変数値です。次のコードに示すように、値は rep で、場所は rep の前です。break before rep / summarize page;

ヒント 値が空の引用符である CONTENTS=オプションが指定されている場合、テーブルリンクは目次に作成されません。このコードの例は CONTENTS=''です。

複数の BREAK BEFORE ステートメントがある場合、リンクテキストはすべての CONTENTS=値またはすべてのデフォルト値の組み合わせです。

PAGE
最終ブレーク行の後に新しいページを開始します。STYLE=オプションをサポートする ODS 出力先では、PAGE オプションは新しいテーブルを開始します。OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

制限事項 OUTPUT 出力先では、このオプションは影響しません。

操作 BREAK ステートメントで PAGE を使用し、レポートの最後にブレークを作成する場合、レポート全体の要約が別のページに表示されます。

例 “例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)

STYLE<*location(s)*>=<*style-override*>
BREAK ステートメントによって作成されるデフォルトの要約行に使用するスタイルの上書きを指定します。

スタイルのオーバーライドは、次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*

<*style-attribute-name-2=style-attribute-value-2* ...>]

制限事項 OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目: “テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)

SUMMARIZE

ブレーク行の各グループの要約行を書き込みます。一連のオブザベーションに対する要約行には、次の値が含まれます。

- ブレーク変数(SUPPRESS オプションで非表示可能)
- ブレーク変数の左側のその他のグループ変数または順序変数
- 統計量
- 分析変数
- 計算変数

次のテーブルに、BREAK ステートメントによって作成される要約行の各種レポート項目の値の PROC REPORT による計算方法を示します。

レポート項目	値
ブレーク変数	変数の現在値(SUPPRESS を使用している場合は欠損値)
ブレーク変数の左側のグループ変数または順序変数	変数の現在値
ブレーク変数の右側のグループ変数または順序変数、またはレポートの任意の場所の表示変数	欠損*。
統計量	セットのすべてのオブザベーションにわたる統計量の値。
分析変数	項目の定義の使い方のオプションとして指定される統計量の値。PROC REPORT は、セットのすべてのオブザベーションにわたる統計量の値

レポート項目	値
	を計算します。デフォルトの使用法は SUM です。
計算変数	対応する計算ブロックのコードに基づく計算結果。“COMPUTE ステートメント” (1795 ページ)を参照してください。

*カスタマイズされた要約行の欠損値を含む変数を参照する場合、PROC REPORT はその変数をブランク(文字変数の場合)またはピリオド(数値変数の場合)として表示します。

注: PROC REPORT は、順序変数または表示変数を含むレポートのグループを作成できません。

アクセシビリティの注 SUPPRESS オプションと SUMMARIZE オプションの両方が指定されている場合、要約行ヘッダーは表示されず、テーブルにアクセスできません。

例 “例 2: レポートでの行の並べ替え” (1853 ページ)

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)

“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)

SUPPRESS
次の印刷を行いません。

- 要約行のブレーク変数の値
- ブレーク変数を含む列のブレーク行の下線および下線

操作 SUPPRESS を使用する場合、ブレーク変数の値は、ブレークと関連付けられている計算ブロックで値を割り当てない限り、カスタマイズされたブレーク行で使用できません。“COMPUTE ステートメント” (1795 ページ)を参照してください。

アクセシビリティの注 SUPPRESS オプションと SUMMARIZE オプションの両方が指定されている場合、要約行ヘッダーは表示されず、テーブルにアクセスできません。

例 “例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)

詳細

ブレーク行の順序

デフォルト要約にブレーク行が複数含まれている場合、次の順序でブレーク行が表示されます。

1 要約行(SUMMARIZE)

2 スキップ行(SKIP)

3 改ページ(PAGE)

注: ブレークに対しカスタマイズされた要約を定義する場合、カスタマイズされたブレーク行は下線または二重下線の後に表示されます。カスタマイズされたブレーク行の詳細については、“COMPUTE ステートメント” (1795 ページ)および “LINE ステートメント” (1814 ページ)を参照してください。

BY ステートメント

出力を BY グループ順に並べ替えます。

制限事項: BY ステートメントではグループ変数を指定しないでください。BY 変数をグループ変数と同じにすることはできません。

操作: BY 処理を使用するレポートで RBREAK ステートメントを使用する場合、PROC REPORT は BY グループごとにデフォルト要約を作成します。この場合、レポート全体に関する情報を要約できません。

ヒント: BY ステートメントを使用しても FIRST.変数と LAST.変数は計算ブロックで利用可能になりません。

構文

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

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

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

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

詳細

BY グループ処理

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

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

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

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

- BY ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。

- PROC SORT の詳細については、55 章, “SORT プロシジャ,” (2027 ページ)を参照してください。
- CAS での BY グループ処理の詳細については、“グループ化と順序付け” (*SAS Cloud Analytic Services: DATA ステッププログラミング*)を参照してください。
- インデックスの作成の詳細については、“INDEX CREATE ステートメント” (435 ページ)を参照してください。

BY 変数値のフォーマティング

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

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

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

2 つのデータセットで長さが異なる BY 変数

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

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

- プロシジャで、BY ステートメントが常にセカンダリデータセットに適用される場合があります。この場合、BY ステートメントの変数が 1 つ以上、セカンダリデータセットに出現している必要があります。
- プロシジャで、BY ステートメントがセカンダリデータセットに一度も適用されない場合もあります。この場合、セカンダリデータセットには、BY ステートメントの変数は不要です。
- プロシジャで、セカンダリデータセットに BY 変数があるかどうかチェックされる場合があります。セカンダリデータセットに BY 変数が 1 つもない場合、セカンダリデータセットに対する BY 処理は発生しません。セカンダリデータセットに BY 変数が 1 つ以上あり、それがプライマリデータセットの BY 変数と一致する場合、セカンダリデータセットに対して BY 処理が行われます。セカンダリデータセットに全部ではなく一部の BY 変数がある場合、プロシジャでエラーメッ

セージが出力され、終了する場合があります。または、その特定プロシジャのドキュメントで説明しているその他のアクションが実行される場合もあります。

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

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

- A
- A、B
- A、B、C
- A、B、C、D

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

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブランクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデータセットの‘ABCD’とセカンダリデータセットの‘ABCD ’は一致します。セカンダリデータセットに‘ABCDEF’が含まれている場合は、一致しません。

BY ステートメントをサポートする Base SAS プロシジャ

COMPARE	SORT (必須)
CORR	STANDARD
FREQ	SUMMARY
MEANS	TABULATE
PRINT	TRANSPOSE

RANK	UNIVARIATE
REPORT (非ウィンドウ環境のみ)	ODSTEXT
ODSLIST	ODSTABLE

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

関連項目

"BY ステートメント" (*SAS DATA ステップステートメント: リファレンス*)

CALL DEFINE ステートメント

現在行の特定の列に対して属性値を設定します。属性 FORMAT、URL、URLBP、および URLP のみが有効です。STYLE=および URL 属性は、ODS を使用して出力を作成する場合にのみ有効です。

制限事項: レポート項目に割り当てられている計算ブロックでのみ有効。

ヒント: OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

例:

```
compute sales;
    if sales.sum>100 and _break_=' ' then
       call define(_col_, "style",
            "style=[backgroundcolor=yellow
                 fontfamily=helvetica
                 fontweight=bold]");
endcomp;

compute weight;
    if weight > 100.0 then
       call define(_row_, "style/merge", "style={font_weight=bold}");
endcomp;
```

例:

"例 5: 複数のオブザベーションをレポートの 1 行にまとめる" (1862 ページ)

"例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する" (1892 ページ)

"例 15: PROC REPORT CALL DEFINE ステートメントでの STYLE/MERGE の使用" (1903 ページ)

"例 16: PROC REPORT CALL DEFINE ステートメントでの STYLE/REPLACE の使用" (1905 ページ)

構文

CALL DEFINE (*column-id* | _ROW_, ' *attribute-name*', *value*);

必須引数

column-id
列名または列番号(レポートの左端からの列の位置)を指定します。列 ID には、次のうちいずれかが可能です。

- 列名である文字リテラル(引用符で囲む)
- 列名を解決する文字式
- 列番号である数値リテラル
- 列番号を解決する数値式
- 形式'_C*n*_'の名前。*n* は列番号です。
- 自動変数_COL_.計算ブロックが割り当てられているレポート項目を含む列を識別します

ROW
現在の行全体を示す自動変数です。

attribute-name
定義する属性です。属性名については、表 51.95 (1791 ページ)を参照してください。

注: 属性 BLINK、HIGHLIGHT および RVSVIDEO は、すべてのデバイスで 機能しません。

value
属性の値を設定します。各属性の値については、表 51.95 (1791 ページ)を参照してください。

表 51.3 属性の説明

属性	説明	値	影響
FORMAT=	列の出力形式を指定します	SAS 出力形式またはユーザー定義の出力形式	非ウィンドウ環境
STYLE=	列または行のスタイルの上書きを指定します	ODS スタイル属性	すべての ODS 出力先
STYLE/MERGE	同じ行または列で既存のスタイルを使用して指定されたスタイルをマージします	ODS スタイル属性	すべての ODS 出力先
STYLE/REPLACE	行または列の既存のスタイルを置き換えます	ODS スタイル属性	すべての ODS 出力先

属性	説明	値	影響
URL	列の各セルのコンテンツを指定した Uniform Resource Locator (URL)へのリンクにします	引用符付きの URL (単一引用符または二重引用符が使用可能)	ODS HTML、RTF、PDF、PowerPoint、EPUB 出力先
URLBP	列の各セルのコンテンツをリンクにします。そのリンクのリンク先は、次の連結である Uniform Resource Locator (URL)です 1 ODS HTML ステートメントの BASE=オプションで指定された文字列 2 ODS HTML ステートメントの PATH=オプションで指定された文字列 3 URLBP 属性の値	引用符付きの URL (単一引用符または二重引用符が使用可能)	ODS HTML 出力先
URLP	列の各セルのコンテンツをリンクにします。そのリンクのリンク先は、次の連結である Uniform Resource Locator (URL)です 1 ODS HTML ステートメントの PATH=オプションで指定された文字列 2 URLP 属性の値	引用符付きの URL (単一引用符または二重引用符が使用可能)	ODS HTML 出力先

1 BASE=オプションと PATH=オプションの詳細については、*SAS Output Delivery System: ユーザーガイド*の ODS HTML ステートメントのドキュメントを参照してください。

詳細

CALL DEFINE ステートメントによるスタイル属性の使用

STYLE 属性は、CALL DEFINE ステートメントによって影響を受けるセルで使用するスタイルの上書きを指定します。

STYLE=値は、PROC REPORT のその他のステートメントの STYLE=オプションのように機能します。ただし、ステートメントでオプションとして機能するかわりに、STYLE 属性の値になります。たとえば、次の CALL DEFINE ステートメントは、特定の列の背景色を黄色に、フォントサイズを 7 に設定します。

```
call define(_col_, "style",
        "style=[backgroundcolor=yellow fontsize=7]");
```

スタイル優先順位の詳細については、“データセルにスタイル属性を適用する際の優先順位” (1832 ページ)を参照してください。

制限: OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

相互作用: PROC REPORT ステートメントで CALLDEF 場所に対しスタイルの上書きを設定し、CALL DEFINE ステートメントでその設定したスタイルの上書きを使用する必要がある場合、次に示すように空の文字列を STYLE 属性の値として使用します。

```
call define (_col_, "STYLE", "" );
```

ヒント: 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

説明: “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1892 ページ)

STYLE/REPLACE および STYLE/MERGE スタイル属性の使用

STYLE/MERGE 属性および STYLE/REPLACE 属性は、CALL DEFINE ステートメントで指定したスタイルのみに対して機能します。STYLE(COLUMN)=で指定したスタイルをマージしたり、置き換えることはできません。STYLE/REPLACE および STYLE/MERGE は、1 つの CALL DEFINE ステートメントが含まれる複数の COMPUTE ブロックがあり、それらの CALL DEFINE ステートメントが同じセルを参照している場合に使用することをお勧めします。

STYLE=属性と STYLE/REPLACE 属性は ODS に使用されるスタイル要素を指定します。このセルまたは行に対しスタイルがすでに存在している場合、これらの STYLE 属性により CALL DEFINE が STYLE=オプションで指定されるスタイルを置き換えます。サンプルプログラムについては、“例 16: PROC REPORT CALL DEFINE ステートメントでの STYLE/REPLACE の使用” (1905 ページ)を参照してください。

STYLE/MERGE 属性により、CALL DEFINE は STYLE=値によって指定されるスタイルと、同じセルまたは行にある既存するスタイル属性をマージします。マージ対象となる既存する STYLE=値がない場合、STYLE/MERGE は STYLE 属性または STYLE/REPLACE 属性と同様に機能します。サンプルプログラムについては、“例 15: PROC REPORT CALL DEFINE ステートメントでの STYLE/MERGE の使用” (1903 ページ)を参照してください。

COLUMN ステートメント

すべての列の調整と複数の列にわたるヘッダーの調整を記述します。

別名: COL
COLS
COLUMNS

制限事項: PROC REPORT ステートメントで REPORT=を使用する場合、COLUMN ステートメントは使用できません。

例:
“例 1: 変数の選択とレポートの要約行の作成” (1849 ページ)
“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)
“例 6: 変数の値ごとに列を作成する” (1866 ページ)
“例 4: 1 つの変数に複数の統計量を表示する” (1860 ページ)
“例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)
“例 9: PROC REPORT での欠損値の処理法” (1878 ページ)

構文

COLUMN *column-specification(s)*;

必須引数

column-specification(s)
次のうちいずれかになります。

- *report-item(s)*
- *report-item-1*, *report-item-2* <..., *report-item-n*>
- ('*header-1* ' < ...'*header-n* '> *report-item(s)*)
- *report-item=name*

この場合、*report-item* は、データセット変数、計算変数または統計量の名前です。利用可能な統計量のリストについては、“PROC REPORT で使用できる統計量” (1821 ページ)を参照してください。

report-item(s)
レポートの列をそれぞれ形成する項目を識別します。

例 “例 1: 変数の選択とレポートの要約行の作成” (1849 ページ)

“例 9: PROC REPORT での欠損値の処理法” (1878 ページ)

report-item-1, report-item-2 <. . . , report-item-n>
列のコンテンツを一括して決定するレポート項目を識別します。これらの項目は、レポートで積み重ねられます。各項目がヘッダーを生成し、ヘッダーが相互に積み重ねられるためです。最左の項目のヘッダーが先頭です。項目のうち 1 つが分析変数、計算変数、グループ変数、または統計量である場合、その値はレポートのその部分のセルに入力されます。その他の場合、PROC REPORT はセルに度数カウントを入力します。

統計量と分析変数を積み重ねる場合、列ステートメントで指定する統計量が分析変数の定義の統計量よりも優先します。たとえば、次の PROC REPORT ステップでは、各セクタに対する売上の最小値を含むレポートを作成します。

```
proc report data=grocery;
column sector sales,min;
define sector/group;
define sales/analysis sum;
run;
```

列変数の下に表示変数を積み重ねる場合、その表示変数のすべての値がレポートに表示されます。

操作 積み重ねられた一連のレポート項目には、分析変数または統計量のいずれかを 1 つだけ含めることができます。複数の分析変数または統計量を含めると、PROC REPORT はエラーを返します。レポートのセルに挿入する値を決定できないためです。

ヒント かっこを使用して、相互に積み重ねるのではなく、ヘッダーを同じレベルで表示する必要があるレポート項目をグループ化することができます。

例 “例 6: 変数の値ごとに列を作成する” (1866 ページ)

“例 4: 1 つの変数に複数の統計量を表示する” (1860 ページ)

“例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)

('*header-1* ' <... '*header-n* '> *report-item(s)*)
複数の列にわたる 1 つ以上のヘッダーを作成します。

header
レポートの 1 つ以上の列にわたる文字の文字列です。PROC REPORT は、各ヘッダーを個別の行に印刷します。ヘッダーで区切り文字を使用して、1 つのヘッダーを複数の行に分割することができます。SPLIT= (1775 ページ)の説明を参照してください。

注: PROC REPORT は、出力が他の出力先に送信されると、拡張文字を単純に削除します。詳細については“ODS 出力先について” (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

report-item(s)
またがる列を指定します。

例 “例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)

report-item=name
レポート項目の別名を指定します。COLUMN ステートメントで同じレポート項目を複数回使用できます。ただし、指定名に使用できるのは、1 つの DEFINE ステートメントだけです。(DEFINE ステートメントは、出力形式、カスタマイズされた列ヘッダーなどの特性を指定します。項目に対して DEFINE ステートメントを省略すると、REPORT プロシジャはデフォルト値を使用します)。COLUMN ステートメントで別名を割り当てても、それだけでレポートは変わりません。ただし、変数または統計量が発生するたびに個別の DEFINE ステートメントを使用できるようになります。

注 常に別名を使用できるわけではありません。計算ブロックの別名を持つレポート項目を参照する場合、その別名を使用する必要があります。ただし、レポート項目と列変数が列を共有する場合は、列番号によって列を参照する必要があります。“計算ブロックのレポート項目を参照するための 4 つの方法” (1756 ページ)を参照してください。

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)

COMPUTE ステートメント

レポートの作成時に PROC REPORT が実行する 1 つ以上のプログラミングステートメントを含む計算ブロックを開始します。

制限事項: レポートを複数の ODS 出力先または後から再表示される ODS ドキュメントに送信している場合は、COMPUTE ブロックで非決定性関数(LAG、DIF、RANUNI、DATETIME など)を使用しないでください。レポートでこのような関数により作成

されたデータを使用する必要がある場合は、DATA ステップで関数を呼び出し、その結果をデータセットに保存してから PROC REPORT を実行します。

操作: ENDCOMP ステートメントは計算ブロックのステートメントグループの終わりをマーク付けする必要があります。

注: 計算ブロックはレポート項目または場所と関連付けることができます(レポートの上下、ページの上下、オブザベーションセットの前後)。COMPUTE ステートメントの 1 つのフォームが計算ブロックをレポート項目と関連付けます。別のフォームが計算ブロックを場所と関連付けます。計算ブロックで使用できる SAS 言語要素のリストについては、“計算ブロックのコンテンツ” (1756 ページ)を参照してください。

ヒント: 計算ブロックの使用法の詳細については、The REPORT Procedure: A Primer for the Compute Block を参照してください

例: “例 2: レポートでの行の並べ替え” (1853 ページ)
“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)
“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)
“例 6: 変数の値ごとに列を作成する” (1866 ページ)
“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)
“例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)
“例 10: 出力データセットの作成と計算変数の保存” (1882 ページ)
“例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1892 ページ)

構文

COMPUTE *location*<*target*>
</STYLE=<*style-override(s)* > >;
LINE *specification(s)*;
.. . select SAS language elements . . . . .
ENDCOMP;
COMPUTE *report-item* </ *type-specification*>;
CALL DEFINE (*column-id*, '*attribute-name*', *value*);
.. . select SAS language elements . . .
ENDCOMP;

必須引数

COMPUTE ステートメントで場所またはレポート項目のいずれかを指定する必要があります。

location
計算ブロックが *target* に関して実行する場所を決定します。

AFTER
次の場所のうちいずれかのブレークで計算ブロックを実行します。

- *target* として指定する変数に対し同じ値を持つ行セットの最後の行の直後、またはその変数にデフォルト要約がある場合は予備要約行の作成直後。結果: REPORT プロシジャ (1838 ページ)を参照してください。
- target を省略する場合はレポートの最後。

BEFORE
次の場所のうちいずれかのブレークで計算ブロックを実行します。

- *target* として指定する変数に対し同じ値を持つ行セットの最初の行の直前、またはその変数にデフォルト要約がある場合は予備要約行の作成直後。結果: REPORT プロシジャ (1838 ページ)を参照してください。
- target を省略する場合、最初の詳細行の直前。

注 レポートに印刷ページに収まる以上の列が含まれている場合、PROC REPORT は残りの列を含めるための追加ページを生成します。この場合、_PAGE_を *target* として指定するとき、COMPUTE ブロックはこれらの各追加ページに対し再実行されません。COMPUTE ブロックが再実行されるのは、すべての行が印刷された後だけです。

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)

“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)

report-item
計算ブロックと関連付けるデータセット変数、計算変数または統計量を指定します。COLUMN ステートメントにレポート項目を含める必要があります。項目が計算変数である場合、それに対し DEFINE ステートメントを含める必要があります。

注 計算変数の位置は重要です。PROC REPORT は値をレポート行の列に、左から右に割り当てます。その結果、レポートの右側に表示される変数に基づいて計算変数の計算を行うことはできません。

例 “例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)

“例 6: 変数の値ごとに列を作成する” (1866 ページ)

オプション引数

target
いつ計算ブロックが実行するかを制御します。COMPUTE ステートメントに対し場所(BEFORE または AFTER)を指定する場合、*target* を指定することもできます。値は、次のうちいずれかになります。

break-variable
グループ変数または順序変数です。

ブレーク変数を指定する場合、PROC REPORT はブレーク変数の値が変わるたびに計算ブロックのステートメントを実行します。

PAGE </ justification>
justification は、テキストと値の配置を制御します。次のいずれかを指定できます。

CENTER
計算ブロックが書き込む各行を中央揃えにします。

LEFT
計算ブロックが書き込む各行を左寄せにします。

RIGHT
計算ブロックが書き込む各行を右寄せにします。

デフォルト CENTER

例 “例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)

type-specification
種類を指定します。(オプション)。*report-item* の長さも指定します。計算ブロックと関連付けられているレポート項目が計算変数である場合、種類の指定を使用してそれが文字変数になるように指定しない限り、PROC REPORT は数値変数であるとみなします。種類の指定には、次の形式があります。

CHARACTER< LENGTH=*length*>

ここで、

CHARACTER
計算変数が文字変数になるように指定します。長さを指定しない場合、変数の長さは 8 です。

別名 CHAR

例 “例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)

LENGTH=*length*
計算文字変数の長さ(文字数)を指定します。

デフォルト 8

範囲 1 から 200

操作 長さを指定する場合、CHARACTER を使用して計算変数が文字変数であることを示す必要があります。

例 “例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)

スラッシュ区切り文字に続くオプション

STYLE<(*location(s)*)>=<*style-override(s)*>
この計算ブロックで LINE ステートメントによって作成されるテキストに使用するスタイルを指定します。

スタイルのオーバーライドは、次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。

- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2* ...>]

別名 S=

制限事項 OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目: “テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)

例 “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1892 ページ)

DEFINE ステートメント

レポート項目の使用方法と表示方法を説明します。

制限事項: 重みは、report-item にも適用しないと、*report-item* の別名に適用できません。WEIGHT=オプションは、DEFINE ステートメントの *report-item* に表示される必要があります。

アクセシビリティの注: DEFINE ステートメントに ORDER または GROUP オプションが含まれている場合、アクセス可能なテーブルを生成するには、SPANROWS オプションも PROCREPORT ステートメントに含める必要があります。

ヒント: DEFINE ステートメントを使用しない場合、PROC REPORT はデフォルトの特性を使用します。

例:
“例 2: レポートでの行の並べ替え” (1853 ページ)
“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)
“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)
“例 6: 変数の値ごとに列を作成する” (1866 ページ)
“例 4: 1 つの変数に複数の統計量を表示する” (1860 ページ)
“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)
“例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)
“例 10: 出力データセットの作成と計算変数の保存” (1882 ページ)
“例 11: 出力形式を使用してグループを作成する” (1886 ページ)
“例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1892 ページ)
“例 12: マルチラベル出力形式の使用” (1889 ページ)
“例 14: セル幅の=スタイル属性を PROC REPORT とともに使用する” (1900 ページ)

構文

DEFINE *report-item* / <*options*>;

オプション引数の要約

Control the placement of values and column headings

CENTER
: 列幅内のレポート項目のフォーマットされた値を中央揃えにし、値にわたる列ヘッダーを中央揃えにします。

column-header
: レポート項目に対し列ヘッダーを定義します。

LEFT
: 列幅内のレポート項目のフォーマットされた値を左寄せにし、値にわたる列ヘッダーを左寄せにします。

RIGHT
: 列幅内のレポート項目のフォーマットされた値を右寄せにし、値にわたる列ヘッダーを右寄せにします。

Customize the appearance of a report item

EXCLUSIVE
: ユーザー定義の出力形式のすでに読み込まれた範囲に見つからない項目のすべての組み合わせを除外します。

FORMAT=*format*
: SAS またはユーザー定義の出力形式を項目に割り当てます。

MISSING
: 欠損値を項目の有効値とみなします。

MLF
: PROC REPORT が出力形式ラベルを使用して、マルチラベル出力形式を含むサブグループの組み合わせを作成できるようにします。

ORDER=DATA | FORMATTED | FREQ | INTERNAL
: グループ変数、順序変数、列変数の値を指定順序に従って並べ替えます。

PRELOADFMT
: 項目に対してすべての 出力形式がすでに読み込まれているように指定します。

statistic
: 統計量と分析変数を関連付けます。

STYLE<(*location(s)*)>=<*style-overrides(s)*>
: レポート項目に対してスタイル要素(Output Delivery System の)を指定します。

WEIGHT=*weight-variable*
: 分析変数の値に重みをつける数値変数を指定します。

Specify how to use a report item

ACROSS
: データセット変数である必要がある項目を列変数として定義します。

ANALYSIS
: データセット変数である必要がある項目を分析変数として定義します。

COMPUTED
: 項目を計算変数として定義します。

DISPLAY
: データセット変数である必要がある項目を表示変数として定義します。

GROUP
: データセット変数である必要がある項目をグループ変数として定義します。

ORDER
: データセット変数である必要がある項目を順序変数として定義します。

Specify options for a report item

CONTENTS='*link-text*'
: 目次でリンクを作成します。

DESCENDING
: PROC REPORT がグループ変数、順序変数、列変数の行または値を表示する順序を逆順にします。

ID
: 定義している項目が ID 変数になるように指定します。

NOPRINT
: レポート項目の表示を非表示にします。

NOZERO
: レポート項目の表示を、その値がすべてゼロまたは欠損値の場合に非表示にします。

PAGE
: レポート項目の値を含む最初の列を印刷する直前に改ページを挿入します。

必須引数

report-item
: 定義対象となるデータセット変数、計算変数または統計量の名前または別名(COLUMN ステートメントで作成)を指定して定義します。*report-item* に使用できる名前の種類を次に示します。

- SAS ID (VALIDVARNAME オプションで決定されます)
- 名前リテラル
- 番号範囲リスト
- 名前範囲リスト
- 特殊名リスト
- 名前接頭辞リスト
- 統計量

注　変数範囲リストの名前は、統計量名、または計算変数名ではなく、入力データセットの変数を表します。DEFINE ステートメントごとに 1 つの名前のみ使用します。ただし、1 つの名前で 1 つの範囲リストを表すことがで

きます。変数範囲リストを使用した構文例は次のとおりです: DEFINE Var1–Var3/ width=10 center “#Visit#Date”;

統計量の定義で使い方のオプションを指定しないでください。PROC REPORT は、統計量の名前でその使用方法を認識します。

参照項目: “SAS 名” (*SAS プログラマガイド: エッセンシャル*)、“変数リスト” (*SAS プログラマガイド: エッセンシャル*)、と“VALIDVARNAME= システムオプション” (*SAS システムオプション: リファレンス*)。

オプション引数

ACROSS
データセット変数である必要がある *report-item* を列変数として定義します。“列変数” (1751 ページ)を参照してください。

例 “例 6: 変数の値ごとに列を作成する” (1866 ページ)

ANALYSIS
数値データセット変数である必要がある *report-item* を順序変数として定義します。“分析変数” (1751 ページ)を参照してください。

デフォルトで、PROC REPORT は分析変数に対する Sum 統計量を計算します。DEFINE ステートメントの *statistic* オプションで代替統計量を指定します。

注: DEFINE ステートメントでの統計量の指定は、ANALYSIS オプションを意味するため、ANALYSIS を指定する必要はありません。ただし、ANALYSIS を指定すると、コードがわかりやすくなります。

注: 特殊欠損値は、ANALYSIS 変数として定義されている場合、欠損値として表示されます。

例 “例 2: レポートでの行の並べ替え” (1853 ページ)

“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)

CENTER
列幅内のレポート項目のフォーマットされた値を中央揃えにし、値にわたる列ヘッダーを中央揃えにします。このオプションは、ページのレポートを中央揃えにする、PROC REPORT ステートメントの CENTER オプションに影響しません。

制限事項 ODS 出力では、このオプションはデータのみに影響します。

column-header
レポート項目に対し列ヘッダーを定義します。各ヘッダーを単一引用符または二重引用符で囲みます。複数の列ヘッダーを指定する場合、PROC REPORT は列ヘッダーごとに別の行を使用します。区切り文字も複数の行にわたる列ヘッダーを分割します。

次のテーブルに、デフォルトの変数と統計量を示します。

項目	ヘッダー
変数(ラベルなし)	変数名
変数(ラベルあり)	変数ラベル
統計量	統計量名

ヒント ラベルが存在するときに名前を使用する場合、PROC REPORT を起動する前に options nolabel;という SAS ステートメントをサブミットします。

参照項目: SPLIT= (1775 ページ)

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)

“例 6: 変数の値ごとに列を作成する” (1866 ページ)

COMPUTED

指定した項目を計算変数として定義します。計算変数は、レポートに対して定義する変数です。入力データセットにはなく、PROC REPORT によって入力データセットに追加されません。

次のように計算変数を追加します。

- 計算された変数を COLUMN ステートメントに含める。
- 変数の使用法を DEFINE ステートメントで COMPUTED として定義する。
- 変数に関連付けられた計算ブロックで変数の値を計算する。

例 “例 6: 変数の値ごとに列を作成する” (1866 ページ)

“例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)

CONTENTS='*link-text*'

デフォルトで、または STYLE=オプションをサポートする ODS 出力先のオプション設定によって作成された目次のエントリのテキストを指定します。DEFINE ステートメントで PAGE=オプションと CONTENTS=オプションが指定され、*link-text* 値が割り当てられている場合、PROC REPORT はディレクトリを目次に追加し、*link-text* の値を目次で作成されたテーブルのリンクとして使用します。

デフォルト DEFINE ステートメントで PAGE オプションが指定され、CONTENTS=オプションが指定されていない場合、ディレクトリはディレクトリテキストとともに COLA-COLB として作成されます。COLA は最左列の名前または別名で、COLB は最右列の名前または別名です。テーブルに 1 列のみ含まれている場合、ディレクトリテキストはその列の名前または別名です。

制限事項 CONTENTS=が指定され、PAGE オプションが指定されていない場合、PROC REPORT は SAS ログファイルに警告メッセージを生成します。

操作 DEFINE ステートメントにページオプションがあり、BREAK BEFORE ステートメントで PAGE オプションが指定され、指定されている CONTENTS=オプションの値が空の引用符以外である場合、PROC REPORT はディレクトリを目次に追加し、リンクをそのディレクトリのテーブルに追加します。

DEFINE ステートメントで PAGE=オプションが指定され、BREAK BEFORE ステートメントで PAGE=オプションが指定されていない場合、PROC REPORT は目次でディレクトリを作成しません。かわりに、PROC REPORT は DEFINE ステートメントの CONTENTS=値を使用して、目次へのリンクを作成します。DEFINE ステートメントに CONTENTS=オプションがない場合、PROC REPORT はデフォルトのテキスト COLA—COLB を使用してリンクを作成します。上記のデフォルトの説明を参照してください。

BREAK BEFORE ステートメントが指定され CONTENTS=' 'オプションと PAGE=オプションが指定されている場合、PROC REPORT は目次でディレクトリを作成しません。かわりに、PROC REPORT は DEFINE ステートメントの CONTENTS=値を使用して、目次へのリンクを作成します。DEFINE ステートメントに CONTENTS=オプションがない場合、PROC REPORT はデフォルトのテキスト COLA—COLB を使用してリンクを作成します。上記のデフォルトの説明を参照してください。

ヒント DEFINE ステートメントで値が空の引用符である CONTENTS=オプションが指定されている場合、目次へのディレクトリは追加されません。このコードの例は、次のとおりです: CONTENTS=''t

複数の BREAK BEFORE ステートメントがある場合、リンクテキストはすべての CONTENTS=値またはすべてのデフォルト値の組み合わせです。

OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

DESCENDING
PROC REPORT がグループ変数、順序変数、列変数の行または値を表示する順序を逆順にします。

ヒント デフォルトで、PROC REPORT はグループ変数、順序変数、列変数をフォーマットされた値によって並び替えます。DEFINE ステートメントの ORDER=オプションを使用して、代替並べ替え順序を指定します。

DISPLAY
データセット変数である必要がある *report-item* を表示変数として定義します。“表示変数” (1749 ページ)を参照してください。

EXCLUSIVE
ユーザー定義の出力形式のすでに読み込まれた範囲に見つからないグループ変数と列変数のすべての組み合わせをレポートと出力データセットから除外します。

要件 変数の出力形式を読み込むために、DEFINE ステートメントで PRELOADFMT オプションを指定する必要があります。

FORMAT=*format*
SAS またはユーザー定義の出力形式を項目に割り当てます。この出力形式は、PROC REPORT で表示されるときに *report-item* に適用されます。出力形式は、データセット内の変数と関連付けられている出力形式を変えません。データセット変数の場合、PROC REPORT は検出されるこれらの出力形式のうち最初のものを有効化します。

- DEFINE ステートメントの FORMAT=と関連付けられている出力形式
- PROC REPORT の起動時に FORMAT ステートメントで割り当てられる出力形式
- データセット内の変数と関連付けられる出力形式

これらフォーマットが一切存在しない場合は、PROC REPORT により、数値変数には BEST*w*.が使用され、文字変数には$*w*.が使用されます。*w* の値は、デフォルトの列幅です。入力データセット内の文字変数の場合、デフォルトの列幅は変数の長さです。

別名 F=

例 “例 2: レポートでの行の並べ替え” (1853 ページ)

“例 4: 1 つの変数に複数の統計量を表示する” (1860 ページ)

GROUP
データセット変数である必要がある *report-item* をグループ変数として定義します。“グループ変数” (1750 ページ)を参照してください。

アクセシビリティの注 DEFINE ステートメントに ORDER または GROUP オプションが含まれている場合、アクセス可能なテーブルを生成するには、SPANROWS オプションも PROCREPORT ステートメントに含める必要があります。

例 “例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)

“例 4: 1 つの変数に複数の統計量を表示する” (1860 ページ)

“例 11: 出力形式を使用してグループを作成する” (1886 ページ)

ID
定義している項目が ID 変数になるように指定します。ID 変数とその左側のすべての列がレポートの各ページの左側に表示されます。ID によって、1 ページに収まる以上の列がレポートに含まれているときにレポートの各行を識別できるようになります。

LEFT
列幅内のレポート項目のフォーマットされた値を左寄せにし、値にわたる列ヘッダーを左寄せにします。出力形式の幅が列幅と同じ場合、LEFT オプションは値の配置に影響しません。

制限事項 ODS 出力では、このオプションはデータのみに影響します。

MISSING
欠損値をレポート項目の有効値とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING オプションを省略すると、PROC REPORT は、グループ変数、順序変数または列変数が欠損値のすべてのオブザベーションをレポートおよび出力データセットから除外します。

MLF
PROC REPORT が指定の範囲や重複する範囲に出力形式ラベルを使用して、マルチラベル出力形式設定を使用するサブグループの組み合わせを作成できるようにします。これらのマルチラベル出力形式は、グループ変数と列変数とのみ使用されます。

MLF は、すべての ODS 出力先およびデータセットでサポートされています。

注: PROC REPORT では、マルチラベル出力形式を持つ数値変数の文字変数への割り当てをサポートしています。

要件 VALUE ステートメントの PROC FORMAT と MULTILABEL オプションを使用して、マルチラベル出力形式を作成します。

ヒント 変数がマルチラベル出力形式と関連付けられていない限り、MLF オプションは影響しません。列と関連付けられている MULTILABEL 出力形式がない場合、既存する出力形式または入力形式を 1 つ以上の変数と関連付けるために追加の FORMAT ステートメントまたは DEFINE ステートメントの FORMAT=オプションが必要になります。MULTILABEL 出力形式がすでに列と関連付けられている場合(出力形式と変数を関連付けるための通常の方法を使用して)、追加の FORMAT ステートメントまたは FORMAT=オプションは不要です。

MLF オプションが省略されると、PROC REPORT は第 1 出力形式ラベルを使用して、サブグループの組み合わせを決定します。第 1 出力形式ラベルは、最初の外部出力形式値に対応しています。

参照項目: FORMAT プロシジャの VALUE ステートメントの MULTILABEL オプション (909 ページ)。

例 “例 12: マルチラベル出力形式の使用” (1889 ページ)

NOPRINT
レポート項目の表示を非表示にします。このオプションは、次の場合に使用します。

- レポートに項目を表示せずに、その値を使用してレポートで使用するその他の値を計算する場合。
- レポートで行の順序を作成する場合。
- 項目を列として使用せずに、要約のその値にアクセスする場合。“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)を参照してください。

操作 NOPRINT を使用して定義する列がレポートに表示されない場合でも、列を番号別に参照する場合はそれらの列をカウントする必要があります。“計算ブロックのレポート項目を参照するための 4 つの方法” (1756 ページ)を参照してください。

PROC REPORT ステートメントの SHOWALL は、NOPRINT のすべての発生を無効にします。

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)

“例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)

NOZERO
レポート項目の表示を、その値がすべてゼロまたは欠損値の場合に非表示にします。

操作 NOZERO を使用して定義する列がレポートに表示されない場合でも、列を番号別に参照する場合はそれらの列をカウントする必要があります。“計算ブロックのレポート項目を参照するための 4 つの方法” (1756 ページ)を参照してください。

PROC REPORT ステートメントの SHOWALL は、NOZERO のすべての発生を無効にします。

ORDER
データセット変数である必要がある *report-item* を順序変数として定義します。“順序変数” (1750 ページ)を参照してください。

アクセシビリティの注 DEFINE ステートメントに ORDER または GROUP オプションが含まれている場合、アクセス可能なテーブルを生成するには、SPANROWS オプションも PROCREPORT ステートメントに含める必要があります。

例 “例 2: レポートでの行の並べ替え” (1853 ページ)

ORDER=DATA | FORMATTED | FREQ | INTERNAL
グループ変数、順序変数、列変数の値を指定順序に従って並べ替えます。

DATA
入力データセットの順序に従って値を並べ替えます。

注 DBMS テーブルの入力データに ORDER=DATA オプションを指定した場合、PROCREPORT からデータベーステーブルに書き込まれる行の順序は保持されない可能性があります。

FORMATTED
フォーマットされた(外部)値によって値を並べ替えます。出力形式がクラス変数に割り当てられていない場合、デフォルトの出力形式 BEST12 が使用されます。

FREQ
昇順の度数カウントによって値を並べ替えます。

INTERNAL
フォーマットされていない値によって値を並べ替えます。これにより PROC SORT が作成するのと同じ順序が作成されます。この順序は、動作環境によって異なります。この並べ替え順序は、日付を年代順に表示する場合に特に便利です。

デフォルト FORMATTED

操作 項目の定義の DESCENDING は、項目の並べ替え順序を逆順にします。デフォルトでは、順序は昇順です。

注 PROC REPORT の ORDER=オプションのデフォルト値は、その他の SAS プロシジャのデフォルト値とは異なります。その他の SAS プロシジャでは、デフォルトは ORDER=INTERNAL です。PROC REPORT のオプションのデフォルトは、その他のプロシジャと整合させるために、今後のリリースで変わることがあります。このため、フォーマットされた値によってレポート項目を並べ替えることが重要な本稼働ジョブでは、現在デフォルトである ORDER=FORMATTED を指定します。これにより、PROC REPORT はデフォルトが変わっても期待どおりのレポートの作成を続行します。

例 “例 2: レポートでの行の並べ替え” (1853 ページ)

PAGE
レポート項目の値を含む最初の列を印刷する直前に改ページを挿入します。

制限事項 このオプションは、OUTPUT 出力先では影響しません。

ヒント レポート内のすべての行が印刷されるまで改ページは発生しません。そのため、PROC REPORT はすべての列に対するすべての行を PAGE 列の左側に印刷してから、レポートの上部に戻って、PAGE 列とその列を右側に印刷します。

PRELOADFMT
変数に対して出力形式がすでに読み込まれているように指定します。

制限事項 PRELOADFMT は、グループ変数と列変数にのみ適用されます。

要件 PRELOADFMT は、EXCLUSIVE または ORDER=DATA を指定し、出力形式を変数に割り当てない限り、影響しません。

操作 レポートを入力データセット内にあるフォーマットされた変数値の組み合わせに制限するには、DEFINE ステートメントの EXCLUSIVE を使用します。

ユーザー定義の出力形式のすべての範囲と値を出力に含めるには、PROC REPORT ステートメントの COMPLETEROWS オプションを使用します。

注 文字出力形式をプリロードする場合、ラベルを生成する代表的な未加工値が少なくとも 1 つ存在しなければなりません。範囲に未加工値がない場合、到達不能と判断され、エラーが発生します。たとえば、Other=と Low-High の両方の範囲を持つマルチラベル文字出力形式をプリロードすることはできません。Other は、すべてのラベルが定義された後に残る値または範囲を参照する特別なキーワードです。文字出力形式の場合、low 範囲に欠損値が含まれるため、その他の域を占める未加工値が残りません。

列変数の定義時に NOCOMPLETECOLS を指定しない場合、レポートには、フォーマットされている値ごとに 1 列が含まれます。グループ変数の定義時に COMPLETEROWS を指定する場合、レポートには、フォーマットされた値ごとに 1 行が含まれます。レポートに列変数のフォーマットされた値ごとに 1 列、グループ変数のフォーマットされた値ごとに 1 行含まれている場合、行と列の組み合わせが意味をなさない場合があります。

例 “例 12: マルチラベル出力形式の使用” (1889 ページ)

RIGHT
列幅内の指定項目のフォーマットされた値を右寄せにし、値にわたる列ヘッダーを右寄せにします。出力形式の幅が列幅と同じ場合、RIGHT は値の配置に影響を及ぼしません。

制限事項 ODS 出力では、このオプションはデータのみに影響します。

statistic
統計量と分析変数を関連付けます。統計量とその定義のすべての分析変数を関連付ける必要があります。PROC REPORT は、レポートの各セルによって表されるオブザベーションに対する分析変数の値を計算するために指定する統計量を使用します。その他の種類の変数の定義で *statistic* を使用することはできません。

利用可能な統計量のリストについては、“PROC REPORT で使用できる統計量” (1821 ページ)を参照してください。

デフォルト SUM

注 PROC REPORT は、分析変数の名前を列のデフォルトヘッダーとして使用します。DEFINE ステートメントの *column-header* オプションを使用して、列ヘッダーをカスタマイズできます。

例 “例 2: レポートでの行の並べ替え” (1853 ページ)

“例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)

“例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)

STYLE<(*location(s)*)>=<*style-overrides(s)*>
このレポート項目に対する列ヘッダーおよびセル内のテキストに使用するスタイル要素を指定します。

スタイルのオーバーライドは、次の 2 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。

style-override の形式は次のとおりです。

style-element-name | [*style-attribute-name-1=style-attribute-value-1*

<*style-attribute-name-2=style-attribute-value-2* ...>]

制限事項 OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目: “テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)

例 “例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する” (1892 ページ)

WEIGHT=*weight-variable*

DEFINE ステートメントで指定される分析変数の値に重みをつける数値変数を指定します。変数値は整数である必要はありません。次のテーブルに、PROC REPORT による WEIGHT 変数のさまざまな値の処理方法を説明します。

重み値	PROC REPORT 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 より小さい	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負とゼロの重みを含むオブザベーションを分析から除外するには、PROC REPORT ステートメントの EXCLNPWGT オプションを使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

別名 WGT=

制限事項 重み付き分位点を比較するには、PROC REPORT ステートメントの QMETHOD=OS を使用します。

重みは、*report-item* にも適用しないと、*report-item* の別名に適用できません。WEIGHT=オプションは、DEFINE ステートメントの *report-item* に表示される必要があります。

ヒント WEIGHT=オプションを使用する場合、PROC REPORT ステートメントの VARDEF=オプションのどの値が適切かを考慮します。

変数に対し異なる重みを指定するため、別の変数定義の WEIGHT=オプションを使用します。

ENDCOMP ステートメント

PROC REPORT がレポートの作成時に実行する 1 つ以上のプログラミングステートメントの最後をマーク付けします。

制限事項: COMPUTE ステートメントは ENDCOMP ステートメントの前に置く必要があります。

参照項目: COMPUTE ステートメント

例: “例 2: レポートでの行の並べ替え” (1853 ページ)

構文

ENDCOMP;

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。“出力形式の取り消し” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、*"PUT ステートメント: リスト" (SAS DATA ステップステートメント: リファレンス)*を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

FREQ ステートメント

オブザベーションを、入力データセットに複数回表示されたように処理します。

ヒント: FREQ ステートメントと WEIGHT ステートメントの影響は、自由度の計算時以外は似ています。

構文

FREQ *variable*;

必須引数

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

詳細

度数情報は保存されません

レポート定義を保存する場合、PROC REPORT は FREQ ステートメントを保存しません。

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=' '>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項 ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
```

```
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: “文字定数と文字変数” (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

LINE ステートメント

カスタマイズされた要約を書き込むための PUT ステートメントの機能のサブセットを提供します。

制限事項: このステートメントは、レポートの場所と関連付けられている計算ブロックでのみ有効です。

LINE ステートメントを条件ステートメント(IF-THEN、IF-THEN/ELSE および SELECT)で使用できません。LINE ステートメントは、PROC REPORT が計算ブロックのその他すべてのステートメントを 実行するまで実行されないためです。

アクセシビリティの注: LINE ステートメントを使用すると、アクセスできないテーブルが生成されます。

例:
- “例 2: レポートでの行の並べ替え” (1853 ページ)
- “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)
- “例 5: 複数のオブザベーションをレポートの 1 行にまとめる” (1862 ページ)
- “例 6: 変数の値ごとに列を作成する” (1866 ページ)
- “例 7: ページごとにカスタマイズされた要約を書き込む” (1869 ページ)

構文

LINE *specification(s)*;

必須引数

specification(s)
次の形式のうちいずれかになります。異なる形式の指定を 1 つの LINE ステートメントに組み合わせることができます。

item item-format
表示する項目と表示に使用する出力形式を指定します。

item
レポートのデータセット変数、計算変数または統計量の名前です。レポート項目の参照の詳細については、“計算ブロックのレポート項目を参照するための 4 つの方法” (1756 ページ)を参照してください。

item-format
SAS 出力形式またはユーザー定義の出力形式です。項目ごとに出力形式を指定する必要があります。

例 “例 2: レポートでの行の並べ替え” (1853 ページ)

'*character-string* '
表示するテキストの文字列を指定します。文字列がブランクで、*specification(s)*にそれ以外何もない場合、PROC REPORT はブランク行を印刷します。

注: *character-string* 内で指定される 16 進値(**'DF'x** など)は引用符内で指定されるため、解決されません。16 進値を解決するには、**num** が 16 進値である**%sysfunc(byte(num))**関数を使用します。マクロ関数が解決するように、*character-string* を二重引用符(" ")で囲みます。

例 “例 2: レポートでの行の並べ替え” (1853 ページ)

***number-of-repetitions**'*character-string* '**
文字列とそれを繰り返す回数を指定します。

例 “例 3: 別名を使用し、同一変数に複数の統計量を求める” (1856 ページ)

RBREAK ステートメント

レポートの始めまたは最後、あるいは各 BY グループの始めと最後にデフォルト要約を作成します。

例: “例 1: 変数の選択とレポートの要約行の作成” (1849 ページ)
“例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)

構文

RBREAK *location*< / *options*>;

オプション引数の要約

CONTENTS='*link-text*'
: 目次で使用されるリンクテキストを指定します。

PAGE
: レポートの始めのブレークの最終ブレーク行の後に新しいページを開始します。

STYLE<(*location(s)*)>=<*style-overrides(s)*>
: デフォルト要約行、カスタマイズされた要約行またはその両方に対し、スタイル要素(Output Delivery System の)を指定します。

SUMMARIZE
: 要約行をブレーク行の 1 つとして含めます。

必須引数

location
ブレーク行の配置を制御します。次のうちいずれかになります。

AFTER
: レポートの最後にブレーク行を置きます。

BEFORE
: レポートの始めにブレーク行を置きます。

オプション引数

CONTENTS='*link-text*'
デフォルトで、または STYLE=オプションをサポートする ODS 出力先のオプション設定によって作成された目次のエントリのテキストを指定します。PAGE オプションと SUMMARIZE オプションが指定されている RBREAK BEFORE ステートメントだけが目次内にテーブルを作成します。CONTENTS=オプションと、PAGE オプション、SUMMARIZE オプションが指定されると、PROC REPORT は *link-text* の値を使用して、そのテキストを作成したテーブルの目次に置きます。CONTENTS=の値が空の引用符の場合、リンクは目次で作成されません。

デフォルト
: RBREAK BEFORE ステートメントが存在し、PAGE オプションと SUMMARIZE オプションが指定され、CONTENTS=オプションが指定されていない場合、目次のデフォルトのリンクテキストには**要約**が表示されます。

制限事項
: CONTENTS=が指定され、PAGE オプションが指定されていない場合、PROC REPORT は SAS ログファイルに警告メッセージを生成します。SUMMARIZE オプションと PAGE オプションが指定されている RBREAK BEFORE/だけが、実際に目次にエントリを作成できます。

ヒント
: HTML 出力には現在、追加のアンカータグを含めることができます。

OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

PAGE
レポートの始めのブレークの最終ブレーク行の後に新しいページを開始します。RBREAK BEFORE で、PAGE オプションは新しいテーブルを開始します。

制限事項 このオプションは、OUTPUT 出力先では影響しません。

STYLE<(*location(s)*)>=<*style-overrides(s)*>
RBREAK ステートメントを使用して作成されるデフォルトの要約行に使用するスタイル要素を指定します。

制限事項 OUTPUT を除くすべての ODS 出力先は、STYLE=オプションをサポートします。

ヒント 文字またはアンダースコア以外の文字を含む FONT 名は、引用符で囲む必要があります。

参照項目: “テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)

SUMMARIZE
要約行をブレーク行の 1 つとして含めます。レポートの始めと最後の要約行には、次の値が含まれます。

- 統計量
- 分析変数
- 計算変数

次のテーブルに、RBREAK ステートメントによって作成される要約行の各種レポート項目の値の PROC REPORT による計算方法を示します。

レポート項目	結果値
統計量	セットのすべてのオブザベーションにわたる統計量の値。
分析変数	DEFINE ステートメントの使い方のオプションとして指定される統計量の値。PROC REPORT は、セットのすべてのオブザベーションにわたる統計量の値を計算します。デフォルトの使用法は SUM です。
計算変数	対応する計算ブロックのコードに基づく計算結果。“COMPUTE ステートメント” (1795 ページ)を参照してください。

例 “例 1: 変数の選択とレポートの要約行の作成” (1849 ページ)

“例 8: レポートに計算されたパーセント列を表示する” (1874 ページ)

詳細

ブレーク行の順序

デフォルト要約に複数のブレーク行が含まれている場合、ブレーク行が表示される順序は次のとおりです。

1 要約行(SUMMARIZE)

2 改ページ(PAGE)

カスタマイズされたブレーク行の詳細については、 COMPUTE ステートメント (1795 ページ) と LINE ステートメント“LINE ステートメント” (1814 ページ)を参照してください。

WEIGHT ステートメント

統計量計算の分析変数に対し重みを指定します。

別名: WGT

参照項目: 重み付き統計量の計算と、WEIGHT ステートメントを使用する例については、“重み付き統計量の計算” (2185 ページ)を参照してください。

構文

WEIGHT *variable*;

必須引数

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

表 51.4 変数値と PROCREPORT の応答方法

重み値	PROC REPORT 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 より小さい	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

制限事項 重み値がアクティブの場合、PROC REPORT は MODE を計算しません。代わりに、MODE の計算が必要で、重み変数がアクティブの場合は、PROC UNIVARIATE を使用しようとします。

ヒント WEIGHT ステートメントを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。詳細については、VARDEF= (1779 ページ)および“キーワードと式” (2410 ページ)の重み付き統計量の計算を参照してください。

詳細

重み情報は保存されない

レポート定義を保存する場合、PROC REPORT は WEIGHT ステートメントを保存しません。

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

使用法: REPORT プロシジャ

PROC REPORT で使用できる統計量

次のキーワードを使用して、KEYWORD ステートメントまたは KEYLABEL ステートメントで統計量キーワードを指定します。

表 51.5 PROC REPORT で使用できる統計量

記述統計量キーワード

CSS	PCTSUM
CV	RANGE
MAX	STD
MEAN	STDERR
MIN	SUM
MODE	SUMWGT
N	USS
NMISS	VAR
PCTN	
四分位範囲統計量キーワード	
MEDIAN | P50	Q3 | P75
P1	P60
P5	P70
P10	P80
P20	P90
P30	P95
P40	P99
Q1 | P25	QRANGE
仮説検定キーワード	
PRT | PROBT	T

これらの統計量、その計算に使用される式、そのデータ要件については、“キーワードと式” (2410 ページ)を参照してください。

標準誤差とスチューデントの **t** 検定を計算するには、VARDEF=のデフォルト値 DF を使用する必要があります。

N 以外のすべての統計量が変数と関連付けられている必要があります。数値表示変数以上または以下の統計量を配置する、または分析変数に対し DEFINE ステートメ

ントで使い方のオプションとして統計量を指定することにより、統計量と変数を関連付けます。

N はどこにでも置くことができます。これは、N がレポートのセルの値に使用する入力データセット内のオブザベーション数であるためです。N の値は、特定の変数に依存していません。

注: PROC REPORT ステートメントで MISSING オプションを使用する場合、N には欠損しているグループ変数、順序変数、列変数を含むオブザベーションが含まれています。

PROC HTTP で ODS スタイルを使用する

Base SAS プロシジャでのスタイルの使用

ODS をサポートする Base SAS プロシジャの多くは、1 つ以上のテーブルテンプレートを使用して出力オブジェクトを生成します。これらのテーブルテンプレートには、テーブル要素(列、ヘッダー、フッター)に対するテンプレートが含まれています。各テーブル要素で、出力のさまざまな部分に 1 つ以上のスタイル要素を使用することを指定できます。これらのスタイル要素はプロシジャの構文内で指定することはできませんが、使用する ODS 出力先に合わせてカスタマイズされたスタイルを使用できます。テーブルとスタイルのカスタマイズの詳細については、“TEMPLATE Procedure: Creating a Style Template” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

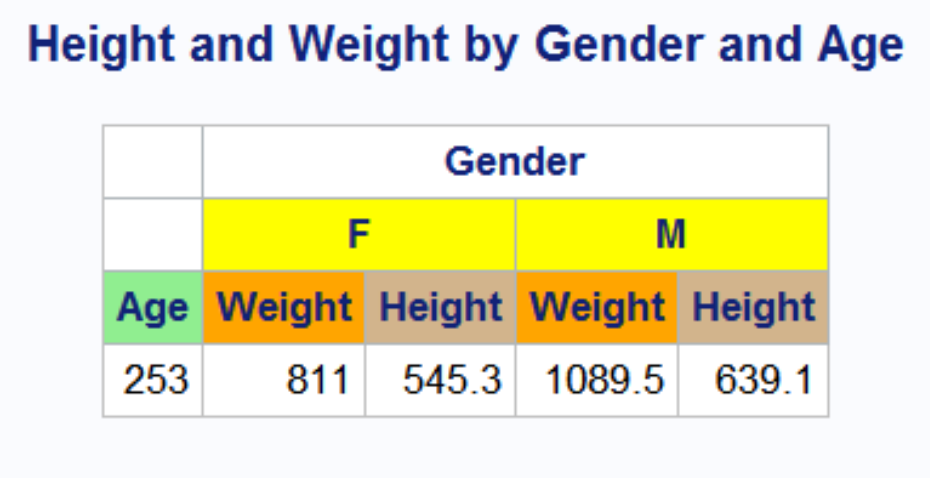
Base SAS レポートプロシジャの PROC PRINT、PROC REPORT および PROC TABULATE を使用してデータをすばやく分析し、読みやすいテーブルに整理することができます。これらのプロシジャステートメントで STYLE=オプションを使用してレポートの表示を変更できます。STYLE=オプションを使用すると、すべての出力のデフォルトスタイルを変更せずに、出力の各セションに変更を加えることができます。プロシジャ内の特定のステートメントで STYLE=オプションを指定して、プロシジャ出力の特定のセクションをカスタマイズすることができます。

次のプログラムでは、STYLE=オプションを使用して以下の PROC REPORT 出力の背景色を作成します。

```
ods _all_ close;
ods html5 file="file-path/customStyle.html";
title "Height and Weight by Gender and Age";
proc report nowd data=sashelp.class
  style(header)=[background=white];
  col age (('Gender' sex),(weight height));
  define age / style(header)=[background=lightgreen];
  define sex / across style(header)=[background=yellow] ' ';
  define weight / style(header)=[background=orange];
  define height / style(header)=[background=tan];
run;
```

```
ods _all_ close;
```

アウトプット 51.1　拡張された PROC REPORT 出力



Height and Weight by Gender and Age

	Gender			
	F		M	
Age	Weight	Height	Weight	Height
253	811	545.3	1089.5	639.1

スタイル、スタイル要素およびスタイル属性

スタイル、スタイル要素、およびスタイル属性について

SAS 出力の外観は、ODS スタイルテンプレート(ODS スタイル)によって制御されます。ODS スタイルは、PROC TEMPLATE スタイルの構文で記述されたコンパイル済みの STYLE テンプレートから生成されます。ODS スタイルテンプレートは、SAS 出力に特定の視覚的属性を提供するスタイル要素のコレクションです。

- スタイル要素は、出力の特定の部分に適用されるスタイル属性の名前付きコレクションです。ODS 出力の各領域には、スタイル要素名が関連付けられています。スタイル要素名は、スタイル属性が適用される場所を指定します。たとえば、スタイル要素には、列見出しの表示やセル内のデータの表示に関する指示が含まれる場合があります。スタイル要素は、スタイルを使用する出力のデフォルトの色とフォントを指定する場合もあります。
- スタイル属性は、予約済みの名前と値を使用して ODS で定義される色、フォントプロパティ、線の特性などの視覚的なプロパティです。Style 属性は、スタイルテンプレート内のスタイル要素によってまとめて参照されます。各*スタイル属性*は、プレゼンテーションの 1 つの側面の値を指定します。たとえば、BACKGROUNDCOLOR=属性は、HTML テーブルの背景色または印刷出力の色付きテーブルの色を指定します。FONTSTYLE=属性は、Roman フォントとイタリックフォントのどちらを使用するかを指定します。

注: スタイルはデータの表示を制御するため DOCUMENT または OUTPUT の出力先に送られる出力オブジェクトには影響しません。

利用可能なスタイルは SASHELP.TMPLMST アイテムストアにあります。PROC TEMPLATE で LIST ステートメントを使用して、使用可能なスタイルテンプレートのリストを表示できます。

```
proc template;
  list styles / store=sashelp.tmplmst;
run;
```

ODS スタイルの表示の詳細については、“Viewing ODS Styles Supplied by SAS” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

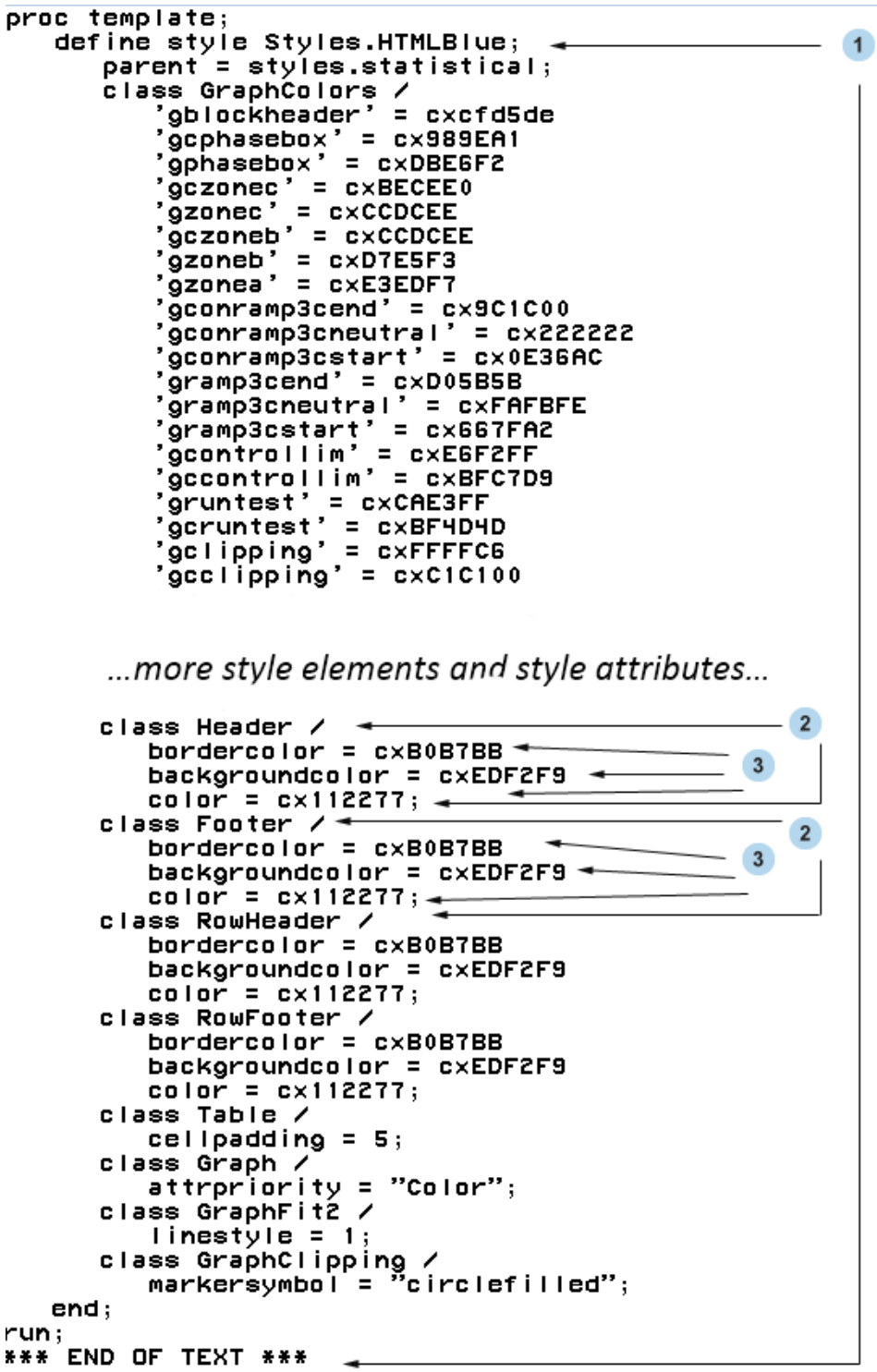
HTML5 出力は、HTMLEncore スタイルテンプレートを使用します。スタイル、スタイル要素、およびスタイル属性に慣れるために、それらの関係に注目してください。

PROC TEMPLATE で SOURCE ステートメントを使用して、スタイルテンプレートの構造を表示できます。次のコードは、HTMLBlue スタイルテンプレートの構造を SAS ログに出力します。

```
proc template;
  source styles.HTMLBlue;
run;
```

次の図は、スタイルの構造を示しています。この図は、スタイル、スタイル要素、およびスタイル属性の関係を示しています。

図 51.9　HtmlBlue スタイルの図



次のリストは、前の図の番号付きの項目に対応しています。

1 Styles.HtmlBlue は*スタイル*です。スタイルは、SAS 出力のプレゼンテーションの側面 (色、フォント、フォント サイズなど) を表示する方法を記述します。スタイルは、それを使用する ODS ドキュメントの全体的な外観を決定します。HTML 出力のデフォルトのスタイルは、HtmlBlue です。各スタイルは、スタイル要素で構成されています。

"DEFINE STYLE Statement" (*SAS Output Delivery System: Procedures Guide*)を使用して新しいスタイルを作成できます。 新しいスタイルは、独立して作成することも、既存のスタイルから作成することもできます。"PARENT= Statement" (*SAS Output Delivery System: Procedures Guide*)を使用して既存のスタイルから新しいスタイルを作成することができます。ODS スタイルの表示の詳細については、"Style Templates" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

2 ヘッダーとフッターは*スタイル要素*の例です。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。たとえば、スタイル要素には、列見出しの表示やテーブルセル内のデータの表示に関する指示が含まれる場合があります。スタイル要素は、そのスタイルを使用する出力のデフォルトの色とフォントを指定する場合もあります。スタイル要素はスタイル内に存在し、1 つ以上のスタイル属性で構成されます。スタイル要素は、ユーザーが定義するか、SAS によって提供されます。ユーザー定義のスタイル要素は、"STYLE Statement" (*SAS Output Delivery System: Procedures Guide*)で作成できます。

注: HTML およびマークアップ言語に使用されるデフォルトのスタイル要素とその継承のリストについては、"Style Elements" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

3 BORDERCOLOR=、BACKGROUNDCOLOR=、および COLOR=は、*スタイル属性*の例です。スタイル属性は、そのスタイル要素が適用される出力領域の 1 つの側面の値を指定します。たとえば、COLOR=属性は、フォントの色に値 **cx112277** を指定します。SAS が提供するスタイル属性のリストについては、"Style Attributes" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

スタイル属性は、スタイル参照で参照できます。スタイル参照の詳細については、"style-reference" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

次の表は、PROC PRINT、PROC TABULATE、および PROC REPORT で STYLE=オプションを使用して設定できる、一般的に使用されるスタイル属性を示しています。これらの属性のほとんどは、セル以外の表の部分に適用されます (たとえば、表の境界線、列と行の間の線など)。すべての出力先ですべての属性が有効であるとは限らないことに注意してください。これらのスタイル属性、その有効な値、およびその適用可能な出力先の詳細については、"Style Attributes Tables" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

表 51.6　PROC REPORT、PROC TABULATE および PROC PRINT のスタイル属性

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
ASIS=	X	X		X		X
BACKGROUNDCOLOR=	X	X	X	X	X	X
BACKGROUNDIMAGE=	X	X	X	X	X	X
BORDERBOTTOMCOLOR=	X	X		X		
BORDERBOTTOMSTYLE=	X	X	X	X		
BORDERBOTTOMWIDTH=	X	X	X	X		
BORDERLEFTCOLOR=	X	X		X		
BORDERLEFTSTYLE=	X	X	X	X		
BORDERLEFTWIDTH=	X	X	X	X		
BORDERCOLOR=	X	X		X	X	X
BORDERCOLORDARK=	X	X	X	X	X	X
BORDERCOLORLIGHT=	X	X	X	X	X	X
BORDERRIGHTCOLOR=	X	X		X		
BORDERRIGHTSTYLE=	X	X	X	X		
BORDERRIGHTWIDTH=	X	X	X	X		

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
BORDERTOPCOLOR=	X	X		X		
BORDERTOPSTYLE=	X	X	X	X		
BORDERTOPWIDTH=	X	X	X	X		
BORDERWIDTH=	X	X	X	X	X	X
CELLPADDING=	X		X		X	
CELLSPACING=	X		X		X	
CELLWIDTH=	X	X	X	X		X
CLASS=	X	X	X	X	X	X
COLOR=	X	X	X			
FLYOVER=	X	X		X		X
FONT=	X	X	X	X	X	X
FONTFAMILY=	X	X	X	X	X	X
FONTSIZE=	X	X	X	X	X	X
FONTSTYLE=	X	X	X	X	X	X
FONTWEIGHT=	X	X	X	X	X	X
FONTWIDTH=	X	X	X	X		X
FRAME=	X		X		X	
HEIGHT=	X	X		X	X	X
HREFTARGET=		X		X		X
HTMLSTYLE=	X	X	X	X	X	
NOBREAKSPACE=[2]	X	X		X		X
OUTPUTWIDTH=	X	X	X	X	X	

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
POSTHTML=[1]	X	X	X	X	X	X
POSTIMAGE=	X	X	X	X	X	X
POSTTEXT=[1]	X	X	X	X	X	X
PREHTML=[1]	X	X	X	X	X	X
PREIMAGE=	X	X	X	X	X	X
PRETEXT=[1]	X	X	X	X	X	X
PROTECTSPECIALCHARS=		X		X	X	X
RULES=	X		X		X	
TAGATTR=	X	X	X	X	X	X
TEXTALIGN=	X	X	X	X	X	X
URL=		X		X		X
VERTICALALIGN=		X		X		X
WIDTH=	X	X	X	X	X	

1 これらの属性をこの場所で使用する場合には、属性 PRETEXT=、POSTTEXT=、PREHTML=、POSTHTML=で指定されるテキストにのみ影響します。表に表示されるテキストの前景色またはフォントを変更するには、表ではなくセルに影響する場所で対応する属性を設定する必要があります。スタイル属性とその値の詳細な説明については、"Style Attributes" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

2 PROC REPORT を ODS RTF 出力先で使用する場合、長いテキスト文字列の予期せぬ折り返しを防ぐために、LINE ステートメントに影響する場所で NOBREAKSPACE=OFF を設定してください。NOBREAKSPACE=OFF 属性は、PROC REPORT コード内の LINE ステートメントまたは style(line)が指定されている PROC REPORT ステートメントのいずれかに設定する必要があります。

テーブル領域のスタイル要素とスタイル属性

次のテーブルに、PROC REPORT 出力の各リージョンのデフォルトのスタイル要素およびスタイル属性を示します。このテーブルの場所は (1776 ページ) の場所に対応します。

このテーブルには、最もよく使用される ODS 出力先のデフォルトを示します。HTML、PDF および RTF のデフォルトがリストされています。それぞれの出力先に

は、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルテンプレートが含まれます。ODS 出力先とそのデフォルトのスタイルに関する詳細なドキュメントは、“Style Templates” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

注:

SAS Studio での ODS 出力のデフォルトのスタイルは、SAS Studio がアクティブに使用しているテーマによって設定されます。テーマに応じて、デフォルトは Illuminate、Ignite、または HighContrast のいずれかです。詳細については、“Customizing SAS Studio” (*SAS Studio: User's Guide*)を参照してください。

- HTML5 出力のデフォルトスタイルは HTMLEncore です。
- PDF の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

表 51.7 テーブルの各場所のデフォルトのスタイル要素とスタイル属性

場所	スタイル要素	HTML5 スタイル属性	PDF スタイル属性	RTF スタイル属性
Header	Header	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb
Column	Data	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 8pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000 BORDERWIDTH = NaN COLOR = cx000000	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000

場所	スタイル要素	HTML5 スタイル属性	PDF スタイル属性	RTF スタイル属性
Summary	DataEmphasis	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 8pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000 BORDERWIDTH = NaN	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = italic COLOR = cx000000

データセルにスタイル属性を適用する際の優先順位

PROC REPORT は、デフォルトの優先順位から、特定のセルに適用するスタイル属性を決定します。優先順位のステップが上がるごとに、よりきめ細かく指定されます。

たとえば、次に示す非要約行にはスタイルの優先順位を使用します。まず、特定のセルについて、PROC REPORT はデフォルトのスタイル属性を使用します。次に、列のセルごとに、PROC REPORT がデフォルトのスタイル属性を上書きします。ステップ 5 までに、適用された以前のスタイルが上書きされますが、 *column-id* で指定されたセルに対してのみです。

要約行と要約行以外の行のスタイルの優先順位を次に示します。

要約行以外の行のスタイル優先順位

1 PROC REPORT は、デフォルトのスタイル要素(Data)のデフォルトのスタイル属性を使用します。
2 PROC REPORT ステートメントの STYLE (COLUMN)=オプションによってデフォルトのスタイル属性は上書きされます。
3 DEFINE ステートメントの STYLE(COLUMN)=オプションは、列のすべてのセルに適用されます。
4 CALL DEFINE ステートメントの_ROW_引数によって指定される行のスタイルは、行のすべてのセルに適用されます。
5 CALL DEFINE ステートメントの *column-id* _COL_によって指定されるスタイルは、列のすべてのセルに適用されます。

要約行のスタイル優先順位

1 PROC REPORT は、要約行のデフォルトのスタイル要素(DataEmphasis)のデフォルトのスタイル属性を使用します。

2 PROC REPORT ステートメントの STYLE (SUMMARY)=オプションは、要約のデフォルトのスタイル属性を上書きします。

3 BREAK ステートメントの STYLE=オプションによって指定されるスタイルは、要約セルに適用されます。

4 行のスタイルは、CALL DEFINE ステートメントで、_ROW_引数を使用して指定されます。

5 CALL DEFINE ステートメントで、column-id _COL_、列名または列番号を使用して指定されたスタイルは、列のすべての要約セルに適用されます。

出力形式を使用したスタイル属性値の割り当て

出力形式を使用して、スタイル属性値を割り当てることができます。たとえば、次のコードを使用して、値が負になっている Difference 列に赤い背景色を割り当てます。

```
proc format;
value proffmt low-<0='red'
0-high='green';
run;

proc report data=sashelp.prdsale;
column country predict actual diff;
define country /group;
define diff /'Difference' computed format=dollar12.2
  style(column)=[backgroundcolor=proffmt.];

compute diff;
diff = predict.sum - actual.sum;
endcomp;
run;
```

Country	Predicted Sales	Actual Sales	Difference
CANADA	$233,019.00	$246,990.00	$-13,971.00
GERMANY	$231,554.00	$245,998.00	$-14,444.00
U.S.A.	$241,722.00	$237,349.00	$4,373.00

行間隔の制御

ユーザーは、ページのより多くの行に合うようにレポートを"縮小"する必要があることがよくあります。レポートの縮小により、フォントサイズや行間隔が変わります。プログラマーに最大の柔軟性を提供するため、ODS では行間隔の計算に REPORT の場所に指定されているフォントサイズを使用します。そのため、テーブルを縮小するには、REPORT の場所と COLUMN の場所のフォントサイズを変更します。次に例を示します。

```
proc report
data=libref.data—set-name
        style(report)=[fontsize=8pt]
        style(column)=[font=(Arial, 8pt)];
```

レポート定義の格納と再利用

PROC REPORT ステートメントの OUTREPT=オプションは、レポート定義を指定したカタログエントリに格納します。

注: レポート定義がレポートを作成する SAS プログラムと異なっている場合があります。 OUTREPT= (1773 ページ)の説明を参照してください。

レポート定義を使用して、レポート定義で使用されるものと同じ名前の変数を含む SAS データセットに対して同一に構築されたレポートを作成できます。PROC REPORT ステートメントの REPORT=オプションを使用して、PROC REPORT の開始時にレポート定義をロードします。詳細については、“REPORT=*libref.catalog.entry*” (1775 ページ)を参照してください。

PROC REPORT の In-database 処理

In-Database 処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。セキュリティの強化は、機密データをデータベース管理システム(DBMS)から抽出する必要がないため可能です。データが、比較的低速なネットワーク接続を介して移送されるかわりに、高速の二次記憶装置を使用して、DBMS 上でローカル操作されます。DBMS が使用されるのは、DBMS に自由に使えるより多くの処理リソースがあり、高度に並列かつスケーラブルな方法で実行するクエリを最適化できるためです。

DATA=入力データセットが DBMS でテーブルまたはビューとして保存される場合、PROC REPORT プロシジャは In-Database 処理を使用して、データベース内の作業のほとんどを実行できます。In-Database 処理は、より迅速な処理と、データベースと SAS ソフトウェア間のデータ転送の減少という利点を提供できます。

PROC REPORT の In-Database 処理では、次のデータベース管理システムがサポートされています。

- Amazon Redshift
- DB2
- Greenplum
- Hadoop
- Impala
- Microsoft SQL Server
- MySQL (2021.1.5 現在)

- Netezza
- Oracle
- PostgreSQL
- SAP HANA
- Snowflake
- Spark
- Teradata
- Vertica

PROC REPORT は、SQL 暗黙のパススルーを使用して In-Database 処理を実行します。プロシジャはステートメントに基づく SQL クエリと、プロシジャで指定される出力統計量と同様に使用される PROC REPORT オプションを生成します。データベースはこれらの SQL クエリを実行し、クエリの結果は PROC REPORT に送信されます。生成された SQL を検証するには、SASTRACE=オプションを設定します。

SAS 出力形式定義がデータベースで配置されていない場合、未加工値に対し In-Database 集計が実行され、結果のセットが PROC REPORT 内部構造と結合されるときに関連する出力形式が適用されます。詳細については、*SAS/ACCESS for Relational Databases: リファレンス*の“Deploying and Using SAS Formats”の章を参照してください。

使い方の種類が DISPLAY または ORDER の変数が PROC REPORT ステップに含まれている場合、In-Database 処理は発生しません。

In-Database 処理に対しサポートされている統計量は、N、NMISS、MIN、MAX、MEAN、RANGE、SUM、SUMWGT、CSS、USS、VAR、STD、STDERR、および CV です。

In-Database 処理の重み付けは、N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN に対してのみサポートされています。

SQLGENERATION システムオプションまたは LIBNAME ステートメントオプションは、In-Database プロシジャがデータベース内で実行されるかどうか、およびその実行方法を制御します。デフォルトで、In-Database プロシジャは可能な場合はデータベース内で実行されます。In-Database 処理を実行しない、多くのデータセットオプションがあります。完全なリストについては、*SAS/ACCESS for Relational Databases: リファレンス*の“In-Database Procedures”を参照してください。

In-Database 処理の詳細については、*SAS/ACCESS for Relational Databases: リファレンス*を参照してください。

PROC REPORT の CAS 処理

あなたの入力データセットが SAS Cloud Analysis Services (CAS)からのものであれば、PROC REPORT 分析の一部を CAS サーバーで実行できます。データの有意な要約を必要とするレポートは、CAS 処理の恩恵を受けることができます。CAS テーブルに対して実行すると、PROC REPORT は分析サーバー上で実行され、適切な統計アクションを使用します。詳細については、4 章, “Base プロシジャの CAS 処理” (75 ページ)を参照してください。

CAS LIBNAME ステートメントオプションは、CAS プロシジャが CAS 内で実行されるかどうか、およびその実行方法を制御します。デフォルトで、CAS プロシジャは可能な場合は CAS 内で実行されます。ただし、CAS 処理を妨げるデータセットオプションが多数あります。

DATA=入力データセットが CAS 内のインメモリテーブルまたはビューを参照する場合、REPORT プロシジャは CAS アクションを使用してサーバー内でその作業のいくつかを実行できます。インメモリテーブルまたはビューを参照するには、CAS エンジンの LIBNAME ステートメントを指定し、CAS エンジンのライブラリ参照オプション IN=または DATA=を指定する必要があります。デフォルトでは、PROC REPORT は CAS エンジンのライブラリ参照名が入力に指定されているときは、常に CAS 処理を使用します。

次の例では、CAS 処理を使用する PROC REPORT を実行する方法を示します。LIBNAME ステートメントは、CAS セッション casauto に接続するために使用する mycas という名前の CAS エンジンライブラリ参照名を割り当てます。

```
option casport=10935 cashost="cloud.example.com";
cas casauto ;

libname mycas cas;
data mycas.cars;
  set sashelp.cars;
run;

proc report data=mycas.cars;
title;
column ('Horsepower by Make and Drivetrain as a Percent of All Horsepower'
origin drivetrain horsepower,(sum pctsum));
define origin / group;
define drivetrain / group;
define horsepower / '';
define sum / 'HP by Make and Drivetrain';
define pctsum / 'Percent of HP' format=percent6.;
rbreak after / summarize style=[font_style=italic];
run;
```

使い方の種類が DISPLAY または ORDER の変数が PROC REPORT ステップに含まれている場合、CAS 処理は発生しません。DISPLAY または ORDER 変数が見つかった場合、すべてのデータがクライアントに 戻され、PROC REPORT が SAS クライアントサーバーで実行されます。

次の統計量が、CAS 処理に対しサポートされています。

CSS	RANGE
CV	STDERR
MAX	SUM
MEAN	SUMWGT
MIN	STD
N	USS
NMISS	VAR

注: CAS 処理の重み付けは、N、NMISS、MIN、MAX、RANGE、SUM、SUMWGT、MEAN に対してのみサポートされています。

計算と処理は CAS で行われます。ただし、ファイナルテーブルのレンダリングは SAS クライアントによって行われます。

CAS LIBNAME ステートメントの使用方法については、“LIBNAME ステートメント: CAS エンジン” (*SAS Cloud Analytic Services: ユーザーガイド*)を参照してください。

テキスト文字列の BY 行値の置換

#BYLINE、#BYVAR、および#BYVAL の置換は、次のオプションで使用できます。

- PROC REPORT ステートメントの CONTENTS=オプション
- PROC REPORT ステートメントの CAPTION=オプション

#BYVAR 置換および#BYVAL 置換を使用するには、代替テキストを表示する位置のテキスト文字列にその項目を挿入します。#BYVAR と#BYVAL の両方の指定の後には、区切り文字を付ける必要があります。文字は、スペース、または引用符などの他の英数字以外の文字にすることができます。区切り文字が指定されていない場合、指定は無視され、そのテキストはそのまま残り、文字列の残りの部分とともに表示されます。#BYVAR および#BYVAL の置換のすぐ後に、区切り文字なしに、他のテキストの直後に続くようにするには、(マクロ変数と同じように)末尾にドットを使用します。解決されたテキストには、末尾のドットは表示されません。解決されたテキストの最後の文字としてピリオドを表示する場合は、＃BYVAR または#BYVAL 置換の後に 2 つのドットを使用します。

次の場合、＃BYVAR または#BYVAL の置換は行われません。

- BY ステートメントで指定されていない変数に#BYVAR または#BYVAL 指定を使用する場合。たとえば、BY 変数が 1 つしかない場合は＃BYVAL2 を使用し、ABC が存在しないか BY 変数ではない場合は#BYVAL(ABC)を使用します。
- BY ステートメントがない場合

結果: REPORT プロシジャ

PROC REPORT でのレポート作成法

イベントの順序

このセクションでは、レポート作成の一般的なプロセスについて説明します。このプロセスの例については、“レポートの作成例” (1840 ページ)を参照してください。.

レポート作成のプロセスを理解するには、レポート変数と一時変数の違いを理解する必要があります。**レポート変数**は、COLUMN ステートメントで指定される変数です。入力データセットから取得することも、計算する(変数の DEFINE ステートメントが COMPUTED オプションを指定)こともできます。計算ブロックに表示されることも、表示されないこともあります。1 つ以上の計算ブロックだけに表示される変数は、**一時変数**です。一時変数はレポートに表示されず、出力データセットに書き込まれません(要求される場合)。

PROC REPORT は、レポートの各行の始めのレポート変数を欠損値に初期化します。**一時変数**の値は、PROC REPORT がレポートの行の構成を開始する前に欠損値に初期化され、値を割り当てるまで欠損値のままです。PROC REPORT は、計算ブロック間の実行からの**一時変数**の値を保持します。

PROC REPORT はレポートを次のように構成します。

1 グループ変数、順序変数、列変数ごとにデータをまとめます。レポートのすべての統計量、詳細行の統計量、およびブレークの要約行の統計量を計算します。統計量には、分析変数に対して計算される統計量が含まれます。PROC REPORT は要約行の統計量を、レポートに表示されるかどうかに関係なく計算します。

2 すべての一時変数を欠損値に初期化します。

3 レポートの行の構成を開始します。

   a 各行の始めに、すべてのレポート変数を欠損値に初期化します。

   b レポート変数の値を左から右に入力します。

      - 計算変数の値は、対応する計算ブロックのステートメント実行によって取得されます。
      - その他すべての変数の値は、データセット、またはレポート作成プロセスの開始時に計算された要約統計量から取得されます。

c ブレークに関しては、PROC REPORT はまず BREAK ステートメントまたは RBREAK ステートメント、または **BREAK** ウィンドウのオプションで作成されるブレーク行を構成します。

d 計算ブロックがブレークに割り当てられている場合、PROC REPORT は計算ブロックのステートメントを実行します。詳細については、“要約行の構成” (1839 ページ)を参照してください。

注: 次のように、PROC REPORT で統計量を使用することもできます。

- グループ変数の前のブレークに対する計算ブロックのグループ統計量を使用します。
- レポートの始めの計算ブロックの全体レポートに対する統計量を使用します。

このドキュメントは、適切な複合名の統計量を参照します。計算ブロックのレポート項目の参照の詳細については、“計算ブロックのレポート項目を参照するための 4 つの方法” (1756 ページ)を参照してください。

注: LINE ステートメントを条件ステートメント(IF-THEN、IF-THEN/ELSE および SELECT)で使用できません。LINE ステートメントは、PROC REPORT が計算ブロックのその他すべてのステートメントを実行するまで実行されないためです。

4 各レポート行が完了すると、PROC REPORT は行を現在オープンなすべての ODS 出力先に送信します。

要約行の構成

次の条件のいずれかが真の場合、PROC REPORT はブレークの要約行を構成します。

- ブレークの数値変数を要約します。
- ブレークの計算ブロックを使用します。(BREAK ステートメントまたは RBREAK ステートメントを使用せずに、または **BREAK** ウィンドウでオプションを選択せずに計算ブロックをブレークに添付できます)。

  計算ブロックの使用の詳細については、“計算ブロックの使用” (1754 ページ)および“COMPUTE ステートメント” (1795 ページ)を参照してください。

PROC REPORT がこの時点で構成する要約行は、予備です。計算ブロックがブレークに割り当てられていない場合、予備要約行は最終要約行となります。ただし、計算ブロックがブレークに割り当てられている場合、計算ブロックのステートメントは予備要約行の値を変える可能性があります。

PROC REPORT が要約行を印刷するのは、ブレークの数値変数を要約する場合だけです。

レポートの作成例

グループを使用したレポートとレポート要約の作成

“グループを含むレポートとレポート要約” (1841 ページ)のレポートには、5 つの列が含まれています。

- Sector と Department はグループ変数です。
- Sales は、Sum 統計量の計算に使用される分析変数です。
- Profit は、値が Department の値に基づく計算変数です。
- N 統計量は、各行が表すオブザベーション数を示します。

レポートの最後で、ブレークはレポートの統計量と計算変数を要約して、Sector に TOTALS:の値を割り当てます。

次のステートメントは“グループを含むレポートとレポート要約” (1841 ページ)を作成します。使用されるユーザー定義の出力形式が PROC FORMAT ステップ (1850 ページ)によって作成されます。

```
libname proclib 'SAS-library';

options nodate pageno=1 linesize=64
      pagesize=60 fmtsearch=(proclib);

proc report data=grocery;
  column sector department sales Profit N ;
  define sector / group format=$sctrfmt.;
  define department   / group format=$deptfmt.;
  define sales  / analysis sum
            format=dollar9.2;
  define profit / computed format=dollar9.2;

  compute before;
  totprof = 0;
  endcomp;

  compute profit;
  if sector ne ' ' or department ne ' ' then do;
    if department='np1' or department='np2'
      then profit=0.4*sales.sum;
    else profit=0.25*sales.sum;
    totprof = totprof + profit;
    end;
  else
    profit = totprof;
  endcomp;

  rbreak after / summarize;
  compute after;
    sector='TOTALS:';
  endcomp;
```

```
  where sector contains 'n';
  title 'Report for Northeast and Northwest Sectors';
run;
ods listing close;
```

例のコード 51.2 グループを含むレポートとレポート要約

```
    Report for Northeast and Northwest Sectors        1

Sector    Department     Sales    Profit        N
-----------------------------------------------------


Northeast  Canned       $840.00   $336.00        2
       Meat/Dairy    $490.00   $122.50        2
       Paper       $290.00    $116.00        2
       Produce      $211.00    $52.75         2
Northwest  Canned      $1,070.00   $428.00        3
       Meat/Dairy  $1,055.00   $263.75        3
       Paper       $150.00    $60.00         3
       Produce      $179.00    $44.75         3
=========          ========= ========= =========
TOTALS:            $4,285.00  $1,423.75       20
=========          ========= ========= =========
```

PROC REPORT によるこのレポートの作成方法についての説明は、次のとおりです。

1 PROC REPORT は、データ(Sector と Department はグループ変数です)をまとめ、レポートの最後に各詳細行およびブレークに対し統計量(Sales.sum と N)を計算することにより、レポートの作成を開始します。

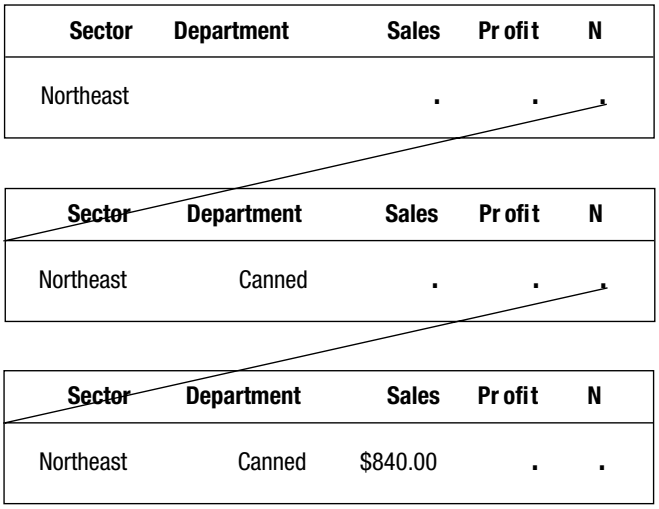
2 これで、PROC REPORT はレポートの最初の行の作成を開始できます。このレポートにはレポートの始めのブレークまたはグループの前のブレークが含まれていないため、レポートの開始行は詳細行になります。次の図にあるように、プロシジャはすべてのレポート 変数を欠損値に初期化します。文字変数の欠損値はブランクで表され、数値変数の欠損値はピリオドで表されます。

図 51.10 値が初期化された最初の詳細行

Sector	Department	Sales	Profit	N
		.	.	.

3 次の図に、行の最初の 3 列の構造を示します。PROC REPORT は、行の値を左から右に入力します。値は、レポート作成プロセスの開始時に計算された統計量から取得されます。

図 51.11　値が左から右に入力された最初の詳細行



Sector	Department	Sales	Profit	N
Northeast		.	.	.

Sector	Department	Sales	Profit	N
Northeast	Canned	.	.	.

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	.	.

4 レポートの次の列には、計算変数 Profit が含まれます。この列に達すると、PROC REPORT は Profit に割り当てられている計算ブロックのステートメントを実行します。非生鮮品目(値が **np1** または **np2**)は利益の 40%を占め、生鮮品目(値が **p1** または **p2**)は利益の 25%を占めています。

```
if department='np1' or department='np2'
   then profit=0.4*sales.sum;
else profit=0.25*sales.sum;
```

行は次の図のようになります。

注: 計算変数の位置は重要です。PROC REPORT は値をレポート行の列に、左から右に割り当てます。その結果、レポートの右側に表示される変数に基づいて計算変数の計算を行うことはできません。

図 51.12　最初の詳細行に追加された計算変数

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	$336.00	.

5 totprof 変数は、総利益の累積合計を保持するためにメモリに保持される一時変数です。各セクターおよび部門の利益が計算されると、変数 totprof に格納されている値が Profit に移動され、TOTALS 要約行に報告されます。

```
totprof = totprof + profit;
     end;
  else
     profit = totprof;
```

6 次に、PROC REPORT は N 統計量に対する値を入力します。値は、レポート作成プロセスの開始時に作成された統計量から取得されます。次の図に、完了した行を示します。

図 51.13 最初の完了詳細行

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	$336.00	2

7 プロシジャは完了した行をレポートに書き込みます。

8 PROC REPORT は、レポートの詳細行ごとにステップ 2、3、4、5、6 を繰り返します。

9 レポートの最後のブレークで、PROC REPORT は RBREAK ステートメントによって記述されるブレーク行を構成します。これらの行には、要約行の暫定版が含まれます。要約行の統計量は前に計算されています。(ステップ 1 を参照)。計算変数の値は、詳細行と同様に、PROC REPORT が適切な列に達すると計算されます。PROC REPORT はこれらの値を使用して、要約行の予備バージョンを作成します。

図 51.14 予備要約行

Sector	Department	Sales	Profit	N
		$4,285.00	$1,423.75	20

10 計算ブロックがブレークに割り当てられていない場合、要約行の予備バージョンは最終バージョンと同じです。ただしこの例では、計算ブロックはブレークに割り当てられています。そのため、PROC REPORT は計算ブロックのステートメントを今実行します。この場合、計算ブロックにはステートメントが 1 つ含まれています。

```
sector='TOTALS:';
```

このステートメントは、デフォルトで要約行にはない Sector の値を TOTALS:という言葉と置き換えます。PROC REPORT はステートメントを実行した後、要約行を変更して、この変更を Sector の値に反映します。要約行の最終バージョンが次の図に表示されます。

図 51.15 最終要約行

Sector	Department	Sales	Profit	N
TOTALS:		$4,285.00	$1,423.75	20

11 最後に、PROC REPORT はすべてのブレーク行と最終要約行とともにレポートに書き込みます。“グループを含むレポートとレポート要約” (1841 ページ)を参照してください。

一時変数を使用するレポートの作成

PROC REPORT は、レポートの各行の始めのレポート変数を欠損値に初期化します。一時変数の値は、PROC REPORT がレポートの行の構成を開始する前に欠損値に初

期化され、値を割り当てるまで欠損値のままです。PROC REPORT は、計算ブロック間の実行からの一時変数の値を保持します。

すべての計算ブロックですべての変数の現在の値を共有しているため、レポートの始めのブレークまたはブレーク変数の前のブレークで一時変数を初期化できます。このレポートは、Sector の前のブレークで一時変数 Sctrtot を初期化します。

注: PROC REPORT は、対応する計算ブロックを実行する前に、ブレークの予備要約行を作成します。要約行に計算変数が含まれている場合、計算は予備要約行の要因となる変数の値に基づいています。計算ブロックで設定した値に基づいて計算変数を再計算する場合は、計算ブロックで明示的に行う必要があります。このレポートでは、この方法について示します。計算ブロックがブレークに割り当てられていない場合、予備要約行は最終要約行となります。

“一時変数を含むレポート” (1845 ページ)のレポートには、5 つの列が含まれています。

- Sector と Department はグループ変数です。
- Sales はこのレポートで二度使用される分析変数です。Sum 統計量の計算と Pctsum 統計量の計算にそれぞれ一度ずつ使用されます。
- Sctrpct は、値が Sales、およびセクタの売上合計である一時変数 Sctrtot の値に基づく計算変数です。

レポートの始めに、カスタマイズされたレポート要約によりすべての店舗の売上が次のようになるように指定されます。部門の各オブザベーショングループの前のブレークで、デフォルト要約はそのセクタのデータを要約します。各グループの最後にブレークはブレーク行を挿入します。

次のステートメントは“一時変数を含むレポート” (1845 ページ)を作成します。使用されるユーザー定義の出力形式が PROC FORMAT ステップ (1850 ページ)によって作成されます。

注: パーセントの計算は、その結果に 100 を掛けません。PROC REPORT はその結果を PERCENT.出力形式で印刷するためです。

```
libname proclib 'SAS-library';

proc report data=grocery ;
column sector department sales Sctrpct sales=Salespct;
define sector / 'Sector' group format=$sctrfmt.;
define department / group format=$deptfmt.;
define sales / 'Sum of/sales' analysis sum format=dollar9.2 ;
define sctrpct / 'Percent/of Sector' computed format=percent9.2 ;
define salespct / 'Percent of/All Stores' pctsum format=percent9.2;
compute before _page_;
line ' ';
line  'Total for all stores is ' sales.sum dollar9.2;
line ' ';
endcomp;
break before sector / summarize ;
compute before sector;
sctrtot=sales.sum;
sctrpct=sales.sum/sctrtot;
endcomp;
```

```
compute sctrpct;
sctrpct=sales.sum/sctrtot;
endcomp;
compute after sector;
line ' ';
endcomp;;
where sector contains 'n';
title 'Report for Northeast and Northwest Sectors';
run;
```

例のコード 51.3 一時変数を含むレポート

```
    Report for Northeast and Northwest Sectors        1


       Total for all stores is $4,285.00

                Sum of    Percent   Percent of
Sector    Department  Sales     of Sector  All Stores
============================================================

Northeast           $1,831.00   100.00%    42.73%
---------           ---------  ---------  ---------
Northeast  Canned      $840.00    45.88%    19.60%
       Meat/Dairy    $490.00    26.76%    11.44%
       Paper         $290.00    15.84%     6.77%
       Produce       $211.00    11.52%     4.92%

Northwest           $2,454.00   100.00%    57.27%
---------           ---------  ---------  ---------
Northwest  Canned     $1,070.00   43.60%    24.97%
       Meat/Dairy  $1,055.00    42.99%    24.62%
       Paper         $150.00     6.11%     3.50%
       Produce       $179.00     7.29%     4.18%
```

PROC REPORT によるこのレポートの作成方法についての説明は、次のとおりです。

1 PROC REPORT は、データ(Sector と Department はグループ変数です)をまとめ、各詳細行、レポートの始めのブレーク、各グループの前のブレーク、および各グループの後のブレークに対し統計量(Sales.sum と Sales.pctsum)を計算することにより、レポートの作成を開始します。

2 PROC REPORT は、一時変数 Sctrtot を欠損値に初期化します。

図 51.16 初期化された一時変数

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
		.	.	.	.

3 この PROC REPORT ステップには COMPUTE BEFORE ステートメントが含まれているため、プロシジャはレポートの始めのブレークに対し予備要約行を構成します。この予備要約行には、統計量(Sales.sum と Sales.pctsum)と計算変数(Sctrpct)に対する値が含まれます。

このブレークで、Sales.sum は全店舗の売上で、Sales.pctsum はこれらの売上が全店舗に対して表すパーセント(100%)です。PROC REPORT は、これらの統計量の値をレポート作成プロセスの開始時に計算された統計量から取得します。

Sctrpct の値は、対応する計算ブロックのステートメントの実行から取得されます。Sctrtot の値が欠損しているため、PROC REPORT は Sctrpct の値を計算できません。そのため、(この場合は印刷されない)予備要約行のこの変数にも欠損値が含まれています(次の図を参照)。

COMPUTE BEFORE ブロックのステートメントは、変数を変えることはありません。そのため、最終要約行は、予備要約行と同じです。

注: COMPUTE BEFORE ステートメントは、レポートの始めにブレークを作成します。RBREAK ステートメントを使用する必要はありません。

図 51.17 レポートの始めのブレークの予備/最終要約行

		Report Variables			Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
		$4,285.00	.	100.00%	.

4 SUMMARIZE オプションを指定した RBREAK ステートメントがプログラムに含まれていないため、PROC REPORT は最終要約行をレポートに書き込みません。かわりに LINE ステートメントを使用して、Sales.sum の値を文に組み込むカスタマイズされた要約を書き込み、カスタマイズされた列ヘッダーを書き込みます。

5 次に、PROC REPORT はオブザベーションの最初のグループの前のブレークに対して予備要約行を構成します。(このブレークは BREAK ステートメントの SUMMARIZE オプションを使用し、計算ブロックが割り当てられています。これらの条件のうちいずれかが要約行を生成します)。予備要約行には、ブレーク変数(Sector)、統計量(Sales.sum と Sales.pctsum)、計算変数(Sctrpct)に対する値が含まれています。このブレークでは、Sales.sum は 1 つのセクタ(北東セクタ)の売上です。PROC REPORT は Sector、Sales.sum、Sales.pctsum の値をレポート作成プロセスの開始時に計算された統計量から取得します。

Sctrpct の値は、対応する計算ブロックのステートメントを実行することにより取得されます。Sctrtot の値がまだ欠損しているため、PROC REPORT は Sctrpct の値を計算できません。そのため、予備要約行では、Sctrpct に欠損値が含まれています。

図 51.18 オブザベーションの最初のグループの前のブレークに対する予備要約行

		Report Variables			Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast		$1,831.00	.	42.73%	.

6 PROC REPORT は、COMPUTE BEFORE SECTOR 計算ブロックのステートメントを実行することにより、要約行の最終バージョンを作成します。これらのステートメントは、Sector の値が変わるたびに一度実行します。

- 最初のステートメントは、レポートのその部分で 1 つの Sector の合計売上を表す Sales.sum の値を変数 Sctrtot に割り当てます。
- 2 番目のステートメントは、Sctrtot の新しい値から Sctrpct を再計算することによって要約行を完了します。次の表に、最終要約行を示します。

注: この例では、最終要約行の Sctrpct の値を再計算する必要があります。再計算しない場合、Sctrpct の値は欠損値になります。COMPUTE Sctrpct ブロックの実行時点で Sctrtot の値が欠損値であるためです。

図 51.19 オブザベーションの最初のグループの前のブレークに対する最終要約行

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast		$1,831.00	100.00%	42.73%	$1,831.00

7 プログラムには SUMMARIZE オプションが指定されている BREAK BEFORE ステートメントが含まれているため、PROC REPORT は最終要約行をレポートに書き込みます。

8 これで、PROC REPORT は最初のレポート行の作成を開始できます。すべてのレポート変数は欠損値に初期化されます。一時変数の値は変わりません。次の表に、この時点での最初の詳細行を示します。

図 51.20 値が初期化された最初の詳細行

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
		.	.	.	$1,831.00

9 次の図に、行の最初の 3 列の構造を示します。PROC REPORT は、行の値を左から右に入力します。値は、レポート作成プロセスの開始時に計算された統計量から取得されます。

図 51.21 左から右への値の入力

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast		.	.	.	$1,831.00

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast	Canned	.	.	.	$1,831.00

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast	Canned	$840.00	.	.	$1,831.00

10 レポートの次の列には、計算変数 Sctrpct が含まれます。この列に達すると、PROC REPORT は Sctrpct に割り当てられている計算ブロックのステートメントを実行します。このステートメントは、この部門が占めるセクタの売上合計のパーセントを計算します。

```
sctrpct=sales.sum/sctrtot;
```

行は次の図のようになります。

図 51.22　最初の計算変数が追加された最初の詳細行

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast	Canned	$840.00	45.88%	.	$1,831.00

11 レポートの次の列には、統計量 Sales.pctsum が含まれます。PROC REPORT はこの値を、レポート作成プロセスの開始時に作成された統計量から取得します。最初の詳細行はこれで完了です。

図 51.23　最初の完了詳細行

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast	Canned	$840.00	45.88%	19.60%	$1,831.00

12 PROC REPORT は詳細行をレポートに書き込みます。グループの詳細行ごとにステップ 8、9、10、11、12 を繰り返します。

13 グループの最後の詳細行をレポートに書き込んだ後、PROC REPORT はデフォルトグループ要約を構成します。計算ブロックがこのブレークに割り当てられておらず、BREAK AFTER ステートメントに SUMMARIZE オプションが含まれていないため、PROC REPORT は要約行を構成しません。

14 これで、ブレーク変数の値が Northeast から Northwest に変わります。PROC REPORT は、このオブザベーショングループの前のブレークに対し予備要約行を構成します。行の始めに、ROC REPORT はすべてのレポート変数を欠損値に初期化しますが、一時変数の値は保持します。次に、ブレーク変数(Sector)、統計量(Sales.sum と Sales.pctsum)、計算変数(Sctrpct)に対する適切な値を含む予備要約行を入力します。このブレークで、Sales.sum は北西セクタの売上です。COMPUTE BEFORE Sector ブロックが未実行のため、Sctrtot の値は北東セクタの値である$1,831.00 のままです。そのため、PROC REPORT がこの予備要約行の Sctrpct に対して計算する値は、正しくありません。このブレークの計算ブロックのステートメントは正しい値を計算します。

図 51.24　オブザベーションの 2 番目のグループの前のブレークに対する予備要約行

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northwest		$2,454.00	134.00%	57.27%	$1,831.00

注意

結果が正しくなるように、ブレーク行の計算変数の値を同期します。 PROC REPORT ステップがそのブレークに割り当てられている計算ブロックの Sctrpct を再計算しない場合、最終要約行の値は要約行のその他の値と同期されず、レポートは正しくなくなります。

15 PROC REPORT は、COMPUTE BEFORE Sector 計算ブロックのステートメントを実行することによって、要約行の最終バージョンを作成します。これらのステートメントは、Sector の値が変わるたびに一度実行します。

- 最初のステートメントは、レポートのその部分で 1 つの北西セクタの売上を表す Sales.sum の値を変数 Sctrtot に割り当てます。
- 2 番目のステートメントは、Sctrtot の新しい適切な値から Sctrpct を再計算することによって要約行を完了します。次の表に、最終要約行を示します。

図 51.25 オブザベーションの 2 番目のグループの前のブレークに対する最終要約行

Report Variables					Temporary Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northwest		$2,454.00	100.00%	57.27%	$2,454.00

プログラムには SUMMARIZE オプションが指定されている BREAK BEFORE ステートメントが含まれているため、PROC REPORT は最終要約行をレポートに書き込みます。

16 これで、PROC REPORT はこのオブザベーショングループの最初の行の作成を開始できます。入力データセットのすべてのオブザベーションが処理されるまで、ステップ 8 から 16 まで(最後のオブザベーショングループについてはステップ 14 まで)が繰り返されます。

例: REPORT プロシジャ

例 1: 変数の選択とレポートの要約行の作成

要素:

PROC REPORT ステートメントオプション
COLUMN ステートメント
RBREAK ステートメントオプション
  AFTER
  STYLE=
  SUMMARIZE
LIBNAME ステートメント
PROC FORMAT ステートメント
DATA ステップ
OPTIONS ステートメント
FORMAT ステートメント
WHERE ステートメント
TITLE ステートメント

詳細

この例では、永久データセットと永久出力形式を使用して、次を含むレポートを作成します。

- オブザベーションごとに 1 行
- レポート全体のデフォルト要約

プログラム

```
libname proclib 'SAS-library';

data grocery;
  input Sector $ Manager $ Department $ Sales @@;
  datalines;
se 1 np1 50   se 1 p1 100   se 1 np2 120   se 1 p2 80
se 2 np1 40   se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60    nw 3 p1 600   nw 3 np2 420   nw 3 p2 30
nw 4 np1 45    nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45    nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53    sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
sw 6 np1 40    sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90    ne 7 p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;

proc format library=proclib;
  value $sctrfmt 'se' = 'Southeast'
                 'ne' = 'Northeast'
                 'nw' = 'Northwest'
                 'sw' = 'Southwest';

  value $mgrfmt '1' = 'Smith'   '2' = 'Jones'
                '3' = 'Reveiz'  '4' = 'Brown'
                '5' = 'Taylor'  '6' = 'Adams'
                '7' = 'Alomar'  '8' = 'Andrews'
                '9' = 'Pelfrey';

  value $deptfmt 'np1' = 'Paper'
                 'np2' = 'Canned'
                 'p1'  = 'Meat/Dairy'
                 'p2'  = 'Produce';
run;

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

proc report data=grocery;

  column manager department sales;

  rbreak after / summarize style=[font_weight=bold];
```

```
  where sector='se';

  format manager $mgrfmt. department $deptfmt.
       sales dollar11.2;

  title 'Sales for the Southeast Sector';
  title2 "for &sysdate";
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

GROCERY データセットを作成します。 GROCERY には、Grocery Mart チェーンの 8 つの店舗の 1 日の売上が含まれています。オブザベーションにはそれぞれ 1 つの店舗の 1 つの部門に対する 1 日の売上データが含まれています。

```
data grocery;
  input Sector $ Manager $ Department $ Sales @@;
  datalines;
se 1 np1 50   se 1 p1 100   se 1 np2 120   se 1 p2 80
se 2 np1 40   se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60    nw 3 p1 600   nw 3 np2 420   nw 3 p2 30
nw 4 np1 45    nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45    nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53    sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
sw 6 np1 40    sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90    ne 7 p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;
```

出力形式$SCTRFMT.、$MGRFMT.、$DEPTFMT.を作成します。 PROC FORMAT は、Sector、Manager、Department に対し永久出力形式を作成します。LIBRARY= オプションは、出力形式が後続の SAS セッションで使用できるように永久保管場所を指定します。これらの出力形式は、このセッション全体にわたって例に使用されます。

```
proc format library=proclib;
  value $sctrfmt 'se' = 'Southeast'
                 'ne' = 'Northeast'
                 'nw' = 'Northwest'
                 'sw' = 'Southwest';

  value $mgrfmt '1' = 'Smith'   '2' = 'Jones'
                '3' = 'Reveiz'  '4' = 'Brown'
                '5' = 'Taylor'  '6' = 'Adams'
                '7' = 'Alomar'  '8' = 'Andrews'
                '9' = 'Pelfrey';

  value $deptfmt 'np1' = 'Paper'
                 'np2' = 'Canned'
                 'p1'  = 'Meat/Dairy'
```

```
                'p2'  = 'Produce';
run;
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 デフォルトでは、REPORT プロシジャはその出力を開いている出力先に送信します。

```
proc report data=grocery;
```

レポート列を指定します。 レポートには、Manager、Department および Sales に対する列が含まれています。これらの変数に対し DEFINE ステートメントがないため、PROC REPORT は表示変数として文字変数(Manager と Department)を、合計統計量の計算に使用される分析変数として数値変数(Sales)を使用します。

```
  column manager department sales;
```

レポートの要約を作成します。 RBREAK ステートメントは、レポートの最後にデフォルト要約を作成します。SUMMARIZE はレポートのすべてのオブザベーションに対する Sales の値を合計します。STYLE=は、要約値を太字にするために使用します。

```
  rbreak after / summarize style=[font_weight=bold];
```

処理するオブザベーションを選択します。 WHERE ステートメントは、南東セクタの店舗のオブザベーションのみレポートに選択します。

```
  where sector='se';
```

レポート列をフォーマットします。 FORMAT ステートメントはレポートで使用する出力形式を割り当てます。FORMAT ステートメントは、データセット変数とのみ使用できます。

```
  format manager $mgrfmt. department $deptfmt.
      sales dollar11.2;
```

タイトルを指定します。 SYSDATE は、SAS ジョブまたは SAS セッション開始時に日付を返す自動マクロ変数です。TITLE2 ステートメントは、マクロ変数が解決するように単一引用符ではなく二重引用符を使用します。

```
  title 'Sales for the Southeast Sector';
  title2 "for &sysdate";
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

HTML 出力

アウトプット 51.2　変数の選択とレポートの要約行の作成

Sales for the Southeast Sector for 03MAY14

Manager	Department	Sales
Smith	Paper	$50.00
Smith	Meat/Dairy	$100.00
Smith	Canned	$120.00
Smith	Produce	$80.00
Jones	Paper	$40.00
Jones	Meat/Dairy	$300.00
Jones	Canned	$220.00
Jones	Produce	$70.00
		$980.00

例 2: レポートでの行の並べ替え

要素:

PROC REPORT ステートメントオプション
COLUMN ステートメント
DEFINE ステートメントオプション
- ANALYSIS
- FORMAT=
- ORDER
- ORDER=
- SUM

BREAK ステートメントオプション
- AFTER
- SUMMARIZE
- STYLE=

LIBNAME ステートメント
OPTIONS ステートメント
WHERE ステートメント
TITLE ステートメント

データセット: GROCERY

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例では、次を行います。

- Manager のフォーマットされた値と Department の内部値によって行をアルファベット順に調整する(非生鮮商品を販売する 2 つの部門の売上が生鮮商品を販売する 2 つの部門の売上の前に来るように)
- デフォルトの列幅および列間のスペースを 制御する
- マネージャーごとに売上のデフォルト要約を作成する
- レポート全体の売上のカスタマイズされた要約を作成する

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

proc report data=grocery;

  column manager department sales;

  define manager / order order=formatted format=$mgrfmt.;
  define department / order order=internal format=$deptfmt.;

  define sales / analysis sum format=dollar7.2;

  break after manager / summarize
                style=[font_style=italic];

  where sector='se';

  title 'Sales for the Southeast Sector';
 run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT はその出力を開いている出力先に送信します。

```
proc report data=grocery;
```

レポート列を指定します。 レポートには、Manager、Department および Sales に対する列が含まれています。

```
 column manager department sales;
```

並べ替え順序変数を定義します。 DEFINE ステートメントで ORDER オプションが指定されているすべての 変数の値によって、レポートの行の順序が決定されます。このレポートで PROC REPORT は最初に Manager の値(これが COLUMN ステートメントの最初の変数のため)、次に Department の値によって行を調整します。ORDER=は変数の並べ替え順序を指定します。このレポートは Manager の出力適用値と Department の内部値(np1、np2、p1、p2)に従って行を調整します。FORMAT=で、レポートに使用する出力形式を指定します。

```
 define manager / order order=formatted format=$mgrfmt.;
 define department / order order=internal format=$deptfmt.;
```

分析変数を定義します。 Sum は、現在の行によって表されるすべてのオブザベーションに対する合計統計量を計算します。このレポートで、行はそれぞれ 1 つのオブザベーションだけを表します。そのため、Sum 統計量は、入力データセットのそのオブザベーションの Sales の値と同じになります。このレポートで Sales を分析変数として使用すると、各グループおよびレポートの最後で値を要約できます。

```
 define sales / analysis sum format=dollar7.2;
```

レポートの要約を作成します。 この BREAK ステートメントは、各マネージャーの最終行の後にデフォルト要約を作成します。PROC REPORT は、Sales が Sum 統計量の計算に使用される分析変数であるため、各マネージャーに対する Sales の値を合計します。

```
 break after manager / summarize
          style=[font_style=italic];
```

処理するオブザベーションを選択します。 WHERE ステートメントは、南東セクタの店舗のオブザベーションのみレポートに選択します。

```
 where sector='se';
```

タイトルを指定します。

```
 title 'Sales for the Southeast Sector';
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

HTML 出力

アウトプット 51.3　レポートでの行の並べ替え

Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	$40.00
	Canned	$220.00
	Meat/Dairy	$300.00
	Produce	$70.00
Jones		*$630.00*
Smith	Paper	$50.00
	Canned	$120.00
	Meat/Dairy	$100.00
	Produce	$80.00
Smith		*$350.00*

例 3: 別名を使用し、同一変数に複数の統計量を求める

要素:

PROC REPORT ステートメントオプション
COLUMN ステートメント
　別名
COMPUTE ステートメント引数
　AFTER
DEFINE ステートメントオプション
　ORDER=
　ANALYSIS
　SUM
　FORMAT=
　MAX
　MIN
　NOPRINT
　列ヘッダーのカスタマイズ
LINE ステートメント
　引用符付きテキスト
　変数値と出力形式
　ブランク行の書き込み

LIBNAME ステートメント
OPTIONS ステートメント
WHERE ステートメント
TITLE ステートメント
自動マクロ変数
SYSDATE

データセット: GROCERY
出力形式: $MGRFMT
出力形式: $DEPTFMT

詳細

このレポートの最後のカスタマイズされた要約には、南東セクタの店舗の全部門に関して売上の最小値と最大値が表示されます。これらの値を決定するため、PROC REPORT はレポートの各行の売上の MIN および MAX 統計量が必要になります。ただし、レポートを簡潔にするため、これらの統計量の表示は非表示になります。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

proc report data=grocery;

  column manager department sales
       sales=salesmin
       sales=salesmax;

  define manager / order
                order=formatted
                format=$mgrfmt.
                'Manager';
  define department    / order
                order=internal
                format=$deptfmt.
                'Department';

  define sales / analysis sum format=dollar7.2 'Sales';

  define salesmin / analysis min noprint;
  define salesmax / analysis max noprint;

compute after;
    line 'Departmental sales ranged from'
         salesmin dollar7.2  " " 'to' " " salesmax dollar7.2;
  endcomp;
```

```
  where sector='se';

  title 'Sales for the Southeast Sector';
  title2 "for &sysdate";
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT はその出力を開いている出力先に送信します。

```
proc report data=grocery;
```

レポート列を指定します。 レポートには Manager と Department の列が含まれています。Sales の 3 列も含まれています。列指定 SALES=SALESMIN と SALES=SALESMAX は、Sales の別名を作成します。これらの別名によって 3 つの列のそれぞれに対し別の Sales の定義を使用できます。

```
column manager department sales
    sales=salesmin
    sales=salesmax;
```

並べ替え順序変数を定義します。 DEFINE ステートメントで ORDER オプションが指定されているすべての 変数の値によって、レポートの行の順序が決定されます。このレポートで PROC REPORT は最初に Manager の値(これが COLUMN ステートメントの最初の変数のため)、次に Department の値によって行を調整します。ORDER=オプションは変数の並べ替え順序を指定します。このレポートは、Manager の値をフォーマットされた値によって調整し、Department の値を内部値(np1、np2、p1、p2)によって調整します。FORMAT=で、レポートに使用する出力形式を指定します。引用符内のテキストは列ヘッダーを指定します。

```
define manager / order
            order=formatted
            format=$mgrfmt.
            'Manager';
define department   / order
            order=internal
            format=$deptfmt.
            'Department';
```

分析変数を定義します。 レポートの行の分析変数の値は、関連付けられている統計量の値(この場合は Sum)で、行によって表されるすべてのオブザベーションに対して計算されます。詳細レポートの各行は 1 つのオブザベーションだけを表します。

そのため、Sum 統計量は、入力データセットのそのオブザベーションの Sales の値と同じになります。

```
define sales / analysis sum format=dollar7.2 'Sales';
```

要約で使用する追加の分析変数を定義します。 これらの DEFINE ステートメントでは、分析変数 Sales の MIN 統計量と MAX 統計量に対し別の列を作成するため COLUMN からの別名を使用します。NOPRINT はこれらの統計量の印刷を行いません。PROC REPORT は列のこれらの値を印刷しませんが、これらの値にアクセスして要約に印刷できます。

```
define salesmin / analysis min noprint;
define salesmax / analysis max noprint;
```

カスタマイズした要約を作成します。 この COMPUTE ステートメントは、レポートの最後に実行する計算ブロックを開始します。LINE ステートメントは引用符で囲まれたテキスト、Salesmin の値(DOLLAR7.2 出力形式)、引用符で囲まれたスペース、引用符で囲まれたテキスト"to"、スペース、Salesmax(DOLLAR7.2 出力形式)を書き込みます .(プログラムは、変数をその別名によって参照する必要があります)ENDCOMP ステートメントは計算ブロックを終了します。

```
compute after;
   line 'Departmental sales ranged from'
        salesmin dollar7.2  " " 'to' " " salesmax dollar7.2;
endcomp;
```

処理するオブザベーションを選択します。 WHERE ステートメントは、南東セクタの店舗のオブザベーションのみレポートに選択します。

```
where sector='se';
```

タイトルを指定します。 SYSDATE は、SAS ジョブまたは SAS セッション開始時に日付を返す自動マクロ変数です。TITLE2 ステートメントは、マクロ変数が解決するように単一引用符ではなく二重引用符を使用します。

```
title 'Sales for the Southeast Sector';
title2 "for &sysdate";
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

HTML 出力

アウトプット 51.4　別名を使用し、同一変数に複数の統計量を求める

Sales for the Southeast Sector for 27MAY14

Manager	Department	Sales
Jones	Paper	$40.00
	Canned	$220.00
	Meat/Dairy	$300.00
	Produce	$70.00
Smith	Paper	$50.00
	Canned	$120.00
	Meat/Dairy	$100.00
	Produce	$80.00
Departmental sales ranged from $40.00 to $300.00		

例 4: 1 つの変数に複数の統計量を表示する

要素:
PROC REPORT ステートメントオプション
COLUMN ステートメント
　積み上げ変数の統計量の指定
DEFINE ステートメントオプション
　FORMAT=
　GROUP
LIBNAME ステートメント
OPTIONS ステートメント
TITLE ステートメント

データセット: GROCERY

出力形式: $MGRFMT

詳細

この例のレポートでは、各マネージャーの店舗の売上に対する 6 つの統計量を表示します。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

proc report data=grocery;

column sector manager (Sum Min Max Range Mean Std),sales;

  define manager / group format=$mgrfmt.;
  define sector / group format=$sctrfmt.;
  define sales / format=dollar11.2 ;

  title 'Sales Statistics for All Sectors';
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。

```
proc report data=grocery;
```

レポート列を指定します。 この COLUMN ステートメントは、Sector、Manager、および Sales と関連付けられている 6 つの統計量のそれぞれに対する列を作成します。

```
column sector manager (Sum Min Max Range Mean Std),sales;
```

グループ変数と分析変数を定義します。 このレポートでは、Sector と Manager がグループ変数です。レポートの各詳細行に、グループ変数の値が同じすべてのオブザベーションに関する情報がまとめられます。FORMAT=で、レポートに使用する出力形式を指定します。

```
  define manager / group format=$mgrfmt.;
  define sector / group format=$sctrfmt.;
  define sales / format=dollar11.2 ;
```

タイトルを指定します。

```
  title 'Sales Statistics for All Sectors';
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

HTML 出力

アウトプット 51.5　1 つの変数に複数の統計量を表示する

Sales Statistics for All Sectors

		Sum	Min	Max	Range	Mean	Std
Sector	Manager	Sales	Sales	Sales	Sales	Sales	Sales
Northeast	Alomar	$786.00	$86.00	$420.00	$334.00	$196.50	$156.57
	Andrews	$1,045.00	$125.00	$420.00	$295.00	$261.25	$127.83
Northwest	Brown	$598.00	$45.00	$250.00	$205.00	$149.50	$105.44
	Pelfrey	$746.00	$45.00	$420.00	$375.00	$186.50	$170.39
	Reveiz	$1,110.00	$30.00	$600.00	$570.00	$277.50	$278.61
Southeast	Jones	$630.00	$40.00	$300.00	$260.00	$157.50	$123.39
	Smith	$350.00	$50.00	$120.00	$70.00	$87.50	$29.86
Southwest	Adams	$695.00	$40.00	$350.00	$310.00	$173.75	$141.86
	Taylor	$353.00	$50.00	$130.00	$80.00	$88.25	$42.65

例 5: 複数のオブザベーションをレポートの 1 行にまとめる

要素:

- PROC REPORT ステートメントオプション
- COLUMN ステートメント
- DEFINE ステートメントオプション
  - ANALYSIS
  - GROUP
  - SUM
  - FORMAT=
- BREAK ステートメントオプション
  - AFTER
  - SUMMARIZE
  - SUPPRESS
- CALL DEFINE ステートメント
- COMPUTE ステートメント引数
  - AFTER
- CALL DEFINE ステートメント

LINE ステートメント
引用符付きテキスト
変数値
LIBNAME ステートメント
OPTIONS ステートメント
TITLE ステートメント
WHERE ステートメント

データセット: GROCERY
出力形式: $MGRFMT
出力形式: $DEPTFMT

詳細

この例では、次を行う要約レポートを作成します。

- Sector と Manager の各組み合わせに関する情報をレポートの 1 行にまとめる
- セクタごとの売上のデフォルト要約を含む
- すべてのセクタの売上のカスタマイズされた要約を含む
- 詳細行と要約行で異なる売上の出力形式を使用する
- カスタマイズされた列ヘッダーを使用する

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

proc report data=grocery;

  column sector manager sales;

  define sector / group format=$sctrfmt.'Sector';
  define manager / group format=$mgrfmt.'Manager';
  define sales / analysis sum format=comma10.2 'Sales';

  break after sector / summarize
                       style=[font_style=italic]
                       suppress;

  compute after;
    line 'Combined sales for the northern sectors were '
         sales.sum dollar9.2 '.';
  endcomp;

  compute sales;
    if _break_ ne ' ' then
```

```
    call define(_col_,"format","dollar11.2");
  endcomp;

  where sector contains 'n';

  title 'Sales Figures for Northern Sectors';
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT はその出力を開いている出力先に送信します。

```
proc report data=grocery;
```

レポート列を指定します。 レポートには、Sector、Manager、Sales の列が含まれます。

```
  column sector manager sales;
```

グループ変数と分析変数を定義します。 このレポートでは、Sector と Manager がグループ変数です。Sales は、Sum 統計量の計算に使用される分析変数です。詳細行はそれぞれ、すべてのグループ変数に対するフォーマットされた値の一意の組み合わせを持つ一連のオブザベーションを表します。各詳細行の Sales の値は、グループのすべてのオブザベーションに 対する Sales の合計です。FORMAT=はレポートで使用する出力形式を指定します。DEFINE ステートメントの引用符内のテキストは、列ヘッダーを指定します。

```
define sector / group format=$sctrfmt.'Sector';
define manager / group format=$mgrfmt.'Manager';
define sales / analysis sum format=comma10.2 'Sales';
```

レポートの要約を作成します。 この BREAK ステートメントは、各セクタの最終行の後にデフォルト要約を作成します。SUMMARIZE は Sales の値を要約行に書き込みます。要約値は、セクターごとの値です。セクタの売上では、すべてのマネージャーの売上を合計します。SUPPRESS は PROC REPORT が要約行に Sector 値を表示するのを防ぎます。要約行は、イタリック体になります。

```
break after sector / summarize
                     style=[font_style=italic]
                     suppress;
```

カスタマイズした要約を作成します。 この計算ブロックは、レポートの最後にカスタマイズされた要約を作成します。LINE ステートメントは引用符付きテキストと

Sales.sum の値(DOLLAR9.2 出力形式)を要約に書き込みます。ENDCOMP ステートメントは計算ブロックを終了する必要があります。

```
compute after;
  line 'Combined sales for the northern sectors were '
       sales.sum dollar9.2 '.';
endcomp;
```

要約行の出力形式を指定します。 詳細行で、PROC REPORT は定義で指定された出力形式(COMMA10.2)を持つ Sales の値を表示します。計算ブロックは、要約行の現在の列で使用する代替出力形式を指定します。要約行は、_BREAK_に対するブランク以外の値として識別されます。

```
compute sales;
  if _break_ ne ' ' then
  call define(_col_,"format","dollar11.2");
endcomp;
```

処理するオブザベーションを選択します。 WHERE ステートメントは、北東セクタと北西セクタの店舗のオブザベーションだけをレポートに選択します。TITLE ステートメントはタイトルを指定します。

```
where sector contains 'n';
```

タイトルを指定します。

```
  title 'Sales Figures for Northern Sectors';
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

HTML 出力

アウトプット 51.6　複数のオブザベーションをレポートの 1 行にまとめる

Sales Figures for Northern Sectors

Sector	Manager	Sales
Northeast	Alomar	786.00
	Andrews	1,045.00
		$1,831.00
Northwest	Brown	598.00
	Pelfrey	746.00
	Reveiz	1,110.00
		$2,454.00

Combined sales for the northern sectors were $4,285.00

例 6: 変数の値ごとに列を作成する

要素:

PROC REPORT ステートメントオプション
- SPLIT=

COLUMN ステートメント
- 積み上げ変数

DEFINE ステートメントオプション
- GROUP
- ACROSS
- ANALYSIS
- COMPUTED
- SUM
- FORMAT=

COMPUTE ステートメント引数
- 計算変数(*report-item*)
- AFTER

LINE ステートメント

ENDCOMP ステートメント

LIBNAME ステートメント

OPTIONS ステートメント

TITLE ステートメント

WHERE ステートメント

データセット: GROCERY

出力形式: $SCTRFMT

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例のレポートでは、次を行います。

- 複数のオブザベーションを 1 行にまとめる
- レポートに選択される部門(生鮮品目を販売する部門)の値ごとの列を含む
- 入力データセットにない変数を含む
- 一部ブランクを含むカスタマイズされた列ヘッダーを使用する
- カスタマイズした要約で変数値を使用する

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

proc report data=grocery split='*';

  column sector manager department,sales perish;

  define sector / group format=$sctrfmt. 'Sector';
  define manager / group format=$mgrfmt. 'Manager* ';

  define department / across format=$deptfmt. 'Department';

  define sales / analysis sum format=dollar11.2 ' ';

  define perish / computed format=dollar11.2
            'Perishable*Total';

  compute perish;
    perish=sum(_c3_, _c4_);
  endcomp;

  compute after;
        line  'Combined sales for meat and dairy : '
        _c3_ dollar11.2 '';
    line  'Combined sales for produce : '
        _c4_ dollar11.2 '';
    line  'Combined sales for all perishables: '
        _c5_ dollar11.2 '';
      endcomp;

  where sector contains 'n'
      and (department='p1' or department='p2');

  title 'Sales Figures for Perishables in Northern Sectors';
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT はその出力を開いている出力先に送信します。デフォルトの区切り文字(/)は部門名の一部であるため、SPLIT=は区切り文字をアスタリスク(*)として定義します。

```
proc report data=grocery split='*';
```

レポート列を指定します。 Department と Sales は定義する列のコンテンツを一括で決定できるように、COLUMN ステートメントでカンマで区切られます。項目はそれぞれヘッダーを生成しますが、Sales のヘッダーは定義でブランクに設定されます。Sales が分析変数であるため、その値はこれらの 2 つの変数によって作成されたセルに入力されます。

```
column sector manager department,sales perish;

define sector / group format=$sctrfmt. 'Sector';
define manager / group format=$mgrfmt. 'Manager* ';
```

列変数を定義します。 PROC REPORT は、列変数 Department の各フォーマットされた値に対し列と列ヘッダーを作成します。PROC REPORT はこれらの値によって列を並べ替えます。PROC REPORT もこれらのすべての列にわたる列ヘッダーを生成します。Department に対する DEFINE ステートメントの引用符付きテキストは、このヘッダーをカスタマイズします。

```
define department / across format=$deptfmt. 'Department';
```

分析変数を定義します。 Sales は、Sum 統計量の計算に使用される分析変数です。いずれの場合も、Sales の値は 1 つのグループの 1 つの部門のすべてのオブザベーションに対する Sales の合計です(この場合、値は 1 つのオブザベーションを表します)。

```
define sales / analysis sum format=dollar11.2 ' ';
```

計算変数を定義します。 COMPUTED オプションは、PROC REPORT が Perish の値を計算する必要があることを示します。Perish と関連付けられている計算ブロックの変数の値を計算します。

```
define perish / computed format=dollar11.2
        'Perishable*Total';
```

計算変数の値を計算します。 この計算ブロックは、肉/乳製品部門と農産物部門の値からの Perish の値を計算します。変数 Sales と Department がこれらの列を一括して定義するため、PROC REPORT への値を名前によって識別することはできません。そのため、割当ステートメントでは列番号を使用して、使用する値を明確に指定します。PROC REPORT は Perish の値が必要となるたび、レポートのその行の 3 番目と 4 番目の列の値を合計します。

```
compute perish;
  perish=sum(_c3_, _c4_);
endcomp;
```

カスタマイズした要約を作成します。 この計算ブロックは、レポートの最後にカスタマイズされた要約を作成します。LINE ステートメントは、指定した列に引用符付きのテキストと変数_C3_、_C4_および_C5_の値(DOLLAR11.2 形式)を書き込みます。

```
compute after;
      line  'Combined sales for meat and dairy : '
      _c3_ dollar11.2 '';
  line  'Combined sales for produce : '
      _c4_ dollar11.2 '';
  line  'Combined sales for all perishables: '
      _c5_ dollar11.2 '';
    endcomp;

where sector contains 'n'
```

```
    and (department='p1' or department='p2');
```

タイトルを指定します。

```
  title 'Sales Figures for Perishables in Northern Sectors';
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

HTML 出力

アウトプット 51.7　変数の値ごとに列を作成する

Sales Figures for Perishables in Northern Sectors

		Department		
		Meat/Dairy	Produce	
Sector	Manager			Perishable Total
Northeast	Alomar	$190.00	$86.00	$276.00
	Andrews	$300.00	$125.00	$425.00
Northwest	Brown	$250.00	$73.00	$323.00
	Pelfrey	$205.00	$76.00	$281.00
	Reveiz	$600.00	$30.00	$630.00

Combined sales for meat and dairy: $1,545.00
Combined sales for produce: $390.00
Combined sales for all perishables: $1,935.00

例 7: ページごとにカスタマイズされた要約を書き込む

要素:

PROC REPORT ステートメントオプション
COLUMN ステートメント
DEFINE ステートメントオプション
- NOPRINT
- GROUP
- COMPUTED
- ANALYSIS

COMPUTE ステートメント引数

BEFORE
AFTER
BEFORE _PAGE_
STYLE=
TEXT=

BREAK ステートメントオプション
PAGE
SUMMARIZE
AFTER
STYLE=

LINE ステートメント

LIBNAME ステートメント

OPTIONS ステートメント

TITLE ステートメント

データセット: GROCERY

出力形式: $SCTRFMT

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例のレポートでは、各店舗に対する 1 日の売上の記録を表示します。行は、1 つの店舗に関するすべての情報がまとめられ、店舗ごとの情報が新しいページで開始されるように調整されます。一部の変数が列に表示されます。その他の変数は、セクタと店舗のマネージャーを識別するページヘッダーにのみ表示されます。

各ページの上部に表示されるヘッダーは、COMPUTE ステートメントの_PAGE_引数によって作成されます。

Profit は、Sales と Department の値に基づく計算変数です。

ページの下部に表示されるテキストは、店舗の Sales の合計によって異なります。ここでは、レポートの最初の 2 ページだけを表示します。

プログラム

```
libname proclib 'SAS-library';
options fmtsearch=(proclib);
ods html file="myhtmlfile.html";
proc report data=grocery;
  title 'Sales for Individual Stores';
  column sector manager department sales Profit;
```

```
define sector / group noprint;
define manager / group noprint;
define profit / computed format=dollar11.2;
define sales / analysis sum format=dollar11.2;
define department / group format=$deptfmt.;

compute profit;
  if department='np1' or department='np2'
    then profit=0.4*sales.sum;
  else profit=0.25*sales.sum;
endcomp;

 compute before _page_ / style={just=left};
  line sector $sctrfmt. ' Sector';
  line 'Store managed by ' manager $mgrfmt.;
    endcomp;

break after manager / summarize style=[font_style=italic] page;

compute after manager;

  length text $ 35;

  if sales.sum lt 500 then
    text='Sales are below the target region.';
  else if sales.sum ge 500 and sales.sum lt 1000 then
    text='Sales are in the target region.';
  else if sales.sum ge 1000 then
    text='Sales exceeded goal!';
  line text $35.;
endcomp;
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT はその出力を開いている出力先に送信します。

```
proc report data=grocery;
```

タイトルを指定します。

```
 title 'Sales for Individual Stores';
```

レポート列を指定します。 レポートには、Sector、Manager、Department、Sales、Profit の列が含まれていますが、NOPRINT オプションは、Sector と Manager の列の印刷を行いません。ページヘッダー(後でプログラムで作成)には、それらの値が含まれています。これらの変数値をページヘッダーに入力するには、Sector と Manager が COLUMN ステートメントにある必要があります。

```
column sector manager department sales Profit;
```

グループ変数、計算変数、分析変数を定義します。 このレポートでは、Sector、Manager、Department はグループ変数です。レポートの各詳細行に、グループ変数の値が同じすべてのオブザベーションに関する情報がまとめられます。Profit は値がプログラムの次のセクションで計算される計算変数です。FORMAT=で、レポートに使用する出力形式を指定します。

```
define sector / group noprint;
define manager / group noprint;
define profit / computed format=dollar11.2;
define sales / analysis sum format=dollar11.2;
define department / group format=$deptfmt.;
```

計算変数を計算します。 利益は Sales のパーセントとして計算されます。非生鮮品目の場合、利益は販売価格の 40%です。生鮮品目の場合、利益は 25%です。計算ブロックでは、計算する変数と統計量の両方を識別する複合名(Sales.sum)を持つ変数 Sales を参照する必要があります。

```
compute profit;
  if department='np1' or department='np2'
    then profit=0.4*sales.sum;
  else profit=0.25*sales.sum;
endcomp;
```

カスタマイズされたページヘッダーを作成します。 この計算ブロックは、PROC REPORT がタイトルを書き込んだ後に各ページの上部で実行します。現在のマネージャーの店舗に対するページヘッダーを書き込みます。LEFT オプションは LINE ステートメントのテキストを左寄せにします。LINE ステートメントは、変数名の直後に指定される出力形式を持つ変数値を書き込みます。

```
compute before _page_ / style={just=left};
  line sector $sctrfmt. ' Sector';
  line 'Store managed by ' manager $mgrfmt.;
    endcomp;
```

レポートの要約を作成します。 この BREAK ステートメントは、各マネージャーの最終行の後にデフォルト要約を作成します。SUMMARIZE は Sales の値(分析変数または計算変数のみ)を要約行に書き込みます。PAGE オプションは、前の計算ブロックで作成されるページヘッダーが常に適切なマネージャーに関連するように、各デフォルト要約の後に新しいページを開始します。STYLE=オプションで、要約情報をイタリック体に設定します。

```
break after manager / summarize style=[font_style=italic] page;
```

カスタマイズした要約を作成します。 この計算ブロックは、各マネージャーに対する最後の詳細行の後に表示される条件テキストをカスタマイズされた要約に置きます。

```
compute after manager;
```

カスタマイズされた要約テキストの長さを指定します。 LENGTH ステートメントは長さ 35 を一時変数 TEXT に割り当てます。こうした特殊な場合、最長バージョン

が最初の IF/THEN ステートメントで表示されるため、LENGTH ステートメントは不要です。ただし LENGTH ステートメントにより、条件ステートメントの順序が変わっても、TEXT は最長バージョンを含めるのに十分な長さになります。

```
length text $ 35;
```

カスタマイズされた要約テキストに対し条件付きロジックを指定します。 LINE ステートメントは条件ステートメント(IF-THEN、IF-THEN/ELSE、SELECT)で使用できません。LINE ステートメントは、PROC REPORT が計算ブロックのその他すべてのステートメントを実行するまで実行されないためです。これらの IF-THEN/ELSE ステートメントは、要約行の Sales.sum の値に基づいて値を TEXT に割り当てます。LINE は値が発生する変数を書き込みます

```
  if sales.sum lt 500 then
    text='Sales are below the target region.';
  else if sales.sum ge 500 and sales.sum lt 1000 then
    text='Sales are in the target region.';
  else if sales.sum ge 1000 then
    text='Sales exceeded goal!';
  line text $35.;
endcomp;
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

HTML 出力

アウトプット 51.8　ページごとにカスタマイズされた要約を書き込む

Sales for Individual Stores

Northeast Sector
Store managed by Alomar

Department	Sales	Profit
Canned	$420.00	$168.00
Meat/Dairy	$190.00	$47.50
Paper	$90.00	$36.00
Produce	$86.00	$21.50
	$786.00	*$196.50*

Sales are in the target region.

Sales for Individual Stores

Northeast Sector
Store managed by Andrews

Department	Sales	Profit
Canned	$420.00	$168.00
Meat/Dairy	$300.00	$75.00
Paper	$200.00	$80.00
Produce	$125.00	$31.25
	$1,045.00	*$261.25*

Sales exceeded goal!

例 8: レポートに計算されたパーセント列を表示する

要素:

PROC REPORT ステートメントオプション
COLUMN ステートメント引数
- PCTSUM
- SUM

DEFINE ステートメントオプション

COMPUTED
STYLE(COLUMN)=
GROUP

COMPUTE ステートメントオプション
CHAR
LENGTH=

RBREAK ステートメントオプション
SUMMARIZE
STYLE=

LIBNAME ステートメント

OPTIONS ステートメント

TITLE ステートメント

データセット: GROCERY

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例の要約レポートでは、店舗ごとの合計売上と、これらの売上の全店舗の売上に対するパーセントを表示します。これらの列にはそれぞれ独自のヘッダーがあります。また、1 つのヘッダーはすべての列にわたります。

レポートには、計算文字変数 COMMENT が含まれます。これは、売上率が非常に高い店舗のフラグをオンにします。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

proc report data=grocery;
  title;

  column ('Individual Store Sales as a Percent of All Sales'
        sector manager sales,(sum pctsum) comment);

  define manager / group
                format=$mgrfmt.;
  define sector / group
               format=$sctrfmt.;
  define sales / format=dollar11.2
              '';
  define sum / format=dollar9.2
            'Total Sales';

  define pctsum / 'Percent of Sales' format=percent6.;
```

```
  define comment / computed style(column)=[cellwidth=2.5in];
  compute comment / char length=40;
    if sales.pctsum gt .15 and _break_ = ' '
    then comment='Sales substantially above expectations.';
    else comment=' ';
  endcomp;
  rbreak after / summarize style=[font_style=italic];
run;
ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 データセットとその他のオプションを指定します。

```
proc report data=grocery;
  title;
```

レポート列を指定します。 COLUMN ステートメントは、引用符内のテキストをヘッダーとして使用します。ヘッダーは、ヘッダーを含む一対のかっこにすべて含まれているため、レポートの全列にわたります。COLUMN ステートメントは、2 つの統計量 Sum と Pctsum を売上と関連付けます。Sum 統計量は、レポートの行に含まれるすべてのオブザベーションに対する Sales の値を合計します。Pctsum 統計量は、集計する Sales のレポートのすべてのオブザベーションに対するパーセントを示します。

```
column ('Individual Store Sales as a Percent of All Sales'
      sector manager sales,(sum pctsum) comment);
```

グループ列と分析列を定義します。 このレポートでは、Sector と Manager がグループ変数です。詳細行はそれぞれ、すべてのグループ変数に対するフォーマットされた値の一意の組み合わせを持つ一連のオブザベーションを表します。Sales はデフォルトで、Sum 統計量の計算に使用される分析変数です。ただし、統計量は COLUMN ステートメントで Sales と関連付けられているため、これらの統計量はデフォルトより優先されます。FORMAT=で、レポートに使用する出力形式を指定します。引用符の間のテキストは列ヘッダーを指定します。

```
define manager / group
            format=$mgrfmt.;
define sector / group
          format=$sctrfmt.;
```

```
define sales / format=dollar11.2
               '';
define sum / format=dollar9.2
             'Total Sales';
```

パーセント列と計算列を定義します。 Pctsum の DEFINE ステートメントで、列のヘッダーと出力形式を指定します。PERCENT.出力形式は、Pctsum の値を小数ではなく、パーセントで表示します。COMMENT の DEFINE ステートメントは、計算変数を定義してそれを列に割り当てます。

```
define pctsum / 'Percent of Sales' format=percent6.;
define comment / computed style(column)=[cellwidth=2.5in];
```

計算変数を計算します。 COMPUTE ステートメントのオプションは、COMMENT を長さ 40 の文字変数として定義します。

```
compute comment / char length=40;
```

計算変数に対し条件付きロジックを指定します。 売上が全店舗の売上の 15%を超えたすべての店舗について、この計算ブロックは **Sales substantially above expectations** というコメントを作成します。レポートの要約行では、Pctsum の値は 100 となります。ただし、この行を例外的な売上があるとしてフラグ設定することは適切ではありません。自動変数_BREAK_によって詳細行と要約行が区別されます。詳細行では、_BREAK_の値はブランクです。THEN ステートメントは Pctsum の値が 0.15 を超える詳細行でのみ実行します。

```
  if sales.pctsum gt .15 and _break_ = ' '
  then comment='Sales substantially above expectations.';
  else comment=' ';
endcomp;
```

レポート要約を作成します。 この RBREAK ステートメントは、レポートの最後にデフォルト要約を作成します。SUMMARIZE は要約行に Sales.sum と Sales.pctsum の値を書き込みます。STYLE=で、要約行をイタリック体に設定します。

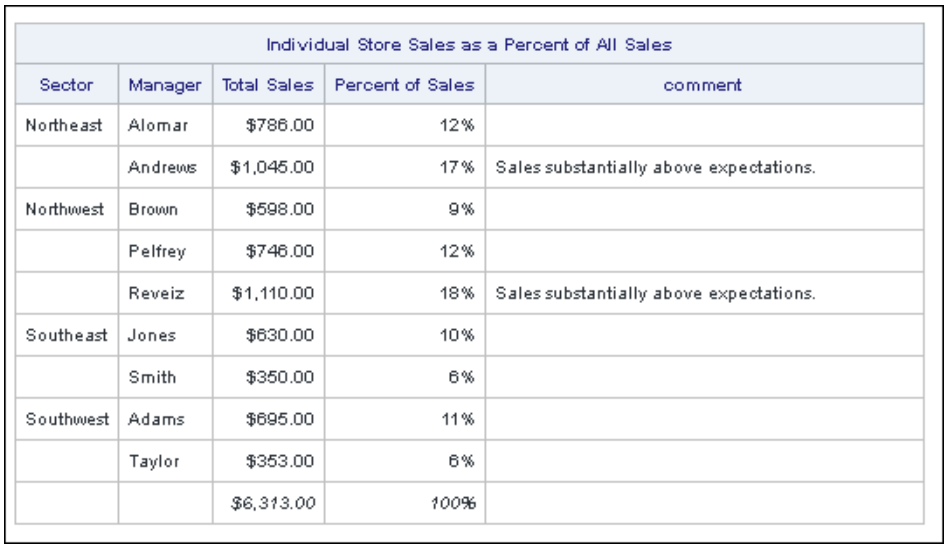
```
  rbreak after / summarize style=[font_style=italic];
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

HTML 出力

アウトプット 51.9　パーセントの計算



Individual Store Sales as a Percent of All Sales

Sector	Manager	Total Sales	Percent of Sales	comment
Northeast	Alomar	$786.00	12%	
	Andrews	$1,045.00	17%	Sales substantially above expectations.
Northwest	Brown	$598.00	9%	
	Pelfrey	$746.00	12%	
	Reveiz	$1,110.00	18%	Sales substantially above expectations.
Southeast	Jones	$630.00	10%	
	Smith	$350.00	6%	
Southwest	Adams	$695.00	11%	
	Taylor	$353.00	6%	
		$6,313.00	*100%*	

例 9: PROC REPORT での欠損値の処理法

要素:

PROC REPORT ステートメントオプション
- MISSING

COLUMN ステートメント
- N 統計量

DEFINE ステートメントオプション
- STYLE(COLUMN)=
- GROUP

RBREAK ステートメントオプション
- AFTER
- SUMMARIZE
- STYLE=

LIBNAME ステートメント

OPTIONS ステートメント

TITLE ステートメント

出力形式: $MGRFMT

詳細

この例では、MISSING オプションを使用した場合と使用しない場合の、PROC REPORT でのグループ(または順序、段組み)変数の欠損値の処理方法を示します。行

ごとに N の値を比較し、レポートの最後のデフォルト要約の合計を比較すると、レポートの違いは明らかです。

プログラム – 欠損値のないデータセット

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

data grocmiss;
  input Sector $ Manager $ Department $ Sales @@;
datalines;
se 1 np1 50   .  1 p1 100   se . np2 120   se 1 p2 80
se 2 np1 40   se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60   nw 3 p1 600   .  3 np2 420   nw 3 p2 30
nw 4 np1 45   nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45   nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53   sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
.  . np1 40   sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90   ne . p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;

ods html file="myhtmlfile.html";

proc report data=grocmiss;

  column sector manager N sales;

  define sector / group format=$sctrfmt.;
  define manager / group format=$mgrfmt.;
  define sales / format=dollar9.2;

  rbreak after / summarize style=[fontstyle=italic];

  title 'Summary Report for All Sectors and Managers';
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

GROCMISS データセットを作成します。 GROCMISS は、Sector、Manager、またはその両方の欠損値を含む一部のオブザベーションを含むという以外は、GROCERY と同じです。

```
data grocmiss;
```

```
  input Sector $ Manager $ Department $ Sales @@;
datalines;
se 1 np1 50   . 1 p1 100   se . np2 120   se 1 p2 80
se 2 np1 40   se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60   nw 3 p1 600   . 3 np2 420   nw 3 p2 30
nw 4 np1 45   nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45   nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53   sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
. . np1 40   sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90   ne . p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 デフォルトでは、PROC REPORT はその出力を開いている出力先に送信します。

```
proc report data=grocmiss;
```

レポート列を指定します。 レポートには、Sector、Manager、N 統計量、Sales の列が含まれています。

```
  column sector manager N sales;
```

グループ変数と分析変数を定義します。 このレポートでは、Sector と Manager がグループ変数です。Sales はデフォルトで、Sum 統計量の計算に使用される分析変数です。詳細行はそれぞれ、すべてのグループ変数に対するフォーマットされた値の一意の組み合わせを持つ一連のオブザベーションを表します。各詳細行の Sales の値は、グループのすべてのオブザベーションに 対する Sales の合計です。この PROC REPORT ステップで、プロシジャにはグループ変数の値が欠損値であるオブザベーションは含まれません。FORMAT=は、レポートで使用する出力形式を指定します。

```
  define sector / group format=$sctrfmt.;
  define manager / group format=$mgrfmt.;
  define sales / format=dollar9.2;
```

レポートの要約を作成します。 この RBREAK ステートメントは、レポートの最後にデフォルト要約を作成します。SUMMARIZE は要約行に N と Sales.sum の値を書き込みます。STYLE=で、要約行をイタリック体に設定します。

```
  rbreak after / summarize style=[fontstyle=italic];
```

タイトルを指定します。

```
  title 'Summary Report for All Sectors and Managers';
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

アウトプット 51.10　欠損値を含まない出力

Summary Report for All Sectors and Managers

Sector	Manager	N	Sales
Northeast	Alomar	3	$596.00
	Andrews	4	$1,045.00
Northwest	Brown	4	$598.00
	Pelfrey	4	$746.00
	Reveiz	3	$690.00
Southeast	Jones	4	$630.00
	Smith	2	$130.00
Southwest	Adams	3	$655.00
	Taylor	4	$353.00
		31	*$5,443.00*

プログラム – 欠損値があるデータセット

```
ods html file="myhtmlfile.html";

proc report data=grocmiss missing;
  column sector manager N sales;
  define sector / group format=$sctrfmt.;
  define manager / group format=$mgrfmt.;
  define sales / format=dollar9.2;
  rbreak after / summarize style=[fontstyle=italic];
run;

ods html close;
```

プログラムの説明

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

欠損値を含めます。 2 番目の PROC REPORT ステップの MISSING オプションには、グループ変数の値が欠損値のオブザベーションが含まれます。

```
proc report data=grocmiss missing;
  column sector manager N sales;
```

```
  define sector / group format=$sctrfmt.;
  define manager / group format=$mgrfmt.;
  define sales / format=dollar9.2;
  rbreak after / summarize style=[fontstyle=italic];
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

アウトプット 51.11　欠損値を含む出力

Summary Report for All Sectors and Managers

Sector	Manager	N	Sales
		1	$40.00
	Reveiz	1	$420.00
	Smith	1	$100.00
Northeast		1	$190.00
	Alomar	3	$596.00
	Andrews	4	$1,045.00
Northwest	Brown	4	$598.00
	Pelfrey	4	$746.00
	Reveiz	3	$690.00
Southeast		1	$120.00
	Jones	4	$630.00
	Smith	2	$130.00
Southwest	Adams	3	$655.00
	Taylor	4	$353.00
		36	*$6,313.00*

例 10: 出力データセットの作成と計算変数の保存

要素:　PROC REPORT ステートメントオプション
　　OUT=
COLUMN

DEFINE ステートメントオプション
    GROUP
    COMPUTED
    ANALYSIS
    SUM
プログラム編集
データセットオプション
    WHERE=
LIBNAME ステートメント
OPTIONS ステートメント
TITLE

データセット: GROCERY

出力形式: $MGRFMT

詳細

この例では、出力データセットの作成時に WHERE 処理を使用します。この方法を使用すると、複数のオブザベーションを 1 つの行にまとめた後に WHERE 処理を実行できます。

注: この方法が必要である理由は、分析変数の結果に関しては部分集合を抽出できないためです。PROC REPORT によって計算された値に関しては部分集合を抽出できません。

最初の PROC REPORT ステップでは、各行が 1 人のマネージャーの入力データセットの全オブザベーションを表すレポートを作成します。2 番目の PROC REPORT ステップでは、出力データセットからのレポートを作成します。

出力データセットを作成するためのプログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

proc report data=grocery
        out=profit( where=(sales gt 1000 and _break_='') );
  column manager sales manager_pct;

  define manager / group;
  define manager_pct / computed;
  compute before;
    total_sales = sales.sum;
  endcomp;
  compute manager_pct;
    manager_pct = sales.sum /total_sales;
```

```
  endcomp;
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションと列を指定します。 デフォルトでは、PROC REPORT はその出力を開いている出力先に送信します。OUT=は出力データセット PROFIT を作成します。出力データセットには、レポートの各列(Manager、Sales および計算列 manager_pct)の変数のほか、この例では使用されていない変数_BREAK_の変数が含まれています。データセット内の各オブザベーションは、レポートの行を表します。Manager がグループ変数、Sales が Sum 統計量の計算に使用される分析変数であるため、レポートの各行(および出力データセット内の各オブザベーション)は、入力データセットからの複数のオブザベーションを表します。特に、出力データセットの Sales の各値は、マネージャーに対する Sales のすべての値の合計です。OUT=オプションの WHERE=データセットオプションは、PROC REPORT が出力データセットを作成するときにこれらの行をフィルターします。売上が$1,000 を超えるオブザベーションだけが出力データセットのオブザベーションになります。

```
proc report data=grocery
        out=profit( where=(sales gt 1000 and _break_='') );
  column manager sales manager_pct;
```

グループ変数および分析変数を定義して、売上のパーセント値を計算します。 全体の合計額が一時変数、total_sales に配置され、売上のパーセント値が計算されます。

```
  define manager / group;
  define manager_pct / computed;
  compute before;
    total_sales = sales.sum;
  endcomp;
  compute manager_pct;
    manager_pct = sales.sum /total_sales;
  endcomp;
run;
```

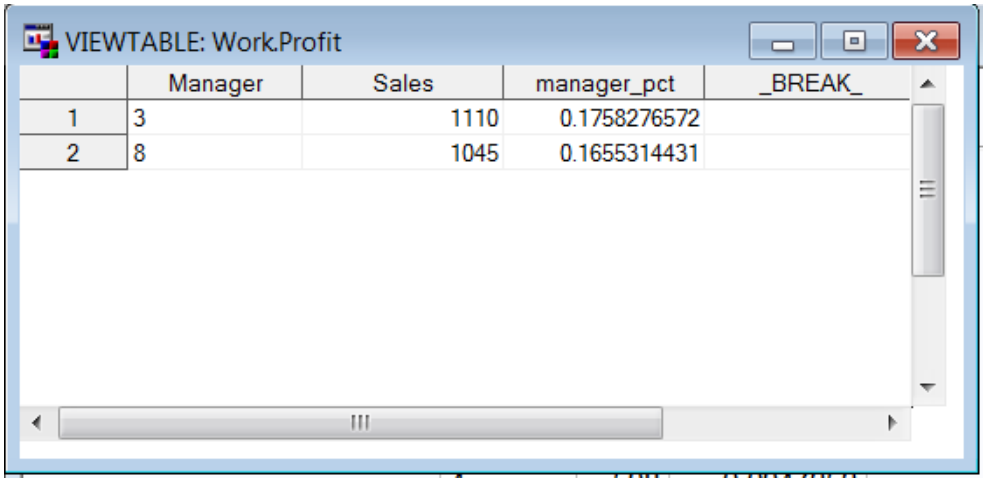
ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

PROC REPORT によって作成されるデータセットを次に示します。このデータセットは、2 番目の PROC REPORT ステップで入力セットとして使用されます。

アウトプット 51.12 出力データセット PROFIT



	Manager	Sales	manager_pct	_BREAK_
1	3	1110	0.1758276572	
2	8	1045	0.1655314431	

出力データセットを使用するプログラム

```
ods html file="myhtmlfile.html";

proc report data=profit;
  column manager sales manager_pct;
  define manager / group format=$mgrfmt.;
  define sales / analysis sum format=dollar11.2;
  define manager_pct /'Percent of Total Sales' format=percent8.2;
  title 'Managers with Daily Sales';
  title2 'of over';
  title3 'One Thousand Dollars';
run;

ods html close;
```

プログラムの説明

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションと列を指定し、グループ列と分析列を定義し、タイトルを指定します。 DATA=は最初の PROC REPORT ステップからの出力データセットをこのレポートの入力データセットとして指定します。TITLE ステートメントはレポートのタイトルを指定します。

```
proc report data=profit;
  column manager sales manager_pct;
```

```
  define manager / group format=$mgrfmt.;
  define sales / analysis sum format=dollar11.2;
  define manager_pct /'Percent of Total Sales' format=percent8.2;
  title 'Managers with Daily Sales';
  title2 'of over';
  title3 'One Thousand Dollars';
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

アウトプット 51.13　出力データセットに基づくレポート

Managers with Daily Sales of over One Thousand Dollars

Manager	Sales	Percent of Total Sales
Andrews	$1,045.00	16.55%
Reveiz	$1,110.00	17.58%

例 11: 出力形式を使用してグループを作成する

要素:

PROC REPORT ステートメント
DEFINE ステートメントオプション
- GROUP
- ORDER
- ACROSS
- ANALYSIS
- SUM
- STYLE=

COMPUTE ステートメントオプション
- AFTER
- STYLE=

LINE ステートメント
FORMAT プロシジャ
LIBNAME ステートメント
OPTIONS ステートメント
TITLE

データセット: GROCERY

出力形式: $MGRFMT

詳細

この例では、出力形式を使用して PROC REPORT が作成するグループ数を制御する方法を示します。プログラムでは、4 つの部門を 2 つの種類 生鮮と非生鮮のいずれかに分類する Department の出力形式を作成します。結果として、Department が列変数の場合、PROC REPORT は 4 つの列ではなく 2 つの列だけを作成します。列ヘッダーは、変数のフォーマットされた値です。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

proc format;
  value $perish 'p1','p2'='Perishable'
                'np1','np2'='Nonperishable';
run;

ods html file="myhtmlfile.html";

proc report data=grocery;


  column manager department,sales sales;

  define manager / group order=formatted
                   format=$mgrfmt.;
  define department / across order=formatted
                format=$perish. ";

  define sales / analysis sum
                 format=dollar9.2 style=[cellwidth=13];

  compute after / style=[font_style=italic];
    line 'Total sales for these stores were: '
         sales.sum dollar9.2;
  endcomp;

title 'Sales Summary for All Stores';
run;

ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

$PERISH.出力形式を作成します。 PROC FORMAT は Department の出力形式を作成します。この変数にはデータセットに 4 つの異なる値がありますが、出力形式には 2 つの値だけがあります。

```
proc format;
  value $perish 'p1','p2'='Perishable'
          'np1','np2'='Nonperishable';
run;
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

レポートオプションを指定します。 デフォルトでは、REPORT プロシジャはその出力を開いている出力先に送信します。

```
proc report data=grocery;
```

レポート列を指定します。 Department と Sales は定義する列のコンテンツを一括で決定できるように、COLUMN ステートメントでカンマで区切られます。Sales が分析変数であるため、その値はこれらの 2 つの変数によって作成されたセルに入力されます。レポートには、単独の Manager の列と Sales の列(すべての部門の売上)も含まれます。

```
column manager department,sales sales;
```

グループ変数と列変数を定義します。 Manager はグループ変数です。レポートの各詳細行には、Manager の値が同じすべてのオブザベーションに関する情報がまとめられます。Department は列変数です。PROC REPORT は Department のフォーマットされた値に対してそれぞれ列と列ヘッダーを作成します。
ORDER=FORMATTED は Manager と Department の値をそのフォーマットされた値に従ってアルファベット順に調整します。FORMAT=は使用する出力形式を指定します。Department の定義のブランクの引用符は、ブランクの列ヘッダーを指定します。そのため、ヘッダーはすべての部門にわたりません。ただし、PROC REPORT は Department の出力形式が定義された値を使用して、個別の部門に対しそれぞれ列ヘッダーを作成します。

```
define manager / group order=formatted
          format=$mgrfmt.;
define department / across order=formatted
         format=$perish. '';
```

分析変数を定義します。 Sales は、Sum 統計量の計算に使用される分析変数です。Sales は COLUMN ステートメントで二度記述され、同じ定義が二度とも適用されます。FORMAT=はレポートで使用する出力形式を指定します。STYLE=は列幅を指定します。Department と Sales が作成する列の列ヘッダーは、Department のヘッダーと Sales の(デフォルト)ヘッダーの組み合わせです。

```
define sales / analysis sum
          format=dollar9.2 style=[cellwidth=13];
```

カスタマイズした要約を作成します。 この COMPUTE ステートメントは、レポートの最後にカスタマイズされた要約を作成する計算ブロックを開始します。LINE ス

テートメントは、引用符付きテキストと Sales.sum の値(DOLLAR9.2 出力形式)を要約に置きます。STYLE=オプションで、要約情報をイタリック体に設定します。ENDCOMP ステートメントは計算ブロックを終了する必要があります。

```
compute after / style=[font_style=italic];
  line 'Total sales for these stores were: '
       sales.sum dollar9.2;
endcomp;
```

タイトルを指定します。

```
title 'Sales Summary for All Stores';
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

アウトプット 51.14　出力形式を使用してグループを 作成する

Sales Summary for All Stores

	Nonperishable	Perishable	
Manager	Sales	Sales	Sales
Adams	$265.00	$430.00	$695.00
Alomar	$510.00	$276.00	$786.00
Andrews	$620.00	$425.00	$1,045.00
Brown	$275.00	$323.00	$598.00
Jones	$260.00	$370.00	$630.00
Pelfrey	$465.00	$281.00	$746.00
Reveiz	$480.00	$630.00	$1,110.00
Smith	$170.00	$180.00	$350.00
Taylor	$173.00	$180.00	$353.00
Total sales for these stores were: $6,313.00			

例 12: マルチラベル出力形式の使用

要素:

PROC REPORT ステートメント
COLUMN ステートメント
DEFINE ステートメントオプション

詳細

この例では、マルチラベル出力形式を使用して、次を行うレポートを作成します。

- PROC FORMAT の VALUE ステートメントでマルチラベル出力形式を指定する方法を示す
- MLF オプションを DEFINE ステートメントと使用して、マルチラベル出力形式処理を有効化する方法を示す
- NOTSORTED および PRELOADFMT で、PROC FORMAT に示されている並べ替え順序を使用する方法を示す

プログラム

```
proc format;
 value agelfmt (multilabel notsorted)
    11='11'
    12='12'
    13='13'
    14='14'
    15='15'
    16='16'
    11-12='11 or 12'
    13-14='13 or 14'
    15-16='15 or 16'
    low-13='13 and below'
    14-high='14 and above' ;
run;

ods html file="example.html";

title "GROUP Variable with MLF Option";

proc report data=sashelp.class;
```

```
  col age ('Mean' height weight);

  define age / group mlf format=agelfmt. 'Age Group' order=data preloadfmt;
  define height / mean format=6.2 'Height (in.)';
  define weight / mean format=6.2 'Weight (lbs.)';
run;

ods html close;
```

プログラムの説明

AGE1FMT.出力形式を作成します。 FORMAT プロシジャは、MULTILABEL オプションを使用して年齢に対するマルチラベル出力形式を作成します。マルチラベル出力形式は、複数のラベルを同一の値に割り当てることができる出力形式です。各値は発生する各範囲のテーブルに表されます。NOTSORTED を使用して、その並べ替え順序が維持されるようにします。

```
proc format;
 value agelfmt (multilabel notsorted)
   11='11'
   12='12'
   13='13'
   14='14'
   15='15'
   16='16'
   11-12='11 or 12'
   13-14='13 or 14'
   15-16='15 or 16'
   low-13='13 and below'
   14-high='14 and above' ;
run;
```

ODS 出力を指定します。

```
ods html file="example.html";
```

タイトルを指定します。

```
title "GROUP Variable with MLF Option";
```

レポートオプションを指定します。 デフォルトでは、REPORT プロシジャはその出力を開いている出力先に送信します。

```
proc report data=sashelp.class;
```

レポート列を指定します。 レポートには、Age、Height、Weight の列が含まれています。Height と Weight の Mean が計算されます。

```
  col age ('Mean' height weight);
```

変数を定義します。 AGE はグループ変数です。AGE 変数は MLF オプションを使用してマルチラベル出力形式処理をアクティブにします。MLF は、グループ変数および列変数のみとともに使用する必要があります。FORMAT=はマルチラベル出力形式 AGE1FMT.が使用されるように指定します。PROC FORMAT で、ORDER=DATA オプションを NOTSORTED オプションとともに使用することにより、望ましい並べ替え順序を維持します。レポートの各詳細行に、グループ変数の値が同じすべてのオブザベーションに関する情報がまとめられます。平均値が HEIGHT と WEIGHT の

列値に対して計算されます。PRELOADFMT オプションで、変数に対して出力形式があらかじめ読み込まれるように指定します。

```
    define age / group mlf format=agelfmt. 'Age Group' order=data preloadfmt;
    define height / mean format=6.2 'Height (in.)';
    define weight / mean format=6.2 'Weight (lbs.)';
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

アウトプット 51.15　マルチラベル出力形式の使用

GROUP Variable with MLF Option

	Mean	
Age Group	Height (in.)	Weight (lbs.)
11	54.40	67.75
12	59.44	94.40
13	61.43	88.67
14	64.90	101.88
15	65.63	117.38
16	72.00	150.00
11 or 12	58.00	86.79
13 or 14	63.41	96.21
15 or 16	66.90	123.90
13 and below	59.03	87.35
14 and above	66.01	114.11

例 13: 複数のステートメントの ODS 出力にスタイル要素を指定する

要素:

PROC REPORT ステートメントオプション STYLE=オプション
　STYLE=

DEFINE ステートメントオプション
　ORDER
　STYLE=

BREAK ステートメントオプション
    AFTER
    SUMMARIZE
COMPUTE ステートメントオプション
    AFTER
    STYLE=
CALL DEFINE
    STYLE=
LINE ステートメント
FORMAT プロシジャ
LIBNAME ステートメント
OPTIONS ステートメント
TITLE ステートメント
WHERE ステートメント
ODS PDF ステートメント
ODS RTF ステートメント

データセット: GROCERY

出力形式: $MGRFMT

出力形式: $DEPTFMT

詳細

この例では、HTML、PDF および RTF ファイルを作成し、PROC REPORT ステートメントでレポートの各場所に対しスタイル要素を設定します。次に、その他のステートメントでスタイル要素を指定することにより、これらの設定のうち一部を無効にします。詳細については、“テーブル領域のスタイル要素とスタイル属性 ” (1830 ページ)を参照してください。

プログラム

```
libname proclib 'SAS-library';

options fmtsearch=(proclib);

ods html file="myhtmlfile.html";

ods pdf file='external-PDF-file';
ods rtf file='external-RTF-file';

proc report data=grocery

style(report)=[cellspacing=5 borderwidth=10 bordercolor=blue]

style(header)=[color=yellow
        fontstyle=italic fontsize=6]

style(column)=[color=moderate brown
```

```
            fontfamily=helvetica fontsize=4]
    style(lines)=[color=white backgroundcolor=black
            fontstyle=italic fontweight=bold fontsize=5]
    style(summary)=[color=cx3e3d73 backgroundcolor=cxaeadd9
             fontfamily=helvetica fontsize=3 textalign=r];
      column manager department sales;
      define manager / order
                  order=formatted
                  format=$mgrfmt.
                  'Manager'
                  style(header)=[color=white
                           backgroundcolor=black];
      define department / order
                    order=internal
                    format=$deptfmt.
                    'Department'
                    style(column)=[fontstyle=italic];
      break after manager / summarize;
      compute after manager
           / style=[fontstyle=roman fontsize=3 fontweight=bold
            backgroundcolor=white color=black];
        line 'Subtotal for ' manager $mgrfmt. 'is '
            sales.sum dollar7.2 '.';
      endcomp;
      compute sales;
        if sales.sum>100 and _break_=' ' then
        call define(_col_, "style",
                  "style=[backgroundcolor=yellow
                       fontfamily=helvetica
                       fontweight=bold]");
      endcomp;
      compute after;
         line 'Total for all departments is: '
            sales.sum dollar7.2 '.';
      endcomp;
      where sector='se';
      title 'Sales for the Southeast Sector';
    run;
    ods pdf close;
    ods rtf close;
    ods html close;
```

プログラムの説明

PROCLIB ライブラリを宣言します。 PROCLIB ライブラリを使用して、ユーザーが作成した出力形式を保存します。

```
libname proclib 'SAS-library';
```

出力形式検索ライブラリを指定します。 SAS システムオプション FMTSEARCH=は、出力形式の検索に使用される検索パスに SAS ライブラリ PROCLIB を追加します。

```
options fmtsearch=(proclib);
```

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

ODS 出力ファイル名を指定します。 複数の ODS 出力先を開くことによって、一度の実行で複数の出力ファイルを作成できます。HTML 出力はデフォルトで作成されます。ODS PDF ステートメントは出力を Portable Document Format (PDF)で作成します。ODS RTF ステートメントは出力を Rich Text Format (RTF)で作成します。PROC REPORT からの出力は、これらのファイルにそれぞれ行われます。

```
ods pdf file='external-PDF-file';
ods rtf file='external-RTF-file';
```

レポートオプションを指定します。 この場合、SAS は出力を従来のプロシジャ出力、HTML Body ファイル、RTF ファイル、PDF ファイルに書き込みます。

```
proc report data=grocery
```

レポートに対しスタイル属性を指定します。 この STYLE=オプションは、レポートの構造部分に対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(report)=[cellspacing=5 borderwidth=10 bordercolor=blue]
```

列ヘッダーに対しスタイル属性を指定します。 この STYLE=オプションは、すべての列ヘッダーに対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(header)=[color=yellow
        fontstyle=italic fontsize=6]
```

レポート列に対しスタイル属性を指定します。 この STYLE=オプションは、すべての列のすべてのセルに対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(column)=[color=moderate brown
        fontfamily=helvetica fontsize=4]
```

計算ブロック行に対しスタイル属性を指定します。 この STYLE=オプションは、すべての計算ブロックのすべての LINE ステートメントに対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(lines)=[color=white backgroundcolor=black
        fontstyle=italic fontweight=bold fontsize=5]
```

レポート要約に対しスタイル属性を指定します。 この STYLE=オプションは、すべてのデフォルト要約行に対しスタイル要素を設定します。スタイル要素は指定されていないため、ここで指定されるものを除き、PROC REPORT では、この場所のデフォルトスタイル要素のスタイル属性をすべて使用します。

```
style(summary)=[color=cx3e3d73 backgroundcolor=cxaeadd9
        fontfamily=helvetica fontsize=3 textalign=r];
```

レポート列を指定します。 レポートには、Manager、Department、Sales の列が含まれています。

```
column manager department sales;
```

最初の並べ替え順序変数を定義します。 このレポートで、Manager は順序変数です。PROC REPORT はまず Manager の値(COLUMN ステートメントの最初の変数であるため)別に行を調整します。ORDER=は Manager の値がそのフォーマットされた値に従って調整されるように指定します。FORMAT=は、この変数に使用する出力形式を指定します。引用符内のテキストは、列ヘッダーを指定します。

```
define manager / order
        order=formatted
        format=$mgrfmt.
        'Manager'
```

最初の並べ替え順序変数列ヘッダーに対しスタイル属性を指定します。 STYLE=オプションは、Manager の列ヘッダーの前景と背景の色を設定します。

```
        style(header)=[color=white
                backgroundcolor=black];
```

2 番目の並べ替え順序変数を定義します。 このレポートで、Department は順序変数です。PROC REPORT はまず Manager の値(COLUMN ステートメントの最初の変数であるため)別に、次に Department の値別に行を調整します。ORDER=は Department の値がその内部値に従って調整されるように指定します。FORMAT=は、この変数に使用する出力形式を指定します。引用符内のテキストは、列ヘッダーを指定します。

```
define department / order
          order=internal
          format=$deptfmt.
          'Department'
```

2 番目の並べ替え順序変数列に対しスタイル属性を指定します。 STYLE=オプションは、列 Department のセルのフォントをイタリックに設定します。セルのスタイル属性は、PROC REPORT ステートメントで COLUMN 場所に対して作成されたものと一致します。

```
          style(column)=[fontstyle=italic];
```

レポートの要約を作成します。 BREAK ステートメントは各マネージャーの最終行の後にデフォルト要約を作成します。SUMMARIZE は Sales の値(レポートの分析変数または計算変数のみ)を要約行に書き込みます。PROC REPORT は、Sales が Sum 統計量の計算に使用される分析変数であるため、各マネージャーに対する Sales の値を合計します。

```
break after manager / summarize;
```

カスタマイズした要約を作成します。 COMPUTE ステートメントは、レポートの最後にカスタマイズされた要約を作成する計算ブロックを開始します。この STYLE=オプションは、この計算ブロックの LINE ステートメントによって作成されるテキストに使用するスタイル要素を指定します。このスタイル要素は、PROC REPORT ステートメントで LINES 場所に対して指定された前景と背景の色を切り替えます。フォントスタイル、フォントの太さ、フォントサイズも変更します。

```
compute after manager
```

```
    / style=[fontstyle=roman fontsize=3 fontweight=bold
      backgroundcolor=white color=black];
```

カスタマイズされた要約に対しテキストを指定します。 LINE ステートメントは引用符付きテキストと Manager と Sales.sum の値(出力形式$MGRFMT.と DOLLAR7.2)を要約に置きます。ENDCOMP ステートメントは計算ブロックを終了する必要があります。

```
  line 'Subtotal for ' manager $mgrfmt. 'is '
      sales.sum dollar7.2 '.';
endcomp;
```

分析列に対するカスタマイズされた背景を作成します。 この計算ブロックは、100 以上の値を含む、要約行にはない Sales 列のすべてのセルに対し、背景色と太字フォントを指定します。

```
compute sales;
  if sales.sum>100 and _break_=' ' then
  call define(_col_, "style",
          "style=[backgroundcolor=yellow
                fontfamily=helvetica
                fontweight=bold]");
endcomp;
```

カスタマイズされたレポートの最後の要約を作成します。 この COMPUTE ステートメントは、レポートの最後に実行する計算ブロックを開始します。LINE ステートメントは引用符付きテキストと Sales.sum の値(出力形式 DOLLAR7.2)を書き込みます。ENDCOMP ステートメントは計算ブロックを終了する必要があります。

```
compute after;
   line 'Total for all departments is: '
      sales.sum dollar7.2 '.';
endcomp;
```

処理するオブザベーションを選択します。 WHERE ステートメントは、南東セクタの店舗のオブザベーションのみレポートに選択します。

```
where sector='se';
```

タイトルを指定します。

```
  title 'Sales for the Southeast Sector';
run;
```

ODS 出力先をクローズします。

```
ods pdf close;
ods rtf close;
ods html close;
```

出力: HTML

アウトプット 51.16　複数のステートメントの ODS 出力のスタイル要素

Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		630
Subtotal for Jones is $630.00.		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		350
Subtotal for Smith is $350.00.		
Total for all departments is: $980.00.		

出力: PDF

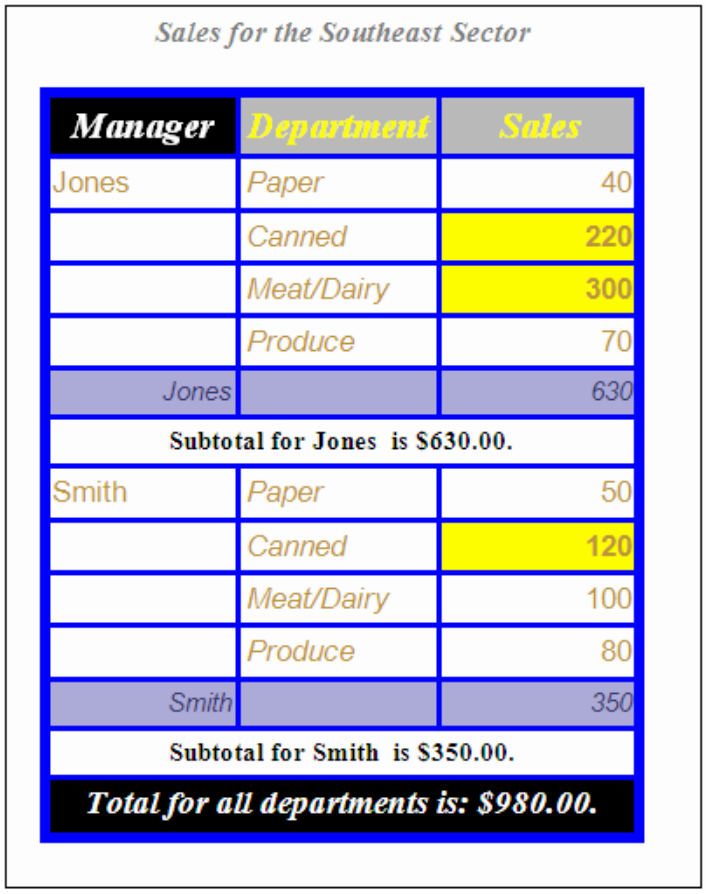
アウトプット 51.17　複数のステートメントの ODS 出力のスタイル要素

Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		630
Subtotal for Jones is $630.00.		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		350
Subtotal for Smith is $350.00.		
Total for all departments is: $980.00.		

出力: RTF

アウトプット 51.18　複数のステートメントの ODS 出力のスタイル要素



Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		630
Subtotal for Jones is $630.00.		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		350
Subtotal for Smith is $350.00.		
Total for all departments is: $980.00.		

例 14: セル幅の=スタイル属性を PROC REPORT とともに使用する

要素:

PROC REPORT ステートメントオプション
　STYLE=
COLUMN ステートメント
DEFINE ステートメントオプション
　STYLE(COLUMN)=
TITLE ステートメント

詳細

この例では、PROC REPORT を使用して、テーブルを作成し、ODS スタイル属性を使用します。この例では、次の操作を実行します。

- レポート全体のセルの幅を設定する
- ODS 出力の列のセル幅を定義する

注: CELLWIDTH=および WIDTH=は、別名です。

詳細については、"Style Attributes Tables" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

プログラム

```
ods html file="myhtmlfile.html";

proc report data=sashelp.class;

  col name age sex;

  define name / style(column)=[cellwidth=1in];
define age / style(column)=[cellwidth=.5in];
define sex / style(column)=[cellwidth=.5in];

title "Using the CELLWIDTH= Style with PROC REPORT";
run;

ods html close;
```

プログラムの説明

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

ODS 出力のすべての列に対してセルの幅を指定します。 デフォルトでは、REPORT プロシジャはその出力を開いている出力先に送信します。ODS HTML はこの例でデフォルトとして使用される出力先です。

```
proc report data=sashelp.class;
```

使用する列を指定します。

```
  col name age sex;
```

CELLWIDTH=スタイル属性を使用して列の幅を定義します。 STYLE=オプションを使用して NAME 列、AGE 列および SEX 列のディメンションを定義します。

```
  define name / style(column)=[cellwidth=1in];
define age / style(column)=[cellwidth=.5in];
```

```
define sex / style(column)=[cellwidth=.5in];
```

テーブルタイトルを指定します。 テーブル名を入力して、SAS プログラムを実行します。

```
title "Using the CELLWIDTH= Style with PROC REPORT";
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

アウトプット 51.19　セル幅の=スタイル属性を PROC REPORT とともに使用する

Using CELLWIDTH= Style with PROC REPORT

Name	Age	Sex
Alfred	14	M
Alice	13	F
Barbara	13	F
Carol	14	F
Henry	14	M
James	12	M
Jane	12	F
Janet	15	F
Jeffrey	13	M
John	12	M
Joyce	11	F
Judy	14	F
Louise	12	F
Mary	15	F
Philip	16	M
Robert	12	M
Ronald	15	M
Thomas	11	M
William	15	M

例 15: PROC REPORT CALL DEFINE ステートメントでの STYLE/MERGE の使用

要素:

PROC REPORT ステートメント
COLUMN ステートメント
COMPUTE ステートメント
CALL DEFINE ステートメントオプション
    STYLE
    STYLE/MERGE
TITLE ステートメント

詳細

この例では、PROC REPORT を使用してテーブルを作成し、CALL DEFINE ステートメントの STYLE/MERGE オプションを使用します。

プログラム

```
ods html file="myhtmlfile.html";

proc report data=sashelp.class;

col name sex age height weight;

define name--weight / display;

  compute sex;
   if sex = 'M' then
      call define('name', "style", "style=[background=cyan]");
   endcomp;

  compute age;
   if age > 13 then
      call define('name', "style/merge", "style=[color=red]");
   endcomp;

title "Using STYLE/MERGE Style with PROC REPORT";
run;

ods html close;
```

プログラムの説明

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

ODS 出力のすべての列に対してセルの幅を指定します。 デフォルトでは、REPORT プロシジャはその出力を開いている出力先に送信します。ODS HTML はこの例でデフォルトとして使用される出力先です。

```
proc report data=sashelp.class;
```

使用する列を指定します。

```
col name sex age height weight;
```

すべての列を表示します。

```
define name--weight / display;
```

スタイル属性を適用します。 男性の名前の背景にシアン色を適用します。

```
compute sex;
 if sex = 'M' then
    call define('name', "style", "style=[background=cyan]");
 endcomp;
```

スタイル属性をマージします。 13 歳を超えている被験者の名前に赤色を適用します。この名前の色を背景色がシアン色のセルにマージします。

```
compute age;
 if age > 13 then
    call define('name', "style/merge", "style=[color=red]");
 endcomp;
```

テーブルタイトルを指定します。 テーブル名を入力して、SAS プログラムを実行します。

```
title "Using STYLE/MERGE Style with PROC REPORT";
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

アウトプット 51.20　STYLE/MERGE の使用

Using STYLE/MERGE

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

例 16: PROC REPORT CALL DEFINE ステートメントでの STYLE/REPLACE の使用

要素:

PROC REPORT ステートメント
COLUMN ステートメント
COMPUTE ステートメント
CALL DEFINE ステートメントオプション
　STYLE

STYLE/REPLACE
TITLE ステートメント

詳細

この例では、PROC REPORT を使用してテーブルを作成し、CALL DEFINE ステートメントの STYLE/REPLACE オプションを使用します。

プログラム

```
ods html file="myhtmlfile.html";

proc report data=sashelp.class;

col name sex age height weight;

define name--weight / display;

  compute sex;
   if sex = 'M' then
      call define('name', "style", "style=[background=cyan]");
   endcomp;

  compute age;
   if age > 13 then
      call define('name', "style/replace", "style=[color=red]");
   endcomp;

title "Using STYLE/REPLACE";
run;

ods html close;
```

プログラムの説明

ODS 出力を指定します。

```
ods html file="myhtmlfile.html";
```

ODS 出力のすべての列に対してセルの幅を指定します。 デフォルトでは、REPORT プロシジャはその出力を開いている出力先に送信します。ODS HTML はこの例でデフォルトとして使用される出力先です。

```
proc report data=sashelp.class;
```

使用する列を指定します。

```
col name sex age height weight;
```

すべての列を表示します。

```
define name--weight / display;
```

スタイル属性を適用します。 男性の名前の背景にシアン色を適用します。

```
compute sex;
 if sex = 'M' then
    call define('name', "style", "style=[background=cyan]");
 endcomp;
```

スタイル属性を置き換えます。 13 歳を超えている被験者の名前に赤色を適用します。13 歳を超えている被験者の名前で、赤い名前の色を背景色シアンに置き換えます。

```
compute age;
 if age > 13 then
    call define('name', "style/replace", "style=[color=red]");
 endcomp;
```

テーブルタイトルを指定します。 テーブル名を入力して、SAS プログラムを実行します。

```
title "Using STYLE/REPLACE";
run;
```

ODS 出力先を閉じます。

```
ods html close;
```

出力: HTML

アウトプット 51.21　STYLE/REPLACE の使用

Using STYLE/REPLACE

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

52

S3 プロシジャ

概要: S3 プロシジャ

S3 プロシジャの動作について

Amazon S3 または MinIO 環境でオブジェクトのオブジェクト管理を実行するには、S3 プロシジャを使用します。たとえば、バケットを作成して S3 にファイルを追加することができます。SAS で使用する一般的なオブジェクトタイプは、ファイルとディレクトリです。

S3 プロシジャを使用するには、その前に、Amazon Web サービス(AWS)のキー ID とシークレットが必要です。一時的な認証情報を使用する場合は、セキュリティトークンも必要です。詳細については、Amazon S3 ドキュメントを参照してください。

S3 に追加または S3 から取得するデータが 5 MB より大きい場合、プロシジャは追加のスレッドを作成します。これらのスレッドは、より高速な転送速度のための並列処理を可能にします。

注: CAS を使用する場合、S3 アクションセットを使用して S3 オブジェクトを操作できます。

概念: S3 プロシジャ

PROC S3 構成

PROC S3 構成の概要

S3 プロシジャは、Amazon S3 または MinIO Enterprise Object Store に保存されているデータオブジェクトを操作します。システム要件の詳細については、“Support

for Data Storage in Amazon S3 and MinIO" (*System Requirements for the SAS Viya Platform*)を参照してください。

S3 プロシジャは、構成ファイル、Amazon EC2 インスタンスでのロール、または PROC S3 ステートメントのオプションから設定およびセキュリティ情報を読み取ります。構成ファイルを使用する場合、これらの構成ファイルは、AWS Command-Line Interface(CLI)の構成ファイルまたはローカルの PROC S3 の構成ファイルのいずれかになります。PROC S3 は、実行時に両方のタイプの構成ファイルを読み取ります。

AWS CLI 構成ファイルで指定されたオプションは、PROC S3 構成ファイルで指定されたオプションを上書きします。S3 プロシジャで指定するオプションは、構成ファイルで設定されているオプションを上書きします。

AWS CLI 構成ファイル

特に指定がない限り、PROC S3 は AWS CLI の構成ファイルをホームディレクトリのデフォルトの場所から読み取り、デフォルトのプロファイルを使用します。AWS CLI 構成ファイルは、config と credentials と呼ばれます。AWSCONFIG=オプションで構成ファイルの別の場所を指定することができます。AWSCREDENTIALS=オプションで認証情報ファイルの別の場所を指定することができます。

構成ファイルには、 *プロファイル*と呼ばれる複数のオプションセットが含まれています。PROC S3 ステートメントの PROFILE=オプションで使用する構成プロファイルを指定します。同様に、認証情報ファイルには、オプションのグループがプロファイルとして含まれています。PROC S3 ステートメントの CREDENTIALSPROFILE=オプションで使用する認証情報プロファイルを指定します。

PROC S3 構成ファイル

接続オプションに使用する PROC S3 の構成ファイルをローカルに作成することができます。デフォルトの PROC S3 構成ファイルは.tks3.conf です。これはホームディレクトリにあります。デフォルト名のローカル構成ファイルが存在する場合、S3 プロシジャは自動的にそれを読み取ります。PROC S3 構成ファイルが別の名前を使用するか、別の場所にある場合は、PROC S3 ステートメントで CONFIG=オプションを使用してその名前とパスを指定します。

注: AWS CLI 構成ファイルとローカル PROC S3 構成ファイルでオプションが指定されている場合、AWS CLI 構成ファイルの値が優先されます。

PROC S3 設定ファイルには、name=value として指定する大文字と小文字を区別する名前と値のペアが含まれています。値は引用符で囲まないでください。このファイルには、次の名前と値のペアの 1 つ以上のセットを含めることができます。

region
: 接続先の AWS リージョンを指定します。有効な region 値のリストについては、Region 引数を参照してください。

keyId
AWS アクセスキー ID を指定します。この値は 20 文字の英数字の文字列です。

secret
AWS シークレットアクセスキーを指定します。この値は 40 文字の文字列です。

注: この情報へのアクセス制限については、“PROC S3 構成ファイルの保護”を参照してください。

sessionToken
セッショントークンを指定します。一時的な AWS のセキュリティ認証情報を使用している場合は、この値を指定します。

注: この情報へのアクセス制限については、“PROC S3 構成ファイルの保護”を参照してください。

ssl
接続に SSL または TLS を使用するかどうかを指定します。SSL または TLS を使用するには、*true* または *yes* を指定します。それ以外の値は、SSL または TLS を無効にします。

次に、PROC S3 構成ファイルのサンプルを示します。

```
ssl=yes
keyID=AKFKI8OMEVIM3XJHKEUQ
secret=wb89GergI/3xejxudQugFj5Wi4iqlFJhGpLvYVv
region=usstd
```

PROC S3 構成ファイルの保護

secret と sessionToken の値は機密情報なので、オペレーティングシステムを使用してファイルへのアクセスを制限します。たとえば、次のコマンドを発行して、自分だけにアクセスを制限できます。

```
chmod 700 /u/$user/.tks3.conf
```

IMDS および IAM ロールのサポート

EC2 Instance Metadata Service バージョン 2 (IMDSv2)のサポートが追加されました。IMDSv1 は今後もサポートされます。Amazon EKS に配置されると、SAS は IMDS を使用して、S3 リソースへのアクセスを 可能にする IAM ロールを取得します。

SAS は、最初に IMDSv2 を使用してセッショントークンを取得しようとします。IMDSv2 の使用が失敗した場合には、IMDSv1 が使用されます。

Amazon 環境で次のオプションを指定します。

- set http-tokens が required に設定されている場合、IMDSv2 を使用する必要があります。http-tokens が optional に設定する場合は、IMDSv1 または IMDSv2 のいずれかが使用できます。
- IMDSv2 を使用する場合は、http_put_response_hop_limit を 2 に設定します。

カスタマイズリージョンの定義

S3 環境で提供される事前定義されたリージョンのリストに加えて、カスタムリージョンを定義できます。カスタムリージョンを管理する 1 つの方法は、PROC S3 で REGION ステートメントを使用することです。PROC S3 を使用してカスタムリージョンを定義する場合、リージョンの定義は SAS セッションの期間中継続します。後続の SAS セッションでリージョンを再定義する必要があります。

カスタムリージョンを管理する別の方法は、TKS3_CUSTOM_REGION 環境変数を使用することです。この環境変数は、Autoexec ファイルで、または SAS Environment Manager を使用して定義できます。このようにすることで、カスタムリージョン定義は、後続の SAS セッションを開始するときに自動的に使用されます。

TKS3_CUSTOM_REGION 環境変数を設定するための構文は次のとおりです。

TKS3_CUSTOM_REGION=<'>*region-name*<'>,*HTTP-host*,*HTTP-port*,*SSL-HTTP-port*,*SSLRequired*,*SSLAllowed*

region-name
カスタムリージョン名を指定します。値に空白または特殊文字が含まれている場合は、その値を引用符で囲みます。

HTTP-host
リージョンのサーバーのホスト名を指定します。

HTTP-port
SSL を使わずに通信する際のサーバーのポート番号を指定します。HTTP プロトコルのデフォルトのポート番号を使用する場合は 0 を使用します。

SSL-HTTP-port
SSL で通信する際のサーバーのポート番号を指定します。HTTPS プロトコルのデフォルトのポート番号を使用する場合は 0 を使用します。

SSLRequired
S3 環境とのすべての通信に SSL を使用するかどうかを指定します。値は TRUE または FALSE です。

SSLAllowed
S3 環境との通信に SSL を使用するかどうかを指定します。値は TRUE または FALSE です。

操作 この値は、*SSLRequired* 値が TRUE の場合は無視されます。

これは、BackBlazeB2 サーバーを使用するこの環境変数のサンプル設定です。

```
TKS3_CUSTOM_REGION=us-west-002,s3.us-west-002.backblazeb2.com,0,0,TRUE,TRUE
```

値のセットをコロン(:)で区切って、複数のリージョンを定義します。

AWS データでのサーバー側暗号化の使用

ENCKEY ステートメントを使用して暗号化キーを管理する

ENCKEY ステートメントを使用して、暗号化キーに名前を付けて管理します。ADD オプションを使用して、新しい名前と暗号化キーを追加します。GET、GETDIR、PUT、および PUTDIR ステートメントを使用して Amazon S3 または MinIO 環境と SAS の間でデータを転送するときに、名前付き暗号化キーを呼び出すことができます。ENCKEY ステートメントの LIST オプションを使用して現在登録されている暗号化キーのリストを要求したり、REMOVE オプションを使用して以前に登録された暗号化キーを削除したりすることもできます。

暗号化キーは、Amazon Key Management Service(KMS)キー、文字列として表される顧客提供のサーバー側暗号化キー(SSE-C)キー、または 16 進値で表される SSE-C キーの 3 種類で指定できます。

Amazon S3 環境から KMS キーを指定できます。このタイプのキーは AWS サーバー側の暗号化を操作するものであり、IAM サービスの一部として S3 環境で管理されます。

SSE-C 暗号化キーの場合、キーを 32 バイトの文字列または 64 桁の 16 進値として指定できます。次に、ユーザー指定のキーが S3 環境で使用され、AES256 暗号化アルゴリズムを使用してデータが暗号化されます。

ベストプラクティスとして、別の SAS プログラムで暗号化キーを管理します。次に、%INCLUDE ステートメントを使用して、名前付き暗号化を読み取り、暗号化キー名を操作できます。例については、“例 3: S3 データによる暗号化の管理” (1936 ページ)を参照してください。

SAS/ACCESS で暗号化キーを使用する

暗号化キーを追加した後、そのキーを SAS/ACCESS インターフェイスで Amazon Redshift に使用できます。LIBNAME ステートメントで、BL_ENCKEY=オプションを使用して、ENCKEY ステートメントで割り当てた名前で暗号化キーを呼び出します。これにより、Amazon Redshift 環境と SAS 間の一括ロードまたはアンロード中の暗号化が可能になります。または、DATA ステップや PROC SQL の呼び出しなど、特定の Amazon Redshift データセットを参照するときに、BL_ENCKEY=データセットオプションを指定することもできます。

注: Amazon Redshift への SAS/ACCESS インターフェイスでは、IAM: KMS キーのみがサポートされています。

詳細については、“Encryption with Amazon Redshift” (*SAS/ACCESS for Relational Databases: Reference*)を参照してください。

転送アクセラレーション

転送アクセラレーションとは、データの読み取りや書き込みを行う際に使用する特別なモードです。このモードでは、代替ホストを使用するとより高速なデータ転送が可能になります。詳細については、AWS のドキュメントを参照してください。

GET、GETDIR、PUT、および PUTDIR ステートメントは、転送アクセラレーションが有効な場合に、この高速データ転送方法を利用します。

構文: S3 プロシジャ

PROC S3<AWSCONFIG="*AWS-CLI-file-path*"> <CONFIG="*local-configuration-file-path*"> <PROFILE="*configuration-profile-name*"> <AWSCREDENTIALS="*AWS-credentials-file-path*"> <CREDENTIALSPROFILE="*credentials-profile-name*"> <AUTHDOMAIN="*AWS-authentication-domain*"> <AUTHSCOPE="*scope-URI*"> <ROLENAME="*IAM-role-name*" | ROLEARN="*IAM-role-ARN*"> <KEYID="*AWS-key-ID*"> <SECRET="*AWS-secret*"> <SESSION="*session-token*"> <SSL | NOSSL> <REGION=*AWS-region*>;

BUCKET"*bucket-name*"ACCELERATE | NOACCELERATE;

COPY<SRCKEY="*key-name*"> <ENCKEY="*key-name*"> <ACL="*canned-ACL-value*">"*source-s3-location*""*destination-s3-location*";

CREATE"*bucket-name*";

DELETE"*s3-location*";

DESTROY"*bucket-name*";

ENCKEYADD | REPLACENAME="*key-name*"ID="*key-ID*" <CONTEXT="*key-context*">;

GET<ENCKEY="*key-name*">"*s3-location*""*local-file-path*";

GETACCEL"*bucket-name*";

GETDIR<ENCKEY="*key-name*">"*s3-location*""*local-path*";

INFO<ENCKEY="*key-name*">"*s3-location*";

LIST<_SHORT_>"*s3-location*"<OUT=*libref.data-set*>;

MKDIR<ACL="*canned-ACL-value*">"*s3-location*";

PUT<ENCKEY="*key-name*"> <ACL="*canned-ACL-value*">"*local-path*""*s3-location*";

PUTDIR<ENCKEY="*key-name*"> <ACL="*canned-ACL-value*">"*local-path*""*s3-location*";

REGIONADDHOST="*AWS-server*"NAME="*region-name*"<PORT=*port-value*>
<SSLPORT=*SSL-port*> <SSLALLOWED> <SSLREQUIRED> <REPLACE>;

RMDIR"*s3-location*";

RUN;

ステートメント	タスク	例
“PROC S3 ステートメント”	接続パラメーターを S3 に指定します。	Ex. 1, Ex. 2, Ex. 3
“BUCKET ステートメント”	バケットの転送アクセラレーションを有効にするかどうかを指定します。	
“COPY ステートメント”	S3 オブジェクトを S3 ターゲットにコピーします。	Ex. 2, Ex. 3
“CREATE ステートメント”	S3 バケットを作成します。	Ex. 1
“DELETE ステートメント”	S3 の場所またはオブジェクトを削除します。	Ex. 2
“DESTROY ステートメント”	S3 バケットを削除します。	
“ENCKEY ステートメント”	暗号化キーを操作できるようにします。	Ex. 3
“GET ステートメント”	S3 オブジェクトを取得します。	Ex. 3
“GETACCEL ステートメント”	バケットの転送アクセラレーションステータスを取得します。	
“GETDIR ステートメント”	S3 ディレクトリの内容を取得します。	Ex. 3
“INFO ステートメント”	S3 の場所またはオブジェクトに関する情報をリスト表示します。	
“LIST ステートメント”	S3 の場所のコンテンツをリスト表示します。	Ex. 1
“MKDIR ステートメント”	S3 の場所に作成するディレクトリを指定します。	Ex. 2
“PUT ステートメント”	S3 の場所に書き込むローカルオブジェクトを指定します。	Ex. 1, Ex. 3
“PUTDIR ステートメント”	S3 の場所に書き込むローカルディレクトリを指定します。	Ex. 3
“REGION ステートメント”	カスタムリージョンの追加、リスト、削除を可能にします。	Ex. 4
“RMDIR ステートメント”	S3 の場所からディレクトリを削除します。	

PROC S3 ステートメント

Amazon S3 または MinIO に接続するための接続パラメーターを指定します。

サポート: スレッド処理

構文

PROC S3<AWSCONFIG="*AWS-CLI-file-path*"> <CONFIG="*local-configuration-file-path*"> <PROFILE="*configuration-profile-name*"> <AWSCREDENTIALS="*AWS-credentials-file-path*"> <CREDENTIALSPROFILE="*credentials-profile-name*"> <AUTHDOMAIN="*AWS-authentication-domain*"> <AUTHSCOPE="*scope-URI*"> <ROLENAME="*IAM-role-name*" | ROLEARN="*IAM-role-ARN*> <KEYID="*AWS-key-ID*"> <SECRET="*AWS-secret*"> <SESSION="*session-token*"> <SSL | NOSSL> <REGION=*AWS-region*>;

オプション引数

AUTHDOMAIN="*AWS-authentication-domain*"
AWS に接続するときに使用する認証ドメインを指定します。認証ドメインは通常、管理者によって定義されます。詳細については、“Credentials Domain: How to (CLI)” (*SAS Viya Platform: External Credentials*)を参照してください。

デフォルト AWSAUTH

注 このオプションのサポートは、2024.02 で追加されました。

AUTHSCOPE="*scope-URI*"
アクセスを許可するために必要な Microsoft Azure リソースと権限を指定します。スコープ URI 値は単一引用符または二重引用符で囲みます。

認証スコープは、SAS Viya プラットフォームのシングルサインオン機能を使用して Amazon S3 に接続するときに使用されます。この場合、管理者が Microsoft Entra ID を OpenID Connect プロバイダーとして構成する必要があります。詳細については、“Azure と AWS のフェデレーション認証の設定” (*SAS Viya プラットフォーム: 認証*)を参照してください。

デフォルト なし

操作 AUTHSCOPE=を指定する際は、ROLEARN=も指定する必要があります。

注 このオプションのサポートは、2024.12 で追加されました。

例 authscope="storage.azure.com/user_impersonation"

AWSCONFIG="*AWS-CLI-file-path*"
config という AWS CLI 構成ファイルのパスを指定します。このファイルには、Amazon S3 または MinIO にアクセスするための接続パラメーターが含まれています。値が AWS CLI 設定ファイルと PROC S3 設定ファイルで指定されている場合は、AWS CLI ファイルの値が優先されます。

AWSCREDENTIALS="*AWS-credentials-file-path*"
AWS 認証ファイルのパスを指定します。このファイルには、Amazon S3 または MinIO へのアクセスに使用される認証情報が含まれます。

CONFIG="*local-configuration-file-path*"
Amazon S3 または MinIO にアクセスするための接続パラメーターを含む PROC S3 構成ファイルのパスとファイル名を指定します。このオプションを指定すると、デフォルトの場所にあるデフォルトの PROC S3 構成ファイルの代わりに、指定された構成ファイルが読み取られます。

デフォルトの PROC S3 構成ファイルは、Linux のホームディレクトリにあるファイル.tks3.conf です。構成ファイルがデフォルトの場所にある場合は、CONFIG=オプションを指定する必要はありません。

詳細については、“PROC S3 構成ファイル” (1911 ページ)を参照してください。

CREDENTIALSPROFILE="*credentials-profile-name*"
Amazon S3 または MinIO へのアクセスに使用する認証情報を含むプロファイルの名前を指定します。認証情報のオプションのセットは、プロファイルにグループ化されます。CREDENTIALSPROFILE=オプションを指定しない場合、PROC S3 はデフォルトプロファイルを使用します。

KEYID="*AWS-key-ID*"
AWS アクセスキー ID を指定します。この値は 20 文字の英数字の文字列です。サンプルキー ID 値は AKIAIOSFODNN7EXAMPLE です。

PROFILE="*profile-name*"
AWS CLI ファイル config で使用するプロファイルを指定します。オプションのセットはプロファイルにグループ化されます。PROFILE=オプションを指定しない場合、PROC S3 はデフォルトプロファイルを使用します。

REGION="*AWS-region*"
接続先の AWS リージョンを指定します。事前定義されたリージョンの値は次のとおりです。

注: この引数には、SAS のリージョン値または AWS リージョン値のいずれかを指定できます。

表 52.1 S3 の事前定義されたリージョンの値

SAS のリージョンの値	説明	対応する Amazon S3 リージョン
afcapetown	アフリカ(ケープタウン)	af-south-1
aphongkong	アジア太平洋(香港)	ap-east-1
aphyderabad	アジア太平洋(ハイデラバード)	ap-south-2

SAS のリージョンの値	説明	対応する Amazon S3 リージョン
apindia	アジア太平洋(インド)	ap-south-1
apjakarta	アジア太平洋(ジャカルタ)	ap-southeast-3
apmelbourne	アジア太平洋(メルボルン)	ap-southeast-4
aposaka	アジア太平洋(大阪)	ap-northeast-3
apseoul	アジア太平洋(ソウル)	ap-northeast-2
apsingapore	アジア太平洋(シンガポール)	ap-southeast-1
apsydney	アジア太平洋(シドニー)	ap-southeast-2
aptokyo	アジア太平洋(東京)	ap-northeast-1
cacalgary	カナダ(カルガリー)	ca-west-1
cacentral	カナダ(セントラル)	ca-central-1
cnbeijing	中国(北京)	cn-north-1
cnningxia	中国(寧夏)	cn-northwest-1
eufrankfurt	EU(フランクフルト)	eu-central-1
euireland	EU(アイルランド)	eu-west-1
eulondon	EU(ロンドン)	eu-west-2
eumilan	EU(ミラノ)	eu-south-1
euparis	EU(パリ)	eu-west-3
euspain	EU(スペイン)	eu-south-2
eustockholm	EU(ストックホルム)	eu-north-1
euzurich	EU(チューリッヒ)	eu-central-2
fips	US Government Cloud (FIPS)	fips-us-gov-west-1
fipseast	US Government Cloud (FIPS East)	fips-us-gov-east-1

SAS のリージョンの値	説明	対応する Amazon S3 リージョン
iltelaviv	イスラエル(テルアビブ)	il-central-1
mebahrain	中東(バーレーン)	me-south-1
meuae	中東(UAE)	me-central-1
sa	南米(サンパウロ)	sa-east-1
useast	US 東(バージニア)	us-east-1
useastoh	US 東(オハイオ)	us-east-2
usgov	US Government Cloud	us-gov-west-1
usgoveast	US Government Cloud (East)	us-gov-east-1
usstd	US 標準	us-east-1
uswest	US 西(オレゴン)	us-west-2
uswestca	US 西(カリフォルニア)	us-west-1

ROLEARN="*IAM-role-ARN*"
IAM ロールを識別する Amazon リソース名(ARN)を指定します。関連する IAM ロールを使用すると、AssumeRole IAM 操作を介して S3 へアクセスするための認証情報を取得できます。

PROC S3 は、デフォルトで EC2 インスタンスと関連する IAM ロールを使用します。使用するロールが、SAS プロセスが実行されている EC2 インスタンスプロファイルにすでに関連付けられている場合は、ROLEARN=を指定する必要はありません。指定するロール ARN は、AWS ですでに定義されている必要があります。

操作 ROLEARN=オプションまたは ROLENAME=オプションのどちらかを指定します。両方を指定しないでください。

注 使用している EC2 インスタンスに関連付けられていないロール ARN を使用する場合にのみ、このオプションを指定します。

例
```
proc s3 rolearn="arn:aws:iam::123456789012:role/s3AccessRole"
    <additional-options>;
```

ROLENAME="*IAM-role-name*"
AssumeRole IAM 操作を介して S3 へアクセスするための認証情報を取得できるようにする IAM ロール名を指定します。

PROC S3 は、デフォルトで EC2 インスタンスと関連する IAM ロールを使用します。使用するロールが、SAS プロセスが実行されている EC2 インスタンスプロフ

ファイルにすでに関連付けられている場合は、ROLENAME=を指定する必要はありません。指定するロール名は、AWS ですでに定義されている必要があります。

操作 ROLENAME=オプションまたは ROLEARN=オプションのどちらかを指定します。両方を指定しないでください。

注 使用している EC2 インスタンスに関連付けられていないロール名を使用する場合にのみ、このオプションを指定します。

例 proc s3 rolename="s3AccessRole" <additional-options>;

SECRET="*AWS-secret*"
AWS シークレットアクセスキーを指定します。この値は 40 文字の文字列です。秘密アクセス値の例は wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY です。

SESSION="*session-token*"
AWS のセッショントークンを指定します。一時的な AWS のセキュリティ認証情報を使用している場合は、この値を使用します。

詳細については、“PROC S3 構成” (1910 ページ)を参照してください。

SSL | NOSSL
SSL は、データ転送中に SSL または TLS を有効にする必要があることを指定します。NOSSL は、データ転送中に SSL または TLS を無効にすることを指定します。

SSL 値を使用して、 PROC S3 設定ファイルの ssl=no の設定を上書きできます PROC S3 ステートメントまたは PROC S3 設定ファイルで SSL に値が指定されていない場合、デフォルトでデータ転送中に SSL または TLS が使用されます。

インタラクション: PROC S3 ステートメントで NOSSL を指定した場合、ENCKEY ステートメントによるサーバー側の暗号化は失敗します。

BUCKET ステートメント

指定されたバケットの転送アクセラレーションモードを設定します。

構文

BUCKET"*bucket-name*"ACCELERATE | NOACCELERATE;

必須引数

bucket-name
転送アクセラレーションモードを設定する S3 バケットの名前を指定します。詳細については、“転送アクセラレーション” (1915 ページ)を参照してください。

ACCELERATE
指定されたバケットの転送アクセラレーションを有効にします。

NOACCELERATE
指定されたバケットの転送アクセラレーションを無効にします。

COPY ステートメント

オブジェクトを S3 ソースの場所から S3 ターゲットにコピーします。

構文

COPY<SRCKEY="*key-name*"> <ENCKEY="*key-name*"> <ACL="*canned-ACL-value*">"*source-s3-location*""*destination-s3-location*";

必須引数

source-s3-location
コピーされるオブジェクトの S3 の場所を指定します。バケット名からオブジェクト名までの S3 の場所を完全修飾します。

destination-s3-location
オブジェクトをコピーする S3 の場所を指定します。バケット名からオブジェクト名までの S3 の場所を完全修飾します。

オプション引数

ACL="*canned-ACL-value*"
出力先の場所にある新しく作成されたオブジェクトに適用される許可された ACL 値を指定します。

有効な値は次のとおりです。

authenticated-read	log-delivery-write
aws-exec-read	private
bucket-owner-full-control	public-read
bucket-owner-read	public-read-write

これらの値の詳細については、Amazon S3 Canned ACL ドキュメントを参照してください。

デフォルト　private

要件　指定する出力先の場所は、ACL が有効な S3 バケット内にある必要があります。

注　このオプションのサポートは、2024.10 で追加されました。

ENCKEY="*key-name*"
作成されたオブジェクトの暗号化を識別するための暗号化キー名を指定します。ENCKEY ステートメントで指定した暗号化キー名と一致する必要があります。

ENCKEY=オプションのデフォルトは、暗号化なしです。値_*DEFAULT*_を使用すると、デフォルトの暗号化キーを使用できます。デフォルトの暗号化キーが自動的に作成されます。

SRCKEY="*key-name*"
ソースオブジェクトの暗号化を指定します。ENCKEY ステートメントで指定した暗号化キー名と一致する必要があります。

注: ENCKEY=オプションが指定されていない場合、この値は作成されたオブジェクトには適用されません。

CREATE ステートメント

バケットを作成します。

構文

CREATE"*bucket-name*";

必須引数

bucket-name
作成する S3 バケットの名前を指定します。作成する S3 バケットの名前は、S3 全体で一意でなければなりません。

DELETE ステートメント

S3 の場所またはオブジェクトを削除します。

構文

DELETE"*s3-location*";

必須引数

s3-location
削除する S3 の場所を指定します。バケット名からオブジェクト名までの S3 の場所を完全修飾します。

DESTROY ステートメント

S3 バケットを削除します。

構文

DESTROY"*bucket-name*";

必須引数

bucket-name
削除する S3 バケットの名前を指定します。指定するバケットは S3 に存在し、空でなければなりません。

ENCKEY ステートメント

S3 の場所の暗号化キーを指定します。

構文

形式 1: **ENCKEY**ADD | REPLACENAME="*key-name*"ID="*key-ID*"<CONTEXT="*key-context*">;

形式 2: **ENCKEY**ADD | REPLACENAME="*key-name*"KEY="*key-string*" <ALGORITHM="*S3-encryption-algorithm*"> <CONTEXT="*key-context*">;

形式 3: **ENCKEY**ADD | REPLACENAME="*key-name*"HEXKEY="*hexadecimal-value*" <ALGORITHM="*S3-encryption-algorithm*"> <CONTEXT="*key-context*">;

形式 4: **ENCKEY**LIST<NAME="*key-name*">;

形式 5: **ENCKEY**REMOVENAME="*key-name*" ;

引数

ADD
暗号化キーを追加していることを示します。

必要要件: 使用するキーの種類に応じて、HEXKEY=、ID=、KEY=のいずれかのオプションを ADD で指定します。

ALGORITHM="*S3-encryption-algorithm*"
S3 の場所で使用される S3 暗号化アルゴリズムを指定します。

現在サポートされている値は、AES256 のみです。この値は、ALGORITHM=オプションを指定しない場合のデフォルト値です。

CONTEXT="*key-context*"
暗号化キーのキーコンテキストを指定します。

HEXKEY="*hexadecimal-value*"
暗号化キーを 64 桁の 16 進数で指定します。

ID="*key-ID*"
暗号化キーのキー ID を指定します。

KEY="*key-string*"
暗号化キーを 32 バイトの文字列で指定します。

LIST
定義された暗号化キーのリストを要求します。

NAME="*key-name*"
暗号化キーの名前を指定します。これは、暗号化キーを参照するために使用できる、ユーザーフレンドリーな名前です。値は大文字と小文字が区別されます。

NAME=は、ADD、REPLACE、REMOVE の各オプションで必須です。LIST オプションと併用する場合、NAME=はオプションです。

REMOVE
定義されたキーのリストから暗号化キーを削除します。

必要要件: REMOVE で NAME=オプションを指定します。

REPLACE
既存の暗号化キーを置き換えることを指定します。設定されていない場合は、暗号化キーを再定義しようとしても失敗します。

必要要件: 使用するキーの種類に応じて、HEXKEY=、ID=、KEY=のいずれかのオプションを REPLACE で指定します。

詳細

ENCKEY ステートメントは、Amazon S3 または MinIO 環境でサーバー側の暗号化をサポートするために使用されます。

- ENCKEY ステートメントのフォーム 1 を使用して、IAM:KMS の暗号化キーを追加または交換します。
- ENCKEY ステートメントのフォーム 2 を使用して、文字列形式の暗号化キーを追加または交換します。文字列の長さは 32 バイトでなければなりません。
- ENCKEY ステートメントのフォーム 3 を使用して、16 進数で表現される暗号化キーを追加または置換します。16 進数のキー文字列を指定する際には、"0x "というプレフィックスを使用しないでください。16 進文字列は 64 桁の長さでなければなりません。
- ENCKEY ステートメントのフォーム 4 を使用して、定義された暗号化キーの一覧を表示します。
- ENCKEY ステートメントのフォーム 5 を使用して、暗号化キーを削除します。

注: PROC S3 ステートメントで NOSSL オプションを指定した場合、ENCKEY ステートメントでの暗号化の使用に失敗します。

ENCKEY ステートメントの中で、次のオプションのいずれかを指定する必要があります: ADD、LIST、REMOVE または REPLACE。

暗号化キーは、SAS セッション中に指定されます。

暗号化キー*_DEFAULT_*は変更できません。*_DEFAULT_*の暗号化キーは、AWS の環境で定義されています。

詳細については、AWS ドキュメントの Protecting Data Using Server-Side Encryption をご覧ください。

GET ステートメント

S3 オブジェクトを取得します。

注: このステートメントは、転送アクセラレーションが利用可能な場合に使用します。詳細については、“転送アクセラレーション” (1915 ページ)を参照してください。

構文

GET<ENCKEY="*key-name*">"*s3-location*""*local-file-path*";

必須引数

s3-location
取得する S3 オブジェクトの名前を指定します。バケット名からオブジェクト名までの場所を完全修飾します。

local-file-path
オブジェクトを保存するローカルファイルを指定します。たとえば、この値を**/u/$USER/myfile.txt** に設定できます。

オプション引数

ENCKEY="*key-name*"
データ検索時に使用する暗号化を識別するための暗号化キー名を指定します。ENCKEY ステートメントで指定した暗号化キー名と一致する必要があります。また、この名前はオブジェクトが S3 環境で作成されたときに使用された暗号化キーにも一致している必要があります。それ以外の場合、S3 はエラーを返します。

GETACCEL ステートメント

バケットの転送アクセラレーションステータスを取得します。

構文

GETACCEL"*bucket-name*";

必須引数

bucket-name
転送アクセラレーションステータスを必要とする S3 バケットの名前を指定します。詳細については、“転送アクセラレーション” (1915 ページ)を参照してください。

GETDIR ステートメント

S3 ディレクトリの内容を取得します。

注: このステートメントは、転送アクセラレーションが利用可能な場合に使用します。詳細については、“転送アクセラレーション” (1915 ページ)を参照してください。

構文

GETDIR<ENCKEY="*key-name*">"*s3-location*""*local-path*";

必須引数

s3-location
取得する S3 ディレクトリの名前を指定します。バケット名からディレクトリまでの場所を完全修飾します。

local-path
SAS クライアントにローカルのディレクトリを指定します。たとえば、この値を**/u/$USER/**に設定できます。

オプション引数

ENCKEY="*key-name*"
データ検索時に使用する暗号化を識別するための暗号化キー名を指定します。ENCKEY ステートメントで指定した暗号化キー名と一致する必要があります。

注: ソースの場所にあるファイルが指定された暗号化キーを使用していない場合、暗号化は失敗します。

INFO ステートメント

S3 の場所に関する情報を SAS ログに要求します。

注: ファイルの情報には、リージョン、パス、ファイルの最終更新日時が含まれます。

構文

INFO<ENCKEY="*key-name*">"*s3-location*";

必須引数

s3-location
バケット名からオブジェクト名への完全修飾パスを指定します。

オプション引数

ENCKEY="*key-name*"
暗号化キーの名前を指定します。ENCKEY ステートメントで指定した暗号化キー名と一致する必要があります。

注: S3 ロケーションのオブジェクトが暗号化されている場合は、ENCKEY=オプションが必要です。

LIST ステートメント

S3 ロケーションのコンテンツに関する情報を SAS ログに要求します。

構文

LIST<_SHORT_>"*s3-location*"<OUT=*libref.data-set* | "*/path/to/output-file*">;

必須引数

s3-location
バケット名からオブジェクト名への完全修飾パスを指定します。通常、指定するオブジェクトは、バケットやディレクトリなどのコンテナーです。

オプション引数

SHORT
オブジェクト名のリストのみ必要であることを指定します。

OUT=*libref.data-set* | "*/path/to/output-file*"
LIST ステートメントの出力を含むデータセットまたはファイルの場所を指定します。

ライブラリ参照名を省略した場合、データセット名を引用符で囲んで指定する必要があります。また、このデータセットは Work ライブラリに保存されます。

例　list "/myS3location" out=sasuser.list;

MKDIR ステートメント

S3 の場所に作成するディレクトリを指定します。

構文

MKDIR<ACL="*canned-ACL-value*">"*s3-location*";

必須引数

s3-location
バケット名から作成するディレクトリ名までの完全修飾パスを指定します。

オプション引数

ACL="*canned-ACL-value*"
指定されたディレクトリで新しく作成されたオブジェクトに適用される保存された ACL 値を指定します。

有効な値は次のとおりです。

authenticated-read	log-delivery-write
aws-exec-read	private
bucket-owner-full-control	public-read
bucket-owner-read	public-read-write

これらの値の詳細については、Amazon S3 Canned ACL ドキュメントを参照してください。

デフォルト　private

要件　指定するディレクトリは、ACL が有効な S3 バケット内にある必要があります。

注　このオプションのサポートは、2024.10 で追加されました。

PUT ステートメント

S3 の場所に書き込むローカルオブジェクトを指定します。

注:　このステートメントは、転送アクセラレーションが利用可能な場合に使用します。詳細については、“転送アクセラレーション” (1915 ページ)を参照してください。

構文

PUT<ENCKEY="*key-name*"> <ACL="*canned-ACL-value*">"*local-path*""*s3-location*";

必須引数

local-path
SAS クライアントにローカルなファイルまたはディレクトリを指定します。

s3-location
S3 の場所を指定します。バケット名からオブジェクト名までのパスを完全修飾します。

オプション引数

ACL="*canned-ACL-value*"
指定された S3 の場所の新しいオブジェクトに適用される範囲内の ACL 値を指定します。

有効な値は次のとおりです。

authenticated-read	log-delivery-write
aws-exec-read	private
bucket-owner-full-control	public-read
bucket-owner-read	public-read-write

これらの値の詳細については、Amazon S3 Canned ACL ドキュメントを参照してください。

デフォルト　private

要件　指定する場所は、ACL が有効な S3 バケット内にある必要があります。

注 このオプションのサポートは、2024.10 で追加されました。

ENCKEY="*key-name*"
S3 にデータを保存する際に使用する、暗号化を識別する暗号化キー名を指定します。ENCKEY ステートメントで指定した暗号化キー名と一致する必要があります。

例 デフォルトの暗号化キーを指定するには:
```
put enckey=_default_ "/home/demouser/s3/foo.txt" "/mybucket/foo.txt";
```

PUTDIR ステートメント

S3 の場所に書き込むローカルディレクトリを指定します。

注: このステートメントは、転送アクセラレーションが利用可能な場合に使用します。詳細については、“転送アクセラレーション” (1915 ページ)を参照してください。

構文

PUTDIR<ENCKEY="*key-name*"> <ACL="*canned-ACL-value*">"*local-path*""*s3-location*";

必須引数

local-path
SAS クライアントにローカルのディレクトリを指定します。

s3-location
S3 の場所を指定します。バケット名からオブジェクト名までのパスを完全修飾します。

オプション引数

ACL="*canned-ACL-value*"
指定された S3 の場所のオブジェクトに適用される範囲内の ACL 値を指定します。

有効な値は次のとおりです。

authenticated-read	log-delivery-write
aws-exec-read	private
bucket-owner-full-control	public-read
bucket-owner-read	public-read-write

これらの値の詳細については、Amazon S3 Canned ACL ドキュメントを参照してください。

デフォルト　private

要件　指定する場所は、ACL が有効な S3 バケット内にある必要があります。

注　このオプションのサポートは、2024.10 で追加されました。

ENCKEY="*key-name*"
S3 への書き込み時に使用する、暗号化を識別する暗号化キー名を指定します。ENCKEY ステートメントで指定した暗号化キー名と一致する必要があります。

注: S3 ロケーションにあるすべてのファイルは、指定された暗号化キーで暗号化されます。

REGION ステートメント

カスタムリージョンの追加、リスト、削除が可能です。

構文

形式 1: **REGION**ADDHOST="*AWS-server*"< REGION="AWS-region-value"> NAME="*region-name*"< PORT=*port-value*> < SSLPORT=*SSL-port*> < SSLALLOWED> < SSLREQUIRED> < REPLACE>;

形式 2: **REGION**LIST;

形式 3: **REGION**REMOVENAME="*region-name*" ;

必須引数

ADD
S3 環境のカスタムリージョンを追加することを指定しています。

注: REGION ADD ステートメントを使用してカスタムリージョンを定義する場合、リージョンの定義は SAS セッションの期間中継続します。後続の SAS セッションでは、リージョンを再定義する必要があります。代替として、TKS3_CUSTOM_REGION 環境変数を使用できます。詳細については、“カスタマイズリージョンの定義” (1913 ページ)を参照してください。

HOST="*AWS-server*"
PROC S3 が AWS 上で接続するサーバーを指定します。

例　host="myhost.amazonaws.com"

NAME="*region-name*"
追加または削除するリージョンの一意の名前を指定します。

LIST
定義されたすべての S3 リージョンのリストを要求します。

REMOVE
NAME=オプションで識別したリージョンを削除するように指定します。

制限事項 事前定義された S3 リージョンを削除することはできません。事前定義されたリージョン値のリストについては、PROC S3 ステートメントの REGION=引数を参照してください。

オプション引数

PORT=*port-value*
SSL なしで S3 に接続するために使用する HTTP ポート値を指定します。

ポート値を指定しない場合は、デフォルトの値が使用されます。

REGION="*AWS-region-code*"
追加または置換するリージョンのリージョンコードを指定します。

REPLACE
カスタムリージョンを追加し、同じ名前のリージョンがある場合はそれを上書きすることを示します。

制限事項 S3 の事前定義されたリージョンのいずれかを置き換えることはできません。事前定義されたリージョン値のリストについては、PROC S3 ステートメントの REGION=引数を参照してください。

SSLALLOWED
S3 環境と通信時に SSL を許可することを示します。

操作 SSLREQUIRED が指定されている場合は、この値は無視されます。

SSLPORT=*SSL-port*
SSL で S3 に接続する場合に使用する HTTP ポート値を指定します。

ポート値を指定しない場合は、デフォルトの値が使用されます。

SSLREQUIRED
S3 環境とのすべての通信に SSL を使用する必要があることを示します。

RMDIR ステートメント

S3 の場所から削除するディレクトリを指定します。

構文

RMDIR"*s3-location*";

必須引数

s3-location
S3 のディレクトリを指定します。バケット名からディレクトリ名までのパスを完全修飾します。指定するディレクトリは空でなければなりません。

例: S3 プロシジャ

例 1: バケットの作成とファイルの追加

要素: PROC S3 ステートメント、CONFIG=オプション
CREATE ステートメント
PUT ステートメント
LIST ステートメント

詳細

この例では、PROC S3 の CREATE ステートメントを使用して、S3 にバケットを作成します。次に、バケットにファイルを追加し、バケットのコンテンツをリスト表示する例を示します。

プログラム

```
proc s3 config="/u/marti/.tks3.conf";
  create "/mybucket";
  put "/u/marti/project/licj.csv" "/mybucket/licj.csv";
  list "/mybucket";
run;
```

プログラムの説明

PROC S3 ステートメントを実行し、PROC S3 構成ファイルの場所を指定します。
指定されたディレクトリの.tks3.conf ファイルに含まれている接続オプションを使用して、S3 に接続します。

```
proc s3 config="/u/marti/.tks3.conf";
```

S3 にバケットを作成します。 CREATE ステートメントを使用して、バケット *mybucket* を作成します。

```
  create "/mybucket";
```

ローカルファイルのコピーを新しい S3 バケットに保存します。 PUT ステートメントを実行して、ローカルファイル licj.csv を、S3 バケット *mybucket* の同じ名前のファイルにコピーします。

```
  put "/u/marti/project/licj.csv" "/mybucket/licj.csv";
```

S3 バケットのコンテンツのリストを表示します。 LIST ステートメントを使用して、*mybucket* のコンテンツのリストを表示します。

```
  list "/mybucket";
run;
```

ファイル licj.csv がバケット内の唯一のファイルである場合、次のメッセージが SAS ログに出力されます。

アウトプット 52.1　バケットのコンテンツのリスト

```
licj.csv    2791403 2015-06-05T19:27:35.000Z
```

例 2: バケットのコンテンツの管理

要素:

PROC S3 ステートメント
MKDIR ステートメント
COPY ステートメント
DELETE ステートメント

詳細

この例は、バケットにディレクトリを作成する方法と、ファイルをコピーおよび削除する方法を示しています。

プログラム

```
proc s3;
  mkdir "/mybucket/csv";
  copy "/mybucket/licj.csv" "/mybucket/csv/licj.csv";
  delete "/mybucket/licj.csv";
```

```
run;
```

プログラムの説明

デフォルトの構成ファイルを使用して S3 に接続します。 PROC S3 ステートメントを使用して、ホームディレクトリの.tks3.conf ファイルに指定されている接続オプションを使用して S3 に接続します。これが、デフォルトの構成ファイルの場所になります。

```
proc s3;
```

新しいディレクトリを作成します。 MKDIR ステートメントを使用して、*mybucket* にディレクトリ csv を作成します。ノートが SAS ログに印刷されます。

```
mkdir "/mybucket/csv";
```

ファイルを新しいディレクトリにコピーします。 COPY ステートメントを使用して、ファイル licj.csv を *mybucket* から csv ディレクトリにコピーします。

```
copy "/mybucket/licj.csv" "/mybucket/csv/licj.csv";
```

ファイルの元のコピーを削除します。 DELETE ステートメントを使用して、licj.csv ファイルの元のコピーを *mybucket* から削除します。ファイルのコピーのみ/mybucket/csv 残ることになります。

```
delete "/mybucket/licj.csv";
run;
```

アウトプット 52.2　バケットのデータの管理

```
NOTE: Created directory /mybucket/csv.

NOTE: Deleted object /mybucket/licj.csv.
```

例 3: S3 データによる暗号化の管理

要素:

PROC S3 ステートメント
COPY ステートメント
ENCKEY ステートメント
GET ステートメント
GETDIR ステートメント
PUT ステートメント
PUTDIR ステートメント

詳細

この例では、暗号化キーの操作方法を説明します。ベストプラクティスとして、暗号化キーは別の SAS プログラムで管理し、AWS 環境と相互作用する別の SAS プログラムに含めます。この例では、暗号化キーを管理するプログラムを keys.sas とします。すべての暗号化は、S3 環境でサポートしているデフォルトの AES256 暗号化アルゴリズムを使用します。

プログラム: Keys.sas

```
/* keys.sas */
proc s3;

  /* IAM:KMS keys */
  enckey add name="foo" id="98v27390-1si1-8sc9-38k0-k893j354nw5g";
  enckey add name="bar" id="22f87852-0ak1-5xy2-01j1-l852g809gx4q";


  /* SSE-C keys (user-specified) as character string */
  enckey add name="foo2" key="ke-sj-eb-ok-qk-zi-nb-lu-fh-wi-zi";
  enckey add name="bar2" key="wl-tk-64-rk-i0-zn-wk-7c-s8-a8-jk";


  /* SSE-C (user-specified) keys as hex data */
  enckey replace name="foo2"
    hexkey="e1d2e3bb4a5f6079889706b3c4d1e2f999f2c3bb4c5f6079888706f6a4b3e2da";
  enckey replace name="bar2"

hexkey="98929402034568920937580293757283940109283746748392018375839201 92";

run;
```

プログラムの説明

IAM:KMS の暗号化キーを追加します。 ENCKEY ステートメントの ADD オプションを使用して、foo と bar という名前の 2 つの暗号化キーを追加します。AWS 環境の IAM サービスによって定義された ID=値を指定します。NAME=オプションを使用して、暗号化キーのわかりやすい識別子を提供します。

```
/* keys.sas */
proc s3;

  /* IAM:KMS keys */
  enckey add name="foo" id="98v27390-1si1-8sc9-38k0-k893j354nw5g";
  enckey add name="bar" id="22f87852-0ak1-5xy2-01j1-l852g809gx4q";
```

文字列を含む SSE-C キーを追加します。 ユーザー指定の暗号化キーを、ENCKEY ステートメントと ADD オプションを使用して追加できます。32 バイトの文字列を KEY=値として指定し、NAME=にわかりやすい値を指定します。

```
/* SSE-C keys (user-specified) as character string */
enckey add name="foo2" key="ke-sj-eb-ok-qk-zi-nb-lu-fh-wi-zi";
enckey add name="bar2" key="wl-tk-64-rk-i0-zn-wk-7c-s8-a8-jk";
```

既存の暗号化キーをユーザー定義の 16 進キーに置き換えます。 REPLACE オプションを NAME=オプションとともに使用して、置き換える暗号化キーを識別します。HEXKEY=オプションを使用して、64 桁の 16 進値で暗号化キーを指定できます。

```
/* SSE-C (user-specified) keys as hex data */
enckey replace name="foo2"
  hexkey="e1d2e3bb4a5f6079889706b3c4d1e2f999f2c3bb4c5f6079888706f6a4b3e2da";
enckey replace name="bar2"

hexkey="98929402034568920937580293757283940109283746748392018375839201 92";

run;
```

```
1          /* keys.sas */
2          proc s3;
3          /* IAM:KMS keys */
4          enckey add name="foo" id="98v27390-1si1-8sc9-38k0-k893j354nw5g";
5          enckey add name="bar" id="22f87852-0ak1-5xy2-01j1-l852g809gx4q";
6          /* user-defined keys as character string */
7          enckey add name="foo2" key="ke-sj-eb-ok-qk-zi-nb-lu-fh-wi-zi";
8          enckey add name="bar2" key="wl-tk-64-rk-i0-zn-wk-7c-s8-a8-jk";
9          /* user-defined keys as hex data */
10         enckey replace name="foo2"
11
hexkey="e1d2e3bb4a5f6079889706b3c4d1e2f999f2c3bb4c5f6079888706f6a4b3e2da";
12         enckey replace name="bar2"
13
hexkey="98929402034568920937580293757283940109283746748392018375839201 92";
14         run;

NOTE:  Created the S3 encryption key "foo".
NOTE:  Created the S3 encryption key "bar".
NOTE:  Created the S3 encryption key "foo2".
NOTE:  Created the S3 encryption key "bar2".
NOTE:  The value for S3 encryption key "foo2" was redefined.
NOTE:  The value for S3 encryption key "bar2" was redefined.
NOTE: PROCEDURE S3 used (Total process time):
      real time           3.02 seconds
      cpu time            0.01 seconds


NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           3.34 seconds
      cpu time            0.11 seconds
```

プログラム: S3data.sas

```
/* s3data.sas */
%include "keys.sas";
```

```
proc s3;
  enckey list;

  put enckey="foo" "/home/demouser/s3/foo.txt" "/mybucket/foo.txt";
  copy srckey="foo" "/mybucket/foo.txt" enckey="bar" "/mybucket/bar.txt";
  get enckey="foo" "/mybucket/foo.txt" "/home/demouser/foo-get.txt";

  putdir enckey="foo2" "/home/demouser/s3" "/mybucket/enctest";
  getdir enckey="foo2" "/mybucket/enctest" "/home/demouser/enctest";

run;
```

プログラムの説明

s3data.sas という別のプログラムでは、S3 環境のデータを扱う際に使用する暗号化を指定できます。

s3data.sas プログラムにアクセスできるように、暗号化キーの定義を含めます。 %INCLUDE ステートメントを使用して、暗号化キーの定義とそれに関連する名前を取得します。

```
/* s3data.sas */
%include "keys.sas";
```

使用可能な暗号化キーのリストを表示します。 ENCKEY ステートメントで LIST オプションを使用して、現在定義されている暗号化キーを確認します。

```
proc s3;
  enckey list;
```

ファイルの暗号化を操作します。 PUT ステートメントは、keys.sas の foo 暗号化キーを使用して、mybucket という S3 バケットにコピーされるローカル foo.txt ファイルを暗号化します。COPY ステートメントは、mybucket の foo.txt ファイルを、bar 暗号化キーを使用して暗号化された bar.txt というファイルにコピーします。GET ステートメントは、ファイル foo.txt を mybucket からローカルディレクトリにコピーし、ファイルの名前を foo-get.txt に変更します。ファイルは foo 暗号化キーを使用して暗号化されます。

```
  put enckey="foo" "/home/demouser/s3/foo.txt" "/mybucket/foo.txt";
  copy srckey="foo" "/mybucket/foo.txt" enckey="bar" "/mybucket/bar.txt";
  get enckey="foo" "/mybucket/foo.txt" "/home/demouser/foo-get.txt";
```

S3 バケットにサブディレクトリを作成し、それを暗号化します。 PUTDIR ステートメントを使用して、mybucket にサブディレクトリ Enctest を作成します。この場所は、ローカルディレクトリ /home/demouser/s3 のコピーです。Enctest の場所は、foo2 暗号化キーを使用して暗号化されます。keys.sas では、foo2 暗号化キーは 16 進値で表されます。これは、その暗号化キーが foo2 の元の暗号化キーを置き換えたためです。

```
  putdir enckey="foo2" "/home/demouser/s3" "/mybucket/enctest";
```

暗号化された S3 ロケーションのコピーである、復号化されたローカルディレクトリを作成します。 GETDIR ステートメントを使用して、S3 ロケーション Enctest の復号化されたコピーを作成します。暗号化キー名は、データの復号化を可能にするために提供されます。データ転送時には、デフォルトでデータが暗号化されます。新しいローカルディレクトリは/home/demouser/enctest です。

```
getdir enckey="foo2" "/mybucket/enctest" "/home/demouser/enctest";

run;
```

```
1          /* s3data.sas */
2          %include "keys.sas";

NOTE:  Created the S3 encryption key "foo".
NOTE:  Created the S3 encryption key "bar".
NOTE:  Created the S3 encryption key "foo2".
NOTE:  Created the S3 encryption key "bar2".
NOTE:  The value for S3 encryption key "foo2" was redefined.
NOTE:  The value for S3 encryption key "bar2" was redefined.
NOTE: PROCEDURE S3 used (Total process time):
      real time           3.02 seconds
      cpu time            0.01 seconds


17         proc s3;
18         enckey list;
19         put enckey="foo" "/home/demouser/foo.txt" "/mybucket/foo.txt";
20         copy srckey="foo" "/mybucket/foo.txt" enckey="bar" "/mybucket/bar.txt";
21         get enckey="foo" "/mybucket/foo.txt" "/home/demouser/foo-get.txt";
22         putdir enckey="foo2" "/home/demouser/s3" "/mybucket/enctest";
23         getdir enckey="foo2" "/mybucket/enctest" "/home/demouser";
24         run;

Defined S3 Encryption Keys
Name      Type       Details
_DEFAULT_ KMS
bar       KMS        22f87852-0ak1-5xy2-01j1-l852g809gx4q
bar2      SSE-C      HEX AES256
foo       KMS        98v27390-1si1-8sc9-38k0-k893j354nw5g
foo2      SSE-C      HEX AES256
NOTE: Writing directory /home/demouser/s3 to S3 path /mybucket/enctest
NOTE:  Put: /home/demouser/s3/s3data.sas -> /mybucket/enctest/s3data.sas
NOTE:  Put: /home/demouser/s3/foo.txt -> /mybucket/enctest/foo.txt
NOTE:  Put: /home/demouser/s3/keys.log -> /mybucket/enctest/keys.doc.log
NOTE:  Put: /home/demouser/s3/keys.sas -> /mybucket/enctest/keys.sas
NOTE: Writing contents of S3 path /mybucket/enctest to local path /home/demouser
NOTE: PROCEDURE S3 used (Total process time):
      real time           5.53 seconds
      cpu time            2.35 seconds


NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           8.82 seconds
      cpu time            2.47 seconds
```

例 4: PROC S3 によるカスタムリージョンの定義

要素: REGION ADD ステートメント

詳細

この例では、PROC S3 を使ってカスタムリージョンを定義する方法を説明します。

プログラム

```
proc s3;

  region add host="s3-us-west-1.amazonaws.com"
    name="south-west"
    sslrequired replace;
run;
```

プログラムの説明

デフォルトの構成ファイルを使用して S3 に接続します。 PROC S3 ステートメントを使用して、ホームディレクトリの .tks3.conf ファイルに指定されている接続オプションを使用して S3 に接続します。これが、デフォルトの構成ファイルの場所になります。

```
proc s3;
```

新しいカスタムリージョンを定義します。 REGION ADD ステートメントを使用して、カスタムリージョン south-west を作成します。SSLREQUIRED 引数は、S3 環境と通信時に SSL が必要であることを示します。REPLACE 引数は、カスタムリージョンがすでに定義されている場合、その定義を上書きすることを指定します。

```
  region add host="s3-us-west-1.amazonaws.com"
    name="south-west"
    sslrequired replace;
run;
```

プログラム

```
proc s3;

  region add host="s3.us-west-004.backblazeb2.com"
    name="backblaze-west"
```

```
    region="us-west-004"
    sslrequired replace;
run;
```

プログラムの説明

デフォルトの構成ファイルを使用して S3 に接続します。 PROC S3 ステートメントは、ホームディレクトリの .tks3.conf ファイルで指定された接続オプションを使用して S3 に接続します。

```
proc s3;
```

新しいカスタムリージョンを定義します。 REGION ADD ステートメントを使用して、カスタムリージョン us-west-004 を作成します。前のプログラムと同様に、SSLREQUIRED 引数は、S3 環境と通信するときに SSL が必要であることを示します。REPLACE 引数は、カスタムリージョンがすでに定義されている場合、その定義を上書きすることを指定します。通常、NAME=に同じ値を使用すると競合が発生する場合は、オプションの REGION=引数を使用してリージョンコードを指定します。

```
  region add host="s3.us-west-004.backblazeb2.com"
    name="backblaze-west"
    region="us-west-004"
    sslrequired replace;
run;
```

53

SCAPROC プロシジャ

概要: SCAPROC プロシジャ

SCAPROC プロシジャの機能

SCAPROC プロシジャは、SAS コードアナライザーを実装します。これは、実行中に SAS ジョブからの入力、出力、マクロシンボルの使用に関する情報を取得します。SAS コードアナライザーは、この情報と元の SAS ファイルにある情報を、ユーザー指定のファイルへ書き込みます。SCAPROC プロシジャは、個別のジョブを同時に実行可能なグリッド対応ジョブを生成することもできます。オペレーティングシステムのコマンドライン、または **SAS エディター**ウィンドウの SAS コードで SCAPROC プロシジャを発行できます。

次のコマンドはオペレーティングシステムのコマンドラインから SAS ジョブと SAS コードアナライザーを実行します。

sas yourjob.sas -initstmt "proc scaproc; record '*yourjob.txt*' ; run;"

sas
SAS を起動するためにサイトで使用されるコマンドです。

yourjob.sas
分析する SAS ジョブの名前です。

yourjob.txt
SAS コードのコピーが含まれるファイルの名前です。ファイルには、入力と出力情報、マクロシンボルの使用、およびジョブの他の情報を示すために挿入されるコメントも含まれます。SAS コードでの PROC SCAPROC の発行の詳細については、例を参照してください。

特記事項

グリッド対応ジョブの一部のタスクは、前のタスクとの依存関係があります。PROC SCAPROC は、前のタスクとの依存関係に基づいて、これらのタスクを組み合わせて並べ替えます。タスクを組み合わせて同一の作業単位でサブミットすることにより、タスクのより迅速な処理が可能です。GRID オプションの NOOPTIMIZE 引数は、グリッド対応ジョブのタスクの組み合わせと並べ替えを無効にします。

GRID ステートメントを実行するには、SAS Grid Manager または SAS/CONNECT のライセンスが必要です。SAS Grid Manager では、分散マシンのグリッドで生成グリッドジョブを実行できます。SAS/CONNECT では、1 つの対称型マルチプロセッシング(SMP)マシンの並列 SAS セッションで生成グリッドジョブを実行できます。

概念: SCAPROC プロシジャ

グロバールステートメントの処理

生成されたグリッドジョブの各リモートセッションにサブミットする一連のステートメントをマークできます。一連のステートメントの前に/* SCAPROC GLOBAL BEGIN */コメントを、後ろに/* SCAPROC GLOBAL END */コメントを加えます。PROC SCAPROC により、ジョブのステップのいずれかがサブミットされる前に、マークしたセッション内のステートメントが各リモートセッションにサブミットされます。

次の例では、PROC SCAPROC は、各 PROC SUMMARY ステートメントで/* SCAPROC GLOBAL BEGIN */コメントと/* SCAPROC GLOBAL END */コメントの間のステートメントをサブミットします。

```
/* SCAPROC GLOBAL BEGIN */;
   libname one "SAS-library-1";
   libname two "SAS-library-1";
  %let year=2018;
/* SCAPROC GLOBAL END */;

Proc summary data=one.sales;
  where year=&year
  class month;
  var sales;
  output out=sasuser.sales;
run;
Proc summary data=two.expenses;
  where year=&year
  class month;
  var expenses;
  output out=sasuser.expenses;
run;
```

生成されたグリッドジョブと併用できるあらゆる SAS ステートメントを付加できます。あらゆる大文字または小文字のテキスト、または大文字と小文字が混在するテキストを使用できます。

構文: SCAPROC プロシジャ

PROC SCAPROC;

RECORD *filespec* <ATTR> <OPENTIMES> <INTCON> <EXPANDMACROS> <GRID *filespec* <RESOURCE "*resource name*"> <INHERITLIB> <NOOPTIMIZE> >;

WRITE;

ステートメント	タスク	例
“PROC SCAPROC ステートメント”	SAS コードアナライザーを実装します。	Ex. 1, Ex. 2
“RECORD ステートメント”	SAS コードアナライザーの出力情報を含めるファイル名またはファイル参照名を指定します。	Ex. 1, Ex. 2
“WRITE ステートメント”	記録ファイルへ情報を出力します。	Ex. 1

PROC SCAPROC ステートメント

SAS が SAS ジョブと SAS コードアナライザーを実行するように指定します。

例: “例 1: 記録ファイルの指定” (1952 ページ)
“例 2: グリッドジョブジェネレーターの指定” (1953 ページ)

構文

PROC SCAPROC;

RECORD ステートメント

SAS コードアナライザーの出力情報を含めるファイル名またはファイル参照名を指定します。

例: “例 1: 記録ファイルの指定” (1952 ページ)
“例 2: グリッドジョブジェネレーターの指定” (1953 ページ)

構文

RECORD*filespec* <ATTR> <OPENTIMES> <INTCON> <EXPANDMACROS>
<GRID *filespec* ><RESOURCE "*resource name*"><INHERITLIB><NOOPTIMIZE> ;

必須引数

filespec
SAS コードアナライザーの出力情報を含めるファイルを指定する、物理ファイル名(引用符で囲む)またはファイル参照名を示します。出力情報は、元の SAS ソースとジョブに関する情報を含むコメントです。出力コメントの詳細については、“結果” (1947 ページ)を参照してください。

オプション引数

ATTR
入力データセットと入力ビューにある変数に関する追加情報を書き込みます。

OPENTIMES
入力データセットが開かれた時間、サイズ、物理ファイル名を書き込みます。

INTCON
テーブル一貫性制約に関する出力情報です。

EXPANDMACROS
マクロ呼び出しを個別のタスクに展開します。

GRID

filespec
グリッドジョブジェネレーターの出力を含めるファイルを指す物理ファイル名(引用符で囲む)またはファイル参照名を示します。

RESOURCE “*resource name*”
grdsvs_enable 関数呼び出しで使用するリソースを示します。デフォルトは SASApp です。

INHERITLIB
グリッド対応ジョブで SIGNON ステートメントに INHERITLIB=(USER)を追加します。デフォルトでは、各リモートセッションに USER ライブラリ(指定されている場合)が割り当てられます。

注: INHERITLIB は、すべてのリモートセッションで USER ライブラリがグローバルにアクセス可能でない場合にのみ使用します。

NOOPTIMIZE
グリッド対応ジョブのタスクの組み合わせと並べ替えを無効にします。

WRITE ステートメント

記録ファイルへの出力情報を示します。

例: “例 1: 記録ファイルの指定” (1952 ページ)

構文

WRITE;

引数なし

WRITE ステートメントは、記録ファイルが RECORD ステートメントにより指定されている場合、SAS コードアナライザーによる記録ファイルへの情報の書き込みを指定します。グリッドジョブジェネレーターが指定されている場合、グリッドジョブジェネレーターもこの時点で実行されます。SAS を終了する場合も、SAS コードアナライザーによって指定の記録ファイルに情報が書き込まれます。

結果: SCAPROC プロシジャ

結果

次のリストに、SAS コードアナライザーにより、PROC SCAPROC で指定される記録ファイルへ書き込まれるコメントの説明を示します。出力コメントは、記録ファイ

ル内で**/***と***/**コメントタグによって区切られています。記録ファイル参照時にわかりやすいように、次にその形式を示します。

/* JOBSPLIT: DATASET INPUT|OUTPUT|UPDATE SEQ|MULTI *name* */
データセットが読み込み、書き込み、更新のために開かれたことを示します。

INPUT
データセットが読み込まれたことを示します。

OUTPUT
データセットが書き込まれたことを示します。

UPDATE
データセットが更新されたことを示します。

SEQ
データセットが順次アクセスのために開かれたことを示します。

MULTI
データセットがマルチパスアクセスのために開かれたことを示します。

name
データセットの名前を示します。

/* JOBSPLIT: CATALOG INPUT|OUTPUT|UPDATE *name* */
カタログが読み込み、書き込み、更新のために開かれたことを示します。

INPUT
カタログが読み込まれたことを示します。

OUTPUT
カタログが書き込まれたことを示します。

UPDATE
カタログが更新されたことを示します。

name
カタログ名を指定します。

/* JOBSPLIT: FILE INPUT|OUTPUT|UPDATE *name* */
外部ファイルが読み込み、書き込み、更新のために開かれたことを示します。

INPUT
ファイルが読み込まれたことを示します。

OUTPUT
ファイルが書き込まれたことを示します。

UPDATE
ファイルが更新されたことを示します。

name
ファイルの名前を示します。

/* JOBSPLIT: ITEMSTOR INPUT|OUTPUT|UPDATE *name* */
ITEMSTOR が読み込み、書き込み、更新のために開かれたことを示します。

INPUT
ITEMSTOR が読み込まれたことを示します。

OUTPUT
ITEMSTOR が書き込まれたことを示します。

UPDATE
ITEMSTOR が更新されたことを示します。

name
ITEMSTOR の名前を示します。

/* JOBSPLIT: OPENTIME *name* DATE:*date* PHYS:*phys* SIZE:*size* */
データセットが入力のために開かれたことを示します。ファイルの OPENTIME と SIZE が SAS によって書き込まれます。

name
データセットの名前を示します。

DATE
データセットが開かれた日付と時刻を示します。DATE に返される値はファイルの作成日時ではありません。

PHYS
開かれたデータセットの完全な物理名を示します。

SIZE
データセットのサイズをバイト単位で示します。

/* JOBSPLIT: ATTR *name* INPUT|OUTPUT VARIABLE:*variable name* TYPE:CHARACTER|NUMERIC LENGTH:*length* LABEL:*label* FORMAT:*format* INFORMAT:*informat* */
データセットをクローズした際、SAS がそれを再度開いて、各変数の属性を書き込むことを指定します。各変数に対して、ATTR 行が 1 つずつ生成されます。

name
データセットの名前を示します。

INPUT
データセットが読み込まれたことを示します。

OUTPUT
データセットが書き込まれたことを示します。

VARIABLE
現在の変数の名前を示します。

TYPE
変数が文字または数値であるか示します。

LENGTH
変数の長さをバイト単位で示します。

LABEL
変数ラベルを示します(ある場合)。

FORMAT
変数の出力形式を示します(ある場合)。

INFORMAT
変数の入力形式を示します(ある場合)。

/* JOBSPLIT: SYMBOL SET|GET *name* */
マクロシンボルがアクセスされたことを示します。

SET
シンボルが設定されたことを示します。たとえば、%let sym1=sym2 というコードではシンボル **sym1** が設定されます。

GET
シンボルが取得されたことを示します。たとえば、a="&sym"というコードではシンボル **sym** が取得されます。

name
シンボルの名前を示します。

/* JOBSPLIT: ELAPSED *number* */
相対的なタスクの実行回数を決定するために使用する数値を示します。

number
相対的なタスクの実行回数を決定するために使用する数値を示します。

/* JOBSPLIT: USER *useroption* */
SAS によりグリッドジョブコードと USER オプションを併用して、単一レベルのデータセット名が Work ライブラリに存在できるようにすることを示します。

useroption
コード実行中に使用される値を示します。

/* JOBSPLIT: _DATA_ */
予約データセット名_DATA_を使用することを示します。

/* JOBSPLIT: _LAST_ */
予約データセット名_LAST_を使用することを示します。

/* JOBSPLIT: PROCNAME procname|DATASTEP */
このステップの SAS プロシジャの名前または DATA ステップを示します。

/* JOBSPLIT: LIBNAME <libname options> */
LIBNAME ステートメントに付与されたか、または内部設定された LIBNAME オプションを示します。

/* JOBSPLIT: SYSSCP <sysscp> */
SAS ジョブが実行されたときの SYSSC 自動マクロ変数の値を示します。

/* JOBSPLIT: JOBSTARTTIME <datetime> */
ジョブ開始時の日時を記録します。

/* JOBSPLIT: JOBENDTIME <datetime> */
ジョブ終了時の日時を記録します。

/* JOBSPLIT: TASKSTARTTIME <datetime> */
タスク開始時の日時を記録します。

/* JOBSPLIT: INTCON <table name> <INPUT | OUTPUT> NAME:<name> TYPE:<UNIQUE|CHECK|NOTNULL|PRIMARY|FOREIGN|REFERENTIAL> VARIABLES:<names> WHERE:<where clause> REFERENCE: ONDELETE: ONUPDATE: MESSAGE:<msg> MESSAGETYPE:<USER> */
table name
テーブルの名前を示します。

INPUT
テーブルが入力のために開かれたことを示します。

OUTPUT
テーブルが出力のために開かれたことを示します。

NAME
一貫性制約の名前を示します。

TYPE
一貫性制約のタイプを示します。

表 53.1 各タイプの一貫性制約

タイプ	一貫性制約
UNIQUE	一意
CHECK	チェック
NOTNULL	null でない
PRIMARY	主キー
FOREIGN	外部キー
REFERENTIAL	このテーブルを参照する別のテーブルの FOREIGN 一貫性制約

VARIABLES
一貫性制約に関連する変数を示します。その他の場合は空欄になります。

WHERE
一貫性制約に対し WHERE 表現を示します。その他の場合は空欄になります。

REFERENCE
次の一貫性制約の中の 1 つを示します。

- FOREIGN 一貫性制約については、外部キーが参照するテーブルを示します。
- REFERENTIAL 一貫性制約については、外部キーを含んでいるテーブルを示します。
- その他の場合は空欄になります。

ONDELETE
FOREIGN および REFERENTIAL 一貫性制約に対する ON DELETE 参照アクションを示します。その他の場合は空欄になります。

ONUPDATE
FOREIGN および REFERENTIAL 一貫性制約に対する ON UPDATE 参照アクションを示します。その他の場合は空欄になります。

MESSAGE
エラーメッセージのテキストがあれば、それを示します。

MESSAGETYPE
MESSAGETYPE=USER が指定された場合、USER を示します。その他の場合は空欄になります。

例: SCAPROC プロシジャ

例 1: 記録ファイルの指定

要素: RECORD ステートメント
WRITE ステートメント

この例では、記録ファイル**'record.txt'**を指定し、SAS コードアナライザーから記録ファイルに情報を書き込みます。

プログラム

```
proc scaproc;
  record 'record.txt';
run;

data a;
  do i = 1 to 100000;
    j = cos(i);
    output;
  end;
run;

proc print data=a(obs=25);
run;

proc means data=a;
run;

proc scaproc;
  write;
run;
```

出力

アウトプット 53.1　record.txt ファイルの内容

```
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS
Temporary Files\_TD1252 */
/* JOBSPLIT: ELAPSED 3984  */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

data a;
    do i = 1 to 1000000;
        j = cos(i);
        output;
    end;
run;


/* JOBSPLIT: ITEMSTOR INPUT SASUSER.TEMPLAT */
/* JOBSPLIT: ITEMSTOR INPUT SASHELP.TMPLMST */
/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS
Temporary Files\_TD1252 */
/* JOBSPLIT: ELAPSED 5187  */
/* JOBSPLIT: PROCNAME PRINT */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
proc print data=a(obs=25);
run;


/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS
Temporary Files\_TD1252 */
/* JOBSPLIT: FILE OUTPUT C:\winnt\profiles\userid\record.txt */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: ELAPSED 2750  */
/* JOBSPLIT: PROCNAME MEANS */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
proc means data=a;
run;


/* JOBSPLIT: END */
```

例 2: グリッドジョブジェネレーターの指定

要素:

RECORD ステートメント
GRID ステートメント

詳細

この例では、SAS コードアナライザーから 1.txt という名前のファイルに情報を書き込みます。サンプルコードでは、グリッドジョブジェネレーターも実行し、その情報を 1.grid という名前のファイルに書き込みます。このサンプルコードには、次のコードを含む終了ステートメントはありません。

```
proc scaproc;
  write;
run;
```

SAS の終了時に、PROC SCAPROC により自動的に保留中の RECORD や GRID ステートメントが実行されます。

GRID ステートメントを実行するには、SAS Grid Manager または SAS/CONNECT のライセンスが必要です。SAS Grid Manager では、分散マシンのグリッドで生成グリッドジョブを実行できます。SAS/CONNECT では、1 つの対称型マルチプロセッシング(SMP)マシンの並列 SAS セッションで生成グリッドジョブを実行できます。

プログラム

```
proc scaproc;
  record '1.txt' grid '1.grid';
run;

data a;
  do i = 1 to 100000;
    j = cos(i);
    output;
  end;
run;

proc print data=a(obs=25);
run;

proc means data=a;
run;
```

出力

アウトプット 53.2　1.txt ファイルの内容

```
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS
Temporary Files\_TD1252 */
/* JOBSPLIT: ELAPSED 375  */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

data a;
    do i = 1 to 1000000;
        j = cos(i);
        output;
    end;
run;


/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS
Temporary Files\_TD1252 */
/* JOBSPLIT: ELAPSED 46  */
/* JOBSPLIT: PROCNAME PRINT */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
proc print data=a(obs=25);
run;


/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK ENGINE V9 PHYS C:\DOCUME~1\userid\LOCALS~1\Temp\SAS
Temporary Files\_TD1252 */
/* JOBSPLIT: FILE OUTPUT C:\WINNT\Profiles\userid\1.txt */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: ELAPSED 81453  */
/* JOBSPLIT: PROCNAME MEANS */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
proc means data=a;
run;

/* JOBSPLIT: END */
```

54

SCOREACCEL プロシジャ

概要: SCOREACCEL プロシジャ

SCOREACCEL プロシジャの機能

PROC SCOREACCEL では、モデルのパブリッシュとスコアリングのために CAS サーバーと外部データソースへのインターフェイスを 提供する対話型プロシジャです。

- モデルは、DATA ステップコード、DS2 コード、または分析ストアの形式でサポートされます。
- モデルコードは、セッションローカルまたはグローバル CAS テーブルにパブリッシュし、CAS で実行できます。セッションローカルとは、PROC SCOREACCEL を実行している現在の CAS セッションにのみアクセスできることを意味します。
- また、モデルを外部データソースにパブリッシュすることもできます。“In-Database モデルスコアリングでサポートされるプラットフォーム” (1959 ページ)を参照してください。モデルが外部データソースにパブリッシュされた後、モデルを実行すると、SAS Embedded Process が呼び出されます。
- モデルは、DELETEMODEL ステートメントを使用して、CAS または外部データソースから削除できます。
- PROC SCOREACCEL は、CAS アクションを呼び出してタスクを完了します。PROC SCOREACCEL を使用する代わりに、モデルのパブリッシュとスコアリングアクションセットを直接使用できます。
- サポートされているスコアリングモデルとその他の情報の一覧については、“In-Database Model Scoring” (*SAS In-Database Products: User’s Guide*)を参照してください。
- PROC SCOREACCEL の代替である PROC ACCELERATOR については、“PROC ACCELERATOR と PROC SCOREACCEL の機能比較” (94 ページ)を参照してください。PROC ACCELERATOR は、Databricks および SingleStore でサポートされています。

概念: SCOREACCEL プロシジャ

In-Database モデルスコアリングでサポートされるプラットフォーム

次のテーブルは、モデルのスコアリングを実行できる CAS 以外の処理環境を示しています。

表 54.1 モデルを実行するためのプラットホーム

RUNMODEL 処理	RUNMODEL EXTTYPE=値 CASLIB 値	PUBLISHMODEL モデルのストレージ	PUBLISHMODEL および DELETEMODEL EXTTYPE=値 CASLIB 値	ドキュメント
AWS Databricks	EXTTYPE=SPARK CASLIB で、srcType=spark と platform=databricks を指定します。	Spark テーブル [1]	EXTTYPE=SPARK CASLIB で、srcType=spark と platform=databricks を指定します。	概要 例
		Amazon S3(非推奨)[2]	EXTTYPE=FILESYSTEM CASLIB で、srcType=spark と platform=databricks を指定します。	概要 例
Azure Databricks	EXTTYPE=SPARK CASLIB で、srcType=spark と platform=databricks を指定します。	Spark テーブル [1]	EXTTYPE=SPARK CASLIB で、srcType=spark と platform=databricks を指定します。	概要 例
		ADLS Gen2(非推奨)[2, 3]	EXTTYPE=FILESYSTEM CASLIB で srcType=adls を指定します。	概要 例
Azure Synapse	EXTTYPE=SPARK CASLIB で、srcType=spark と platform=Synapse を指定します。	SQL Server テーブル [4]	EXTTYPE=SYNAPSE	概要 例

RUNMODEL 処理	RUNMODEL EXTTYPE=値 CASLIB 値	PUBLISHMODEL モデルのストレージ	PUBLISHMODEL および DELETEMODEL EXTTYPE=値 CASLIB 値	ドキュメント
			CASLIB で srcType=jdbc を指定し、その他必要な引数を指定します。	
		ADLS Gen2(非推奨)[3]	EXTTYPE=FILESYSTEM CASLIB で srcType=adls を指定します。	概要 例
Hadoop	EXTTYPE=HADOOP CASLIB で srcType=hadoop を指定します。	Hive テーブル [5]	EXTTYPE=HADOOP CASLIB で srcType=hadoop を指定します。	概要 例
		Hadoop(非推奨)	EXTTYPE=HADOOP CASLIB で srcType=hadoop を指定します。	概要
SingleStore(SAS SpeedyStore)[6]	EXTTYPE=SINGLESTORE CASLIB で srcType=singlestore を指定します。	SingleStore	EXTTYPE=SINGLESTORE CASLIB で srcType=singlestore を指定します。	概要 例
Teradata	EXTTYPE=TERADATA CASLIB で srcType=teradata を指定します。	Teradata	EXTTYPE=TERADATA CASLIB で srcType=teradata を指定します。	概要 例

1 Databricks の場合、2023.10 以降、モデルを Spark テーブルにパブリッシュできるようになり、この方法が推奨されます。
2 DBFS REST API を使用してモデルを Databricks にパブリッシュすることはお勧めしません。
3 ADLS Gen1 と Azure BLOB ストレージはサポートされていません。
4 Synapse の場合、2023.12 以降、モデルを SQL Server テーブル(専用 SQL プールテーブルとも呼ばれます)にパブリッシュできるようになり、この方法が推奨されます。
5 Hadoop の場合、2023.09 以降、モデルを Hive テーブルにパブリッシュできるようになり、この方法が推奨されます。
6 SAS SpeedyStore 製品のライセンスを取得して配置した場合、In-database モデルスコアリングがサポートされます。詳細については、*"SAS SpeedyStore データコネクタ" (SAS Cloud Analytic Services: ユーザーガイド)*を参照してください。In-database モデルスコアリングは、SingleStore 標準データコネクタではサポートされていません。

データソースおよびステートメント別の引数リスト

EXTTYPE=CAS

- DELETEMODEL ステートメント (1965 ページ)

  CASLIB=, DATABASE=, DELETEGLOBAL=, EXTTYPE=, MODELNAME=, MODELTABLE=, PERSISTTABLE=, PROMOTETABLE=, REPLACETABLE=, SCHEMA=

- PUBLISHMODEL ステートメント (1973 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORMATFILE=, FORMATITEMSTOREFILE=, KEEPLIST=, MODELNAME=, MODELNOTES=, MODELTABLE=, MODELTYPE=, MODELUUID=, OUTDIR=, PERSISTTABLE=, PROGRAMFILE=, PROMOTETABLE=, PUBLISHGLOBAL=, REPLACEMODEL=, REPLACETABLE=, SCHEMA=, STORETABLES=, VARXMLFILE=

- RUNMODEL ステートメント (1987 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORMATFILE=, FORMATITEMSTOREFILE=, KEEPLIST=, MODELNAME=, MODELNOTES=, MODELTABLE=, MODELTYPE=, MODELUUID=, OUTDIR=, PERSISTTABLE=, PROGRAMFILE=, PROMOTETABLE=, PUBLISHGLOBAL=, REPLACEMODEL=, REPLACETABLE=, SCHEMA=, STORETABLES=, VARXMLFILE=

EXTTYPE=DATABRICKS

- DELETEMODEL ステートメント (1965 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, MODELDATABASE=, MODELNAME=, SCHEMA=

- PUBLISHMODEL ステートメント (1973 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORMATFILE=, FORMATITEMSTOREFILE=, KEEPLIST=, MODELDATABASE=, MODELNAME=, MODELTYPE=, OUTDIR=, PROGRAMFILE=, REPLACEMODEL=, SCHEMA=, STOREFILES=, STORETABLES=, VARXMLFILE=

EXTTYPE=FILESYSTEM (非推奨)

- DELETEMODEL ステートメント (1965 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, MODELDIR=, MODELNAME=, PASSWORD=, SCHEMA=, TIMEOUT=

- PUBLISHMODEL ステートメント (1973 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORMATFILE=, FORMATITEMSTOREFILE=, KEEPLIST=, MODELDIR=, MODELNAME=, MODELTYPE=, OUTDIR=, PASSWORD=, PROGRAMFILE=, REPLACEMODEL=, SCHEMA=, STOREFILES=, STORETABLES=, TIMEOUT=, VARXMLFILE=

EXTTYPE=HADOOP

- DELETEMODEL ステートメント (1965 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, MODELDATABASE=, MODELDIR=, MODELNAME=, SCHEMA=

- PUBLISHMODEL ステートメント (1973 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORMATFILE=, FORMATITEMSTOREFILE=, KEEPLIST=, MODELDATABASE=, MODELDIR=, MODELNAME=, MODELTYPE=, OUTDIR=, PROGRAMFILE=, REPLACEMODEL=, SCHEMA=, STOREFILES=, STORETABLES=, VARXMLFILE=

- RUNMODEL ステートメント (1987 ページ)

  CASLIB=, DATABASE=, DBMAXTEXT=, EXTTYPE=, FORCEOVERWRITE=, INDATASET=, INTABLE=, KEEPLISTCOLUMNS=, KEEPLISTFILE=, MODELDATABASE=, MODELDIR=, MODELNAME=, OUTDATASET=, OUTSCHEMA=, OUTTABLE=, PROPERTIES=, SCHEMA=, TRACE

EXTTYPE=SINGLESTORE

- DELETEMODEL ステートメント (1965 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, MODELNAME=, SCHEMA=

- PUBLISHMODEL ステートメント (1973 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORMATFILE=, FORMATITEMSTOREFILE=, KEEPLIST=, MODELNAME=, MODELNOTES=, MODELTYPE=, MODELUUID=, OUTDIR=, PROGRAMFILE=, REPLACEMODEL=, SCHEMA=, STOREFILES=, STORETABLES=, VARXMLFILE=

- RUNMODEL ステートメント (1987 ページ)

  CASLIB=, DATABASE=, EPOPTIONS=, EXTTYPE=, INQUERY=, INTABLE=, MODELNAME=, OUTTABLE=, OUTTABLEOPTIONS=, SCHEMA=

EXTTYPE=SPARK

- RUNMODEL ステートメント (1987 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORCEOVERWRITE= , INDATASET=, INTABLE=, KEEPLISTCOLUMNS=, KEEPLISTFILE=, MODELDATABASE=, MODELDIR=, MODELNAME=, OUTDATASET=, OUTSCHEMA=, OUTTABLE=, PROPERTIES=, SCHEMA=, TRACE

EXTTYPE=SYNAPSE

- DELETEMODEL ステートメント (1965 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, MODELNAME=, MODELSCHEMA=, SCHEMA=

- PUBLISHMODEL ステートメント (1973 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORMATFILE=, FORMATITEMSTOREFILE=, KEEPLIST=, MODELNAME=, MODELSCHEMA=, MODELTYPE=, OUTDIR=, PROGRAMFILE=, REPLACEMODEL=, SCHEMA=, STOREFILES=, STORETABLES=, VARXMLFILE=

EXTTYPE=TERADATA

- DELETEMODEL ステートメント (1965 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, MODELNAME=, MODELTABLE=, SCHEMA=

- PUBLISHMODEL ステートメント (1973 ページ)

  CASLIB=, DATABASE=, EXTTYPE=, FORMATFILE=, FORMATITEMSTOREFILE=, KEEPLIST=, MODELNAME=, MODELNOTES=, MODELTABLE=, MODELTYPE=, MODELUUID=, OUTDIR=, PROGRAMFILE=, REPLACEMODEL=, SCHEMA=, STOREFILES=, STORETABLES=, VARXMLFILE=

- RUNMODEL ステートメント (1987 ページ)

CASLIB=, DATABASE=, EPOPTIONS=, EXTTYPE=, INQUERY=, INTABLE=, MODELNAME=, MODELTABLE=, OUTTABLE=, OUTTABLEOPTIONS=, SCHEMA=,

CAS サーバー上のモデルテーブルへのモデルのパブリッシュ

モデルを CAS にパブリッシュするには、モデルテーブルの行を追加または置換する必要があります。モデルを既存のモデルテーブルにパブリッシュするには、まずモデルテーブルを CAS サーバーにロードする必要があります。指定されたモデルテーブルが CAS サーバーにロードされている場合は、モデルテーブルが更新されます。それ以外の場合は、新しいモデルテーブルが作成されます。

モデルは、CAS のグローバルモデルテーブルに直接パブリッシュできます。PUBLISHMODEL PUBLISHGLOBAL=パラメーターが TRUE に設定されている場合、モデルは CAS のグローバルモデルテーブルにパブリッシュされます。

命名要件

データソースの列とテーブルの名前は、ベンダーの命名規則に従う必要があります。

構文: SCOREACCEL プロシジャ

PROC SCOREACCEL<*optional-argument*>;

PUBLISHMODEL*required-arguments*<*publish-model-optional-arguments*>;

RUNMODEL*required-arguments*<*run-model-optional-arguments*>;

DELETEMODEL*required-arguments*<*delete-model-optional-arguments*>;

ステートメント	タスク	例
“PROC SCOREACCEL ステートメント”	CAS または外部データソースでモデルをパブリッシュ、実行、または削除	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10
“PUBLISHMODEL ステートメント”	CAS または外部データソースでモデルをパブリッシュ	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10

ステートメント	タスク	例
“RUNMODEL ステートメント”	CAS または外部データソースでモデルを実行	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10
“DELETEMODEL ステートメント”	CAS または外部データソースからモデルを削除	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10

PROC SCOREACCEL ステートメント

CAS または外部データソースでモデルをパブリッシュ、実行、または削除します。

例:

“例 1: CAS で DATA ステップモデルをパブリッシュ、実行、および削除する” (2000 ページ)
“例 2: Teradata でモデルをパブリッシュ、実行、および削除する” (2003 ページ)
“例 3: モデルを Hive テーブルにパブリッシュして Cloudera で実行する” (2006 ページ)
“例 4: モデルを Spark テーブルにパブリッシュして Databricks on AWS で実行する” (2009 ページ)
“例 5: モデルを Spark テーブルにパブリッシュして Azure Databricks で実行する” (2011 ページ)
“例 6: モデルを Amazon S3 にパブリッシュして Databricks on AWS で実行する” (2013 ページ)
“例 7: モデルを ADLS にパブリッシュして Azure Databricks で実行する” (2015 ページ)
“例 8: モデルを SQL Server テーブルにパブリッシュして Azure Synapse で実行する” (2017 ページ)
“例 9: モデルを ADLS にパブリッシュして Azure Synapse で実行する” (2019 ページ)
“例 10: SingleStore でモデルをパブリッシュ、実行、および削除する” (2023 ページ)

構文

PROC SCOREACCEL<SESSREF=*session-reference* | SESSUUID="*session-uuid*">;

引数

SESSREF=*session-reference* | SESSUUID="*session-uuid*"
CAS セッションの名前、または接続する既存の CAS セッションのユニバーサル一意識別子(UUID)を指定します。オプションが指定されていない場合、自動 CAS セッション CASAUTO が使用されます。

SESSREF=*session-reference*
接続先の CAS セッションの名前を指定します。

SESSUUID="*session-uuid*"
既存の CAS セッションの汎用一意識別子(UUID)を指定します。このオプションで指定する前に、既存のセッションから SESSUUID を取得する必要があります。エンジンは、UUID で識別されるセッションに接続します。

DELETEMODEL ステートメント

CAS または外部データソースからモデルを削除。

要件 Hadoop の場合、2023.09 以降、新しいバージョンの Java ランタイム環境によってセキュリティ更新が導入されました。配置、構成、使用法における変更の概要については、"Publishing and Running Models in Hadoop" (*SAS In-Database Products: User's Guide*)を参照してください。

構文

DELETEMODEL
CASLIB="*caslib*"
MODELNAME="*model-name*"
MODELTABLE="*model-table*" | "*caslib.model-table*" | "*schema.model-table*"
<*delete-model-optional-arguments*>;

必須引数

CASLIB="*casref*"
モデルがパブリッシュされるデータソースに関連付けられている caslib の名前を指定します。 PROC SCOREACCEL は、caslib を使用して、データソースに固有のオプションを取得します。たとえば、caslib は、ユーザーの資格情報やその他の接続オプションを指定できます。

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

操作 caslib 割り当てで接続オプションを指定します。プロシジャでの接続オプションの指定は非推奨です。CASLIB=オプションが推奨されており、将来的にはすべての EXTTYPE=データソースで必須になります。現在、EXTTYPE=値が FILESYSTEM、SINGLESTORE、SPARK、および SYNAPSE のモデルをパブリッシュおよび削除するには、CASLIB=オプションが必要です。

CAS または Teradata の場合、MODELTABLE=で 2 部構成のモデルテーブル名の最初の部分として指定された caslib は、このオプションの値よりも優先されます。

MODELNAME="*model-name*"
削除するモデル名を指定します。

別名 MODEL=

適用対象 CAS

Filesystem

Databricks

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

注 SingleStore の場合、SAS は各モデルをテーブルとしてパブリッシュします。Hadoop、Databricks、、および Synapse の場合、MODELDIR=オプションを省略すると、各モデルはテーブル(データソースに応じて Hive テーブル、Spark テーブル、または SQL Server テーブル)としてパブリッシュされます。これらのすべてのケースで、SAS はモデル名をテーブル名として使用し、データベース内の他のテーブルと区別するために **sasmodel_** という接頭辞を追加します。モデルを参照してパブリッシュ、削除、または実行する場合、**sasmodel_**接頭辞を含めないでください。MODELTABLE=オプションは使用しないでください。

MODELTABLE="*model-table*" | "*caslib.model-table*" | "*schema.model-table*"
CAS または Teradata からモデルを削除するときに、削除するモデルを含むテーブルの名前を指定します。

CAS の場合、モデルテーブルは 1 部または 2 部の名前で指定できます。2 部構成の名前の 1 部はモデルテーブルを含む caslib の名前として解釈されます。

適用対象 CAS

Teradata

要件 このパラメーターは、CAS または Teradata からモデルを削除するときに必要です。

モデルを CAS に DELETE するには、モデルテーブルから行を削除する必要があります。モデルを削除するには、まず CAS サーバーにモデルテーブルをロードする必要があります。

オプション引数

AUTHDOMAIN="*authentication-domain*"
Teradata へのアクセスに使用される認証情報(ユーザー名とパスワード)を含む認証ドメインの名前を指定します。

適用対象 Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

AUTHTOKEN="*authentication-token*"
Spark クラスターとの接続を確立するために使用される認証トークンを指定します。

適用対象 Hadoop

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

CLASSPATH="*class_path*"
Hadoop 呼び出しコンテキストで使用されるクラスパスを指定します。クラスパスは、フォルダーまたは個別の JAR ファイルにすることができます。Hadoop 構成フォルダーは、クラスパスに含める必要があります。

適用対象 Hadoop

注意 **このオプションはサポートされなくなりました。** 2023.09 以降、コードを変更してこのオプションを削除してください。

CLUSTERID="*cluster-id*"
Spark クラスターの識別子を指定します。

適用対象 Hadoop

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

DATABASE="*database-name*"
データベース名を指定します。

デフォルト "" (長さゼロの文字列)

適用対象 Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

注 Teradata では、このオプションはアクセスする Teradata データベースの名前を指定する接続オプションを提供します。

Hadoop、SingleStore、Spark、または Synapse の場合、DATABASE=オプションは SCHEMA=オプションの別名です。

DELETEGLOBAL=TRUE | FALSE
モデルを CAS のグローバルモデルテーブルから削除するかどうかを指定します。

TRUE
グローバルモデルテーブルからモデルを削除します。

別名 YES

FALSE
ローカル CAS セッションのモデルテーブルからモデルを削除し、グローバルテーブルは変更しないままにします。

別名 NO

デフォルト FALSE

適用対象 CAS

EXTTYPE=CAS | DATABRICKS | FILESYSTEM | HADOOP | SINGLESTORE | SYNAPSE | TERADATA
モデルが削除されるターゲット環境を指定します。

CAS
CAS のモデルテーブルからモデルを削除することを指定します。

DATABRICKS
モデルを Databricks から削除することを指定します。

FILESYSTEM
クラウドベースのファイルシステムからモデルを削除することを指定します。FILESYSTEM は、Microsoft Azure Data Lake Storage Gen2 (ADLS)および Amazon S3 をサポートしています。

注: FILESYSTEM 値は、2023.12 以降非推奨になりました。Databricks 、、および Synapse の場合、モデルをファイルシステムに発行する必要がなくなりました。

HADOOP
モデルを Hadoop サーバーから削除することを指定します。

SINGLESTORE
モデルを SingleStore クラスターから削除することを指定します。この値は 2023.05 以降でサポートされています。

SYNAPSE
モデルを Azure Synapse から削除することを指定します。この値は、SQL Server テーブルにパブリッシュされたモデルを削除するために 2023.12 以降サポートされています。

TERADATA
Teradata データベースのモデルテーブルからモデルを削除することを指定します。

別名 TARGET=

デフォルト CAS

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

MODELDATABASE="*model-database*"
モデルが保存されているデータベースを指定します。 このオプションが指定されていない場合は、caslib の SCHEMA=オプションで指定されたデータベースが使用されます。このオプションは、モデルを Hive テーブルまたは Spark テーブルに保存するためにサポートされています。MODELDIR=を指定すると、MODELDATABASE=は無視されます。

適用対象 Databricks

Hadoop

要件 モデルをパブリッシュするときに MODELDATABASE=が指定されている場合は、データベースがすでに存在している必要があります。

注 このオプションは 2023.09 以降でサポートされています。

MODELDIR="*model-directory*"
モデルディレクトリが保存されるルートフォルダーを指定します。

Hadoop の場合、2023.09 より前のリリースでは、モデルは常に HDFS にパブリッシュされていました。現在は、WebHDFS を使用して、HDFS に保存されているモデルにアクセスしています。HDFS 内のモデルにアクセスする場合は、Hadoop caslib で webHdfsUrl=オプションを指定する必要があります。ただし、この方法はお勧めできません。将来のリリースでは、HDFS へのパブリッシュはサポートされなくなります。

Hadoop の場合、2023.09 以降、モデルを Hive テーブルにパブリッシュできるようになり、この方法が推奨されます。Hive テーブルに格納されているモデルをパブリッシュ、削除、または実行するには、PROC SCOREACCEL から MODELDIR=オプションを省略します。MODELDIR=オプションを省略する場合は、Hadoop caslib の SERVER=パラメーターで Hive サーバーを指定する必要があります。

適用対象 Filesystem

Hadoop

MODELSCHEMA="*model-schema*"
保存されたているモデルのスキーマを指定します。 このオプションは、SQL Server テーブルにモデルを保存するためにサポートされています。MODELDIR=を指定すると、MODELSCHEMA=は無視されます。

適用対象 Synapse

要件 Synapse の場合、SQL Server テーブルに格納されているモデルを削除、パブリッシュ、または実行するときに、MODELSCHEMA=オプションを指定する必要があります。

注 このオプションは 2023.12 以降でサポートされています。

PASSWORD="*password*"
ADLS、Hadoop サーバーまたは Teradata サーバーのユーザー ID のパスワードです。ADLS では、これはシークレットと呼ばれる値です。

別名 PASS=

PASSWD=

PWD=

適用対象 Filesystem

Hadoop

Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

PERSISTTABLE=TRUE | FALSE
モデルの削除の結果として更新されたモデルテーブルを、テーブルに関連付けられた caslib データソースに保存するかどうかを指定します。

TRUE
更新されたモデルテーブルをデータソースに保存します。

別名 YES

FALSE
更新されたモデルテーブルをデータソースに保存しません。

別名 NO

デフォルト FALSE

適用対象 CAS

要件 モデルテーブルが caslib データソースにすでに存在する場合は、テーブルを保存するために REPLACETABLE=TRUE も指定する必要があります。

参照項目: REPLACETABLE=オプション

PROMOTETABLE=TRUE | FALSE
モデルの削除の結果として更新されたモデルテーブルを CAS サーバーのグローバルスコープにプロモートする必要があるかどうかを指定します。

TRUE
更新されたモデルテーブルをグローバルスコープにプロモートします。

別名 YES

FALSE
更新されたモデルテーブルをグローバルスコープにプロモートしません。

別名 NO

デフォルト FALSE

適用対象 CAS

REPLACETABLE=TRUE | FALSE
モデルの削除の結果として生じる既存のモデルテーブルが、更新されたモデルテーブルが caslib データソースに保存される際に置換されるのを許可するかどうかを指定します。

TRUE
モデルテーブルを置き換えます。

別名 YES

FALSE
モデルテーブルを置き換えません。

別名 NO

デフォルト TRUE

適用対象 CAS

要件 モデルテーブルを caslib データソースに保存するには、PERSISTTABLE=TRUE も指定する必要があります。

参照項目: PERSISTTABLE=オプション

SCHEMA="*schema-name*"
データベースまたはスキーマの名前を指定します。

適用対象 Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

注 Teradata では、このオプションは、Teradata データベースが Teradata テーブルを修飾するために使用する接続オプションを提供します。

Hadoop、SingleStore、Spark、または Synapse の場合、DATABASE=オプションは SCHEMA=オプションの別名です。

SERVER="*server*"
Teradata サーバーの名前を指定します。

適用対象 Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

TIMEOUT="*n*"
プロシジャが操作を試みる秒数を指定します。 操作が *n* 秒以内に完了しない場合、操作は終了します。

適用対象 Filesystem

制限事項 このオプションは ADLS の場合にのみサポートされます。

USERNAME="*id*"
Hadoop サーバーまたは Teradata サーバーの認証ユーザー ID です。

別名 USER=

USERID=

UID=

適用対象 Hadoop

Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

WEBHDFSURL="*webhdfs-url*"
REST API を介して Hadoop 分散ファイルシステムにアクセスするのに使用する URL を指定します。

適用対象 Hadoop

注 この引数は、REST API を介して分散ファイルシステムにアクセスするようにプラットフォームが構成されている場合に使用します。

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

PUBLISHMODEL ステートメント

CAS または外部データソースでモデルをパブリッシュします。

要件 Hadoop の場合、2023.09 以降、新しいバージョンの Java ランタイム環境によってセキュリティ更新が導入されました。配置、構成、使用法における変更の概要については、“Publishing and Running Models in Hadoop” (*SAS In-Database Products: User's Guide*)を参照してください。

構文

PUBLISHMODEL
CASLIB="*caslib*"
MODELNAME="*model-name*"
MODELTYPE=DATASTEP | DS2
MODELTABLE="*model-table*" | "*caslib.model-table*" | "*schema.model-table*"
STOREFILES=*file-specification* | PROGRAMFILE=*file-specification*
VARXMLFILE=*file-path*
<*publish-model-optional-arguments*>;

必須引数

CASLIB="*casref*"
モデルがパブリッシュされるデータソースに関連付けられている caslib の名前を指定します。 PROC SCOREACCEL は、caslib を使用して、データソースに固

有のオプションを取得します。たとえば、caslib は、ユーザーの資格情報やその他の接続オプションを指定できます。

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

操作 caslib 割り当てで接続オプションを指定します。プロシジャでの接続オプションの指定は非推奨です。CASLIB=オプションが推奨されており、将来的にはすべての EXTTYPE=データソースで必須になります。現在、EXTTYPE=値が FILESYSTEM、SINGLESTORE、SPARK、および SYNAPSE のモデルをパブリッシュおよび削除するには、CASLIB=オプションが必要です。

CAS または Teradata の場合、MODELTABLE=で 2 部構成のモデルテーブル名の最初の部分として指定された caslib は、このオプションの値よりも優先されます。

MODELNAME="*model-name*"
パブリッシュするモデルの名前を指定します。

別名 MODEL=

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

注 SingleStore の場合、SAS は各モデルをテーブルとしてパブリッシュします。Hadoop、Databricks、、および Synapse の場合、MODELDIR=オプションを省略すると、各モデルはテーブル(データソースに応じて Hive テーブル、Spark テーブル、または SQL Server テーブル)としてパブリッシュされます。これらのすべてのケースで 、SAS はモデル名をテーブル名と

して使用し、データベース内の他のテーブルと区別するために **sasmodel_** という接頭辞を追加します。モデルを参照してパブリッシュ、削除、または実行する場合、**sasmodel_**接頭辞を含めないでください。
MODELTABLE=オプションは使用しないでください。

MODELTABLE="*model-table*" | "*caslib.model-table*" | "*schema.model-table*"
CAS または Teradata でモデルをパブリッシュするテーブルの名前を指定します。

CAS の場合、モデルテーブルは 1 部または 2 部の名前で指定できます。2 部構成の名前の 1 部はモデルテーブルを含む caslib の名前として解釈されます。

適用対象 CAS

Teradata

操作 2 部構成モデルテーブル名の 1 部として指定された caslib は、CASLIB=オプションで指定された caslib よりも優先されます。

注 モデルを CAS にパブリッシュするには、モデルテーブルの行を追加または置換する必要があります。モデルを既存のモデルテーブルにパブリッシュするには、まずモデルテーブルを CAS サーバーにロードする必要があります。指定されたモデルテーブルが CAS サーバーにロードされている場合は、モデルテーブルが更新されます。それ以外の場合は、新しいモデルテーブルが作成されます。

MODELTYPE=DATASTEP | DS2
入力モデルプログラムの種類を指定します。

DATASTEP
入力モデルプログラムが DATA ステップコードであることを指定します。

別名 DS

DS2
入力モデルプログラムが DS2 コードであることを指定します。

デフォルト DS2

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

注 MODELTYPE=オプションは、DATA ステップモデルの場合のみ必要です。DATA ステップコードは、アイテムストアにバンドルされる前に DS2 コードに変換されます。

STOREFILES=("*file-path-1*" | *fileref-1* <,"*file-path-2*" | *fileref-2*, ...>)
分析ストア.sasast ファイルのリストを指定します。ファイルは、分析ストアごとに 1 つずつパブリッシュされます。 ファイルのリストには、ファイルパス名とファイル参照名が混在している場合があります。

STOREFILES=または PROGRAMFILE=またはその両方のオプションを指定する必要があります。一部の分析ストアモデルでは、STOREFILES=オプションのみが必要です。一部の DATA ステップおよび DS2 モデルでは、PROGRAMFILE=オプションのみが必要です。一部のモデルは、DATA ステップコード、DS2 コード、および 1 つ以上の ASTORE ファイルで構成されています。その場合、STOREFILES=オプションと PROGRAMFILE=オプションの両方を指定します。

モデルプログラムなしで分析ストアをパブリッシュすると、DS2 モデルプログラムが分析ストアから生成されます。

関連するモデルプログラムを指定するには、PROGRAMFILE=オプションを使用します。分析ストアがパブリッシュされるとき、関連付けられたモデルプログラムは通常、DS2 コードです。

別名 STOREFILE=

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

操作 DS2 プログラムを伴わずに単一の分析ストアモデルが指定されている場合は、KEEPLIST オプションも指定できます。

STOREFILES=または PROGRAMFILE=またはその両方のオプションを指定する必要があります。

STOREFILES=オプションは MODELTYPE=DS2 の場合にのみ使用されます。MODELTYPE=DATASTEP の場合はこのオプションは使用されません。

PROGRAMFILE="*file-path*" | *fileref*
パブリッシュするモデルプログラムを含むファイルを指定します。
STOREFILES=または PROGRAMFILE=またはその両方のオプションを指定する必要があります。一部の分析ストアモデルでは、STOREFILES=オプションのみが必要です。一部の DATA ステップおよび DS2 モデルでは、PROGRAMFILE=オプションのみが必要です。一部のモデルは、DATA ステップコード、DS2 コード、お

よび 1 つ以上の ASTORE ファイルで構成されています。その場合、STOREFILES=オプションと PROGRAMFILE=オプションの両方を指定します。

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

操作 STOREFILES=または PROGRAMFILE=またはその両方のオプションを指定する必要があります。

VARXMLFILE="*file-path*" | *fileref*
入力 DATA ステップモデルプログラムを DS2 に変換する際に使用される変数メタデータ XML を含むファイルを指定します。

別名 XMLFILE=

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

操作 MODELTYPE=DATASTEP の場合、一部のモデルでは VARXMLFILE=オプションが必要です。MODELTYPE=DS2 の場合はこのオプションは使用されません。

オプション引数

AUTHDOMAIN="*authentication-domain*"
Teradata へのアクセスに使用される認証情報(ユーザー名とパスワード)を含む認証ドメインの名前を指定します。

適用対象 Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

AUTHTOKEN="*authentication-token*"
Spark クラスターとの接続を確立するために使用される認証トークンを指定します。

適用対象 Hadoop

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

CLASSPATH="*class_path*"
Hadoop 呼び出しコンテキストで使用されるクラスパスを指定します。クラスパスは、フォルダーまたは個別の JAR ファイルにすることができます。Hadoop 構成フォルダーは、クラスパスに含める必要があります。

適用対象 Hadoop

注意 **このオプションはサポートされなくなりました。** 2023.09 以降、コードを変更してこのオプションを削除してください。

CLUSTERID="*cluster-id*"
Spark クラスターの識別子を指定します。

適用対象 Hadoop

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

DATABASE="*database-name*"
データベース名を指定します。

デフォルト "" (長さゼロの文字列)

適用対象 Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

注 Teradata では、このオプションはアクセスする Teradata データベースの名前を指定する接続オプションを提供します。

Hadoop、SingleStore、Spark、または Synapse の場合、DATABASE=オプションは SCHEMA=オプションの別名です。

EXTTYPE=CAS | DATABRICKS | FILESYSTEM | HADOOP | SINGLESTORE | SYNAPSE | TERADATA
モデルがパブリッシュされるターゲット環境を指定します。

CAS
CAS のモデルテーブルにパブリッシュすることを指定します。

DATABRICKS
モデルを Databricks にパブリッシュすることを指定します。この値は、Spark テーブルにモデルをパブリッシュするために 2023.10 以降サポートされています。

FILESYSTEM
モデルをクラウドベースのファイルシステムにパブリッシュすることを指定します。FILESYSTEM は、Microsoft Azure Data Lake Storage Gen2 (ADLS) および Amazon S3 をサポートしています。

注: FILESYSTEM 値は、2023.12 以降非推奨になりました。Databricks 、、および Synapse の場合、モデルをファイルシステムに発行する必要がなくなりました。

HADOOP
モデルを Hadoop サーバーにパブリッシュするように指定します。

SINGLESTORE
モデルを SingleStore クラスターにパブリッシュすることを指定します。この値は 2023.05 以降でサポートされています。

SYNAPSE
モデルを Azure Synapse にパブリッシュすることを指定します。この値は、SQL Server テーブルにモデルをパブリッシュするために 2023.12 以降サポートされています。

TERADATA
Teradata データベースのモデルテーブルをパブリッシュすることを指定します。

別名 TARGET=

デフォルト CAS

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

FORMATFILE="*file-path*" | *fileref*
パブリッシュするユーザー定義の形式 XML 定義を含むファイルを指定します。

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

注 このオプションは、DS2 モデルまたは DATA ステップモデルに有効です。

FORMATITEMSTOREFILE="*file-path*" | *fileref*
パブリッシュする出力形式アイテムストアを含むファイルを指定します。

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

KEEPLIST=TRUE | FALSE
分析ストアモデルから自動的に生成された DS2 モデルプログラムに KEEP ステートメントを含めるかどうかを指定します。

TRUE
KEEP ステートメントを含みます。

別名 YES

FALSE
KEEP ステートメントは含まれません。

別名 NO

デフォルト FALSE

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

操作 KEEPLIST=は、STOREFILES=または STORETABLES=オプションに単一の分析ストアモデルが含まれていて、付随する DS2 モデルプログラムが指定されていない場合に指定できます。

MODELDATABASE="*model-database*"
モデルが保存されているデータベースを指定します。 このオプションが指定されていない場合は、caslib の SCHEMA=オプションで指定されたデータベースが使用されます。このオプションは、モデルを Hive テーブルまたは Spark テーブルに保存するためにサポートされています。MODELDIR=を指定すると、MODELDATABASE=は無視されます。

適用対象 Databricks

Hadoop

要件 モデルをパブリッシュするときに MODELDATABASE=が指定されている場合は、データベースがすでに存在している必要があります。

注 このオプションは 2023.09 以降でサポートされています。

MODELDIR="*model-directory*"
モデルディレクトリが保存されるルートフォルダーを指定します。

Hadoop の場合、2023.09 より前のリリースでは、モデルは常に HDFS にパブリッシュされていました。現在は、WebHDFS を使用して、HDFS に保存されているモデルにアクセスしています。HDFS 内のモデルにアクセスする場合は、Hadoop caslib で webHdfsUrl=オプションを指定する必要があります。ただし、この方法はお勧めできません。将来のリリースでは、HDFS へのパブリッシュはサポートされなくなります。

Hadoop の場合、2023.09 以降、モデルを Hive テーブルにパブリッシュできるようになり、この方法が推奨されます。Hive テーブルに格納されているモデルをパブリッシュ、削除、または実行するには、PROC SCOREACCEL から MODELDIR=オプションを省略します。MODELDIR=オプションを省略する場合は、Hadoop caslib の SERVER=パラメーターで Hive サーバーを指定する必要があります。

適用対象 Filesystem

Hadoop

MODELNOTES="*model-notes*"
モデルテーブルに書き込まれるモデルノートを指定します。

適用対象 CAS

SingleStore(SAS SpeedyStore)

Teradata

MODELSCHEMA="*model-schema*"
保存されたているモデルのスキーマを指定します。 このオプションは、SQL Server テーブルにモデルを保存するためにサポートされています。 MODELDIR=を指定すると、MODELSCHEMA=は無視されます。

適用対象 Synapse

要件 Synapse の場合、SQL Server テーブルに格納されているモデルを削除、パブリッシュ、または実行するときに、MODELSCHEMA=オプションを指定する必要があります。

注 このオプションは 2023.12 以降でサポートされています。

MODELUUID="*model-uuid*"
モデル UUID がモデルテーブルに書き込まれることを指定します。

適用対象 CAS

SingleStore(SAS SpeedyStore)

Teradata

OUTDIR="*work-directory*"
DATA ステップから DS2 に変換されたプログラムファイルを含むローカル出力ディレクトリを指定します。

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

PASSWORD="*password*"
ADLS、Hadoop サーバーまたは Teradata サーバーのユーザー ID のパスワードです。ADLS では、これはシークレットと呼ばれる値です。

別名 PASS=

PASSWD=

PWD=

適用対象 Filesystem

Hadoop

Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

PERSISTTABLE=TRUE | FALSE
更新されたモデルテーブルを、テーブルに関連付けられた caslib データソースに保存するかどうかを指定します。

TRUE
更新されたモデルテーブルをデータソースに保存します。

別名 YES

FALSE
更新されたモデルテーブルをデータソースに保存しません。

別名 NO

デフォルト FALSE

適用対象 CAS

要件 モデルテーブルが caslib データソースにすでに存在する場合は、テーブルを保存するために REPLACETABLE=TRUE も指定する必要があります。

参照項目: REPLACETABLE=オプション

PROMOTETABLE=TRUE | FALSE
更新されたモデルテーブルを CAS サーバーのグローバルスコープにプロモートする必要があるかどうかを指定します。

TRUE
更新されたモデルテーブルをグローバルスコープにプロモートします。

別名 YES

FALSE
更新されたモデルテーブルをグローバルスコープにプロモートしません。

別名 NO

デフォルト FALSE

適用対象 CAS

PUBLISHGLOBAL=TRUE | FALSE
モデルを CAS のグローバルモデルテーブルにパブリッシュするかどうかを指定します。

TRUE

別名 YES

FALSE
ローカル CAS セッションのモデルテーブルにモデルをパブリッシュし、グローバルモデルテーブルは変更しないままにします。

別名 NO

デフォルト FALSE

適用対象 CAS

REPLACEMODEL=TRUE | FALSE
モデルテーブル内の既存モデルをパブリッシュされるモデルに置換することを許可するかどうかを指定します。

TRUE
モデルテーブルのモデルを置き換えます。

別名 YES

FALSE
モデルテーブルのモデルを置き換えません。

別名 NO

デフォルト TRUE

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

REPLACETABLE=TRUE | FALSE
既存のモデルテーブルが、更新されたモデルテーブルが caslib データソースに保存される際に置換されるのを許可するかどうかを指定します。

TRUE
モデルテーブルを置き換えます。

別名 YES

FALSE
モデルテーブルを置き換えません。

別名 NO

デフォルト TRUE

適用対象 CAS

要件 モデルテーブルを caslib データソースに保存するには、PERSISTTABLE=TRUE も指定する必要があります。

参照項目: PERSISTTABLE=オプション

SCHEMA="*schema-name*"
データベースまたはスキーマの名前を指定します。

適用対象 Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

注 Teradata では、このオプションは、Teradata データベースが Teradata テーブルを修飾するために使用する接続オプションを提供します。

Hadoop、SingleStore、Spark、または Synapse の場合、DATABASE=オプションは SCHEMA=オプションの別名です。

SERVER="*server*"
Teradata サーバーの名前を指定します。

適用対象 Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

STORETABLES=("*store-table-1*" | "*caslib.store-table-1*" <,"*store-table-2*" | "*caslib.store-table-2*", ...>)
パブリッシュする分析ストアを含む 1 つ以上の CAS BLOB テーブル名を指定します。各テーブルには、分析ストア BLOB を含む_state_という名前の VARBINARY 列が含まれている必要があります。 モデルプログラムなしで分析ストアをパブリッシュすると、DS2 モデルプログラムが分析ストアから生成されます。

関連するモデルプログラムを指定するには、PROGRAMFILE=オプションを使用します。分析ストアがパブリッシュされるとき、関連付けられたモデルプログラムは通常、DS2 コードです。

適用対象 CAS

Databricks

Filesystem

Hadoop

SingleStore(SAS SpeedyStore)

Synapse

Teradata

操作 DS2 プログラムを伴わずに単一の分析ストアモデルが指定されている場合は、KEEPLIST オプションも指定できます。

STORETABLES=オプションを使用して単一の分析ストアが指定されている場合、PROGRAMFILE=オプションは必要ありません。

TIMEOUT="*n*"
プロシジャが操作を試みる秒数を指定します。 操作が *n* 秒以内に完了しない場合、操作は終了します。

適用対象 Filesystem

制限事項 このオプションは ADLS の場合にのみサポートされます。

USERNAME="*id*"
Hadoop サーバーまたは Teradata サーバーの認証ユーザー ID です。

別名 USER=

USERID=

UID=

適用対象 Hadoop

Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

WEBHDFSURL="*webhdfs-url*"
REST API を介して Hadoop 分散ファイルシステムにアクセスするのに使用する URL を指定します。

適用対象 Hadoop

注 この引数は、REST API を介して分散ファイルシステムにアクセスするようにプラットフォームが構成されている場合に使用します。

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

RUNMODEL ステートメント

CAS または外部データソースでモデルを実行します。

要件 Hadoop の場合、2023.09 以降、新しいバージョンの Java ランタイム環境によってセキュリティ更新が導入されました。配置、構成、使用法における変更の概要については、"Publishing and Running Models in Hadoop" (*SAS In-Database Products: User's Guide*)を参照してください。

構文

RUNMODEL
CASLIB="*caslib*"
MODELNAME="*model-name*"
MODELTABLE="*model-table*" | "*caslib.model-table*" | "*schema.model-table*"
<*run-model-optional-arguments*>;

必須引数

CASLIB="*casref*"
モデルが実行されるデータソースに関連付けられている caslib の名前を指定します。 PROC SCOREACCEL は、caslib を使用して、データソースに固有のオプションを取得します。たとえば、caslib は、ユーザーの資格情報やその他の接続オプションを指定できます。

適用対象 CAS

Hadoop

SingleStore(SAS SpeedyStore)

Spark

Teradata

要件 EXTTYPE=SPARK の場合、Spark caslib で次のパラメーターを使用します。platform=オプションは外部ターゲットを指定します。schema=は、Synapse SQL データベース名または Spark データベース名のいずれかです。username=はアプリケーション ID、password=はアプリケーション ID のシークレットです。caslib ですべての接続オプションを指定する必要があります。

操作 caslib 割り当てで接続オプションを指定します。プロシジャでの接続オプションの指定は非推奨です。CASLIB=オプションが推奨されており、将来的にはすべての EXTTYPE=データソースで必須になります。現在、

EXTTYPE=値が SINGLESTORE および SPARK のモデルをパブリッシュおよび削除するには、CASLIB=オプションが必要です。

CAS または Teradata の場合、MODELTABLE=で 2 部構成のモデルテーブル名の最初の部分として指定された caslib は、このオプションの値よりも優先されます。

MODELNAME="*model-name*"
実行するモデルの名前を指定します。

別名 MODEL=

適用対象 CAS

Hadoop

SingleStore(SAS SpeedyStore)

Spark

Teradata

注 SingleStore の場合、SAS は各モデルをテーブルとしてパブリッシュします。Hadoop、Databricks、、および Synapse の場合、MODELDIR=オプションを省略すると、各モデルはテーブル(データソースに応じて Hive テーブル、Spark テーブル、または SQL Server テーブル)としてパブリッシュされます。これらのすべてのケースで、SAS はモデル名をテーブル名として使用し、データベース内の他のテーブルと区別するために **sasmodel_** という接頭辞を追加します。モデルを参照してパブリッシュ、削除、または実行する場合、**sasmodel_**接頭辞を含めないでください。
MODELTABLE=オプションは使用しないでください。

MODELTABLE="*model-table*" | "*caslib.model-table*" | "*schema.model-table*"
実行するモデルを含むテーブルの名前を指定します。 CAS でモデルを実行する場合、モデルテーブルは 1 部または 2 部の名前で指定できます。2 部構成の名前の 1 部はモデルテーブルを含む caslib の名前として解釈されます。

適用対象 CAS

Teradata

操作 2 部構成モデルテーブル名の 1 部として指定された caslib は、CASLIB=オプションで指定された caslib よりも優先されます。

オプション引数

AUTHDOMAIN="*authentication-domain*"
Teradata へのアクセスに使用される認証情報(ユーザー名とパスワード)を含む認証ドメインの名前を指定します。

適用対象 Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

AUTHTOKEN="*authentication-token*"
Spark クラスターとの接続を確立するために使用される認証トークンを指定します。

適用対象 Hadoop

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

CLASSPATH="*class_path*"
Hadoop 呼び出しコンテキストで使用されるクラスパスを指定します。クラスパスは、フォルダーまたは個別の JAR ファイルにすることができます。CONFIGPATH=オプションを指定しない場合は、Hadoop 構成フォルダーをクラスパスに含める必要があります。 Spark を使用している場合、Hadoop JAR ファイルと Spark JAR ファイルは別々のフォルダーに配置する必要があり、Spark JAR フォルダーをクラスパスの最後のエントリとして指定する必要があります。

適用対象 Hadoop

注意 **このオプションはサポートされなくなりました。** 2023.09 以降、コードを変更してこのオプションを削除してください。

CLUSTERID="*cluster-id*"
Spark クラスターの識別子を指定します。

適用対象 Hadoop

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

CONFIGPATH="*configuration-path*"
すべての Hadoop および Spark 構成ファイルが存在する単一のフォルダーを指定します。

適用対象 Hadoop

Spark

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

CUSTOMJAR="*file-path*"
ユーザー提供のカスタムリーダーを含むローカル JAR ファイルを指定します。カスタム JAR ファイルは、ジョブのサブミット時に自動的に Hadoop クラスターにコピーされます。

適用対象 Hadoop

注意 **このオプションはサポートされなくなりました。** 2023.09 以降、コードを変更してこのオプションを削除してください。

DATABASE="*database-name*"
データベース名を指定します。

デフォルト "" (長さゼロの文字列)

適用対象 Hadoop

Spark

SingleStore(SAS SpeedyStore)

Teradata

注 Teradata では、このオプションはアクセスする Teradata データベースの名前を指定する接続オプションを提供します。

Hadoop、SingleStore、または Spark の場合、DATABASE=オプションは SCHEMA=オプションの別名です。

DBMAXTEXT=*number-of-bytes*
STRING データ型列に割り当てる最大バイト数を指定します。 このオプションは、CHAR 列または VARCHAR 列には適用されません。

デフォルト 32767

範囲 1–32767

適用対象 Hadoop

EPOPTIONS="*options-string*"
SAS Embedded Process に渡されるプロパティを指定します。 SingleStore の場合、プロパティ ds2.dbMaxText=*number* と logger.level=*value* は有効です。これらのプロパティの詳細については、"epProperties= データコネクタオプション" (*SAS Cloud Analytic Services: ユーザーガイド*)を参照してください。

Teradata では、このオプションはレガシープログラムでのみサポートされます。プロパティは、OPTIONS パラメーターで SAS_SCORE_EP ストアドプロシジャに渡されます。詳細については、*SAS In-Database Products: ユーザーガイド*を参照してください。

適用対象 SingleStore(SAS SpeedyStore)

Teradata

例 epOptions="ds2.dbMaxText=2048,logger.level=allon"

EXTTYPE=CAS | HADOOP | SINGLESTORE | SPARK | TERADATA
モデルを実行するターゲット環境を指定します。

CAS
CAS でモデルを実行することを指定します。

HADOOP
SAS Embedded Process for Hadoop を使用してモデルを実行することを指定します。

SINGLESTORE
SAS Embedded Process for SingleStore を使用してモデルを実行することを指定します。この値は 2023.05 以降でサポートされています。

SPARK
SAS Embedded Process for Spark を使用してモデルを実行することを指定します。CASLIB=オプションは必須です。caslib では、PLATFORM=オプションを使用して外部ターゲットを指定します。詳細については、*"Interaction with SAS Data Connector to Spark" (SAS In-Database Products: User's Guide)*を参照してください。

注: DATABRICKS 値と SYNAPSE 値は非推奨になりました。代わりに EXTTYPE=SPARK を使用してください。

TERADATA
SAS Embedded Process for Teradata を使用してモデルを実行することを指定します。

別名 TARGET=

デフォルト CAS

適用対象 CAS

Hadoop

SingleStore(SAS SpeedyStore)

Spark

Teradata

FORCEOVERWRITE=TRUE | FALSE
Hadoop MapReduce ジョブを実行する前に出力データディレクトリを強制的に削除するかどうかを指定します。

TRUE
出力データディレクトリの削除を強制します。

別名 YES

FALSE
出力データディレクトリの削除は強制されません。

別名 NO

デフォルト FALSE

適用対象 Hadoop

Spark

制限事項 Synapse の SQL テーブルに対してモデルを実行する場合、FORCEOVERWRITE=FALSE はサポートされていません。出力テーブルは存在してはなりません。

HIVEPORT=*integer*
Hive サーバーのポート番号を指定します。

適用対象 Hadoop

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

INDATASET="*input-dataset*"
SAS Embedded Process への入力として渡される Spark データセットの名前を指定します。

適用対象 Hadoop

Spark

要件 Hadoop でモデルを実行する場合は、INDATASET=、INHDMD=または INTABLE=のいずれかを指定する必要があります。

Databricks、、または Synapse でモデルを実行する場合は、INDATASET=または INTABLE=のいずれかを指定する必要があります。

参照項目: INHDMD= オプション

INTABLE=オプション

INHDMD="*file-path*"
HDFS 上の入力 Spark データセットファイルの名前を指定します。

適用対象 Hadoop

要件 Hadoop でモデルを実行する場合は、INDATASET=、INHDMD=または INTABLE=のいずれかを指定する必要があります。

参照項目: INDATASET=オプション

INTABLE=オプション

注意 **このオプションは非推奨です。** 将来的にはサポートされなくなります。コードを変更してこのオプションを削除します。

INQUERY="*sql-query*"
SAS Embedded Process への入力を定義する SQL SELECT ステートメントを指定します。SQL クエリはデータソースでサポートされている必要があります。

適用対象 SingleStore(SAS SpeedyStore)

Teradata

要件 Teradata、または SingleStore でモデルを実行する場合は、INTABLE=または INQUERY=のいずれかを指定する必要があります。

参照項目: INTABLE=オプション

INTABLE="*input-table*" | "*caslib.input-table* " | "*schema.input-table*"
入力テーブルの名前を示します。 入力テーブルは、1 部構成または 2 部構成の名前で指定できます。

- CAS でモデルを実行する場合、2 部構成の名前の 1 部は入力テーブルを含む caslib の名前として解釈されます。
- SingleStore で実行する場合、1 部構成の名前を使用する必要があります。DATABASE=引数を使用してデータベース名を指定します。
- Synapse で SQL Server テーブルに対して実行する場合は、2 部構成の名前を使用します。スキーマ名を指定し、その後にドット、テーブル名を続けます。
- Teradata でモデルを実行する場合、2 部構成の名前の 1 部は、入力テーブルを含むデータベーススキーマの名前として解釈されます。

別名 INPUTTABLE=

適用対象 CAS

Hadoop

Spark

Teradata

要件 Hadoop でモデルを実行する場合は、INDATASET=、INHDMD=または INTABLE=のいずれかを指定する必要があります。

Databricks、、または Synapse でモデルを実行する場合は、INDATASET=または INTABLE=のいずれかを指定する必要があります。

Teradata、または SingleStore でモデルを実行する場合は、INTABLE=または INQUERY=のいずれかを指定する必要があります。

参照項目: INDATASET=オプション

INHDMD= オプション

INQUERY=オプション

JOBMANAGEMENTURL="*rest-url*"
Apache Livy や Databricks Notebooks などのサービスに REST インターフェイスを介して実行要求をサブミットするために使用される URL を指定します。

別名 RESTURL=

適用対象 Hadoop

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

KEEPLISTCOLUMNS="*column-1 <column-2 ...* >"
KEEPLISTCOLUMNS=(*column-1 <column-2 ...* >)
KEEPLISTCOLUMNS=("*column-1*" <"*column-2*" ... >)
DS2 プログラムによって保持される 1 つまたは複数の列の名前を指定します。

別名 KEEPLISTCOLS=

適用対象 Hadoop

Spark

操作 パラメーター KEEPLISTFILE と KEEPLISTCOLUMNS は相互排他的です。

KEEPLISTFILE="*file-path*"
DS2 プログラムによって保持される列のリストを含むファイルの名前を指定します。

適用対象 Hadoop

Spark

操作 パラメーター KEEPLISTFILE と KEEPLISTCOLUMNS は相互排他的です。

注意 **このオプションは非推奨です。** 将来的にはサポートされなくなります。コードを変更してこのオプションを削除します。

MODELDATABASE="*model-database*"
モデルが保存されているデータベースを指定します。 このオプションが指定されていない場合は、caslib の SCHEMA=オプションで指定されたデータベースが使用されます。このオプションは、モデルを Hive テーブルまたは Spark テーブルに保存するためにサポートされています。MODELDIR=を指定すると、MODELDATABASE=は無視されます。

適用対象 Databricks

Hadoop

要件 モデルをパブリッシュするときに MODELDATABASE=が指定されている場合は、データベースがすでに存在している必要があります。

注 このオプションは 2023.09 以降でサポートされています。

MODELDIR="*model-directory*"
モデルディレクトリが保存されるルートフォルダーを指定します。

2023.10 時点の Databricks、、および 2023.12 時点の Synapse では、モデルをファイルシステムにパブリッシュする必要はなくなりました。ADLS または Amazon S3 にパブリッシュされたモデルを実行するには、MODELDIR=オプションのみを指定します。

Databricks の場合、MODELDIR=を指定すると、モデルディレクトリは接頭辞 **dbfs:**を含んでおり、Databricks File System (DBFS)にマウントされたストレージオブジェクトを参照します。モデルを Amazon S3 または Microsoft Azure Data Lake Storage Gen2 (ADLS)にパブリッシュし、そのモデルを Databricks で実行できます。次に例を示します。

```
modeldir="dbfs:/mnt/mycontainer/mymodels"
```

Hadoop の場合、2023.09 より前のリリースでは、モデルは常に HDFS にパブリッシュされていました。現在は、WebHDFS を使用して、HDFS に保存されているモデルにアクセスしています。HDFS 内のモデルにアクセスする場合は、Hadoop caslib で webHdfsUrl=オプションを指定する必要があります。ただし、この方法はお勧めできません。将来のリリースでは、HDFS へのパブリッシュはサポートされなくなります。

Hadoop の場合、2023.09 以降、モデルを Hive テーブルにパブリッシュできるようになり、この方法が推奨されます。Hive テーブルに格納されているモデルをパブリッシュ、削除、または実行するには、PROC SCOREACCEL から MODELDIR=オプションを省略します。MODELDIR=オプションを省略する場合は、Hadoop caslib の SERVER=パラメーターで Hive サーバーを指定する必要があります。

適用対象 Hadoop

Spark

MODELSCHEMA="*model-schema*"
保存されたているモデルのスキーマを指定します。 このオプションは、SQL Server テーブルにモデルを保存するためにサポートされています。MODELDIR=を指定すると、MODELSCHEMA=は無視されます。

適用対象 Synapse

要件 Synapse の場合、SQL Server テーブルに格納されているモデルを削除、パブリッシュ、または実行するときに、MODELSCHEMA=オプションを指定する必要があります。

注 このオプションは 2023.12 以降でサポートされています。

OUTDATASET="*output-dataset*"
SAS Embedded Process によって作成される出力 Spark データセットの名前を指定します。

適用対象 Hadoop

Spark

要件 Hadoop でモデルを実行する場合は、OUTHDMD=、OUTDATASET=、または OUTTABLE=のいずれかを指定する必要があります。

Databricks、、HDInsight、または Synapse でモデルを実行する場合は、OUTDATASET=または OUTTABLE=のいずれかを指定する必要があります。

参照項目: OUTHDMD=オプション

OUTTABLE=オプション

OUTHDMD="*file-path*"
SAS Embedded Process によって作成される出力 Hadoop メタデータファイルの名前を指定します。

適用対象 Hadoop

要件 Hadoop でモデルを実行する場合は、OUTHDMD=、OUTDATASET=、または OUTTABLE=のいずれかを指定する必要があります。

参照項目: OUTDATASET=オプション

OUTTABLE=オプション

注意 **このオプションは非推奨です。** 将来的にはサポートされなくなります。コードを変更してこのオプションを削除します。

OUTKEY="*column-1 <column-2 ...)*>"
OUTKEY=(*column-1 <column-2 ...)*>
OUTKEY=("*column-1*" <"*column-2*" ...)>
SAS Embedded Process によって作成される出力テーブルのプライマリインデックスに使用される 1 つ以上の列の名前を指定します。

適用対象 SingleStore(SAS SpeedyStore)

Teradata

OUTPUTFOLDER="*directory-path*"
出力ファイルが格納されているディレクトリの名前を指定します。

適用対象 Hadoop

注意 **このオプションは非推奨です。** 将来的にはサポートされなくなります。コードを変更してこのオプションを削除します。

OUTPUTFORMATCLASS="*class-name*"
出力レコードの書き込みに使用される出力形式クラスの名前をドット表記で指定します。

適用対象 Hadoop

注意 **このオプションは非推奨です。** 将来的にはサポートされなくなります。コードを変更してこのオプションを削除します。

OUTRECORDFORMAT=BINARY | DELIMITED
Hadoop 用 SAS Embedded Process が作成する出力レコードの形式を指定します。

BINARY
出力レコードがバイナリであることを指定します。

DELIMITED
出力レコードが区切られることを指定します。

デフォルト DELIMITED

適用対象 Hadoop

注意 **このオプションは非推奨です。** 将来的にはサポートされなくなります。コードを変更してこのオプションを削除します。

OUTSCHEMA="*output-schema*"
モデルを実行するときの Hive 出力データソーススキーマ名を指定します。 この引数は、Databricks、Hadoop、 、または Synapse に適しています。

適用対象 Hadoop

Spark

OUTTABLE="*output-table*" | "*caslib.output-table* " | "*schema.output-table*"
出力テーブルの名前を示します。 出力テーブルは、1 部構成または 2 部構成の名前で指定できます。

- CAS でモデルを実行する場合、2 部構成の名前の 1 部は出力テーブルを含む caslib の名前として解釈されます。
- SingleStore で実行する場合、1 部構成の名前を使用する必要があります。DATABASE=引数を使用してデータベース名を指定します。
- Synapse で SQL Server テーブルに対して実行する場合は、2 部構成の名前を使用します。スキーマ名を指定し、その後にドット、テーブル名を続けます。
- Teradata でモデルを実行する場合、出力テーブルは入力テーブルと同じデータベースにある必要があります。2 部構成の名前の 1 部は出力テーブルを含むデータベーススキーマ名前として解釈されます。

別名 OUTPUTTABLE=

適用対象 CAS

Hadoop

SingleStore(SAS SpeedyStore)

Spark

Teradata

要件 Hadoop でモデルを実行する場合は、OUTHDMD=、OUTDATASET=、または OUTTABLE=のいずれかを指定する必要があります。

Databricks、、HDInsight、または Synapse でモデルを実行する場合は、OUTDATASET=または OUTTABLE=のいずれかを指定する必要があります。

SingleStore または Teradata でモデルを実行する場合、OUTTABLE=を指定する必要があります。

参照項目: OUTDATASET=オプション

OUTHDMD=オプション

OUTTABLEOPTIONS="*options-string*"
Hive CREATE TABLE ステートメントに追加されるユーザー指定のオプションを提供します。

適用対象 Hadoop

Spark

注意 **このオプションは非推奨です。** 将来的にはサポートされなくなります。コードを変更してこのオプションを削除します。

PASSWORD="*password*"
Hadoop サーバーまたは Teradata サーバーのユーザー ID のパスワードです。

別名 PASS=

PASSWD=

PWD=

適用対象 Hadoop

Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

PLATFORM=MAPRED | SPARK
SAS Embedded Process for Hadoop が実行されるプラットフォームを指定します。

MAPRED
MapReduce でモデルを実行することを指定します。

SPARK
Spark でモデルを実行することを指定します。

デフォルト SPARK

適用対象 Hadoop

注意 **このオプションは非推奨です。** SPARK 値が自動的に使用されるようになりました。コードを変更してこのオプションを削除します。

PROPERTIES="*name1=value*" <PROPERTIES="*name2=value*"> |
PROPERTIES=("*name=value*"<, "*name=value*", ...>)
名前と値のペアとして Hadoop 構成プロパティを指定します。複数のプロパティの場合は、かっこで囲まれた名前と値のペアのカンマ区切りリストを含む単一の引数を指定します。このオプションを使用して、任意の Hadoop 構成プロパティを割り当てることができます。PROPERTIES パラメーターは、複数回(プロパティごとに 1 回)指定することもできます。

適用対象 Hadoop

Spark

要件 出力先が Hive テーブルであり、Hive データベースフォルダーが HDFS 暗号化ゾーンの下にある場合は、SAS Embedded Process 一時フォルダーを同じ暗号化ゾーンの下にある場所に設定する必要があります。これを

行うには、**sas.ep.tempdir** 構成プロパティを設定します。次に例を示します。

properties="sas.ep.tempdir=*yourSASEPTemporaryFolder*"

SCHEMA="*schema-name*"
データベースまたはスキーマの名前を指定します。

適用対象 Hadoop

Spark

SingleStore(SAS SpeedyStore)

Teradata

注 Teradata では、このオプションは、Teradata データベースが Teradata テーブルを修飾するために使用する接続オプションを提供します。

Hadoop、SingleStore、または Spark の場合、DATABASE=オプションは SCHEMA=オプションの別名です。

SENDPROGRAM=TRUE | FALSE
モデルプログラムのソースコードをクライアントに送り返してユーザーに表示するかどうかを指定します。

TRUE
ソースコードをクライアントに送り返します。

別名 YES

FALSE
ソースコードをクライアントに送り返しません。

別名 NO

デフォルト FALSE

適用対象 CAS

SERVER="*server*"
Teradata サーバーまたは Hive サーバーの名前を指定します。

適用対象 Hadoop

Spark

Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

TRACE
トレースをオンにして SAS Embedded Process for Hadoop を実行します。

適用対象 Hadoop

Spark

USERNAME="*id*"
Hadoop サーバーまたは Teradata サーバーの認証ユーザー ID です。

別名 USER=

USERID=

UID=

適用対象 Hadoop

Teradata

注意 **プロシジャ内の接続オプションは非推奨です。** 代わりに、caslib 割り当てでこのオプションを指定してください。

VERBOSE
SAS Embedded Process が追加のロギング情報を提供することを指定します。

適用対象 Hadoop

Spark

注意 **このオプションは非推奨です。** 将来的にはサポートされなくなります。コードを変更してこのオプションを削除します。

例: SCOREACCEL プロシジャ

例 1: CAS で DATA ステップモデルをパブリッシュ、実行、および削除する

要素: CAS LIBNAME ステートメント
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この PROC SCOREACCEL の例では、DATA ステップモデルがパブリッシュされ、CAS で実行されます。その後、このモデルは削除されます。

PROC SCOREACCEL は、DATA ステップコードを SAS クライアントの DS2 コードに変換します。

プログラム

```
cas mysess1;
libname mycaslib cas casref=mysess1;

proc delete data=mycaslib.model01_score_data; run;
libname eminput "/mydir/scoring/model01";
data mycaslib.model01_score_data;
  set eminput.traindata;
  id=_n_; run;
quit;

proc scoreaccel sessref=mysess1;
  publishmodel
    modelname="model01"
    modeltype=datastep
    modeltable="modeltable1"
    programfile="/mydir/scoring/model01/score.sas"
    xmlfile="/mydir/scoring/model01/score.xml"
    outdir="/user1/score/work/"
    modelnotes="Simple model01 test model"
    promotetable=no
    persisttable=no
    ;
quit;

proc delete data=mycaslib.model01out_data run;
proc scoreaccel sessref=mysess1;
  runModel
    modelname="model01"
    modeltable="modeltable1"
    intable="model01_score_data"
    outtable="model01_out_data"
    ;
quit;

proc scoreaccel sessref=mysess1;
  deletemodel
    modelname="model01"
    modeltable="modeltable1"
    ;
quit;
```

プログラムの説明

CAS LIBNAME を割り当てます。 CAS LIBNAME は、PROC DELETE の DATA ステップに渡されます。

```
cas mysess1;
libname mycaslib cas casref=mysess1;
```

CAS で入力スコアリングテーブルを作成します。 DELETE プロシジャは、既存の入力スコア表を除去します。

```
proc delete data=mycaslib.model01_score_data; run;
libname eminput "/mydir/scoring/model01";
data mycaslib.model01_score_data;
  set eminput.traindata;
  id=_n_; run;
quit;
```

CAS でモデルをパブリッシュします。 DATA ステップモデルは、このステップで DS2 に変換され、CAS でパブリッシュされます。

```
proc scoreaccel sessref=mysess1;
  publishmodel
    modelname="model01"
    modeltype=datastep
    modeltable="modeltable1"
    programfile="/mydir/scoring/model01/score.sas"
    xmlfile="/mydir/scoring/model01/score.xml"
    outdir="/user1/score/work/"
    modelnotes="Simple model01 test model"
    promotetable=no
    persisttable=no
    ;
quit;
```

CAS でモデルを実行します。 DELETE プロシジャは、モデルを実行する前に既存の CAS テーブルを削除します。

```
proc delete data=mycaslib.model01out_data run;
proc scoreaccel sessref=mysess1;
  runModel
    modelname="model01"
    modeltable="modeltable1"
    intable="model01_score_data"
    outtable="model01_out_data"
    ;
quit;
```

CAS のモデルテーブルからモデルを削除します。

```
proc scoreaccel sessref=mysess1;
  deletemodel
    modelname="model01"
    modeltable="modeltable1"
    ;
quit;
```

例 2: Teradata でモデルをパブリッシュ、実行、および削除する

要素: CAS プロシジャ
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この PROC SCOREACCEL の例では、Teradata で DS2 モデルをパブリッシュして実行しています。PROC SCOREACCEL は、CAS でモデルのパブリッシュとスコアリングのアクションセットを呼び出して、モデルの削除、モデルのパブリッシュ、またはモデルの実行を行います。

プログラム

```
libname mytdlib teradata server=mytdserver user=myuser password=XXXXX
  database=model;

proc delete data=mytdlib.model01_score_data;
run;
libname eminput "/mydir/scoring/model01";
data mytdlib.model01_score_data;
  set eminput.traindata;
  id=_n_;
run;
quit;

proc cas;
  session mysess1;
  action addCaslib /
    caslib="tdlib1"
    datasource={
    srcType="teradata",
    server="mytdserver",
    username="myuser",
    password="XXXXX",
    database="mymodel"
  };
run;

proc scoreaccel sessref=mysess1;
```

```
  publishmodel
    caslib="tdlib1"
    exttype=teradata
    modelname="model01"
    modeltable="modeltable1"
    programfile="/mydir/scoring/model01/score.ds2"
    modelnotes="Simple model01 test model"
    ;
run;
quit;

proc scoreaccel sessref=mysess1;
  runmodel
    caslib="tdlib1"
    exttype=teradata
    modelname="model01"
    modeltable="modeltable1"
    intable="model01_score_data"
    outtable="model01_out_data"
    outkey="id"
    ;
run;
quit;

proc scoreaccel sessref=mysess1;
  deletemodel
    caslib="tdlib1"
    exttype=teradata
    modelname="model01"
    modeltable="modeltable1"
    ;
run;
quit;
```

プログラムの説明

Teradata LIBNAME 参照名を作成します。 Teradata ライブラリ参照名は、PROC DELETE および DATA ステッププログラムに必要です。

```
libname mytdlib teradata server=mytdserver user=myuser password=XXXXX
  database=model;
```

入力スコアリングテーブルを作成します。 DELETE プロシジャは、既存の入力スコアテーブルが存在する場合、それを削除します。

```
proc delete data=mytdlib.model01_score_data;
run;
libname eminput "/mydir/scoring/model01";
data mytdlib.model01_score_data;
  set eminput.traindata;
  id=_n_;
run;
quit;
```

Teradata caslib を定義します。 addCaslib アクションは、現在の mysess1 セッションに CAS ライブラリを追加します。

```
proc cas;
  session mysess1;
  action addCaslib /
     caslib="tdlib1"
     datasource={
     srcType="teradata",
     server="mytdserver",
     username="myuser",
     password="XXXXX",
     database="mymodel"
  };
run;
```

DS2 モデルを Teradata にパブリッシュします。 PROGRAMFILE ステートメントには、DS2 モデルが含まれています。

```
proc scoreaccel sessref=mysess1;
  publishmodel
     caslib="tdlib1"
     exttype=teradata
     modelname="model01"
     modeltable="modeltable1"
     programfile="/mydir/scoring/model01/score.ds2"
     modelnotes="Simple model01 test model"
     ;
run;
quit;
```

Teradata で DS2 モデルを実行します。 入力テーブルと出力テーブルは同じ Teradata データベースに存在する必要があります。

```
proc scoreaccel sessref=mysess1;
  runmodel
     caslib="tdlib1"
     exttype=teradata
     modelname="model01"
     modeltable="modeltable1"
     intable="model01_score_data"
     outtable="model01_out_data"
     outkey="id"
     ;
run;
quit;
```

必要に応じて、Teradata のモデルテーブルからモデルを削除します。

```
proc scoreaccel sessref=mysess1;
  deletemodel
     caslib="tdlib1"
     exttype=teradata
     modelname="model01"
     modeltable="modeltable1"
     ;
run;
quit;
```

例 3: モデルを Hive テーブルにパブリッシュして Cloudera で実行する

要素:

CASLIB ステートメント
PROC CAS
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この例では、Hive テーブルに DS2 モデルがパブリッシュされます。次に、モデルは Spark Livy インターフェイスを使用して実行されます。

注: 2023.10 において SAS は、SAS/ACCESS Interface to Hadoop および SAS/ACCESS Interface to Spark によって使用される再配布された JDBC ドライバーを変更しました。接続オプションで変更が必要になる場合があります。詳細については、“Updates Might Be Needed Due to Driver Change” (*SAS In-Database Products: User’s Guide*)を参照してください。

プログラム

```
cas mysess1;
caslib myhadoop datasource=(
      srcType="hadoop",
      server="Hive-server",
      port="port-number",
      userName="user-name",
      password="password",
      schema="database-name",
      hadoopConfigDir="Hadoop_config_path_directory",
      jobManagementURL="http://server.company.com:8998"
      );

proc scoreaccel sessref=mysess1;
  publishmodel
    exttype=hadoop
    caslib="myhadoop"
    modelname="simple01"
    modeldatabase="mydatabase"
```

```
    programfile="/score/simple/simple.ds2";
run;
quit;

proc cas;
  sparkEmbeddedProcess.startSparkEP caslib="myhadoop"
    executorInstances=2 executorCores=2
    executorMemory=10 driverMemory=4;
run;
quit;

proc scoreaccel sessref=mysess1;
  runmodel
    exttype=hadoop
    caslib="myhadoop"
    modelname="simple01"
    modeldatabase="mydatabase"
    intable="mytable"
    outtable="myoutput"
    ;
run;
quit;

proc scoreaccel sessref=mysess1;
  deletemodel
    exttype=hadoop
    caslib="myhadoop"
    modelname="simple01"
    modeldatabase="mydatabase";
run;
quit;

proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myhadoop";
run;
quit;
```

プログラムの説明

CAS セッションを開始し、caslib を Hadoop に割り当てます。 hadoopConfigDir= オプションは、sparkep-defaults.config ファイルが保存されるフォルダーを指定します。server=オプションは、Hive サーバーを識別します。Apache Spark Livy インターフェイスを使用してジョブをディスパッチするには、スコアリングのために jobManagementURL=オプションを設定する必要があります。Kerberos が有効な場合は、userName=オプションと password=オプションを省略します。この caslib は、並列データ転送に必要なオプションを示していないことに注意してください。

```
cas mysess1;
caslib myhadoop datasource=(
    srcType="hadoop",
    server="Hive-server",
    port="port-number",
    userName="user-name",
    password="password",
    schema="database-name",
    hadoopConfigDir="Hadoop_config_path_directory",
```

```
    jobManagementURL="http://server.company.com:8998"
    );
```

モデルをパブリッシュします。 次の PUBLISHMODEL ステートメントでは、MODELDIR=オプションが指定されていません。したがって、上記の caslib 割り当てでは webHdfsUrl=オプションは必要ありません。MODELDIR=オプションを指定しないと、モデルは Hive テーブルにパブリッシュされます。この方法をお勧めします。オプションの MODELDATABASE=オプションは、モデルが保存されている既存のデータベースを指定します。

```
proc scoreaccel sessref=mysess1;
  publishmodel
    exttype=hadoop
    caslib="myhadoop"
    modelname="simple01"
    modeldatabase="mydatabase"
    programfile="/score/simple/simple.ds2";
run;
quit;
```

Spark 連続セッション用の SAS Embedded Process を開始します。 CAS プロシジャを使用して、startSparkEP アクションをサブミットします。caslib=パラメーターで、Hadoop caslib を指定します。

```
proc cas;
  sparkEmbeddedProcess.startSparkEP caslib="myhadoop"
    executorInstances=2 executorCores=2
    executorMemory=10 driverMemory=4;
run;
quit;
```

モデルを実行します。

```
proc scoreaccel sessref=mysess1;
  runmodel
    exttype=hadoop
    caslib="myhadoop"
    modelname="simple01"
    modeldatabase="mydatabase"
    intable="mytable"
    outtable="myoutput"
    ;
run;
quit;
```

必要に応じてモデルを削除します。

```
proc scoreaccel sessref=mysess1;
  deletemodel
    exttype=hadoop
    caslib="myhadoop"
    modelname="simple01"
    modeldatabase="mydatabase";
run;
quit;
```

SAS Embedded Process for Spark の連続セッションを停止するには、stopSparkEP アクションをサブミットします。 CAS セッションが終了すると、セッションも終了します。

```
proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myhadoop";
run;
quit;
```

例 4: モデルを Spark テーブルにパブリッシュして Databricks on AWS で実行する

要素:
CASLIB ステートメント
CAS プロシジャ
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この例では、PROC SCOREACCEL はモデルを Spark テーブルにパブリッシュします。次に、PROC SCOREACCEL は、SAS Embedded Process for Spark を使用して、Databricks on AWS でモデルを実行します。caslib の詳細については、*“Spark データコネクタ” (SAS Cloud Analytic Services: ユーザーガイド)*を参照してください。

注: 2023.10 において SAS は、SAS/ACCESS Interface to Hadoop および SAS/ACCESS Interface to Spark によって使用される再配布された JDBC ドライバーを変更しました。接続オプションで変更が必要になる場合があります。詳細については、*“Updates Might Be Needed Due to Driver Change” (SAS In-Database Products: User’s Guide)*を参照してください。

CAS セッションを開始します。caslib を Spark に割り当てます。 Databricks ワークスペースを 調べて、clusterId=、httpPath=、jobManagementUrl=、server=オプションの値を見つけます。

```
cas mysess;
caslib myspark datasource=(
    srcType="spark",
    platform="databricks",
    schema="default"
    clusterId="cluster-id",
    userName="token",
    authToken="authentication-token",
    password="authentication-token",
    jobManagementUrl="https://nnnnn.cloud.databricks.com/",
    server="https://nnnnn.cloud.databricks.com/",
    httpPath="my-http-path"
```

```
    );
```

モデルを Spark テーブルにパブリッシュします。 MODELDIR=オプションを指定しないでください。オプションの MODELDATABASE=オプションは、モデルが保存されている既存のデータベースを指定します。

```
proc scoreaccel sessref=mysess;
  publishmodel
    exttype=databricks
    caslib="myspark"
    modelname="mymodel"
    storefiles="/myfiles/mystore.ast"
    programfile="/myfiles/myprogram.sas"
    modeldatabase="mydatabase";
  run;
quit;
```

モデルを実行する前に、Databricks 対話型セッションを開始します。 対話型セッションを使用しない場合、SAS Embedded Process for Spark は Databricks ノートブックで実行され、実行要求ごとに Embedded Process が開始および停止されます。

```
proc cas;
  sparkEmbeddedProcess.startSparkEP
  caslib="myspark";
run;
quit;
```

モデルを実行します。 INTABLE=オプションは、モデルの実行対象となるデータを含む入力テーブルを指定します。

```
proc scoreaccel sessref=mysess;
  runmodel
    exttype=spark
    caslib="myspark"
    modelname="mymodel"
    modeldatabase="mydatabase"
    intable="mytable"
    outtable="mytable_out";
run;
quit;
```

必要に応じてモデルを削除します。

```
proc scoreaccel sessref=mysess;
  deletemodel
    exttype=databricks
    caslib="myspark"
    modelname="mymodel"
    modeldatabase="mydatabase";
run;
quit;
```

SAS Embedded Process for Spark の連続セッションを停止するには、stopSparkEP アクションをサブミットします。 CAS セッションが終了すると、セッションも終了します。

```
proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myspark";
run;
```

```
quit;
```

例 5: モデルを Spark テーブルにパブリッシュして Azure Databricks で実行する

要素:

CASLIB ステートメント
CAS プロシジャ
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL

詳細

この例では、PROC SCOREACCEL はモデルを Spark テーブルにパブリッシュします。次に、PROC SCOREACCEL は、SAS Embedded Process for Spark を使用して、Azure 上の Databricks でモデルを実行します。caslib の詳細については、"Spark データコネクタ" (*SAS Cloud Analytic Services: ユーザーガイド*)を参照してください。

注: 2023.10 において SAS は、SAS/ACCESS Interface to Hadoop および SAS/ACCESS Interface to Spark によって使用される再配布された JDBC ドライバーを変更しました。接続オプションで変更が必要になる場合があります。詳細については、"Updates Might Be Needed Due to Driver Change" (*SAS In-Database Products: User's Guide*)を参照してください。

CAS セッションを開始します。caslib を Spark に割り当てます。 Azure Databricks ワークスペースを 調べて、clusterId=、httpPath=、jobManagementUrl=、server=オプションの値を見つけます。

```
cas mysess;
caslib myspark datasource=(
     srcType="spark",
     platform="databricks",
     server="myserver.azuredatabricks.net",
     userName="token",
     password="authentication-token",
     clusterId="cluster-id",
     jobManagementUrl="https://nnnnn.cloud.databricks.net/",
     httpPath="my-http-path",
     schema="my-schema"
     );
```

モデルを Spark テーブルにパブリッシュします。 MODELDIR=オプションを指定しないでください。オプションの MODELDATABASE=オプションは、モデルが保存されている既存のデータベースを指定します。

```
proc scoreaccel sessref=mysess;
  publishmodel
    exttype=databricks
    caslib="myspark"
    modelname="mymodel"
    storefiles="/myfiles/mystore.ast"
    programfile="/myfiles/myprogram.sas"
    modeldatabase="mydatabase";
  run;
quit;
```

モデルを実行する前に、Databricks 対話型セッションを開始します。 対話型セッションを使用しない場合、SAS Embedded Process for Spark は Databricks ノートブックで実行され、実行要求ごとに Embedded Process が開始および停止されます。

```
proc cas;
  sparkEmbeddedProcess.startSparkEP
  caslib="myspark";
run;
quit;
```

モデルを実行します。 INTABLE=オプションは、モデルの実行対象となるデータを含む入力テーブルを指定します。

```
proc scoreaccel sessref=mysess;
  runmodel
    exttype=spark
    caslib="myspark"
    modelname="mymodel"
    modeldatabase="mydatabase"
    intable="mytable"
    outtable="mytable_out";
run;
quit;
```

必要に応じてモデルを削除します。

```
proc scoreaccel sessref=mysess;
  deletemodel
    exttype=databricks
    caslib="myspark"
    modelname="mymodel"
    modeldatabase="mydatabase";
run;
quit;
```

SAS Embedded Process for Spark の連続セッションを停止するには、stopSparkEP アクションをサブミットします。 CAS セッションが終了すると、セッションも終了します。

```
proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myspark";
run;
quit;
```

例 6: モデルを Amazon S3 にパブリッシュして Databricks on AWS で実行する

要素:

CASLIB ステートメント
CAS プロシジャ
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この例では、PROC SCOREACCEL はモデルを Amazon S3 にパブリッシュします。次に、PROC SCOREACCEL は、SAS Embedded Process for Spark を使用して、Amazon Web Services (AWS)上の Databricks でモデルを実行します。Amazon S3 の詳細については、*"Amazon S3 データソース" (SAS Cloud Analytic Services: ユーザーガイド)*を参照してください。

CAS セッションを開始します。パブリッシュ用に caslib を Amazon S3 に割り当てます。

```
cas casauto;
caslib mys3 datasource=(
      srctype="s3",
      bucket="bucket-name",
      accesskeyid="access-key-id",
      secretaccesskey="secret-access-key",
      region="region-value"
      );
```

モデルを Amazon S3 にパブリッシュします。

```
proc scoreaccel sessref=casauto;
  publishmodel
    exttype=filesystem
    caslib="mys3"
    modelname="mymodel"
    storefiles="/myfiles/mystore.ast"
    programfile="/myfiles/myprogram.sas"
    modeldir="/scoring/models";
  run;
quit;
```

Databricks on AWS でモデルを実行するには、Spark caslib が必要です。

```
caslib myspark datasource=(
```

```
    srctype="spark",
    platform="databricks",
    schema="default",
    clusterid="cluster-id",
    username="token",
    authtoken="authentication-token",
    password="authentication-token",
    jobmanagementurl="https://nnnnn.cloud.databricks.com/",
    hadoopjarpath="path-to-hadoop-jar-files"
    );
```

Databricks 対話型セッションを開始します。 対話型セッションを使用しない場合、SAS Embedded Process for Spark は Databricks ノートブックで実行され、実行要求ごとに Embedded Process が開始および停止されます。

```
proc cas;
  sparkEmbeddedProcess.startSparkEP
  caslib="myspark";
run;
quit;
```

Databricks on AWS でモデルを実行します。 INTABLE=オプションは、モデルの実行対象となるデータを含む入力テーブルを指定します。

```
proc scoreaccel sessref=casauto;
  runmodel
    exttype=spark
    caslib="myspark"
    modelname="mymodel"
    modeldir="dbfs:/mnt/s3-bucket-mount-point/scoring/models"
    intable="mytable"
    outtable="mytable_out";
run;
quit;
```

必要に応じてモデルを削除します。

```
proc scoreaccel sessref=casauto;
  deletemodel
    exttype=filesystem
    caslib="mys3"
    modelname="mymodel"
    modeldir="/scoring/models";
run;
quit;
```

SAS Embedded Process for Spark の連続セッションを停止するには、stopSparkEP アクションをサブミットします。 CAS セッションが終了すると、セッションも終了します。

```
proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myspark";
run;
quit;
```

例 7: モデルを ADLS にパブリッシュして Azure Databricks で実行する

要素:

CASLIB ステートメント
CAS プロシジャ
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この例では、PROC SCOREACCEL がモデルを Microsoft Azure Data Lake Storage Gen2 (ADLS)にパブリッシュします。次に、PROC SCOREACCEL は、SAS Embedded Process for Spark を使用して、Azure Databricks でモデルを実行します。

モデルを Microsoft Azure Data Lake Storage Gen2 (ADLS)にパブリッシュする準備をします。 CAS セッションを開始します。azuretenantid=セッションオプションを含める必要があります。

```
cas casauto sessopts=(azuretenantid="tenant-ID");
```

パブリッシュ用に caslib を ADLS に割り当てます。

```
caslib myadls datasource=(
    srctype="adls"
    accountname="account-name",
    applicationid="application-id",
    filesystem="mystorage",
    dnssuffix="dfs.core.windows.net"
    );
```

PUBLISHMODEL ステートメントで EXTTYPE=FILESYSTEM を指定して、モデルを ADLS にパブリッシュします。 CASLIB=引数で、ADLS caslib を指定します。PASSWORD=引数には、シークレットとも呼ばれる RBAC からのパスワードを使用します。

```
proc scoreaccel sessref=casauto;
  publishmodel
    exttype=filesystem
    caslib="myadls"
    password="RBAC-client-secret"
    modelname="mymodel"
    modeldir="/scoring/models"
    storefiles="/myfiles/mystore.ast"
```

```
    programfile="/myfiles/myprogram.sas";
run;
quit;
```

Azure Databricks でモデルを実行するには、Spark caslib が必要です。

```
caslib myspark datasource=(
    srctype='spark',
    platform=databricks,
    schema="default",
    clusterid="cluster-id",
    username="token",
    authtoken="authentication-token",
    password="authentication-token",
    jobmanagementurl="https://nnnnn.cloud.databricks.com/",
    hadoopjarpath="path-to-hadoop-jar-files"
    );
```

Databricks 対話型セッションを開始します。 対話型セッションを使用しない場合、SAS Embedded Process for Spark は Databricks ノートブックで実行され、実行要求ごとに Embedded Process が開始および停止されます。ほとんどのお客様は、パフォーマンスを向上させるために対話型セッションを使用することを選択しています。

```
proc cas;
  sparkEmbeddedProcess.startSparkEP
  caslib="myspark";
run;
quit;
```

Azure Databricks でモデルを実行します。 INTABLE=オプションは、モデルの実行対象となるデータを含む入力テーブルを指定します。

```
proc scoreaccel sessref=casauto;
  runmodel
    exttype=spark
    caslib="myspark"
    modelname="mymodel"
    modeldir="dbfs:/mnt/blob-storage-mount-point/scoring/models"
    intable="mytable"
    outtable="mytable_out";
run;
quit;
```

必要に応じてモデルを削除します。

```
proc scoreaccel sessref=casauto;
  deletemodel
    exttype=filesystem
    caslib="myadls"
    password="RBAC-client-secret"
    modelname="mymodel"
    modeldir="/scoring/models";
run;
quit;
```

SAS Embedded Process for Spark の連続セッションを停止するには、stopSparkEP アクションをサブミットします。 CAS セッションが終了すると、セッションも終了します。

```
proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myspark";
run;
quit;
```

例 8: モデルを SQL Server テーブルにパブリッシュして Azure Synapse で実行する

要素:

CASLIB ステートメント
CAS プロシジャ
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL

詳細

この例では、PROC SCOREACCEL がモデルを SQL Server テーブルにパブリッシュします。次に、PROC SCOREACCEL は、SAS Embedded Process for Spark を使用して Synapse でモデルを実行します。

モデルを SQL Server テーブルにパブリッシュする前の要件として、SQL Server 用の Microsoft JDBC ドライバーをダウンロードする必要があります。詳細は、"Configuration for In-Database Model Scoring" (*SAS Viya In-Database Technologies: Deployment and Administration Guide*)を参照してください。

CAS セッションを開始します。モデルをパブリッシュするには、JDBC caslib を割り当てます。 Synapse の場合、次の方法で caslib 内のスキーマとデータベースの両方を識別する必要があります。 caslib の SCHEMA= オプションでスキーマ値を指定します。データベース値の場合、JDBC caslib は DATABASE=オプションをサポートしていないため、URI=オプションを使用して Microsoft JDBC Driver for SQL Server の接続文字列を指定する必要があります。(接続文字列にはデータベース値が含まれますが、スキーマ値は含まれません。)上で述べたように、前提条件は SQL Server 用 Microsoft JDBC ドライバーをダウンロードすることです。詳細は、"Configuration for In-Database Model Scoring" (*SAS Viya In-Database Technologies: Deployment and Administration Guide*)を参照してください。次に、Synapse ワークスペースの JDBC 接続文字列に、JDBC (SQL 認証)用に提供されている接続文字列をコピーします。caslib の URI=オプションでその接続文字列を指定します。さらに、接続文字列にプロパティ useBulkCopyForBatchInsert=true を追加する必要があります。

```
cas mysess;
caslib myjdbc datasource=(
    srcType="jdbc",
    userName="user-name",
    password="password",
    uri="JDBC-connection-options;
```

```
    useBulkCopyForBatchInsert=true;BatchSize=400",
  schema="SQL-server-schema-name"
  );
```

モデルを SQL Server テーブルにパブリッシュします。 モデルを SQL Server テーブルに保存するには、MODELDIR=オプションを指定しないでください。Synapse の場合、SQL Server テーブルに格納されているモデルを削除、パブリッシュ、または実行するときに、MODELSCHEMA=オプションが必要です。EXTTYPE=SYNAPSE を指定し、JDBC caslib を参照します。

```
proc scoreaccel sessref=mysess;
  publishmodel
    exttype=synapse
    caslib="myjdbc"
    modelname="mymodel"
    modelschema="myschema"
    storefiles="/myfiles/mystore.ast"
    programfile="/myfiles/myprogram.sas"
    ;
run;
quit;
```

対話型セッションを開始してモデルを実行するには、Spark caslib を割り当てます。 caslib で、platform="synapse"を指定します。jobManagementUrl=パラメーターには、接続先の Spark プールに固有の Livy REST URL を指定します。jobManagementUrl=値の詳細については、“Forming the Livy REST URL for Synapse” (*SAS In-Database Products: User’s Guide*)を参照してください。

```
cas mysess;
caslib myspark datasource=(
    srcType="spark",
    platform="synapse",
    userName="application-id",
    password="secret",
    server="myserver.xxxx.net",
    database="my-database-name",
    jobManagementUrl="Livy-rest-url"
    );
```

モデルを実行する前に対話型セッションを開始します。 対話型セッションを使用しない場合は、クラスターに構成されているデフォルトのリソース設定を使用して、SAS Embedded Process for Spark が自動的に開始されます。

```
proc cas;
  sparkEmbeddedProcess.startSparkEP
  caslib="myspark";
run;
quit;
```

モデルを実行します。 INTABLE=値と OUTTABLE=値には、2 部構成の名前を使用します。SQL Server のスキーマ名を指定し、その後にドット、テーブル名を続けます。

```
proc scoreaccel sessref=mysess;
  runmodel
    exttype=spark
    caslib="myspark"
    modelname="mymodel"
```

```
    modelschema="myschema"
    intable="myschema.mytable"
    outtable="myschema.mytable_out";
run;
quit;
```

必要に応じてモデルを削除します。

```
proc scoreaccel sessref=mysess;
  deletemodel
    exttype=synapse
    caslib="myjdbc"
    modelname="mymodel"
    modelschema="myschema";
run;
quit;
```

SAS Embedded Process for Spark の連続セッションを停止するには、stopSparkEP アクションをサブミットします。 CAS セッションが終了すると、セッションも終了します。

```
proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myspark";
run;
quit;
```

例 9: モデルを ADLS にパブリッシュして Azure Synapse で実行する

要素:

- CASLIB ステートメント
- CAS プロシジャ
- PROC SCOREACCEL ステートメント
- PUBLISHMODEL ステートメント
- RUNMODEL ステートメント
- DELETEMODEL ステートメント

詳細

この例では、PROC SCOREACCEL がモデルを Microsoft Azure Data Lake Storage Gen2 (ADLS)にパブリッシュします。次に、PROC SCOREACCEL は、SAS Embedded Process for Spark を使用して Azure Synapse でモデルを実行します。モデルは SQL Server テーブルに対して実行されます。

Synapse の場合、PROC SCOREACCEL は Azure Active Directory (Azure AD)アプリケーション登録とロールベースのアクセスコントロール(RBAC)を使用します。Synapse Spark プールの Livy REST URL を指定する方法や、SQL Server の要件など

の重要な情報については、“In-Database Model Scoring” (*SAS In-Database Products: User's Guide*)を参照してください。

プログラム

```
cas mysess sessopts=(azuretenantid="tenant-id");

caslib myadls datasource=(
      srctype="adls"
      accountname="myaccount",
      applicationid="application-id",
      filesystem="mystorage",
      dnssuffix="dfs.core.windows.net"
      );

proc scoreaccel sessref=mysess;
  publishmodel
     exttype=filesystem
     caslib="myadls"
     password="RBAC-client-secret"
     modelname="mymodel"
     modeldir="/scoring/models"
     programfile="/myfiles/myprogram.ds2";
run;
quit;

caslib myspark datasource=(
      srctype="spark",
      platform="synapse",
      username="application-id",
      password="RBAC-client-secret",
      schema="SQL-server-database-name",
      hadoopjarpath="path-to-hadoop-jar-files",
      jobmanagementurl="Livy-rest-url"
      );

proc cas;
  sparkEmbeddedProcess.startSparkEP caslib="myspark"
     executorInstances=2 executorCores=2
     executorMemory=10 driverMemory=4;
run;
quit;

proc scoreaccel sessref=mysess;
  runmodel
     exttype=spark
     caslib="myspark"
     modelname="mymodel"
     modeldir="/scoring/models"
     intable="SQL-server-database-schema-name.in-table-name"
     outtable="SQL-server-database-schema-name.out-table-name";
run;
quit;

proc scoreaccel sessref=mysess;
  deletemodel
     exttype=filesystem
```

```
    caslib="myadls"
    password="RBAC-client-secret"
    modelname="mymodel"
    modeldir="/scoring/models";
run;
quit;

proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myspark";
run;
quit;
```

プログラムの説明

モデルを Microsoft Azure Data Lake Storage Gen2 (ADLS)にパブリッシュする準備をします。 CAS セッションを開始します。azuretenantid=セッションオプションを含める必要があります。

```
cas mysess sessopts=(azuretenantid="tenant-id");
```

caslib を ADLS に割り当てます。

```
caslib myadls datasource=(
    srctype="adls"
    accountname="myaccount",
    applicationid="application-id",
    filesystem="mystorage",
    dnssuffix="dfs.core.windows.net"
    );
```

PUBLISHMODEL ステートメントで EXTTYPE=FILESYSTEM を指定して、モデルを ADLS にパブリッシュします。 CASLIB=引数で、ADLS caslib を指定します。PASSWORD=引数には、シークレットとも呼ばれる RBAC からのパスワードを使用します。

```
proc scoreaccel sessref=mysess;
  publishmodel
    exttype=filesystem
    caslib="myadls"
    password="RBAC-client-secret"
    modelname="mymodel"
    modeldir="/scoring/models"
    programfile="/myfiles/myprogram.ds2";
run;
quit;
```

Synapse でモデルを実行するには、Spark caslib が必要です。 username=はアプリケーション ID です。password=パラメーターには、シークレットとも呼ばれる RBAC のパスワードを使用します。SQL Server テーブルの場合、スキーマ名はデータベース名です。jobmanagementurl=パラメーターには、Synapse Spark プールの Livy REST URL を指定します。jobManagementUrl=値の詳細については、"Forming the Livy REST URL for Synapse" (*SAS In-Database Products: User's Guide*) を参照してください。Hadoop JAR パスは必須です。Hadoop 構成パスは必要ありません。

```
caslib myspark datasource=(
```

```
    srctype="spark",
    platform="synapse",
    username="application-id",
    password="RBAC-client-secret",
    schema="SQL-server-database-name",
    hadoopjarpath="path-to-hadoop-jar-files",
    jobmanagementurl="Livy-rest-url"
    );
```

Spark 連続セッション用の SAS Embedded Process を開始します。 CAS プロシジャを使用して、startSparkEP アクションをサブミットします。caslib=パラメーターで、Spark caslib を指定します。

```
proc cas;
  sparkEmbeddedProcess.startSparkEP caslib="myspark"
    executorInstances=2 executorCores=2
    executorMemory=10 driverMemory=4;
run;
quit;
```

RUNMODEL ステートメントで EXTTYPE=SPARK を指定して、Synapse でモデルを実行します。 CASLIB=引数で、Spark caslib を指定します。

```
proc scoreaccel sessref=mysess;
  runmodel
    exttype=spark
    caslib="myspark"
    modelname="mymodel"
    modeldir="/scoring/models"
    intable="SQL-server-database-schema-name.in-table-name"
    outtable="SQL-server-database-schema-name.out-table-name";
run;
quit;
```

必要に応じてモデルを削除します。

```
proc scoreaccel sessref=mysess;
  deletemodel
    exttype=filesystem
    caslib="myadls"
    password="RBAC-client-secret"
    modelname="mymodel"
    modeldir="/scoring/models";
run;
quit;
```

SAS Embedded Process for Spark の連続セッションを停止するには、stopSparkEP アクションをサブミットします。 CAS セッションが終了すると、セッションも終了します。

```
proc cas;
  sparkEmbeddedProcess.stopSparkEP caslib="myspark";
run;
quit;
```

例 10: SingleStore でモデルをパブリッシュ、実行、および削除する

要素:

CASLIB ステートメント
PROC SCOREACCEL ステートメント
PUBLISHMODEL ステートメント
RUNMODEL ステートメント
DELETEMODEL ステートメント

詳細

この例では、PROC SCOREACCEL が SingleStore でモデルをパブリッシュして実行します。SAS SpeedyStore 製品のライセンスを取得して配置した場合のみ、In-database モデルスコアリングがサポートされます。詳細については、*"SAS SpeedyStore データコネクタ" (SAS Cloud Analytic Services: ユーザーガイド)*を参照してください。In-database モデルスコアリングは、SingleStore 標準データコネクタではサポートされていません。

配置の詳細については、*SAS SpeedyStore: 管理および構成ガイド*を参照してください。

プログラム

```
cas casauto;

caslib mycaslib datasource=(
    srcType="singlestore"
    database="my_database"
    host="svc-sas-singlestore-cluster-dml"
    port="3306"
    password="my_password"
    userName="my_username"
    );

proc scoreaccel;
  publishmodel
    exttype=singlestore
    caslib="mycaslib"
    modelname="mymodel"
    programfile="/myfiles/myprogram.ds2"
   ;
run;
quit;
```

```
proc scoreaccel;
  runmodel
    exttype=singlestore
    caslib="mycaslib"
    modelname="mymodel"
    intable="in-table-name"
    outtable="out-table-name";
   ;
run;
quit;
```

```
proc scoreaccel;
  deletemodel
    exttype=singlestore
    caslib="mycaslib"
    modelname="mymodel"
   ;
run;
quit;
```

プログラムの説明

モデルを SingleStore にパブリッシュする準備をします。 CAS セッションがまだ開始されていない場合は、開始します。

```
cas casauto;
```

caslib を SingleStore に割り当てます。 シングルサインオンが有効な場合は、password=オプションと userName=オプション、または authenticationDomain=オプションを含めないでください。

```
caslib mycaslib datasource=(
   srcType="singlestore"
   database="my_database"
   host="svc-sas-singlestore-cluster-dml"
   port="3306"
   password="my_password"
   userName="my_username"
   );
```

PUBLISHMODEL ステートメントで EXTTYPE=SINGLESTORE を指定して、モデルを SingleStore にパブリッシュします。 CASLIB=引数で、SingleStore caslib を指定します。

```
proc scoreaccel;
  publishmodel
    exttype=singlestore
    caslib="mycaslib"
    modelname="mymodel"
    programfile="/myfiles/myprogram.ds2"
   ;
run;
quit;
```

PUBLISHMODEL ステートメントで EXTTYPE=SINGLESTORE を指定してモデルを実行します。 CASLIB=引数で、SingleStore caslib を指定します。

```
proc scoreaccel;
  runmodel
    exttype=singlestore
    caslib="mycaslib"
    modelname="mymodel"
    intable="in-table-name"
    outtable="out-table-name";
   ;
run;
quit;
```

必要に応じてモデルを削除します。

```
proc scoreaccel;
  deletemodel
    exttype=singlestore
    caslib="mycaslib"
    modelname="mymodel"
   ;
run;
quit;
```

55

SORT プロシジャ

概要: SORT プロシジャ

SORT プロシジャの機能

SORT プロシジャは、SAS データセットオブザベーションを 1 つ以上の文字変数または数値変数の値を基準にして並べ替えます。SORT プロシジャは、元のデータセットを置き換えるか、新しいデータセットを作成します。PROC SORT は、出力データセットのみを作成します。詳細については、“プロシジャの出力” (2068 ページ)を参照してください。

注: 入力データセットに拡張属性が定義されている場合、PROC SORT は拡張属性を出力データセットにプロパゲートします。拡張属性の詳細については、“拡張属性” (365 ページ)を参照してください。

動作環境の情報: 本章で説明する並べ替え機能は、すべての動作環境で使用できます。また、SAS システムオプション SORTPGM=の HOST 値を使用すると、この他にも使用する動作環境で使用可能な並べ替えオプションを使用できる場合があります。他の並べ替え機能の詳細については、動作環境に関する SAS ドキュメントを参照してください。

SAS データセットの並べ替え

次の例では、元のデータセットは姓のアルファベット順に並べられていました。PROC SORT を実行すると、元のデータセットを従業員 ID 番号順に並べ替えられたデータセットに置き換えることができます。次のログは、この PROC SORT ステップの実行結果を示しています。アウトプット 55.286 (2029 ページ)は、PROC PRINT ステップの結果を示しています。出力を生成するステートメントは次のとおりです。

```
proc sort data=employee;
  by idnumber;
run;

proc print data=employee;
run;
```

例のコード 55.1　PROC SORT により生成される SAS ログ

```
NOTE: There were six observations read from the data set WORK.EMPLOYEE.
NOTE: The data set WORK.EMPLOYEE has six observations and three variables.
NOTE: PROCEDURE SORT used:
      real time         0.01 seconds
      cpu time          0.01 seconds
```

アウトプット 55.1　1 つの変数の値によるオブザベーションの並べ替え

```
                          The SAS System                        1

                 Obs     Name          IDnumber

                  1      Belloit         1988
                  2      Wesley          2092
                  3      Lemeux          4210
                  4      Arnsbarger      5466
                  5      Pierce          5779
                  6      Capshaw         7338
```

次の出力には、3 つの変数を基準にしたより複雑な並べ替えの結果が表示されます。この例の企業は、町、債務(金額の高い順)、アカウント番号の順に並べ替えられます。この出力を作成するプログラムの説明については、“例 2: 降順の並べ替え” (2072 ページ)を参照してください。

アウトプット 55.2　3 つの変数値によるオブザベーションの並べ替え

```
                    Customers with Past-Due Accounts                   1
                  Listed by Town, Amount, Account Number

                                                                 Account
      Obs    Company                   Town               Debt    Number

        1    Paul's Pizza              Apex              83.00     1019
        2    Peter's Auto Parts        Apex              65.79     7288
        3    Watson Tabor Travel       Apex              37.95     3131
        4    Tina's Pet Shop           Apex              37.95     5108
        5    Apex Catering             Apex              37.95     9923
        6    Deluxe Hardware           Garner           467.12     8941
        7    Boyd & Sons Accounting    Garner           312.49     4762
        8    World Wide Electronics    Garner           119.95     1122
        9    Elway Piano and Organ     Garner            65.79     5217
       10    Ice Cream Delight         Holly Springs    299.98     2310
       11    Tim's Burger Stand        Holly Springs    119.95     6335
       12    Strickland Industries     Morrisville      657.22     1675
       13    Pauline's Antiques        Morrisville      302.05     9112
       14    Bob's Beds                Morrisville      119.95     4998
```

概念: SORT プロシジャ

スレッド化された並べ替え

THREADS システムオプションを使用すると、スレッド化された並べ替えを有効にできます。スレッド化された並べ替えにより、処理操作における一定の並列処理が達成されます。この並列処理の目的は、所定の操作を完了するための処理時間を削減し、それによって追加 CPU リソースのコストを抑えることです。詳細については、"THREADS システムオプション" (*SAS システムオプション: リファレンス*)を参照してください。

マルチスレッド SAS の並べ替えは、PROC SORT ステートメントで THREADS オプションを指定しているときにも起動できます。マルチスレッド並べ替えでは、UTILLOC=システムオプションで指定される場所のいずれかにある 1 つのユーティリティファイルにすべての一時データが保存されます。このユーティリティファイルのサイズは、入力データセットから読み込まれるデータ量に比例します。入力データセットから読み込まれるデータ量が大きいとき、または SORT プロシジャで使用可能なメモリ量が小さいとき、同じサイズの 2 つ目のユーティリティファイルをこれらの場所のいずれかに作成できます。詳細については、"UTILLOC= システムオプション" (*SAS システムオプション: リファレンス*)を参照してください。

注: TAGSORT オプション (2054 ページ)は、スレッド化された並べ替えをサポートしていません。

マルチスレッド SAS の並べ替えは、THREADS システムオプションが指定されており、CPUCOUNT=システムオプションの値が 1 より大きいときに起動できます。SAS システムオプション CPUCOUNT=の値はスレッド化された並べ替えのパフォーマンスに影響します。CPUCOUNT=は、スレッド化されたプロシジャで使用できるシステム CPU の数を示します。

計算された統計は、オブザベーションが処理される順序に応じてわずかに異なる場合があります。このような変動は、浮動小数点演算によって導入される数値エラーによるものであり、その結果は概算であり、正確ではないと見なす必要があります。オブザベーション処理の順序は、マルチスレッド処理または並列処理の非決定論的影響を受ける可能性があります。処理の順序は、ACCESS エンジンを介してクエリ結果を提供する DBMS などのデータソースによって生成されるオブザベーションの一貫性のない、または非決定論的な並べ替えによっても影響を受ける可能性があります。詳細については、"Base SAS のスレッド処理" (*SAS プログラマガイド: エッセンシャル*)を参照してください。

詳細については、"THREADS システムオプション" (*SAS システムオプション: リファレンス*) および、"CPUCOUNT= システムオプション" (*SAS システムオプション: リファレンス*)を参照してください。

数値変数の並べ替え順序

数値変数について、昇順の比較順序を次に示します。

1 SAS 欠損値(ピリオドや特殊欠損値で表示)

2 負数値

3 ゼロ

4 正数値

文字変数の並べ替え順序

デフォルトの照合シーケンス

英数字の並べ替え順序は、照合シーケンスと呼ばれます。この並べ替え順序は、セッションエンコーディングによって 決定されます。

デフォルトでは、PROC SORT は、SAS Viya プラットフォームで文字値を比較するときに ASCII 照合順序を使用します。

さまざまな照合シーケンスとそれらの使用タイミングについては、 “照合順序” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

注: ASCII と EBCDIC は、セッションエンコーディングのファミリ名を表します。並べ替え順序は、エンコーディングを参照して決定できます。

ASCII 順序

Linux 動作環境では、ASCII 照合順序が使用されます。

英語の ASCII シーケンスは、表示可能な文字を昇順に並べると、次のテーブルに示す順序と一致します。

表 55.1 ASCII 並べ替え順序の例

blank !"# $ % & ' ( ) * + , - ./0 1 2 3 4 5 6 7 8 9 : ; < = > ? @

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ◦ _
a b c d e f g h i j k l m n o p q r s t u v w x y z { } ~

ASCII シーケンスの主な特徴は、数字の後に大文字が並べ替えられ、その後に小文字が並べ替えられることです。表示可能な最小文字はブランクです。

EBCDIC 順序

英語の EBCDIC シーケンスの並べ替え順序は、次の並べ替え順序の例と一致しています。

表 55.2 EBCDIC 並べ替え順序の例

空白 。 < (+ | & ! $ *) ; ¬ - / , % _ > ? : # @ ' = "
a b c d e f g h i j k l m n o p q r ~ s t u v w x y z
{ A B C D E F G H I } J K L M N O P Q R S T
U V W X Y Z
0 1 2 3 4 5 6 7 8 9

EBCDIC シーケンスの主な特徴は、小文字の後に大文字が並べ替えられ、その後に数字が並べ替えられることです。また、アルファベット順序の途中に特殊文字がいくつか割り込んでいるので注意してください。表示可能な最小文字はブランクです。

文字変数の並べ替え順序の指定

オプション ASCII、EBCDIC、DANISH、NATIONAL、SWEDISH、および REVERSE を使用すると、HOST カタログに保存されている照合シーケンスを指定できます。

独自の照合シーケンスの作成や提供されている照合シーケンスの変更を行う必要がある場合は、TRANTAB プロシジャを使用して変換テーブルを作成または変更します。独自の変換テーブルを作成すると、その変換テーブルは PROFILE カタログに保存され、HOST カタログの同名の変換テーブルよりも優先されます。 詳細については、*"TRANTAB プロシジャ" (SAS 各国語サポート(NLS): リファレンスガイド)*を参照してください。

注: システム管理者は、新規作成したテーブルを PROFILE カタログから HOST カタログにコピーして、HOST カタログを変更できます。その場合、すべてのユーザーが新規または修正された変換テーブルにアクセスできます。

言語照合では、言語のルールに従ってデータが並べ替えられます。 言語照合の詳細については、“照合順序” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

保存された並べ替え情報

PROC SORT は、データセットの並べ替えに使用する BY 変数、照合シーケンスおよび文字セットを記録します。この情報は、不要な並べ替えを回避するためにデータセットとともに保存されます。

PROC SORT は、データセットを並べ替える前に、保存されている並べ替え情報を確認します。現在の並べ替え方法でデータセットを並べ替えようとすると、PROC SORT は並べ替えを実行せず、その旨を示すメッセージをログに書き込みます。この動作を無効にするには、FORCE オプションを使用します。現在の並べ替えと同じ方法でデータセットを並べ替える場合に OUT=データセットを指定しても、PROC SORT は DATA=データセットのコピーを作成するだけです。

PROC SORT が保存した並べ替え情報を無効にするは、_NULL_値と SORTEDBY=データセットオプションを使用します。“SORTEDBY= データセットオプション” (*SAS データセットオプション: リファレンス*)を参照してください。

既存データセットの並べ替え情報を変更する場合は、DATASETS プロシジャの MODIFY ステートメントで SORTEDBY=データセットオプションを使用します。詳細については、“MODIFY ステートメント” (441 ページ)を参照してください。

データセットと保存されている並べ替え情報にアクセスするには、PROC DATASETS で CONTENTS ステートメントを使用します。詳細については、“CONTENTS ステートメント” (400 ページ)を参照してください。

PROC SORT でデータセットを並べ替えるときの基準となる変数の数は、利用可能なメモリでのみ制限されます。PROC SQL を使用して結果セットの行を並べ替えるときの基準となる列の数も利用可能なメモリでのみ制限されます。ソートインジケーターは、基本データセットのメタデータに格納されるか、メモリで表されるかにかかわらず、127 個の変数に制限されます。このため、ソートインジケーターに保存できる変数、または SORTEDBY=データセットオプションでリストできる変数は最大 127 個です。127 を超える数の変数で並べ替えすると、最初の 127 個だけがソートインジケーターに記録されます。データセットを BY 変数のリスト全体でもう一度並べ替えると、(127 を超える)追加の変数はソートインジケーター内に見つからないため、データセットは並べ替えられているものと認識されません。詳細な説明については、“並べ替えられたデータセット” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

事前に並べ替えられた入力データセット

“PRESORTED” (2053 ページ)オプションを指定すると、すでに並べ替えられたデータセットは並べ替えられません。並べ替えの前に、入力データセット内のオブザベーションの順序がチェックされ、オブザベーションが順番どおりかどうかが確認されます。データセットがすでに BY ステートメントで指定したキー変数に従った順序であることがわかっている、またはそう思われる場合に、PRESORTED オプション

を使用します。データセット内のオブザベーションの順序をチェックするには、データセットを読み取り、読み取った各オブザベーションの BY 変数と、その前のオブザベーションの BY 変数を比較します。この処理は、すべてのデータセットが読み取られるか、順序の外れたオブザベーションが検出されるまで続行されます。

データセットをすべて読み取っても順序を外れたオブザベーションが見つからなかった場合は、2 つのアクションのどちらかが実行されます。出力データセットを指定していない場合は、入力データセットの並べ替え順序メタデータが更新され、順序が検証済みであることが示されます。この検証には、データセットが指定された BY 変数に従って正当に並べ替えられていることが示されます。それ以外の場合、データセットは並べ替え済みと見なされ、入力データセットのメタデータが更新されるか、OUT=が指定されている場合は、データが出力データセットにコピーされます。

データセット内のオブザベーションが順序どおりでない場合、データセットが並べ替えられます。

“NODUPKEY” (2049 ページ)オプションを指定した場合は、順序チェックによって、データセットにキーが重複するオブザベーションが存在するかどうかが判別されます。キーが重複しているオブザベーションが見つかった場合、データセットは並べ替えられていないと見なされ、並べ替えが実行されます。それ以外の場合、データセットは並べ替え済みと見なされ、入力データセットのメタデータが更新されるか、OUT=が指定されている場合は、データが出力データセットにコピーされるアクションが実行されます。実行されるアクションについては、前の段落で詳しく説明しています。

入力データセットのメタデータで、データがすでに BY ステートメントに表示されるキー変数に従って並べ替え済みで、その入力データセットが検証済みであることが示されている場合は、順序チェックも並べ替えも実行されません。

詳細については、“並べ替えられたデータセット” (*SAS プログラマガイド: エッセンシャル*) および“SORTVALIDATE システムオプション” (*SAS システムオプション: リファレンス*)との相互作用を参照してください。

データセットの言語的な並べ替えと ICU

言語照合では、言語とロケールに関連付けられたルールに従って、文化的に配慮した方法で文字が並べ替えられます。ルールおよびデフォルトの照合順序は、現在のロケール設定で指定された言語に基づきます。実装は、International Components for Unicode (ICU)ライブラリによって提供されています。その結果、Unicode 照合アルゴリズム(UCA)との互換性が高くなります。

言語オプション(SORTSEQ=LINGUISTIC)が Base SAS プロシジャの PROC SORT で指定された場合、SAS システムは ICU 照合を提供します。SQL プロシジャで SORTSEQ=オプションを使用し、SORTSEQ=LINGUISTIC システムオプションを指定して言語照合を指定できるようになりました。

注: SORTSEQ=LINGUISTIC システムオプションが指定された場合、PROC SORT と PROC SQL のみが影響を受けます。

SORTSEQ=LINGUISTIC オプションを指定すると、SAS は Unicode 照合アルゴリズム(UCA)の参照実装として、またデファクトスタンダードとして ICU ライブラリに依存します。

SAS Viya プラットフォームでは、SAS に組み込まれ、PROC SORT で使用される ICU ライブラリのバージョンは ICU 63 です。詳細については、 ICU 63 のダウンロード を参照してください。

注: ICU ライブラリは 70.1 にアップグレードされました。アップグレードにより、zh_CN や zh_TW などの中国語ロケールの言語照合に非互換性が見られる場合があります。

中国語ロケールの言語照合機能によって並べ替えられた既存のデータセットは、もう一度並べ替えする必要があります。

言語照合のために PROC SORT で使用されている ICU のバージョンを変更すると、別のバージョンの SAS で並べ替えられたデータセットの解釈に影響する可能性があります。2 つの SAS バージョンが異なるバージョンの ICU を使用しているときに、一方のバージョンの SAS でデータセットが 1 つ以上の文字変数により言語的に並べ替えされている場合、そのたデータセットはもう一方のバージョンの SAS でアクセスされたときに並べ替えられているものと認識されます。照合ルールは ICU バージョン間で変更されることがあるため、ルールの変更によって、PROC SORT で作成されたオブザベーションの順序が異なるものになる場合があります。順序の違いが無視されると、処理中に予期しない結果が生じる可能性があります。

言語的に並べ替える際に、SAS で使用されている ICU バージョンは、データセットヘッダーに保存されているソートインジケーターに記録されます。データセットが並べ替え済みと見なすかどうかを判定するときに、ICU バージョンが確認されます。使用中の ICU バージョンと、データセットのソートインジケーターに記録された ICU バージョンに違いがあると、示された並べ替え順序は無視され、データセットが並べ替えられていないものと見なされます。

注: PROC CONTENTS 出力には、使用中の ICU バージョンが示されます。“例 5: ALTERNATE_HANDLING=を使用した言語的な並べ替え” (2079 ページ)を参照してください。

永久データセットのソートインジケーターを無視する場合は、処理を容易にするため、順序を再度指定し、データセットでソートインジケーターを再確立した方が望ましい場合があります。これを行うには、PRESORTED オプションを指定した PROC SORT を使用します。多くの場合、データセット内のオブザベーションの順序は乱れておらず、正しい可能性があるため、SORT プロシジャは完全な並べ替えを実行しなくても、データセットを順番に読み込むだけでインジケーターを再確立できます。オブザベーションの順序が正しくない場合、SORT プロシジャが必要に応じてオブザベーションを並べ替えます。

COPY プロシジャと MIGRATE プロシジャの両方で、入力データセットで記録された ICU バージョンが SAS で使用中のバージョンと異なる場合、入力データセットのソートインジケーターは無視され、出力データセットは並べ替え済みとしてマークされず、メッセージが SAS ログに書き込まれます。ただし、どちらのプロシジャも、入力から読み込んだときと同じ順序でオブザベーションを出力データセットに書き込みます。物理的な順序が OUT=出力先ライブラリに使用されているエンジンでサポートされている場合、この順序は保持されます。このため、新しいリリースの SAS

に移行するときは、PRESORTED オプションを指定した PROC SORT を使用して永久データセットの並べ替え順序を再確立することを検討してください。

SAS での言語照合の使用方法の詳細については、次のドキュメントおよび PROC SORT SORTSEQ=LINGUISTIC システムオプションで説明されています。

- “SORTSEQ=sort-table | LINGUISTIC” (*SAS SQL プロシジャユーザーガイド*)を参照してください。
- PROC SORT オプション“ LINGUISTIC*<(collating-options)>*” (2042 ページ)を参照してください。
- “言語照合の指定” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。
- 12 章, “CONTENTS プロシジャ,” (283 ページ)を参照してください。
- 13 章, “COPY プロシジャ,” (303 ページ)を参照してください。
- 36 章, “MIGRATE プロシジャ,” (1335 ページ)を参照してください。
- 付録 3, “UNICODE ライセンス” (2525 ページ)を参照してください。

次の SAS の資料には、言語照合に関する詳細な情報が記載されています。

- Creating Order out of Character Chaos: Collation Capabilities of the SAS System
- Linguistic Collation: Everyone Can Get What They Expect
- Processing Multilingual Data with the SAS 9.2 Unicode Server
- New Language Features in SAS 9.2 for the Global Enterprise

次は、言語照合の詳細について参照する必要のあるサードパーティの資料のリストです。

- Unicode Collation Algorithm (UCA) Specification を参照してください。
- ICU User Guide の Collation セクションを参照してください。
- 照合規則の詳細については、ICU Documentation: Collation

システムオプション SORTPGM を使用して SAS にデータセットの並べ替えを強制する

システムオプション SORTPGM=SAS を設定することにより、データセットを SAS で並べ替えるように強制できます。このシステムオプションは、PROC SORT を使用する前に設定する必要があります。システムオプション SORTPGM=SAS を設定すると、データセットが SAS が期待する順序で並べ替えられます。

デフォルトでは、SORTPGM のデフォルトは BEST であり、SAS が決定するのはおそらく最良のパフォーマンスの選択です。SORTPGM が BEST に設定されている場合、並べ替えは次のように実行される可能性があります。

- システムまたはサードパーティのホスト並べ替えユーティリティプログラム(インストールされていて利用可能な場合)。

- 入力データがデータベースに存在し、SAS/ACCESS エンジンを使用して読み取る場合は DBMS。

正しく機能するには、ホストの並べ替えまたは DBMS と SAS が互換性のある操作用に構成されている必要があります。ホストの並べ替えまたは DBMS は、SAS とは異なる方法でデータを順序付けるように構成できます。SAS に返されるオブザベーションが SAS の期待どおりに順序付けられていない場合、SORTPGM=システムオプションを SAS に設定して、SAS が必要とする順序でデータを並べ替えるように SAS に指示できます。

"SORTPGM システムオプション" (*SAS システムオプション: リファレンス*)と"PROC SORT: In-Database 処理" (2065 ページ)を参照してください。

構文: SORT プロシジャ

要件 BY ステートメント

サポート: スレッド処理

注: 一部の SAS ステートメントおよびオプションは、CAS で実行する場合はサポートされておらず、PROC SORT が CAS で実行するかどうかを決定するときにエラーが発生する可能性があります。

一部の SAS ステートメントおよびオプションは、CAS で実行する場合はサポートされませんが、CAS での操作を妨げることはありません。PROC SORT が CAS で実行することを決定すると、そのようなオプションについてのメモが印刷されます。

ほとんどの PROC SORT 固有のオプションは効果がなく、CAS での実行を妨げることはありません。

ヒント: ステートメント ATTRIB、FORMAT、LABEL、WHERE を PROC SORT プロシジャと使用できます。

発生する In-Database 処理に対し、データは SAS In-Database 処理用に適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、"PROC SORT: In-Database 処理" (2065 ページ)を参照してください。

PROC SORT <*collating-sequence-option*> <*other options*>;

ATTRIB *variable-list(s) attribute-list(s)*;

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

FORMAT *variable-1* <...*variable-n*> <*format*> <DEFAULT=*default-format*>;

KEY *variable(s)* </ *option*>;

LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク	例
"PROC SORT ステートメント"	SAS データセットオブザベーションを 1 つ以上の文字変数または数値変数の値を基準にして並べ替えます	Ex. 1, Ex. 3, Ex. 4

ステートメント	タスク	例
“BY ステートメント”	並べ替え変数を指定します。	Ex. 1, Ex. 2, Ex. 4
“KEY ステートメント”	並べ替えキーと変数を指定します。	

PROC SORT ステートメント

SAS データセットオブザベーションを 1 つ以上の文字変数または数値変数の値を基準にして並べ替えます。

例:

“例 1: 複数の変数の値による並べ替え” (2069 ページ)
“例 3: BY グループ内でオブザベーションの相対順序を維持する” (2074 ページ)
“例 4: 各 BY グループの最初のオブザベーションを維持する” (2077 ページ)
“例 7: NODUPKEY を使用してすべての 重複するオブザベーションを排除する” (2085 ページ)

構文

PROC SORT <*collating-sequence-option*> <*other options*>;

オプション引数の要約

DATA=*SAS-data-set*
: 入力データセットを指定します。

DATECOPY
: 作成日と変更日を変更せずに SAS データセットを並べ替えます。

FORCE
: 強制的に冗長な並べ替えを行います。

OVERWRITE
: 置換出力データセットが作成される前に、入力データセットを削除します。

PRESORTED
: データセットがすでに並べ替えられた可能性があるかどうかを指定します。

SORTSIZE=*memory-specification*
: 利用可能なメモリを指定します。

TAGSORT
: 一時的なディスク使用量を減らします。

Create output data sets

DUPOUT= *SAS-data-set*
重複オブザベーションの書き込み先となる出力データセットを指定します。

OUT= *SAS-data-set*
出力データセットを指定します。

UNIQUEOUT=*SAS-data-set*
除外されたオブザベーションの出力データセットを指定します。

Eliminate duplicate observations

NODUPKEY
BY 値が重複するオブザベーションを削除します。

Eliminate unique observations

NOUNIQUEKEY
一意の並べ替えキーがある出力データセットからオブザベーションを除外します。

Override SAS system option THREADS

NOTHREADS
スレッド化された並べ替えは実行されません。

THREADS
スレッド化された並べ替えの有効化を可能にするか行わないようにします。

Specify the collating sequence

ASCII
ASCII を指定します。

DANISH
デンマーク語を指定します。

EBCDIC
EBCDIC を指定します。

FINNISH
フィンランド語を指定します。

NATIONAL
カスタマイズされた順序を指定します。

NORWEGIAN
ノルウェー語を指定します。

REVERSE
文字変数の照合シーケンスを逆順にします。

SORTSEQ=*collating-sequence*
照合順序を指定します。

SWEDISH
スウェーデン語を指定します。

Specify the output order

EQUALS
BY グループ内の相対順序を指定します。

NOEQUALS
BY グループ内の相対順序を保持しません。

Collating-Sequence-Options

PROC SORT ステップでは、1 つの *collating-sequence-option* と複数の *other options* を指定できます。2 種類のオプションの順序は重要でなく、両方の種類を同じ PROC SORT ステップに含める必要はありません。

ASCII
ASCII 照合シーケンスを使用して文字変数を並べ替えます。EBCDIC がネイティブ照合シーケンスであるシステムで ASCII 順序に並べ替える場合のみ、このオプションが必要になります。

参照項目: “ASCII 順序” (2031 ページ)

DANISH
デンマーク語とノルウェー語の規則に従って文字を並べ替えます。

デンマーク語とノルウェー語の照合シーケンスを図 55.58 (2041 ページ)に示します。

EBCDIC
EBCDIC 照合シーケンスを使用して文字変数を並べ替えます。ASCII がネイティブ照合シーケンスであるシステムで EBCDIC 順序に並べ替える場合のみ、このオプションが必要になります。

参照項目: “EBCDIC 順序” (2032 ページ)

FINNISH
フィンランド語とスウェーデン語の規則に従って文字を並べ替えます。

フィンランド語とスウェーデン語の照合シーケンスを図 55.58 (2041 ページ)に示します。

NATIONAL
インストール時に設定した、各国の国別規則を反映した代替照合シーケンスを使用して文字変数を並べ替えます。このオプションを使用するには、カスタマイズされた各国並べ替え順序をサイトに定義しておく必要があります。サイトの SAS 導入担当者に連絡して、カスタマイズされた各国並べ替え順序が使用可能かどうかを確認します。

NORWEGIAN
デンマーク語とノルウェー語の規則に従って文字を並べ替えます。

ノルウェー語の照合シーケンスを図 55.58 (2041 ページ)に示します。

REVERSE
通常の照合シーケンスを逆にした照合シーケンスを使用して、文字変数を並べ替えます。

お使いの動作環境の通常の照合順序については、“EBCDIC 順序” (2032 ページ)および “ASCII 順序” (2031 ページ)を参照してください。

制限事項　指定できる照合シーケンスは、1 つのみです。

操作　REVERSE を BY ステートメントの DESCENDING オプションとともに使用すると、順序が通常の順序に戻ります。

参照項目: BY ステートメントの"DESCENDING" (2058 ページ)オプション。相違点は、DESCENDING オプションが文字変数と数値変数の両方に使用できる点です。

SWEDISH
フィンランド語とスウェーデン語の規則に従って文字を並べ替えます。

フィンランド語とスウェーデン語の照合シーケンスを図 55.58 (2041 ページ)に示します。

SORTSEQ=*collating-sequence*
collating-sequence は、次のうちのいずれかになります。

- collating-sequence-option (2041 ページ)
- translation_table (2041 ページ)
- encoding-value (2042 ページ)
- LINGUISTIC (2042 ページ)

詳細については、"照合順序" (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

図 55.1 英数字の各国照合シーケンス

```
Danish:     0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅabcdefghijklmnopqrstuvwxyzæøå
Finnish:    0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖabcdefghijklmnopqrstuvwxyzåäö
Italian:    0123456789AÀBCÇDEÉÈFGHIÌJKLMNOÒPQRSTUÙVWXYZaàbcçdeéèfghiìjklmnoòpqrstuùvwxyz
Norwegian:  0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅabcdefghijklmnopqrstuvwxyzæøå
Spanish:    0123456789AÁaáBbCcDdEÉeéFfGgHhIÍiíJjKkLlMmNnÑñOÓoóPpQqRrSsTtUÚuúÜ\OOfcVvWwXxYyZz
Swedish:    0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖabcdefghijklmnopqrstuvwxyzåäö
```

次に、照合シーケンスの種類について説明します。

collating-sequence-option
translation_table
PROC SORT ステートメントの *collating-sequence-option* (ASCII、DANISH、EBCDIC、FINNISH、NORWEGIAN、POLISH、REVERSE、SWEDISH)のいずれか 1 つを指定する、POLISH か、SAS 提供の変換テーブルかユーザー定義の変換テーブルを指定します。SAS 提供の変換テーブルは、ASCII、DANISH、EBCDIC、FINNISH、ITALIAN、NORWEGIAN、POLISH、REVERSE、SPANISH、SWEDISH です。

制限事項: SORTSEQ=オプションに 1 つの *collating-sequence-option* か、または 1 つの変換テーブルを指定できます。

操作: SORTSEQ=オプションを指定した場合、In-Database 処理行われません。

ヒント: SORTSEQ=照合順序オプションはかっこなしで指定され、引数は関連付けられません。

参照項目: 各 *collating-sequence-option* の詳細については、"Collating-Sequence-Options" (2040 ページ)を参照してください。

文字変数の並べ替え順序を確認するには、 “EBCDIC 順序” (2032 ページ)、“ASCII 順序” (2031 ページ)、および他のすべてについては 図 55.58 (2041 ページ)を参照してください。

例 proc sort data=mydata SORTSEQ=ASCII;

例 SORTSEQ=とともに PROC TRANTAB と PROC SORT を使用した例については、“並べ替えへのさまざまな変換テーブルの使用” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。

encoding-value
エンコーディング値を指定します。この結果は、指定したエンコーディングで表示される文字データのバイナリ照合と同じです。サポートされているエンコーディング値については、*SAS 各国語サポート(NLS): リファレンスガイド* を参照してください。

制限事項 SORTSEQ=オプションで指定したエンコーディングを認識する SAS System の一部またはプロシジャは、PROC SORT のみです。

ヒント エンコーディング値に英数字またはアンダースコア以外の文字が含まれる場合、その値を引用符で囲む必要があります。

参照項目: 指定できるエンコーディングのリスト(*SAS 各国語サポート(NLS): リファレンスガイド*)。

LINGUISTIC<*(collating-options)*>
言語照合を指定します。言語照合では、言語とロケールに関連付けられたルールに従い、文化的に配慮した方法で文字が並べ替えられます。ルールとデフォルトの *collating-sequence* オプションは、現在のロケール設定で指定された言語に基づきます。実装は、International Components for Unicode (ICU) ライブラリによって提供されています。その結果、Unicode 照合アルゴリズム(UCA)との互換性が高くなります。詳細については、“データセットの言語的な並べ替えと ICU” (2034 ページ)を参照してください。

注: 言語照合システムオプションの指定時、PROC SORT および PROC SQL にのみ影響します。

SORTSEQ=LINGUISTIC を指定するときに使用できるオプションを次に示します。これらのオプションで、言語照合シーケンスを変更します。

ALTERNATE_HANDLING=SHIFTED
スペース、句読点、記号などの変数文字の処理を制御します。このオプションを指定しない(デフォルト値の Non-Ignorable を使用する)場合、これらの変数文字間の違いは、通常の文字間の違いと同様に重要です。ALTERNATE_HANDLING オプションを指定する場合、これらの変数文字はあまり重要ではありません。

デフォルト NON_IGNORABLE

ヒント SHIFTED 値は、QUATERNARY に設定された STRENGTH=と組み合わせてよく使用されます。このような場合、文字列の比較時に、

スペース、句読点および記号が考慮されますが、それは文字列のその他の側面(基本文字、アクセント、および大文字と小文字の区別)がすべて一致する場合に限ります。

参照項目: “例 5: ALTERNATE_HANDLING=を使用した言語的な並べ替え” (2079 ページ) および“例 6: ALTERNATE_HANDLING=およびSTRENGTH=を使用した言語的な並べ替え” (2082 ページ).

CASE_FIRST=
大文字と小文字の順序を指定します。この引数は、TERTIARY、QUATERNARY または IDENTICAL レベルでのみ有効です。次の表に、CASE_FIRST 引数の値と情報を示します。

表 55.3 CASE_FIRST=の引数

値	説明
UPPER	大文字、小文字の順序で並べ替えます。
LOWER	小文字、大文字の順序で並べ替えます。

COLLATION=
文字の順序を指定します。次のテーブルに、指定可能な COLLATION=値を示します。

注: 照合値を選択しない場合、ユーザーのロケールのデフォルト照合が選択されます。

表 55.4 COLLATION=の値

値	説明
BIG5HAN	ラテン語の場合はピンイン順序を指定し、中国語、日本語、韓国語文字の場合は bug5 文字セット順序を指定します。
DIRECT	ヒンディー語の異形を指定します。
GB21312HAN	ラテン語の場合はピンイン順序を指定し、中国語、日本語、韓国語文字の場合は gb2312han 文字セット順序を指定します。
PHONEBOOK	文字の順序について電話帳スタイルを指定します。PHONEBOOK はドイツ語でのみ選択します。
PINYIN	文字ごとのピンインへの音訳に基づいて、中国語、日本語、韓国語文字の順序を指定します。この並べ替えは、簡体字中国語でよく使用されます。

値	説明
POSIX	Portable Operating System Interface です。このオプションでは、“C”ロケールの文字順序を指定します。
STROKE	非アルファベット書き込みスタイルの文字順序を指定します。STROKE は中国語、日本語、韓国語、またはベトナム語で選択します。この並べ替えは、繁体字中国語でよく使用されます。
TRADITIONAL	文字の順序に対して従来のスタイルを指定します。たとえば、スペイン語で TRADITIONAL を選択します。

LOCALE= ***locale_name***
POSIX 名の形式(ja_JP など)でロケール名を指定します。PROC SORT でサポートされているロケール値と POSIX 値のリストについては、*“ロケールからエンコーディングへのマッピング” (SAS 各国語サポート(NLS): リファレンスガイド)*を参照してください。

制限事項 次のロケールは PROC SORT でサポートされません。
- Afrikaans_SouthAfrica, af_ZA
- Cornish_UnitedKingdom, kw_GB
- ManxGaelic_UnitedKingdom, gv_GB

NUMERIC_COLLATION=
数を示す文字のかわりに、数値でテキスト内の整数値を並べ替えます。

表 55.5 NUMERIC_COLLATION の値

値	説明
ON	数値で番号を並べ替えます。たとえば、"8 Main St."は"45 Main St."より前になります。
OFF	文字の値で数を並べ替えます。たとえば、"45 Main St."は"8 Main St."より前になります。

デフォルト OFF

STRENGTH=
強度の値は、照合レベルに関連付けられます。5 つの照合レベル値があります。次の表に、5 つのレベルの情報を示します。強度のデフォルト値は、ロケールに関連付けられます。

表 55.6 STRENGTH=の値

値	照合の種類	説明
PRIMARY または 1	PRIMARY では、基本文字間の差異を指定します("a" < "b"など)。	この差異は最強です。たとえば、辞書は基本文字により異なるセクションに分割されます。
SECONDARY または 2	文字のアクセント記号が第 2 差異となります("as" < "às" < "at"など)。	文字列内に第 1 差異がある場合は、第 2 差異は無視されます。文字間のその他の差異も、言語に応じて第 2 差異と考慮される場合があります。
TERTIARY または 3	大文字と小文字の差異は、第 3 レベルで区別されます("ao" < "Ao" < "aò"など)。例については、“例 5: ALTERNATE_HANDLING=を使用した言語的な並べ替え” (2079 ページ)を参照してください。	文字列内に第 1 または第 2 差異がある場合は、第 3 差異は無視されます。別の例として、通常の仮名と小文字の仮名の違いがあります。
QUATERNARY または 4	レベル 1 から 3 までで句読点を無視する場合、追加のレベルを使用して句読点のある単語とない単語を区別できます(a-b" < "ab" < "aB"など)。例については、“例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語的な並べ替え” (2082 ページ)を参照してください。	句読点の無視が必須の場合や日本語テキストを処理する場合は、第 4 レベルを使用する必要があります。第 1、第 2 または第 3 差異がある場合は、この差異は無視されます。
IDENTICAL または 5	他のすべてのレベルが等しい場合、最終的な決定をするために同一レベルが使用されます。各文字列の正規化形式 D (NFD)の Unicode コードポイント値がこのレベルで比較され、レベル 1 から 4 で違いがないことが確認されます。	2 つの文字列でコードポイント値が異なることはほとんどないため、このレベルは慎重に使用する必要があります。たとえば、へ

値	照合の種類	説明
		ブライ語の詠唱マークのみがこのレベルで区別されます。

別名 LEVEL=

別名 UCA

制限事項 SORTSEQ=LINGUISTIC オプションは、PROC SORT SORTSEQ=オプションでのみ使用可能で、システムオプション SORTSEQ には使用できません。

操作 ICU バージョンは新しい SAS リリースで変更される可能性があります。2 つのリリースが別のバージョンの ICU を使用しているときに、一方の SAS リリースを使用してデータセットの言語的に並べ替える時に作成されるオブザベーションの順序は、もう一方のリリースで作成される順序と異なる場合があります。新しいリリースの SAS に移行するときは、PRESORTED オプションを指定した PROC SORT を使用して永久データセットの並べ替え順序を再確立することを検討してください。詳細については、“データセットの言語的な並べ替えと ICU” (2034 ページ)を参照してください。

ヒント CONTENTS プロシジャまたは CONTENTS ステートメントの出力に、言語的に並べ替えられたデータセットの ICU バージョン番号が示されます。

collating-options は、かっこで囲む必要があります。複数の照合オプションを指定できます。

言語照合によって並べ替えられたデータセットで BY 処理を実行する場合、データセットを適切に処理するために NOBYSORTED システムオプションの指定が必要になる場合があります。BY 処理は、照合順序処理とは異なる方法で実行されます。

参照項目: ICU ライセンス契約については、付録 3, “UNICODE ライセンス” (2525 ページ)を参照してください。

詳細については、“言語照合の指定” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照してください。詳細については、“データセットの言語的な並べ替えと ICU” (2034 ページ)を参照してください。

注意

ホストの並べ替えユーティリティを使用してデータを並べ替える際に、SORTSEQ=オプションで変換テーブルに基づく照合シーケンスを指定すると、文

字 BY 変数が破損する可能性があります。 詳細については、現在の作環境向けのドキュメントの PROC SORT を参照してください。

操作　SORTSEQ=オプションを指定した場合、In-Database 処理行われません。

ヒント　SORTSEQ= *collating-sequence* オプションはかっこなしで指定され、引数は関連付けられません。照合シーケンスの指定例は次のとおりです。
proc sort data=mydata SORTSEQ=ASCII;

その他のオプション

オプションには、*collating-sequence-option* とその他複数のオプションが含まれます。2 種類のオプションの順序は重要でなく、両方の種類を同じ PROC SORT ステップに含める必要はありません。

DATA=*SAS-data-set*
入力 SAS データセットを識別します。

制限事項　In-Database 処理を実行するためには、データセットが DBMS に存在するテーブルを参照する必要があります。

SAS データセットオプション DROP=、KEEP=、RENAME=は CAS ではサポートされていません。これらのオプションがデータセットで指定されている場合、CAS 操作は発生しません。

注　PROC SORT は、拡張属性を入力データセットから出力データセットにコピーして拡張属性をサポートします。

オプション NODUPKEY または NOUNIKEY が指定され、DATA=オプションが CAS テーブルを参照し、OUT=、DUPOUT=、または UNIOUT=オプションが CAS テーブルを参照する場合、PROC SORT は CAS サーバーで CAS deduplicate アクションを呼び出します。詳細については、deduplicate アクションを参照してください。

参照項目:　“入力データセット” (24 ページ)

SAS データセットオプション: リファレンス

DATECOPY
SAS データセットが作成された SAS 内部日時と並べ替え前に最後に変更を加えた日時を、並べ替えた結果のデータセットにコピーします。動作環境の日時は保持されません。

制限事項　DATECOPY は、結果のデータセットで V8 Engine または V9 Engine が使用される場合にのみ使用できます。

ヒント　ファイル作成日時は、PROC DATASETS の MODIFY ステートメントの DTC=オプションで変更できます。詳細については、“MODIFY ステートメント” (441 ページ)を参照してください。

DUPOUT= *SAS-data-set*
重複オブザベーションの書き込み先となる出力データセットを指定します。

操作 DUPOUT=オプションを指定した場合、In-Database 処理は行われません。

DUPOUT=オプションと UNIQUEOUT=オプションは両立しないため、同時には指定できません。

注 オプション NODUPKEY または NOUNIKEY が指定され、DATA=オプションが CAS テーブルを参照し、OUT=、DUPOUT=、または UNIOUT=オプションが CAS テーブルを参照する場合、PROC SORT は CAS サーバーで CAS deduplicate アクションを呼び出します。詳細については、deduplicate アクションを参照してください。

ヒント DUPOUT=オプションは、NODUPKEY オプションとのみ使用できます。DUPOUT=オプションを NOUNIQUEKEY オプションと組み合わせることはできません。

指定した DUPOUT=データセット名が INPUT データセット名と同じ場合、INPUT データセットの並べ替えも上書きも行われません。かわりに、SAS でエラーメッセージが生成されます。INPUT データセットを同名の DUPOUT=データセットで上書きするには、FORCE オプションを指定する必要があります。

参照項目: *SAS データセットオプション: リファレンス*

EQUALS

出力データセットのオブザベーションの順序を指定します。BY 変数値が同一のオブザベーションについて、EQUALS は、出力データセットで入力データセット内のオブザベーションの相対順序を維持します。NOEQUALS は、必ずしも出力データセットの相対順序を保持するとは限りません。

デフォルト EQUALS

操作 NODUPKEY を使用して出力データセットのオブザベーションを削除する場合、EQUALS または NOEQUALS の選択によって、削除されるオブザベーションが影響を受ける可能性があります。

EQUALS | NOEQUALS プロシジャオプションは、SORTEQUALS | NOSORTEQUALS システムオプションで設定されたデフォルトの並べ替え安定動作よりも優先されます。

EQUALS オプションは、スレッド化された並べ替えによってサポートされています。ただし、スレッド化された並べ替えで EQUALS オプションを使用すると、区分データセットが単一区分構成のように処理されるため、I/O パフォーマンスが低下する場合があります。

NOEQUALS オプションは、スレッド化された並べ替えでサポートされています。スレッド化された並べ替えによって返される BY グループ内のオブザベーションの順序は、実行間で一致しない場合があります。

ヒント NOEQUALS を使用すると、CPU 時間とメモリを節約できます。

参照項目: “NOEQUALS” (2051 ページ).

FORCE
OUT=オプションを指定しない場合に、インデックス付きデータセットの並べ替えと置き換えを行います。FORCE オプションを指定しない場合、PROC SORT はインデックス付きデータセットの並べ替えや置き換えは行いません。並べ替えによってデータセットのユーザー作成インデックスが破損するためです。
FORCE を指定すると、PROC SORT がデータセットの並べ替えと置き換えを行い、データセットのユーザー作成インデックスがすべて 破損します。一貫性制約によって作成された、または必要とされたインデックスは保持されます。

制限事項 古い互換性エンジンや、テープ形式エンジンなどの順次エンジンで作成されたデータセットに対して、FORCE オプションを指定して PROC SORT を使用する場合、OUT=オプションも指定する必要があります。

ヒント PROC SORT は、データセットを並べ替える前に、データの不要な再並べ替えを行わないように、ソートインジケーターをチェックします。デフォルトでは、PROC SORT は、並べ替え情報と要求された並べ替えが一致する場合、データセットを並べ替えません。FORCE オプションを使用すると、この動作を無効にできます。データセットオプション SORTEDBY=の並べ替え指定を確認できない場合は、FORCE の使用が必要になることがあります。SORTEDBY= の詳細については、*SAS データセットオプション: リファレンス*の SAS データセットオプションの章を参照してください。

NODUPKEY
BY 値が重複するオブザベーションのチェックと除外を行います。このオプションを指定すると、PROC SORT は各オブザベーションのすべての BY 値を出力データセットに先に書き込まれたオブザベーションの BY 値と比較します。完全に一致する値が見つかると、そのオブザベーションは出力データセットに書き込まれません。

並べ替えにより、BY 変数値が等しいオブザベーションがグループ化されて出力されます。通常、PROC SORT は、アセンブルする各 BY グループのすべてのオブザベーションを出力データセットに書き込みます。NODUPKEY オプションは、SORT プロシジャに、各 BY グループの最初のオブザベーションのみを出力データセットに書き込み、その BY グループに含まれる追加のオブザベーションを破棄するように指示します。

いくつかの要因により、PROC SORT が出力用にアセンブルした BY グループ内で最初にオブザベーションが発生するかどうかが決まります。これらの要因には、入力データセットからオブザベーションが読み取られる順序、使用されている並べ替えプログラム、および並べ替えアルゴリズムによって安定性が提供されるかどうかが含まれます。

SORT プロシジャの入力が V9 エンジンデータセットであり、並べ替えが SAS によって行われる場合、出力 BY グループ内のオブザベーションの順序は予測可能です。グループ内の観測の順序は、データセットが作成されたときにデータセットに書き込まれた順序と同じです。V9 エンジンは、データセットに書き込まれた順序でオブザベーションを維持するため、PROC SORT によって同じ順序で読み取られます。PROC SORT は、安定した並べ替えアルゴリズムを使用しているため、処理中、オブザベーションの順序を維持します。EQUALS オプションがデフォルトで設定されているため、安定並べ替えアルゴリズムが使用されます。し

たがって、特定の BY グループの出力データセットに書き込まれるように PROC SORT によって選択されたオブザベーションは、グループを定義する BY 変数値を持つデータセットの最初のオブザベーションです。

SORT プロシジャが、予測可能なオブザベーション順序または代替の並べ替えプログラム(ホストの並べ替えが並べ替えを実行する場合)を提供しないエンジンから入力を読み取る場合、削除されたオブザベーションと出力データセットに書き込まれたオブザベーションは十分に定義されていない可能性があります。たとえば、並列処理のためにクエリ結果を非決定的な順序で表示する DBMS から SAS にデータが読み込まれている場合、どのオブザベーションが保持され、どのオブザベーションが破棄されるかを予想することが困難な場合があります。

出力データセットの各オブザベーションが一意であることを確認するには、PROC SORT で NODUPKEY オプションを使用し、入力データセットの_ALL_変数で並べ替えることができます。データセット内の他のオブザベーションに同じ変数値の組み合わせがない場合、データセット内のオブザベーションは一意です。サンプルプログラム“例 7: NODUPKEY を使用してすべての 重複するオブザベーションを排除する” (2085 ページ)を参照してください。

注: NODUPKEY を使用しているときに出力データセットから 1 つ以上の BY 変数を削除すると、各オブザベーションに一意の BY 変数値のセットがあるという保証が無効になります(削除されます)。同様に、NODUPKEY を使用し、_ALL_変数で並べ替えて一意のオブザベーションを生成する場合、出力データセットから 1 つ以上の変数を削除すると、出力データセット内のオブザベーションが一意であることが保証されなくなります。

出力データセット内のオブザベーションが一意であることを確認する別の方法は、DISTINCT キーワードを指定して PROC SQL を使用することです。PROC SQL の簡単な例を次に示します:

```
PROC SQL;
 CREATE TABLE DL (keep division league) AS SELECT DISTINCT *
 FROM SASHELP.BASEBALL;
QUIT;
```

操作

V9 エンジンは一貫した順序を提供し、最初のオブザベーション(最初に書き込まれて保存されたオブザベーション)が通常、PROC SORT によって読み取られる最初のオブザベーションになります。並べ替えられたデータセットには、PROC SORT が読み取る各 BY グループの最初のオブザベーションのみが含まれます。

NODUPKEY を使用して BY 値の重複するオブザベーションを削除する際は、EQUALS または NOEQUALS の選択により、削除されるオブザベーションが影響を受ける可能性があります。出力データセットで整合性のある結果を得るためには、EQUALS オプション(デフォルト値)と NODUPKEY オプションを一緒に使用します。

NODUPKEY オプションが指定され、システムオプション SQLGENERATION=に DBMS が割り当てられ、システムオプションが SORTPGM=BEST の場合に、In-Database 並べ替えが行われます。

オプション NODUPKEY と NOUNIQUEKEY には互換性がありません。これらのオプションを一緒に指定すると、SAS ログにエラーが書き込まれます。

注 オプション NODUPKEY または NOUNIKEY が指定され、DATA=オプションが CAS テーブルを参照し、OUT=、DUPOUT=、または UNIOUT=オプションが CAS テーブルを参照する場合、PROC SORT は CAS サーバーで CAS deduplicate アクションを呼び出します。詳細については、deduplicate アクションを参照してください。

ヒント DUPOUT=オプションは NODUPKEY オプションと一緒に使用できます。ただし、これを NOUNIQUEKEY オプションと一緒に使用することはできません。

例 “例 4: 各 BY グループの最初のオブザベーションを維持する” (2077 ページ)

“例 7: NODUPKEY を使用してすべての 重複するオブザベーションを排除する” (2085 ページ)

NOEQUALS
“EQUALS” (2048 ページ)を参照してください。

NOTHREADS
“Base SAS のスレッド処理” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

NOUNIQUEKEY
一意の並べ替えキーがある出力データセットのオブザベーションのチェックと除外を行います。キーを含むオブザベーションが BY グループ内の唯一のオブザベーションである場合、その並べ替えキーは一意です。

注: NODUPKEY は、BY グループの 1 変数を出力データセットに書き込み、BY グループの他のすべてのオブザベーションを破棄しますが、NOUNIQUEKEY はそれとは異なり、BY グループの一貫性を維持します。BY グループが 2 つ以上のオブザベーションから構成されている場合に BY グループのすべてのオブザベーションが出力データセットに書き込まれるか、BY グループが単一のオブザベーションから構成されている場合に BY グループのすべてのオブザベーションが破棄されるかのいずれかになります。

別名 NOUNIKEY

NOUNIKEYS

NOUNIQUEKEYS

操作 オプション NODUPKEY と NOUNIQUEKEY は互換しません。NODUPKEY と NOUNIQUEKEY を同時に指定すると、SAS ログにエラーが書き込まれます。

注 オプション NODUPKEY または NOUNIKEY が指定され、DATA=オプションが CAS テーブルを参照し、OUT=、DUPOUT=、または UNIOUT=オプションが CAS テーブルを参照する場合、PROC SORT は CAS サーバーで CAS deduplicate アクションを呼び出します。詳細については、deduplicate アクションを参照してください。

ヒント UNIQUEOUT= オプションは NOUNIQUEKEY オプションと一緒に使用できます。UNIQUEOUT= オプションを NODUPKEY オプションと組み合わせることはできません。

参照項目: UNIQUEOUT= (除外されたオブザベーションを出力データセットに送る)

OUT= *SAS-data-set*
出力データセットを指定します。*SAS-data-set* が存在しない場合、PROC SORT が作成します。

注意
OUT=を指定せずに PROC SORT を使用する際は、注意が必要です。 OUT=オプションを指定しない場合、PROC SORT では、プロシジャがエラーなしで実行されると、元のデータセットが並べ替えたオブザベーションに置き換えられます。

デフォルト OUT=を指定しない場合、PROC SORT は元のデータセットを上書きします。

注 オプション NODUPKEY または NOUNIKEY が指定され、DATA=オプションが CAS テーブルを参照し、OUT=、DUPOUT=、または UNIOUT=オプションが CAS テーブルを参照する場合、PROC SORT は CAS サーバーで CAS deduplicate アクションを呼び出します。詳細については、deduplicate アクションを参照してください。

ヒント In-Database 並べ替えでは、出力データセットは、DBMS の入力テーブルを参照できません。

データセットオプションは OUT=と使用できます。

参照項目: *SAS データセットオプション: リファレンス*

例 “例 1: 複数の変数の値による並べ替え” (2069 ページ)

OVERWRITE
同じ名前の置換出力データセットにオブザベーションが作成される前に、入力データセットを削除できます。

注意
OVERWRITE オプションは、バックアップされているデータセットか、再構築が可能なデータセットにのみ使用してください。 入力データセットは削除されるため、出力データセットの書き込み中にエラーが発生した場合はデータが失われます。

制限事項 OVERWRITE および OUT=オプションを指定し、OUT=データセット名が INPUT データセット名と同じではない場合、SAS で INPUT データセットは上書きされません。

TAGSORT オプションも指定した場合、OVERWRITE オプションは無効になります。TAGSORT は出力データセットの作成時に入力データセットを再度読み込む必要があるため、入力データセットは上書きできません。

OVERWRITE オプションは、SAS の並べ替えおよび SAS のスレッド化された並べ替えによってのみサポートされています。このオプションは、ホストの並べ替えを使用する場合は無効になります。

ヒント OVERWRITE オプションを使用すると、必要なディスクスペースを 減らすことができます。

PRESORTED
並べ替えの前に、入力データセット内がチェックされ、オブザベーション順序が順番どおりかが判定されます。データセットがすでに BY ステートメントで指定したキー変数に従った順序であることがわかっている、またはそう思われる場合に、PRESORTED オプションを使用します。このオプションを指定すると、データセットの並べ替えのコストを回避できます。

操作 “FORCE” (2049 ページ)オプションが指定された場合、順序チェックは実行されません。

ヒント SORT オプションと BY ステートメントに含められたキー変数との組み合わせによって指定された並べ替え順序に従って、SAS 処理に対応した順序で、外部テキストファイルからデータをインポートするには、DATA ステップを使用します。次に、データが適切に並べ替えられているとわかっている、またはそう思われる場合に、PRESORTED オプションを指定します。

PRESORTED オプションを ACCESS Engine および DBMS データと使用することはお勧めしません。これらの外部データベースでは、クエリで ORDER BY 句を指定しない限り、オブザベーションが並べ替えられた順序で返される保証はありません。通常、外部データベースでは、物理的な順序という概念は使用されません。そのため、これらのデータベースでは、クエリを複数回実行した場合、オブザベーションが同一の順序で返される保証はありません。データの処理時に整合性のある反復可能な結果を得るためには、物理的な順序が重要になります。PROC SORT では、反復可能なデータ取得順序を指定しなければ、並べ替えの安定性を要求して“EQUALS” (2048 ページ)オプションを使用したとしても、PROC SORT を実行するたびに同一の順序でオブザベーションが返される保証はありません。反復可能なデータ取得順序を指定しなければ、PROC SORT による隣接する重複レコードの検出と除外も、PROC SORT を実行するたびに変わる可能性があります。

参照項目: システムオプション“SORTVALIDATE システムオプション” (*SAS システムオプション: リファレンス*)。

SORTSIZE=*memory-specification*
PROC SORT で利用可能な最大メモリ量を指定します。*memory-specification* の有効値は次のとおりです。

MAX
利用可能なメモリをすべて使用できるよう指定します。

n
メモリ量をバイト単位で指定します。この場合、*n* は実数です。

***n*K**
メモリ量をキロバイト単位で指定します。この場合、*n* は実数です。

***n*M**
メモリ量をメガバイト単位で指定します。*n* は実数です。

***n*G**
メモリ量をギガバイト単位で指定します。*n* は実数です。

PROC SORT ステートメントで SORTSIZE=オプションを指定すると、一時的に SAS システムオプションよりも優先されます。詳細については、"SORTSIZE= システムオプション" (*SAS システムオプション: リファレンス*)を参照してください。

動作環境の情報: システムの並べ替えユーティリティによっては 、このオプションの扱いが異なる場合もあります。ご使用の動作環境の SAS ドキュメントを参照してください。

別名　SIZE=

デフォルト　SAS システムオプション SORTSIZE=の値

ヒント　PROC SORT ステートメントで SORTSIZE=オプションを MAX または 0 に設定するか、もしくは SORTSIZE=オプションを設定しない場合、PROC SORT の利用可能な物理メモリは、SAS システムオプション REALMEMSIZE および MEMSIZE の設定に基づいて制限されます。

SAS システムオプション REALMEMSIZE および MEMSIZE については、動作環境の SAS ドキュメントを参照してください。

TAGSORT
BY 変数とオブザベーション番号のみを一時ファイルに保存します。この BY 変数とオブザベーション番号は *tags* と呼ばれます。PROC SORT は、並べ替え処理の完了時に、タグを使用して入力データセットから並べ替え順序でレコードを取得します。

注: 作成されるユーティリティファイルは、TAGSORT オプションを指定しなかった場合よりもずっと小さくなります。

制限事項　TAGSORT オプションと OVERWRITE オプションは両立しません。

操作　TAGSORT オプションは、スレッド化された並べ替えではサポートされていません。

ヒント　BY 変数の合計長がレコード長に比べて小さい場合、TAGSORT によって一時的なディスク使用量が大幅に削減されます。ただし、処理時間がずっと長くなる場合があります。

THREADS
スレッド化された並べ替えの有効化を可能にするか行わないようにします。

デフォルト
THREADS | NOTHREADS SAS システムオプションの値。デフォルトは、SORT プロシジャの THREADS | NOTHREADS オプションを使用して無効にできます。

制限事項
サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

SPD Engine による THREADS | NOTHREADS プロシジャオプションの追加時にエラーが発生した場合、PROC SORT は処理を停止して、SAS ログにメッセージを書き込みます。

操作
システムオプションが制限されない限り、PROC SORT THREADS | NOTHREADS オプションは SAS の THREADS | NOTHREADS オプションを無効にします(制約を参照)。詳細については、“THREADS システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

PROC SORT がスレッド処理が有効であると判断した場合、THREADS システムオプションが有効になります。SAS システムオプションの値が CPUCOUNT=1 の場合、スレッド処理は有効ではありません。ただし、システムオプションが NOTHREADS に設定されている場合、またはシステムオプションが THREADS でプロシジャオプションが NOTHREADS の場合は、PROC SORT THREADS オプションを指定してスレッド処理を強制できます。このオプションの組み合わせによって、スレッド処理は実行されず、システムオプションに基づくアクションが無効になります。スレッド化された並べ替えが有効で、NOEQUALS が指定されている場合、BY グループ内のオブザベーションが予想外の順序で返されることがあります。

スレッド化された SAS の並べ替えを使用している場合、UTILLOC=システムオプションがユーティリティファイルの配置に影響します。スレッドの有効な SAS アプリケーションでは、別々のスレッドからの並列アクセスが可能な一時ファイルを作成できます。詳細については、“UTILLOC= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

PROC SORT で使用されるユーティリティファイルのページサイズは、新しい STRIPESIZE=システムオプションの影響を受けます。詳細については、“STRIPESIZE= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

TAGSORT オプションは、スレッド化された並べ替えではサポートされていません。TAGSORT オプションを指定すると、スレッド処理は実行されません。

参照項目:
“NOTHREADS” (2051 ページ)。

“スレッド化された並べ替え” (2030 ページ)と“Base SAS のスレッド処理” (*SAS プログラマガイド: エッセンシャル*)。

UNIQUEOUT=*SAS-data-set*
NOUNIQUEKEY オプションで除外されたオブザベーションの出力データセットを指定します。

別名 UNIOUT=

操作 DUPOUT=オプションと=オプションは両立しないため、同時には指定できません。

注 オプション NODUPKEY または NOUNIKEY が指定され、DATA=オプションが CAS テーブルを参照し、OUT=、DUPOUT=、または UNIOUT=オプションが CAS テーブルを参照する場合、PROC SORT は CAS サーバーで CAS deduplicate アクションを呼び出します。詳細については、deduplicate アクションを参照してください。

ヒント UNIQUEOUT= オプションは NOUNIQUEKEY オプションと一緒に使用できます。UNIQUEOUT= オプションを NODUPKEY オプションと組み合わせることはできません。

参照項目: “NOUNIQUEKEY” (2051 ページ)

SAS データセットオプション: リファレンス

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

BY ステートメント

並べ替え変数を指定します。

例: “例 1: 複数の変数の値による並べ替え” (2069 ページ)
“例 2: 降順の並べ替え” (2072 ページ)
“例 4: 各 BY グループの最初のオブザベーションを維持する” (2077 ページ)

構文

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

必須引数

variable
PROC SORT がオブザベーションの並べ替えの基準にする変数を指定します。PROC SORT はデフォルトで、まずデータセットを最初の BY 変数の値を基準にして昇順で並べます。PROC SORT は、次に、最初の BY 変数の値が同じオブザベーションを、2 番目の BY 変数の値を基準にして昇順で並べます。指定したすべての BY 変数に対して、この並べ替えが続行されます。

オプション引数

DESCENDING
ステートメントの直後に続く変数の並べ替え順序を逆順にして、オブザベーションを降順に並べ替えるようにします。DESCENDING キーワードは、後に続く変数を変更します。

ヒント PROC SORT BY ステートメントでは、DESCENDING キーワードは後に続く変数を変更します。

PROC SORT THREADS | NOTHREADS オプションが指定されていない限り、THREADS SAS システムオプションがデフォルトになります。

例 “例 2: 降順の並べ替え” (2072 ページ)

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。“出力形式の取り消し” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、“PUT ステートメント: リスト” (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

KEY ステートメント

並べ替えキーと変数を指定します。KEY ステートメントは、BY ステートメントの代替です。KEY ステートメント構文では、将来 KEY 変数ごとに異なる照合オプションを指定する可能性が考慮されています。現在使用できるオプションは、ASCENDING と DESCENDING のみです。

制限事項: BY ステートメントは、KEY ステートメントと使用できません。

ヒント: KEY ステートメントは、複数指定できます。

構文

KEY *variable(s)* </ *option*> ;

必須引数

variable(s)
PROC SORT でオブザベーションの並べ替えの基準にする変数を指定します。変数は複数指定できます。その変数は、それぞれスペースで区切る必要があります。変数の範囲も指定できます。たとえば、次のコードは、複数の変数と値の範囲の指定方法を示しています。

```
data sortKeys;
  input x1 x2 x3 x4 ;
cards;
  7 8 9 8
  0 0 0 0
  1 2 3 4 ;
run;
proc sort data=sortKeys out=sortedOutput;
  key x1 x2-x4;
run;
```

複数の KEY ステートメントも指定できます。すべての並べ替えキーのうち、最初に発生した並べ替えキーが第 1 並べ替えキーとみなされます。指定されているすべての KEY ステートメントとその変数に対して、並べ替えが続行されます。たとえば、次のコードは、複数の KEY ステートメントの指定方法を示しています。

```
proc sort data=sortKeys out=sortedOutput;
  key x2;
  key x3;
run;
```

次のコード例では、BY ステートメントを使用して、前述の例と同じ種類の並べ替えを実行しています。

```
proc sort data=sortKeys out=sortedOutput;
  by x2 x3;
```

```
run;
```

オプション引数

ASCENDING
この後に続く 1 つまたは複数の変数を昇順で並べ替えます。オブザベーションは、昇順に並べ替えられます。ASCENDING キーワードは、KEY ステートメントで前に置かれる変数をすべて変更します。

別名 ASC

デフォルト ASCENDING がデフォルト並べ替え順序です。

ヒント PROC SORT KEY ステートメントでは、ASCENDING オプションは後に続く変数をすべて変更します。このオプションは、*/*の後に置く必要があります。次の例では、入力データセットの x1 変数が昇順で並べ替えられます。

```
proc sort data=sortVar out=sortedOutput;
  key x1 / ascending;
run;
```

DESCENDING
ステートメントで後に続く変数の並べ替え順序を逆順にして、オブザベーションを降順に並べ替えるようにします。DESCENDING キーワードは、KEY ステートメントで前に置かれる変数をすべて変更します。

別名 DESC

デフォルト ASCENDING (ASC)がデフォルト並べ替え順序です

ヒント PROC SORT KEY ステートメントでは、DESCENDING オプションは後に続く変数を変更します。このオプションは、*/*の後に置く必要があります。次の例では、入力データセットの x1 変数と x2 変数が降順で並べ替えられます。

```
proc sort data=sortVar out=sortedOutput;
  key x1 x2 / descending;
run;
```

次の例では、BY ステートメントを使用して、前述の例と同じ種類の並べ替えを実行しています。

```
proc sort data=sortVar out=sortedOutput;
  by descending x1 descending x2 ;
run;
```

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=' '>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項 ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL

ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: *"文字定数と文字変数" (SAS プログラマガイド: エッセンシャル)*.

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

- COMPARE プロシジャと、PROC DATASETS の APPEND ステートメントは、複数の入力データセットを受け入れます。詳細については、特定のプロシジャのドキュメントを参照してください。

- 出力データセットをサブセット化するには、WHERE=データセットオプションを使用します。

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

使用法: SORT プロシジャ

PROC SORT: In-Database 処理

In-Database 処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。セキュリティの強化は、機密データを DBMS から抽出する必要がないため可能です。より迅速な処理は、データが比較的遅いネットワーク接続から転送される代わりに高速二次記憶装置を使用して DBMS でローカルで操作されるため、DBMS に自由に使えるより多くの処理リソースがあるため、DBMS は高度に並列かつスケーラブルな方法で実行するクエリを最適化できるため可能です。

DATA=入力データセットがデータベース管理システム(DBMS)のテーブルまたはビューとして保存される場合、PROC SORT プロシジャは In-Database 処理を使用して、データを並べ替えられます。In-Database 処理は、より迅速な処理と、データベースと SAS ソフトウェア間のデータ転送の減少という利点を提供できます。

PROC SORT の In-Database 処理では、次のデータベース管理システムがサポートされています。

- Amazon Redshift
- DB2
- Google BigQuery

  注: PROC SORT の In-database 処理には、FLOAT64 型ではない BY 変数が必要です。

- Greenplum
- Hadoop
- Impala
- Microsoft SQL Server
- MySQL
- Netezza
- Oracle
- PostgreSQL
- SAP HANA
- Snowflake

- Spark
- Teradata
- Vertica

PROC SORT は、SQL 明示パススルーを使用して In-Database 処理を実行します。パススルー機能は、SAS/ACCESS を使用して DBMS に接続し、ステートメントを実行するために DMBS に直接送信します。この機能により、DMBS の SQL 構文を使用できます。詳細については、*SAS/ACCESS for Relational Databases: リファレンス* の "Pass-Through Facility for Relational Dabases"を参照してください。

In-Database 処理は、プロシジャとシステムオプションの組み合わせが正しく設定されている場合に PROC SORT によって使用されます。システムオプション SORTPGM=BEST を指定し、システムオプション SQLGENERATION=を In-Database 処理を行うように設定して、PROC SORT NODUPKEY オプションを指定した場合、PROC SORT はデータを並べ替える DBMS SQL クエリを生成します。並べ替え結果は、DBMS 内に新しいテーブルとして残すことも、SAS に返すこともできます。生成された SQL クエリを表示するには、SASTRACE=オプションを設定します。

また、SAS システムオプション SORTPGM=を使用すると、SQLGENERATION オプションを設定せずに、PROC SORT に対し DBMS、SAS、HOST のいずれかを使用して並べ替えを実行するように命令できます。SORTPGM=BEST を指定すると、DBMS、SAS、HOST のいずれかが並べ替えを実行します。PROC SORT により生成されるオブザベーション順序は、DBMS または SAS のどちらが並べ替えを実行するかによって異なります。

DBMS が並べ替えを実行する場合、DMBS 並べ替えプログラムの構成と特性が、結果のデータ順序に影響します。データ順序に影響する可能性がある DBMS の構成設定と特性には、文字照合、NULL 値の順序および並べ替えの安定性があります。ほとんどのデータベース管理システムでは、並べ替えの安定性が保証されていません。並べ替えは、SORTEQUALS/NOSORTEQUALS システムオプションおよび EQUALS/NOEQUALS プロシジャオプションの状態に関係なく、DMBS によって実行される場合があります。

SAS システムオプション SORTPGM=を SAS に設定した場合、並べ替え前のデータが DBMS から SAS に配信され、SAS で並べ替えが実行されます。ただし、DBMS からのオブザベーションの配信順序は保証されていません。そのため、DBMS データに対して安定した並べ替えを実行できるとしても、出力 BY グループ内のオブザベーションの順序が PROC SORT の実行のたびに同じになるという保証はありません。BY グループ内のオブザベーション順序の整合性を実現するには、まず DBMS データを含む SAS データセットを作成し、EQUALS または SORTEQUALS オプションを使用して安定した並べ替えを実行します。

In-Database 処理は、次の状況によって影響を受けます。

- PROC SORT オプションの SORTSEQ=または DUPOUT=を指定すると、In-Database 処理は実行されません。
- In-Database 処理では、OUT=プロシジャオプションの指定が必要です。このとき、出力データセットは DBMS の入力テーブルを参照できません。
- LIBNAME オプションとデータセットオプションも、In-Database 処理の発生の有無や、生成されるクエリの種類に影響します。これらのすべてのオプションのリストについては、*SAS/ACCESS for Relational Databases: リファレンス*の"In-Database Procedures"を参照してください。SAS で OPTIONS MSGLEVEL=I を設定すると、In-Database 処理を行わないオプション、In-Database 処理に影響するオプションを確認できます。

PROC SORT の SAS Cloud Analytic Services 処理

PROC SORT は CAS で重複データ検出および操作オプション(NODUPKEY、NOUNIKEY、DUPOUT=、UNIOUT=)を実行する機能をサポートしています。この機能は、SAS Viya プラットフォームのユーザーのコードの移行を容易にするために提供されています。de-duplicate アクションは、CAS 内のデータセット内の重複エントリを排除するために使用されます。

入力データセットが CAS ビューまたはインメモリテーブルを参照する場合、SORT プロシジャは、CAS de-duplicate アクションを使用して、PROCSORT の NODUPKEY または NOUNIKEY オプションと同等の機能を実行します。オプション NODUPKEY または NOUNIKEY が PROC SORT ステートメントで指定され、PROC SORT への入力が CAS によって読み取られ、PROC SORT からのすべての出力(オプションの DUPOUT=および=UNIOUT データセットを含む)が CAS テーブルに書き込まれると、PROC SORT は CAS サーバーでの CAS de-duplicate アクションを呼び出します。これらの条件下で、アクションが呼び出され、CAS サーバーでデータの出入りがない状態で CAS で作業が実行されます。

CAS アクションについては、deduplicate アクションを参照してください。

CAS で実行されるプロシジャの概念的な情報については、4 章, “Base プロシジャの CAS 処理” (75 ページ)を参照してください。

整合性制約: SORT プロシジャ

入力データセットを並べ替え、並べ替えたデータセットと置き換えると、参照一貫性制約と一般一貫性制約の両方、および必要になる可能性があるすべてのインデックスが保持されます。新しいデータセットを作成する並べ替えでは、一貫性制約もインデックスも保持されません。OUT=オプションを指定する場合としない場合の暗黙置換、明示置換および置換なしの詳細については、“出力データセット” (2068 ページ)を参照してください。一貫性制約の詳細については、“Integrity Constraints” (*SAS V9 LIBNAME Engine: Reference*)を参照してください。

結果: SORT プロシジャ

プロシジャの出力

PROC SORT は、出力データセットのみを作成します。出力データセットを確認するには、PROC PRINT、PROC REPORT、または SAS で使用可能な多くの印刷方法のいずれかを使用します。

出力データセット

OUT=オプションを指定しない場合、PROC SORT では、プロシジャがエラーなしで実行されると、元のデータセットが並べ替えたオブザベーションに置き換えられます。新しいデータセット名を使用して OUT=オプションを指定すると、PROC SORT は並べ替えられたオブザベーションを含む新しいデータセットを作成します。

表 55.7　データセット置換オプション

タスク	オプション
入力データセットの暗黙置換	proc sort data=names;
入力データセットの明示置換	proc sort data=names out=names;
入力データセットの置換なし	proc sort data=names out=namesbyid;

3 つの置換オプション(暗黙置換、明示置換および置換なし)すべてに関して、出力ライブラリで、少なくとも元のデータセットをコピーするのに十分なスペースが必要になります。

また、圧縮されているデータセットも並べ替えできます。圧縮データセットを入力データセットとして指定し、OUT=オプションを省略した場合、入力データセットは並べ替えられ、圧縮された状態は維持されます。OUT=データセットを指定すると、COMPRESS=データセットオプションで圧縮方法を選択した場合に限り、結果のデータセットが圧縮されます。詳細については、“COMPRESS= データセットオプション” (*SAS データセットオプション: リファレンス*)を参照してください。

また、PROC SORT はメモリ内の圧縮されていないオブザベーションを操作し、並べ替えを完了するためのメモリが不足している場合は、ユーティリティファイルに解凍データを保存します。これらの理由から圧縮されたデータセットの分類は集中的

でかつ予想よりより多くのストレージが必要です。ＴＡＧＳＯＲＴの使用を考慮すると、圧縮データセットの分類のときのオプションです。

注: SAS システムオプション NOREPLACE が有効な場合は、元の永久データセットを並べ替えられたバージョンに置き換えることはできません。OUT=オプションを使用するか、OPTIONS ステートメントで SAS システムオプション REPLACE を指定する必要があります。SAS システムオプション NOREPLACE は、一時 SAS データセットには影響しません。

例: SORT プロシジャ

例 1: 複数の変数の値による並べ替え

要素:

PROC SORT ステートメントオプション
OUT=

BY ステートメント

PROC PRINT

詳細

この例では、次を行います。

- 2 つの変数を基準にしてオブザベーションを並べ替えます。
- 並べ替えられたオブザベーションの出力データセットを作成します。
- 結果を印刷します。

プログラム

```
data account;
  input Company $ 1-22 Debt 25-30 AccountNumber 33-36
      Town $ 39-51;
  datalines;
Paul's Pizza           83.00  1019  Apex
World Wide Electronics  119.95  1122  Garner
Strickland Industries   657.22  1675  Morrisville
Ice Cream Delight       299.98  2310  Holly Springs
```

```
Watson Tabor Travel     37.95  3131  Apex
Boyd & Sons Accounting  312.49  4762  Garner
Bob's Beds             119.95  4998  Morrisville
Tina's Pet Shop         37.95  5108  Apex
Elway Piano and Organ    65.79  5217  Garner
Tim's Burger Stand      119.95  6335  Holly Springs
Peter's Auto Parts       65.79  7288  Apex
Deluxe Hardware          467.12  8941  Garner
Pauline's Antiques      302.05  9112  Morrisville
Apex Catering           37.95  9923  Apex
;

proc sort data=account out=bytown;

  by town company;
run;

proc print data=bytown;

  var company town debt accountnumber;

  title  'Customers with Past-Due Accounts';
  title2 'Listed Alphabetically within Town';
run;
```

プログラムの説明

入力データセット ACCOUNT を作成します。 ACCOUNT には、借金をしている各企業の名前、そのアカウントの借金額、アカウント番号、企業が所在する町が含まれます。

```
data account;
  input Company $ 1-22 Debt 25-30 AccountNumber 33-36
       Town $ 39-51;
  datalines;
Paul's Pizza           83.00  1019  Apex
World Wide Electronics  119.95  1122  Garner
Strickland Industries   657.22  1675  Morrisville
Ice Cream Delight       299.98  2310  Holly Springs
Watson Tabor Travel     37.95  3131  Apex
Boyd & Sons Accounting  312.49  4762  Garner
Bob's Beds             119.95  4998  Morrisville
Tina's Pet Shop         37.95  5108  Apex
Elway Piano and Organ    65.79  5217  Garner
Tim's Burger Stand      119.95  6335  Holly Springs
Peter's Auto Parts       65.79  7288  Apex
Deluxe Hardware          467.12  8941  Garner
Pauline's Antiques      302.05  9112  Morrisville
Apex Catering           37.95  9923  Apex
;
```

出力データセット BYTOWN を作成します。 OUT=は並べ替えられたオブザベーションに対し、新しいデータセットを作成します。

```
proc sort data=account out=bytown;
```

2 つの変数を基準にして並べ替えます。 BY ステートメントは、オブザベーションが町のアルファベット順、次に会社のアルファベット順で並べ替えられるよう指定します。

```
  by town company;
run;
```

出力データセット BYTOWN を印刷します。 PROC PRINT は、データセット BYTOWN を印刷します。

```
proc print data=bytown;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
  var company town debt accountnumber;
```

タイトルを指定します。

```
  title  'Customers with Past-Due Accounts';
  title2 'Listed Alphabetically within Town';
run;
```

出力: HTML

アウトプット 55.3　複数の変数の値による並べ替え

Customers with Past-Due Accounts
Listed Alphabetically within Town

Obs	Company	Town	Debt	AccountNumber
1	Apex Catering	Apex	37.95	9923
2	Paul's Pizza	Apex	83.00	1019
3	Peter's Auto Parts	Apex	65.79	7288
4	Tina's Pet Shop	Apex	37.95	5108
5	Watson Tabor Travel	Apex	37.95	3131
6	Boyd & Sons Accounting	Garner	312.49	4762
7	Deluxe Hardware	Garner	467.12	8941
8	Elway Piano and Organ	Garner	65.79	5217
9	World Wide Electronics	Garner	119.95	1122
10	Ice Cream Delight	Holly Springs	299.98	2310
11	Tim's Burger Stand	Holly Springs	119.95	6335
12	Bob's Beds	Morrisville	119.95	4998
13	Pauline's Antiques	Morrisville	302.05	9112
14	Strickland Industries	Morrisville	657.22	1675

例 2: 降順の並べ替え

要素: この例の BY ステートメントオプション
DESCENDING
PROC PRINT

データセット: Account

詳細

この例では、次を行います。

- 3 つの変数の値を基準にしてオブザベーションを並べ替えます。
- 変数の 1 つは、降順で並べ替えます。
- 結果を印刷します。

プログラム

```
proc sort data=account out=sorted;
  by town descending debt accountnumber;
run;
proc print data=sorted;
  var company town debt accountnumber;
  title  'Customers with Past-Due Accounts';
  title2 'Listed by Town, Amount, Account Number';
run;
```

プログラムの説明

出力データセット SORTED を作成します。 OUT=は並べ替えられたオブザベーションに対し、新しいデータセットを作成します。

```
proc sort data=account out=sorted;
```

3 つの変数を基準に、そのうち 1 つを降順にして並べ替えます。 BY ステートメントは、オブザベーションが町のアルファベット順、借金額の降順、アカウント番号の昇順の順番で並べ替えられるよう指定します。

```
  by town descending debt accountnumber;
run;
```

出力データセット SORTED を印刷します。 PROC PRINT は、データセット SORTED を印刷します。

```
proc print data=sorted;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
var company town debt accountnumber;
```

タイトルを指定します。

```
title  'Customers with Past-Due Accounts';
title2 'Listed by Town, Amount, Account Number';
run;
```

出力: HTML

最後に AccountNumber 順に並べ替えると、$37.95 の債務がある Apex の企業がアカウント番号順に並べられます。

アウトプット 55.4　降順の並べ替え

Customers with Past-Due Accounts
Listed by Town, Amount, Account Number

Obs	Company	Town	Debt	AccountNumber
1	Paul's Pizza	Apex	83.00	1019
2	Peter's Auto Parts	Apex	65.79	7288
3	Watson Tabor Travel	Apex	37.95	3131
4	Tina's Pet Shop	Apex	37.95	5108
5	Apex Catering	Apex	37.95	9923
6	Deluxe Hardware	Garner	467.12	8941
7	Boyd & Sons Accounting	Garner	312.49	4762
8	World Wide Electronics	Garner	119.95	1122
9	Elway Piano and Organ	Garner	65.79	5217
10	Ice Cream Delight	Holly Springs	299.98	2310
11	Tim's Burger Stand	Holly Springs	119.95	6335
12	Strickland Industries	Morrisville	657.22	1675
13	Pauline's Antiques	Morrisville	302.05	9112
14	Bob's Beds	Morrisville	119.95	4998

例 3: BY グループ内でオブザベーションの相対順序を維持する

要素: PROC SORT ステートメントオプション
EQUALS | NOEQUALS
PROC PRINT

詳細

この例では、次を行います。

- 最初の変数の値を基準にしてオブザベーションを並べ替えます。
- EQUALS オプションを使用して、相対順序を維持します。
- NOEQUALS オプションを使用して相対順序を維持しないようにします。

プログラム

```
data insurance;
  input YearsWorked 1 InsuranceID 3-5;
  datalines;
5 421
5 336
1 209
1 564
3 711
3 343
4 212
4 616
;
proc sort data=insurance out=byyears1 equals;
  by yearsworked;
run;
proc print data=byyears1;
  var yearsworked insuranceid;
  title 'Sort with EQUALS';
run;
proc sort data=insurance out=byyears2 noequals;
  by yearsworked;
run;
```

```
proc print data=byyears2;
  var yearsworked insuranceid;
  title 'Sort with NOEQUALS';
run;
```

プログラムの説明

入力データセット INSURANCE を作成します。 INSURANCE には、全被保険従業員の勤続年数とその保険証 ID が含まれます。

```
data insurance;
  input YearsWorked 1 InsuranceID 3-5;
  datalines;
5 421
5 336
1 209
1 564
3 711
3 343
4 212
4 616
;
```

EQUALS オプションを使用して、出力データセット BYYEARS1 を作成します。 OUT=は並べ替えられたオブザベーションに対し、新しいデータセットを作成します。EQUALS オプションは、相対的なオブザベーション順序を維持します。

```
proc sort data=insurance out=byyears1 equals;
```

最初の変数順に並べ替えます。 BY ステートメントは、オブザベーションが勤続年数の数値順で並べ替えられるよう指定します。

```
  by yearsworked;
run;
```

出力データセット BYYEARS1 を印刷します。 PROC PRINT は、データセット BYYEARS1 を印刷します。

```
proc print data=byyears1;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
  var yearsworked insuranceid;
```

タイトルを指定します。

```
  title 'Sort with EQUALS';
run;
```

出力データセット BYYEARS2 を作成します。 OUT=は並べ替えられたオブザベーションに対し、新しいデータセットを作成します。NOEQUALS オプションは、相対的なオブザベーション順序を維持しません。

```
proc sort data=insurance out=byyears2 noequals;
```

最初の変数順に並べ替えます。 BY ステートメントは、オブザベーションが勤続年数の数値順で並べ替えられるよう指定します。

```
    by yearsworked;
run;
```

出力データセット BYYEARS2 を印刷します。 PROC PRINT は、データセット BYYEARS2 を印刷します。

```
proc print data=byyears2;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
    var yearsworked insuranceid;
```

タイトルを指定します。

```
    title 'Sort with NOEQUALS';
run;
```

出力: HTML

EQUALS オプションを指定した並べ替えと、NOEQUALS オプションを指定した並べ替えとを比較すると、YearsWorked=3 のオブザベーションの並べ替え順序が異なることに注意してください。

アウトプット 55.5　EQUALS オプションでの並べ替え

Sort with EQUALS

Obs	YearsWorked	InsuranceID
1	1	209
2	1	564
3	3	711
4	3	343
5	4	212
6	4	616
7	5	421
8	5	336

アウトプット 55.6　NOEQUALS オプションでの並べ替え

Sort with NOEQUALS

Obs	YearsWorked	InsuranceID
1	1	209
2	1	564
3	3	343
4	3	711
5	4	212
6	4	616
7	5	421
8	5	336

例 4: 各 BY グループの最初のオブザベーションを維持する

要素: PROC SORT ステートメントオプション
NODUPKEY
BY ステートメント
PROC PRINT

データセット: Account

注: EQUALS オプションは、各 BY グループの最初のオブザベーションが NODUPKEY オプションによって維持されるように、有効である必要があります。EQUALS オプションはデフォルトです。NOEQUALS オプションが指定されている場合、NODUPKEY オプションによって BY グループごとに 1 つのオブザベーションが維持されますが、それが必ずしも最初のオブザベーションとは限りません。

詳細

この例では、V9 エンジンが使用されています。V9 エンジンは一貫した順序を提供し、最初のオブザベーション(最初に書き込まれて保存されたオブザベーション)が通常、PROC SORT によって読み取られる最初のオブザベーションになります。並べ替えられたデータセットには、各 BY グループの最初のオブザベーションのみが含まれます。

並べ替えにより、BY 変数値が等しいオブザベーションがグループ化されて出力されます。通常、PROC SORT は、アセンブルする各 BY グループのすべてのオブザベー

ションを出力データセットに書き込みます。NODUPKEY オプションは、SORT プロシジャに、各 BY グループの最初のオブザベーションのみを出力データセットに書き込み、その BY グループに含まれる追加のオブザベーションを破棄するように指示します。この例の結果レポートには、オブザベーションが企業の所在する町ごとに 1 つずつ含まれます。

プログラム

```
proc sort data=account out=towns nodupkey;
  by town;
run;
proc print data=towns;
  var town company debt accountnumber;
  title 'Towns of Customers with Past-Due Accounts';
run;
```

プログラムの説明

出力データセット TOWNS を作成します。 ただし、ここに含まれるのは各 BY グループの最初のオブザベーションのみです。

```
proc sort data=account out=towns nodupkey;
```

1 つの変数を基準にして並べ替えます。 BY ステートメントは、オブザベーションが町順で並べ替えられるよう指定します。

```
  by town;
run;
```

出力データセット TOWNS を印刷します。 PROC PRINT は、データセット TOWNS を印刷します。

```
proc print data=towns;
```

印刷する変数を指定します。 VAR ステートメントは印刷する変数と、出力でのそれらの変数の列の順序を指定します。

```
  var town company debt accountnumber;
```

タイトルを指定します。

```
  title 'Towns of Customers with Past-Due Accounts';
run;
```

出力: HTML

出力データセットには、入力データセットの町ごとに 1 つずつ、全部で 4 つのオブザベーションのみが含まれます。

アウトプット 55.7　各 BY グループの最初のオブザベーションを維持する

Towns of Customers with Past-Due Accounts

Obs	Town	Company	Debt	AccountNumber
1	Apex	Paul's Pizza	83.00	1019
2	Garner	World Wide Electronics	119.95	1122
3	Holly Springs	Ice Cream Delight	299.98	2310
4	Morrisville	Strickland Industries	657.22	1675

例 5: ALTERNATE_HANDLING=を使用した言語的な並べ替え

要素:

PROC SORT ステートメントオプション
sortseq=linguistic
ALTERNATE_HANDLING=SHIFTED
STRENGTH=3

BY ステートメント

VAR ステートメント

PROC PRINT

PROC CONTENTS

注: 文字列の言語的な並べ替えの強さの指定に関する詳細については、“例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語的な並べ替え” (2082 ページ)を参照してください。

詳細

この例では、PROC SORT は各 BY グループの最初のオブザベーションのみを含む出力データセットを作成します。"a-b" を"ab"および"aB"の近くになるように並べ替えるため、ALTERNATE_HANDLING=SHIFTED を指定しました。つまり、"a-b"がそのハイフンのせいで"ab"および"aB"と離れた場所に表示されないようにします。

注: この例では、このロケールのデフォルトの STRENGTH は 3 です。

次の例では、"a-b"と"ab"は同じものと見なされている点に注意してください。比較の最初の 3 つのレベル(アルファベット、付加記号、および大文字小文字)以外で並べ替える場合は、比較の 4 つ目のレベルを使用し、STRENGTH=4 と指定できます。“例

6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語的な並べ替え” (2082 ページ)は、文字列をさらに区別する方法を示しています。

PROC CONTENTS は、出力に Sort Information セクションを表示します。ICU バージョンも並べ替え情報に表示されます。

プログラム

```
data a;
  length x $ 10;
  x='a-b'; output;
  x='ab'; output;
  x='a-b'; output;
  x='aB'; output;
run;

proc sort data=a sortseq=linguistic( ALTERNATE_HANDLING=SHIFTED );
  by x;
run;

title1 "Linguistic Collation with ALTERNATE_HANDLING=SHIFTED";
proc print data=a;
run;

title1 "Linguistic Collation with ALTERNATE_HANDLING=SHIFTED and BY Processing";
proc print data=a;
  var x;
  by x;
run;

proc contents data=a;
run;
```

プログラムの説明

データセットを作成します。

```
data a;
  length x $ 10;
  x='a-b'; output;
  x='ab'; output;
  x='a-b'; output;
  x='aB'; output;
run;
```

言語的な並べ替えを使用してデータセットを並べ替えます。 言語的な並べ替えと ALTERNATE_HANDLING=SHIFTED オプションを使用してデータセットを並べ替えます。このロケールのデフォルトの STRENGTH は 3 です。また、BY ステートメントも使用してオブザベーションを x で並べ替えます。

```
proc sort data=a sortseq=linguistic( ALTERNATE_HANDLING=SHIFTED );
  by x;
run;
```

データセット A を印刷します。 TITLE1 ステートメントで、PRINT プロシジャに出力に使用するタイトルを伝えます。PROC PRINT で、データセット A が印刷されます。

```
title1 "Linguistic Collation with ALTERNATE_HANDLING=SHIFTED";
proc print data=a;
run;
```

By 処理を使用してデータセット A を印刷します。 TITLE1 ステートメントで、PRINT プロシジャに出力に使用するタイトルを伝えます。PROC PRINT では、By 処理を使用してデータセット A が印刷されます。

```
title1 "Linguistic Collation with ALTERNATE_HANDLING=SHIFTED and BY Processing";
proc print data=a;
  var x;
  by x;
run;
```

言語的な並べ替えが使用されている場合は、Sort Information を出力します。 PROC SORT が言語照合で使用される場合、PROC CONTENTS 出力には Sort Information セクションが含まれます。この並べ替え情報には、使用されている ICU バージョンも含まれます。

```
proc contents data=a;
run;
```

出力: HTML

1 つ目の PROC PRINT では、"a-b"と"ab"の順序が適切に定義されていないことが示されます。2 つ目の PROC PRINT では、BY 処理を使用してこれらの値が等価と見なされることが示されます。“例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語的な並べ替え” (2082 ページ)は、文字列をさらに区別する方法を示しています。

アウトプット 55.8　ALTERNATE_HANDLING オプションを使用した言語的な並べ替え

Linguistic Collation with ALTERNATE_HANDLING=SHIFTED

Obs	x
1	a-b
2	ab
3	a-b
4	aB

Linguistic Collation with ALTERNATE_HANDLING=SHIFTED and BY Processing

x=a-b

Obs	x
1	a-b
2	ab
3	a-b

x=aB

Obs	x
4	aB

PROC CONTENTS は、言語的な並べ替えを使用した場合に並べ替え情報を出力します。使用されている ICU バージョンに関する情報も提供されます。

Sort Information	
Sortedby	x
Validated	YES
Character Set	ASCII
Collating Sequence	LINGUISTIC
Locale	en_US
Strength	3
Alternate Handling	SHIFTED
ICU Version Number	63

例 6: ALTERNATE_HANDLING=および STRENGTH=を使用した言語的な並べ替え

要素:

PROC SORT ステートメントオプション
sortseq=linguistic
ALTERNATE_HANDLING=SHIFTED
STRENGTH=4

BY ステートメント

VAR ステートメント

PROC PRINT

詳細

この例では、PROC SORT は各 BY グループの最初のオブザベーションのみを含む出力データセットを作成します。この例では、"a-b"をハイフンに関係なく"ab"および"aB"に近づくように並べ替えるため、ALTERNATE_HANDLING=SHIFTED が指定されています。

次の例では、"a-b"と"ab"は同じものと見なされている点に注意してください。ただし、これらをさらに区別して、2 つの個別の BY グループに表示する場合は、文字列をさらに並べ替える必要があります。比較の最初の 3 つのレベル(アルファベット、付加記号、および大文字小文字)以外で並べ替える場合は、比較の 4 つ目のレベル、STRENGTH=4 を指定できます。

プログラム

```
data a;
  length x $ 10;
  x='a-b'; output;
  x='ab'; output;
  x='a-b'; output;
  x='aB'; output;
run;

proc sort data=a sortseq=linguistic( ALTERNATE_HANDLING=SHIFTED STRENGTH=4);
  by x;
run;

title1 "Linguistic Collation with STRENGTH=4";
proc print data=a;
run;

Title1 "Linguistic Collation with STRENGTH=4 and BY Processing";
proc print data=a;
  var x;
  by x;
run;
```

プログラムの説明

データセットを作成します。

```
data a;
  length x $ 10;
  x='a-b'; output;
  x='ab'; output;
  x='a-b'; output;
  x='aB'; output;
run;
```

言語的な並べ替えを使用してデータセットを並べ替えます。 言語的な並べ替えと ALTERNATE_HANDLING=SHIFTED オプションを使用してデータセットを並べ替えます。このロケールのデフォルトの STRENGTH は 4 です。BY ステートメントでは、オブザベーションが x 順に並べ替えられるよう指定します。

```
proc sort data=a sortseq=linguistic( ALTERNATE_HANDLING=SHIFTED STRENGTH=4);
  by x;
run;
```

出力データセット A を印刷します。 TITLE1 ステートメントで、PRINT プロシジャに出力に使用するタイトルを伝えます。PROC PRINT で、データセット A が印刷されます。

```
title1 "Linguistic Collation with STRENGTH=4";
proc print data=a;
run;
```

By 処理を使用して出力データセット A を印刷します。 TITLE ステートメントで、PRINT プロシジャにこの出力に使用するタイトルを伝えます。PROC PRINT では、By 処理を使用してデータセット A が印刷されます。

```
Title1 "Linguistic Collation with STRENGTH=4 and BY Processing";
proc print data=a;
  var x;
  by x;
run;
```

出力: HTML

1 つ目の PROC PRINT では、"a-b"と"ab"の順序が適切に定義されていないことが示されます。STRENGTH=4 を設定してこの 2 つを区別します。2 番目の PROC PRINT では、BY 処理を使用して優先順位とその区別の方法を示します。

アウトプット 55.9　ALTERNATE_HANDLING および STRENGTH オプションを使用した言語的な並べ替え

Linguistic Collation with STRENGTH=4

Obs	x
1	a-b
2	a-b
3	ab
4	aB

Linguistic Collation with STRENGTH=4 and BY Processing

x=a-b

Obs	x
1	a-b
2	a-b

x=ab

Obs	x
3	ab

x=aB

Obs	x
4	aB

例 7: NODUPKEY を使用してすべての 重複するオブザベーションを排除する

要素:

PROC SORT ステートメントオプション
- NODUPKEY
- OUT=

BY ステートメント

PROC PRINT

KEEP=データセットオプション

詳細

この例では、NODUPKEY を使用した PROC SORT は、重複するオブザベーションを持たない出力データセットを作成します。これらのオブザベーションはそれぞれ固有のものです。変数値の特定のセットの出力データセットには、1 つのオブザベーションしかありません。BY _ALL_変数は、保持されている変数(KEEP=オプション)、DIVISION、および LEAGUE で並べ替えを行います。

プログラム

```
proc sort data=sashelp.baseball(keep=division league)out=DL NODUPKEY;
  by _ALL_;
run;
```

```
proc print data=DL;
title 'Baseball Leagues and Divisions';
run;
```

プログラムの説明

入力データセット(KEEP=)から分割変数とリーグ変数のみを処理し、DL 出力データセットを作成して、重複するすべてのエントリを削除します。

```
proc sort data=sashelp.baseball(keep=division league)out=DL NODUPKEY;
```

すべての変数による並べ替えを行います。

```
  by _ALL_;
run;
```

出力データセット DL を印刷します。

```
proc print data=DL;
title 'Baseball Leagues and Divisions';
run;
```

出力: HTML

出力データセットには、野球のディビジョンとリーグごとに 1 つずつ、全部で 4 つのオブザベーションのみが含まれます。ディビジョンとリーグの重複エントリは保持されません。

アウトプット 55.10　データセットから重複オブザベーションを削除する

Baseball Leagues and Divisions

Obs	League	Division
1	American	East
2	American	West
3	National	East
4	National	West

56

SQL プロシジャ

概要

SQL プロシジャは、Base SAS における構造化照会言語の実装です。PROC SQL は、Base SAS ソフトウェアに含まれており、任意の SAS データセット(テーブル)と共に使用できます。多くの場合、PROC SQL は、他の SAS プロシジャまたは DATA ステップの代替として使用できます。グローバルステートメント、データセットオプション、関数、入力形式、出力形式などの SAS 言語要素を、他の SAS プロシジャと同様に PROC SQL で使用できます。PROC SQL によって次のタスクを実行できます。

- レポートの生成
- 要約統計量の生成
- テーブルまたはビューからのデータ検索
- テーブルまたはビューのデータの結合
- テーブル、ビューおよびインデックスの作成
- PROC SQL テーブルのデータ値の更新
- データベース管理システム(DBMS)テーブルのデータの更新および検索
- 列の追加、変更または削除による PROC SQL テーブルの変更

詳細については、*SAS SQL プロシジャ ユーザーガイド* を参照してください。

例: SQL プロシジャの使用

次の例では、Cars データセットから Type="Sports"および MSRP が 100,000 を超えるすべての行を選択します。

```
proc sql;
  select * from sashelp.cars
    where Type='Sports' and MSRP>100000;
quit;
```

アウトプット 56.1　SQL プロシジャ結果

Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	Engine Size (L)	Cylinders	Horsepower	MPG (City)	MPG (Highway)	Weight (LBS)	Wheelbase (IN)	Length (IN)
Mercedes-Benz	SL55 AMG 2dr	Sports	Europe	Rear	$121,770	$113,388	5.5	8	493	14	21	4235	101	179
Mercedes-Benz	SL600 convertible 2dr	Sports	Europe	Rear	$126,670	$117,854	5.5	12	493	13	19	4429	101	179
Porsche	911 GT2 2dr	Sports	Europe	Rear	$192,465	$173,560	3.6	6	477	17	24	3131	93	175

57

SQOOP プロシジャ

概要: SQOOP プロシジャ

SQOOP プロシジャの機能

Hadoop とリレーショナルデータベース管理システム(RDBMS)との間でのデータ転送に、Apache Sqoop (*scoop* と発音します)を使用できます。SQOOP プロシジャを使用すると、SAS セッションから Apache Sqoop にアクセスしてデータベースと HDFS 間でデータを転送できます。これにより、SAS アプリケーション内から Hadoop クラスターに Sqoop コマンドをサブミットできます。

Sqoop コマンドは、Hadoop 向けの Apache Oozie Workflow Scheduler を使ってクラスターに渡されます。PROC SQOOP で Sqoop タスク用の Oozie ワークフローを定義すると、そのワークフローは RESTful API を使用して Oozie サーバーにサブミットされます。

PROC SQOOP は Apache Sqoop コマンド行インターフェイス(CLI)と同様に機能します。同じ構文を使用して、SAS / ACCESS Interface to Hadoop のライセンスを取

得したユーザーは、データベースと HDFS 間でデータを転送できます。ユーザーは、SQOOP プロシジャの COMMAND ステートメントで Sqoop CLI コマンドをサブミットできます。このプロシジャは、ジョブが正常に完了したかどうか、および Sqoop タスクが失敗した場合に Hadoop クラスターからの詳細を確認できる場所についてフィードバックを提供します。

Hadoop ディストリビューションによっては、Sqoop コマンドのさまざまなバージョンがサポートされています。特定の Sqoop コマンド構文については、ディストリビューションのドキュメントを参照してください。

Apache Sqoop の詳細については、http://sqoop.apache.org および http://sqoop.apache.org/docs/1.4.5/index.html にあるオンラインドキュメントを参照してください。

Sqoop に関する考慮事項と使用法については、*Apache Sqoop Cookbook* を参照してください。

Oozie の詳細については、http://oozie.apache.org を参照してください。

構文: SQOOP プロシジャ

要件: この手順を使用するには、Hadoop への SAS/ACCESS インターフェイスへの個別のライセンスが必要です。

Sqoop コマンドではなく、PROC SQOOP ステートメントの DBUSER および DBPWD オプションとともに個別に--username または--password オプションを指定します。

サポート: Linux のみの Kerberos

注: デフォルトでは、SQOOP プロシジャは hive-site.xml ファイルを使用して Zookeeper サービス検出を検索します。Zookeeper サービス検出が利用できない場合、SQOOP プロシジャで、HIVE_SERVER=オプションを使用してサーバー名とポートを設定するか、HIVE_URI=オプションを使用して JDBC URI を設定できます。

HIVE_SERVER=オプションと HIVE_URI=オプションのサポートは、SAS Viya 3.5 に追加されました

参照項目: SERVER=LIBNAME オプション、URI=LIBNAME オプション

例: SQL クエリを使用した Teradata から HDFS へのインポート

PROC SQOOP *required-arguments* <*additional-sqoop-options*>;

ステートメント	タスク	例
“PROC SQOOP ステートメント”	データ転送のための Apache Sqoop へのアクセスを可能にします。	Ex. 1

PROC SQOOP ステートメント

Apache Sqoop へアクセスし、データベースと HDFS の間でデータ転送を可能にするオプションを使います。

構文

PROC SQOOP
COMMAND='*command-to-sqoop*'
DBPWD='*database-password*'
DBUSER='*database-user-name*'
<DELETEWF>
HADOOPUSER='*hadoop-user-name*'
HADOOPPWD='*hadoop-password*'
<HIVE_SERVER='*server-name-and-port*'>
<HIVE_URI='*JDBC-URI*'>
<JOBTRACKER='*job-tracker-URL*'>
<NAMENODE='*name-node-URL*'>
OOZIEURL='*oozie-URL*'
<PASSWORDFILE='*password-file*'>
<WFHDFSPATH='*Oozie-workflow-path*'>;

run;

必須引数

COMMAND='*command-to-sqoop*'
Apache Sqoop コマンドを指定します。次のようにコマンドを指定する必要があります。

SQOOP-command --option ... --option

制限事項 sqoop 起動コマンドは使用しないでください。

注 SQOOP プロシジャを使用して Sqoop コマンドをサブミットする場合、$CONDITIONS でエスケープ文字のスラッシュ(\)を使用する必要はありません。

ヒント Apache Sqoop コマンドの詳細については、 http://sqoop.apache.org にあるオンラインドキュメントを参照してください。

DBPWD='*database-password*'
DBUSER オプションに関連付けられているデータベースパスワードを指定します。このオプションは PASSWORDFILE オプションと相互排他的であるため、どちらか一方のオプションのみ指定する必要があります。

DBUSER='*database-user-name*'
インポートまたはエクスポートに使用するデータベースユーザー名を指定します。

HADOOPPWD='*hadoop-password*'
HADOOPUSER=オプションに関連付けられている Hadoop パスワードを指定します。

制限事項 Kerberos 向けに有効になっているクラスターに接続するときは、このオプションを指定しないでください。

HADOOPUSER='*hadoop-user-name*'
インポートまたはエクスポートに使用する Hadoop ユーザー名を指定します。

制限事項 Kerberos 向けに有効になっているクラスターに接続するときは、このオプションを指定しないでください。

OOZIEURL='*oozie-URL*'
Oozie サーバーへの URL を指定します。

オプション引数

DELETEWF
WFHDFSPATH 内の場所で指定された Oozie ワークフローファイルが存在する場合に、そのファイルを削除することを指定します。SQOOP プロシジャで、その場所に新しいワークフローファイルが作成されます。

HIVE_SERVER='*server-name-and-port*'
Hive サービスを実行する Hive サーバー名と、指定された Hive サービスへの接続に使用するポート番号を指定します。

制限事項 一回に 1 つのオプションのみ指定できます: HIVE_SERVER=または HIVE_URI=。

要件 サーバー名にスペースまたは英数字以外の文字が含まれる場合、引用符で囲む必要があります。

hive-site.xml ファイルに Zookeeper サービスが含まれていない場合、Kerberos 化された HDP 3.1 クラスターに接続するときは、このオプションが必要です。

HIVE_URI=jdbc:hive2://*HiveServerHost*:*HiveServerPort*</Schema:*Property1=Value*;
<... *PropertyN=ValueN*;>>
JDBC URI を指定します。

制限事項 一回に 1 つのオプションのみ指定できます: HIVE_SERVER=または HIVE_URI=。

要件 hive-site.xml ファイルに Zookeeper サービスが含まれていない場合、Kerberos 化された HDP 3.1 クラスターに接続するときは、このオプションが必要です。

例 jdbc:hive2://MyHiveServer:10000

JOBTRACKER='*job-tracker-URL*'
JobTracker または ResourceManager サービスへの URL を指定します。

デフォルト SAS_HADOOP_CONFIG_PATH は、このオプションの値を指定しない場合、このオプションの値を決定します。

NAMENODE='*name-node-URL*'
NameNode サービスへの URL を指定します。

デフォルト SAS_HADOOP_CONFIG_PATH は、このオプションの値を指定しない場合、このオプションの値を決定します。

PASSWORDFILE='*password-file*'
インポートまたはエクスポート用のデータベースパスワードを含む、HDFS 内のファイルの名前を指定します。このパスワードファイルはこのプロシジャを実行する前に存在している必要があります。また、常に安全に保管するよう注意してください。

WFHDFSPATH='*Oozie-workflow-path-and-filename*'
以下に示すように、Oozie ワークフローをアップロードする場所のパスとファイル名を指定します。

```
/user/myID/mydir/myfile.xml
```

デフォルト 値を指定しない場合、生成されたパスが使用されます。

ヒント 必須ではありませんが、hdfs://server 構文を使用できます。

使用法: SQOOP プロシジャ

一般的な使用方法

Oracle JDBC Connector の要件として、Sqoop の--table オプションに使用する値は大文字で指定する必要があります。テーブルでの大文字と小文字の区別の詳細については、特定の DBMS のドキュメントを参照してください。

接続文字列には、インポートするデータに適した文字セットオプションを含める必要があります。詳細については、コネクタのドキュメントを参照してください。

ワークフローの使用

ワークフローは、必要なときにのみ作成されます。一部の SQOOP ジョブは、ワークフローファイルを生成しない Oozie プロキシサブミットを使用できます。Oozie 4.1 以降を実行していて、Hive テーブルを出力先として使用していない場合は、プロキシサブミットが選択されます。PROC SQOOP がプロキシサブミットを使用する場合、SAS ログにはメモが含まれます。

SQOOP プロシジャの使用要件

SQOOP プロシジャの使用を開始する前に必要な要件は次のとおりです。

必要な内容	PROC SQOOP オプション	確認する場所
データベースコネクタ	COMMAND: Sqoop 構文には、データベースに固有の--connection オプションが必要です。	Hadoop ディストリビューションのドキュメントを参照してください。
データベースユーザー ID	DBUSER	データベース管理者にお問い合わせください。
データベースパスワード	DBPWD	
データベースパスワードを含む HDFS ファイル	PASSWORDFILE	データベース管理者と Hadoop クラスター管理者にお問い合わせください。
Hadoop ユーザー ID	HADOOPUSER	Hadoop クラスター管理者にお問い合わせください。
Hadoop パスワード	HADOOPPWD	
Oozie URL	OOZIEURL	
名前ノード	NAMENODE	
Job Tracker (MR1)または Resource Manager (MR2)	JOBTRACKER	
Oozie ワークフロー出力パス	WFHDFSPATH	Hadoop クラスター管理者にお問い合わせください。一般的に、Oozie ワークフローは

必要な内容	PROC SQOOP オプション	確認する場所
		HDFS 内のユーザーディレクトリに書き込まれます。
Sqoop コマンド	COMMN	Hadoop ディストリビューションの Sqoop ドキュメントを参照してください。

例: SQL クエリを使用した Teradata から HDFS へのインポート

この例では、既存のワークフローをこのタスクの新しいワークフローに置き換えるために、DELETEWF が含まれています。

注: 環境に適したデフォルトの NAMENODE および JOBTRACKER ポート値については、特定のディストリビューションの構成を確認するか、Hadoop ドキュメントを参照してください。

```
proc sqoop dbuser='mydbusr1' dbpwd='mydbpwd1'
  hadoopuser='sashdpusr1' hadooppwd='sashdppwd1'
  oozieurl='http://myoozie-04:55000/oozie'
  namenode='hdfs://myoozie-04.unx.srvr.com:8020'
  jobtracker='myoozie-04.unx.srvr.com:8032'
  wfhdfspath='hdfs://myoozie-04.unx.srvr.com:8020/user/mydbusr1/myworkflow.xml'
  deletewf
  command='import
    --connection-manager com.cloudera.connector.teradata.TeradataManager
    --connect jdbc:teradata://myconnecti/Database=sqoop
    --query "SELECT * FROM sales where ($CONDITIONS and I < 25)" -m 1
    --split-by i --delete-target-dir
    --target-dir /user/mydbusr1/sales2';
run;
```

58

STANDARD プロシジャ

概要: STANDARD プロシジャ

STANDARD プロシジャの動作について

STANDARD プロシジャは、SAS データセットの変数を指定の平均と標準偏差に標準化し、標準化された値を含む新しい SAS データセットを作成します。

データの標準化

次の出力に、標準化された学生の試験のスコアが出力データセットに含まれる単純な標準化を示します。出力を生成するステートメントは次のとおりです。

```
proc standard data=score mean=75 std=5
        out=stndtest;
run;

proc print data=stndtest;
run;
```

アウトプット 58.1　PROC STANDARD を使用して標準化されたテストのスコア

```
                        The SAS System                          1

                Obs    Student       Test1

                  1    Capalleti    80.5388
                  2    Dubose       64.3918
                  3    Engles       80.9143
                  4    Grant        68.8980
                  5    Krupski      75.2816
                  6    Lundsford    79.7877
                  7    McBane       73.4041
                  8    Mullen       78.6612
                  9    Nguyen       74.9061
                 10    Patel        71.9020
                 11    Si           73.4041
                 12    Tanaka       77.9102
```

次の出力に、BY グループ処理を使用した、より複雑な例を示します。PROC STANDARD は平均寿命データを平均 0、標準偏差 1 に標準化して、2 つの BY グループに対し別々に Z スコアを計算します。データは、16 か国の 1950 年と 1993 年の出生時平均寿命です。stable または rapid に分類される各国の出生率が、2 つの BY グループを形成しています。分析を生成するステートメントは、次も行います。

- 標準化する変数ごとに統計量を印刷します。
- 欠損値を指定の平均と置き換えます。
- 指定の平均と標準偏差を使用して標準化された値を計算します。
- データセットを標準化された値とともに印刷します。

この出力を作成するプログラムの説明については、“例 2: BY グループの標準化と欠損値の置換” (2122 ページ)を参照してください。

アウトプット 58.2　PROC STANDARD を使用した BY グループごとの Z スコア

```
          Life Expectancies by Birth Rate           2

-------------------- PopulationRate=Stable --------------------

              The STANDARD Procedure

                        Standard
   Name          Mean      Deviation          N
   Label

   Life50     67.400000     1.854724          5
   1950 life expectancy
   Life93     74.500000     4.888763          6
   1993 life expectancy


-------------------- PopulationRate=Rapid --------------------

                        Standard
   Name          Mean      Deviation          N
   Label

   Life50     42.000000     5.033223          8
   1950 life expectancy
   Life93     59.100000     8.225300         10
   1993 life expectancy
```

```
     Standardized Life Expectancies at Birth          3
          by a Country's Birth Rate

Population
 Rate      Country          Life50      Life93

 Stable    France          -0.21567     0.51138
 Stable    Germany          0.32350     0.10228
 Stable    Japan           -1.83316     0.92048
 Stable    Russia           0.00000    -1.94323
 Stable    United Kingdom   0.86266     0.30683
 Stable    United States    0.86266     0.10228
 Rapid     Bangladesh       0.00000    -0.74161
 Rapid     Brazil           1.78812     0.96045
 Rapid     China           -0.19868     1.32518
 Rapid     Egypt            0.00000     0.10942
 Rapid     Ethiopia        -1.78812    -1.59265
 Rapid     India           -0.59604    -0.01216
 Rapid     Indonesia       -0.79472    -0.01216
 Rapid     Mozambique       0.00000    -1.47107
 Rapid     Philippines      1.19208     0.59572
 Rapid     Turkey           0.39736     0.83888
```

構文: STANDARD プロシジャ

ヒント: ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。詳細については、“複数のプロシジャで同じ機能を提供するステートメント” (23 ページ)を参照してください。

PROC STANDARD<*options*>;
  ATTRIB *variable-list(s) attribute-list(s)*;
  BY<DESCENDING>*variable-1*
    <... <DESCENDING> *variable-n*>
    <NOTSORTED>;
  FREQ *variable*;
  FORMAT *variable-1*<...*variable-n*> <*format*> <DEFAULT=*default-format*>;
  LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;
  VAR *variable(s)*;
  WEIGHT *variable*;
  WHERE *where-expression-1*
    <*logical-operator where-expression-n*>;

ステートメント	タスク	例
“PROC STANDARD ステートメント”	変数を指定の平均と標準偏差に標準化します。	Ex. 1, Ex. 2

ステートメント	タスク	例
“BY ステートメント”	BY グループごとに個別の標準化された値を計算します。	Ex. 2
“BY ステートメント”	値が各オブザベーションの度数を表す変数を識別します。	
“VAR ステートメント”	標準化する変数を選択して、印刷出力に表示される順序を決定します。	Ex. 1
“WEIGHT ステートメント”	値が統計量計算で各オブザベーションを重み付ける変数を識別します。	

PROC STANDARD ステートメント

SAS データセットの変数を指定の平均と標準偏差に標準化し、標準化された値を含む新しい SAS データセットを作成します。

例: “例 2: BY グループの標準化と欠損値の置換” (2122 ページ)
“例 1: 指定された平均と標準偏差への標準化” (2119 ページ)

構文

PROC STANDARD <*options*>;

オプション引数の要約

DATA=*SAS-data-set*
入力データセットを指定します。

EXCLNPWGT
非正の重みが付いたオブザベーションを除外します。

MEAN=*mean-value*
平均値を指定します。

OUT=*SAS-data-set*
出力データセットを指定します。

REPLACE
欠損値を変数の平均または MEAN=値と置き換えます。

STD=*std-value*
標準偏差値を指定します。

VARDEF=*divisor*
分散および標準偏差の計算に使用する除数を指定します。*divisor* に可能な値と、関連する除数の選択肢を次に示します。

Control printed output

NOPRINT
: すべての印刷出力を抑制します。

PRINT
: 標準化する変数ごとに統計量を印刷します。

Preserve values

PRESERVERAWBYVALUES
: 未加工の BY 値を保持します。

引数なし

MEAN=、REPLACE、STD=を指定しない場合、出力データセットは入力データセットの同一のコピーです。

オプション引数

DATA=*SAS-data-set*
: 入力 SAS データセットを指定します。

制限事項 PROC STANDARD は、別のユーザーが同時にデータセットを更新している場合に同時アクセス権をサポートするエンジンと使用できません。

参照項目: “入力データセット” (24 ページ)

EXCLNPWGT
: 非正(ゼロまたは負)の重みが付いたオブザベーションを除外します。プロシジャは平均と標準偏差の計算にオブザベーションを使用しませんが、オブザベーションは標準化されます。デフォルトで、プロシジャは負の重みの付いたオブザベーションをゼロの重みが付いたオブザベーションのように扱い、それらをオブザベーションの合計数にカウントします。

別名 EXCLNPWGTS

MEAN=*mean-value*
: 変数を *mean-value* の平均に標準化します。

別名 M=

デフォルト 入力値の平均

例 “例 1: 指定された平均と標準偏差への標準化” (2119 ページ)

NOPRINT
: プロシジャ出力の印刷を抑制します。NOPRINT はデフォルト値です。

OUT=*SAS-data-set*
: 出力データセットを指定します。*SAS-data-set* が存在しない場合、PROC STANDARD が作成します。OUT=を省略すると、データセットの名前は DATA*n* となります。*n* は、名前を一意のものにする最小整数です。

デフォルト Data*n*

例 “例 1: 指定された平均と標準偏差への標準化” (2119 ページ)

PRESERVERAWBYVALUES
未加工の BY 値を保持します。 BY 変数が出力データセットに伝搬する場合、すべての BY 変数の未加工値を保持します。

PRINT
標準化する変数ごとに元の度数、平均、標準偏差を印刷します。

例 “例 2: BY グループの標準化と欠損値の置換” (2122 ページ)

REPLACE
欠損値を変数の平均と置き換えます。

操作 MEAN=を使用すると、PROC STANDARD は欠損値を指定の平均と置き換えます。

例 “例 2: BY グループの標準化と欠損値の置換” (2122 ページ)

STD=*std-value*
変数を標準偏差 *std-value* に標準化します。

別名 S=

デフォルト 入力値の標準偏差

例 “例 1: 指定された平均と標準偏差への標準化” (2119 ページ)

VARDEF=*divisor*
分散および標準偏差の計算に使用する除数を指定します。*divisor* に可能な値と、関連する除数の選択肢を次に示します。

DF
自由度。除数の式は *n* −1 です。

N
オブザベーション数。除数の式は *n* です。

WDF
重みの合計 - 1。除数の式は$(\Sigma_i w_i) - 1$です。

WEIGHT
重みの合計。除数の式は $\Sigma_i w_i$ です。

別名 WGT

プロシジャは分散を$CSS/divisor$として計算します。CSSは修正平方和で、$\Sigma (x_i - \bar{x})^2$と等しくなります。分析変数を重み付けする場合、CSSは$\Sigma\ w_i(x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト DF

ヒント WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。*i* 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$、およびw_iは *i*

番目のオブザベーションの重みです。これにより、オブザベーションの分散の推定が単位重みとともに生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな *n* に対し)$\sigma^2/\bar{w}$の推定になります。$\bar{w}$は平均重みです。これにより、オブザベーションの分散の漸近的推定が平均重みとともに生成されます。

参照項目: “キーワードと式” (2410 ページ)

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

"ATTRIB ステートメント" (*SAS DATA ステップステートメント: リファレンス*)。

BY ステートメント

出力を BY グループ順に並べ替えます。

制限事項: BY ステートメントではグループ変数を指定しないでください。BY 変数をグループ変数と同じにすることはできません。

構文

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

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

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

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

詳細

BY グループ処理

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

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

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

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

- BY ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。
- PROC SORT の詳細については、55 章, "SORT プロシジャ," (2027 ページ)を参照してください。
- CAS での BY グループ処理の詳細については、"グループ化と順序付け" (*SAS Cloud Analytic Services: DATA ステッププログラミング*)を参照してください。
- インデックスの作成の詳細については、"INDEX CREATE ステートメント" (435 ページ)を参照してください。

BY 変数値のフォーマティング

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

1 プロシジャで、値を BY 変数の内部(未フォーマット)値順に並べ替えるどうかが決定されます。

2 プロシジャで、BY 変数に出力形式を適用されたかどうかが決定されます。BY 変数が数値で、ユーザー適用の出力形式がない場合は、BY グループ処理のために BEST12.出力形式が適用されます。

3 プロシジャは、BY 変数または変数の内部値と未フォーマット値が両方とも変更されるまで、現在の BY グループにオブザベーションを追加し続けます。

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

2 つのデータセットで長さが異なる BY 変数

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

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

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

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

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

- A

- A、B
- A、B、C
- A、B、C、D

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

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブランクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデータセットの'ABCD'とセカンダリデータセットの'ABCD 'は一致します。セカンダリデータセットに'ABCDEF'が含まれている場合は、一致しません。

BY ステートメントをサポートする Base SAS プロシジャ

COMPARE	SORT (必須)
CORR	STANDARD
FREQ	SUMMARY
MEANS	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE
REPORT (非ウィンドウ環境のみ)	ODSTEXT
ODSLIST	ODSTABLE

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

関連項目

"BY ステートメント" (*SAS DATA ステップステートメント: リファレンス*)

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。*“出力形式の取り消し” (SAS DATA ステップステートメント: リファレンス)*を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、*“PUT ステートメント: リスト” (SAS DATA ステップステートメント: リファレンス)*を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

FREQ ステートメント

オブザベーションを、入力データセットに複数回表示されたように処理します。WEIGHT ステートメントと FREQ ステートメントは、両方のステートメントをサポートする任意のプロシジャの同一ステップ内で使用できます。

構文

FREQ *variable*;

必須引数

variable
値がオブザベーションの度数を表す数値変数を指定します。FREQ ステートメントを使用する場合、プロシジャは各オブザベーションが

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

度数変数の合計は、オブザベーションの合計数を表します。

例

例 1

FREQ ステートメントをサポートするプロシジャ

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

PROC MEANS で度数変数を使用する

この例のデータは、毎時記録された船の針路と速度(海里/時間)を表します。度数変数 Hours は船が同じ針路と速度を維持した時間数を表します。次の各 PROC MEANS ステップでは、平均進路と平均速度が計算されます。さまざまな結果によって、Hours を度数変数として使用する効果を示します。1 つ目の PROC MEANS ステップでは、度数変数が使用されていません。度数変数がなければ、各オブザベーションの度数は 1 で、オブザベーションの合計数は 8 です。2 つ目の PROC MEANS ステップでは、度数変数として Hours が使用されます。Hours を度数変数として使用

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

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

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

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

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

例 2

出力

Average Course and Speed

The MEANS Procedure

Variable	N	Mean
Course	8	48.50
Speed	8	7.88

Average Course and Speed

The MEANS Procedure

Variable	N	Mean
Course	141	49.28
Speed	141	8.06

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=' '>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項
ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注
ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL

ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: “文字定数と文字変数” (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

VAR ステートメント

標準化する変数と、印刷出力での順序を指定します。

別名: VARIABLE
VARIABLES

デフォルト: VAR ステートメントを省略すると、PROC STANDARD はその他のステートメントでリストされないすべての数値変数を標準化します。

例: “例 1: 指定された平均と標準偏差への標準化” (2119 ページ)

構文

VAR *variable(s)*;

必須引数

variable(s)
標準化する変数を識別します。

WEIGHT ステートメント

統計量計算の分析変数に対し重みを指定します。

別名: WGT

参照項目: 重み付き統計量の計算と、WEIGHT ステートメントを使用する例については、“重み付き統計量の計算” (2185 ページ)を参照してください。

構文

WEIGHT *variable*;

必須引数

variable
分析変数の値に重みをつける数値変数を指定します。変数の値は、整数である必要はありません。次のテーブルに、重み値に基づいたアクションを示します。

表 58.1 WEIGHT ステートメント値および PROC STANDARD アクション

重み値	PROC STANDARD アクション
0	オブザベーションをオブザベーションの合計数にカウントします。
0 より小さい	重み値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントする
欠損値	オブザベーションを平均および標準偏差の計算から除外します

負およびゼロの重みを含むオブザベーションを平均および標準偏差の計算から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

ヒント WEIGHT ステートメントを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。詳細については、“VARDEF=*divisor* ” (2103 ページ) および重み付き統計量の計算(“キーワードと式” (2410 ページ))を参照してください。

詳細

注: バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

使用法: STANDARD プロシジャ

統計計算: STANDARD プロシジャ

値の標準化により、場所とスケールの属性がデータのセットから削除されます。標準化された値を計算する式は、

$$x_i' = \frac{S * (x_i - \bar{x})}{s_x} + M$$

ここで、

x_i'
新しい標準化された値です。

S
STD=の値です

M
MEAN=の値です。

x_i
オブザベーションの値です。

$\overline{x}$
変数の平均です。

s_x
変数の標準偏差です。

PROC STANDARD は、平均($\overline{x}$)と標準偏差(s_x)を入力データセットから計算します。結果として標準化された変数は、平均が **M**、標準偏差が **S** となります。

データが正規分布している場合、結果のデータはスチューデントの **t** 分布になるため、標準化はスチューデント化でもあります。

結果: STANDARD プロシジャ

欠損値

デフォルトで、PROC STANDARD は分析変数に対する欠損値を標準化プロセスから除外します。値は出力データセットで欠損のままです。REPLACE オプションを指定すると、プロシジャは欠損値を変数の平均または MEAN=値と置き換えます。

WEIGHT 変数または FREQ 変数の値が欠損している場合、プロシジャは平均と標準偏差の計算にオブザベーションを使用しません。ただし、オブザベーションは標準化されます。

出力データセット

PROC STANDARD は、OUT=オプションを指定するかどうかに関係なく、標準化された値を VAR ステートメント変数に保存する出力データセットを常に作成します。出力データセットには、標準化されていないものを含む、すべての入力データセット変数が含まれています。PROC STANDARD は、出力データセットを印刷しません。出力データセットを印刷するには、PROC PRINT、PROC REPORT または別の SAS レポートツールを使用します。

例: STANDARD プロシジャ

例 1: 指定された平均と標準偏差への標準化

要素:

PROC STANDARD ステートメントオプション
- MEAN=
- OUT=
- STD=

VAR ステートメント

PRINT プロシジャ

詳細

この例では、次を行います。

- 2 つの変数を平均 75、標準偏差 5 に標準化します。
- 出力データセットを指定します。
- 標準化された変数を元の変数と結合します。
- 出力データセットを印刷します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

data score;
  length Student $ 9;
  input Student $ StudentNumber Section $
        Test1 Test2 Final @@;
  format studentnumber z4.;
  datalines;
Capalleti 0545 1 94 91 87  Dubose    1252 2 51 65 91
Engles    1167 1 95 97 97  Grant     1230 2 63 75 80
Krupski   2527 2 80 69 71  Lundsford 4860 1 92 40 86
McBane    0674 1 75 78 72  Mullen    6445 2 89 82 93
Nguyen    0886 1 79 76 80  Patel     9164 2 71 77 83
Si        4915 1 75 71 73  Tanaka    8534 2 87 73 76
;
```

```
proc standard data=score mean=75 std=5 out=stndtest;

  var test1 test2;
run;

proc sql;
  create table combined as
  select old.student, old.studentnumber,
       old.section,
       old.test1, new.test1 as StdTest1,
       old.test2, new.test2 as StdTest2,
       old.final
  from score as old, stndtest as new
  where old.student=new.student;

proc print data=combined noobs round;
  title 'Standardized Test Scores for a College Course';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションで、SAS ジョブの開始日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Score データセットを作成します。 このデータセットには、2 つのテストと期末試験を受けた学生のテストの成績が含まれます。FORMAT ステートメントは、Z*w.d* 出力形式を StudentNumber に割り当てます。この出力形式は、右寄せ出力にブランクではなく、0 を埋め込みます。LENGTH ステートメントは、Student の値の保存に使用するバイト数を指定します。

```
data score;
  length Student $ 9;
  input Student $ StudentNumber Section $
       Test1 Test2 Final @@;
  format studentnumber z4.;
  datalines;
Capalleti 0545 1 94 91 87  Dubose    1252 2 51 65 91
Engles    1167 1 95 97 97  Grant     1230 2 63 75 80
Krupski   2527 2 80 69 71  Lundsford 4860 1 92 40 86
McBane    0674 1 75 78 72  Mullen    6445 2 89 82 93
Nguyen    0886 1 79 76 80  Patel     9164 2 71 77 83
Si        4915 1 75 71 73  Tanaka    8534 2 87 73 76
;
```

標準化されたデータを生成し、Stndtest 出力データセットを作成します。 PROC STANDARD は平均 75 と標準偏差 5 を使用して、値を標準化します。OUT=は、標準化された値を含むデータセットとして Stndtest を識別します。

```
proc standard data=score mean=75 std=5 out=stndtest;
```

標準化する変数を指定します。 VAR ステートメントは標準化する変数と、その出力順序を指定します。

```
  var test1 test2;
run;
```

元の値と標準化された値を結合するデータセットを作成します。 PROC SQL は Score と Stndtest を結合して、学生ごとの標準化されたテストの成績と元のテストの成績を含む Combined データセット(テーブル)を作成します。AS を使用して標準化された変数 NEW.TEST1 の名前を StdTest1 に、NEW.TEST2 の名前を StdTest2 に変更すると、変数名が一意のものになります。

```
proc sql;
  create table combined as
  select old.student, old.studentnumber,
       old.section,
       old.test1, new.test1 as StdTest1,
       old.test2, new.test2 as StdTest2,
       old.final
  from score as old, stndtest as new
  where old.student=new.student;
```

データセットを出力します。 PROC PRINT で、Combined データセットを印刷します。ROUND は、標準化された値を小数点以下 2 桁に丸めます。TITLE ステートメントでタイトルを指定します。

```
proc print data=combined noobs round;
  title 'Standardized Test Scores for a College Course';
run;
```

出力: リスト

次のデータセットには、標準化された値を持つ変数と元の値を持つ変数の両方が含まれています。StdTest1 と StdTest2 には、PROC STANDARD で計算される標準化されたテストの成績が保存されています。

アウトプット 58.3　大学の講座の標準化した試験得点

Standardized Test Scores for a College Course 1

Student	Student Number	Section	Test1	Std Test1	Test2	Std Test2	Final
Capalleti	0545	1	94	80.54	91	80.86	87
Dubose	1252	2	51	64.39	65	71.63	91
Engles	1167	1	95	80.91	97	82.99	97
Grant	1230	2	63	68.90	75	75.18	80
Krupski	2527	2	80	75.28	69	73.05	71
Lundsford	4860	1	92	79.79	40	62.75	86
McBane	0674	1	75	73.40	78	76.24	72
Mullen	6445	2	89	78.66	82	77.66	93
Nguyen	0886	1	79	74.91	76	75.53	80
Patel	9164	2	71	71.90	77	75.89	83
Si	4915	1	75	73.40	71	73.76	73
Tanaka	8534	2	87	77.91	73	74.47	76

例 2: BY グループの標準化と欠損値の置換

要素:

PROC STANDARD ステートメントオプション
- PRINT
- REPLACE

BY ステートメント

FORMAT プロシジャ

PRINT プロシジャ

SORT プロシジャ

詳細

この例では、次を行います。

- 平均 0 と標準偏差 1 を使用して BY グループごとに Z スコアを個別に計算します。
- 欠損値を指定の平均と置き換えます。
- 標準化する変数の平均と標準偏差を印刷します。
- 出力データセットを印刷します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=60;

proc format;
  value popfmt 1='Stable'
               2='Rapid';
run;

data lifexp;
  input PopulationRate Country $char14. Life50 Life93 @@;
  label life50='1950 life expectancy'
        life93='1993 life expectancy';
  datalines;
2 Bangladesh     .  53 2 Brazil        51 67
2 China         41 70 2 Egypt         42 60
2 Ethiopia      33 46 1 France        67 77
1 Germany       68 75 2 India         39 59
2 Indonesia     38 59 1 Japan         64 79
2 Mozambique     . 47 2 Philippines    48 64
1 Russia         . 65 2 Turkey        44 66
1 United Kingdom 69 76 1 United States  69 75
;
```

```
proc sort data=lifexp;
  by populationrate;
run;

proc standard data=lifexp mean=0 std=1 replace
           print out=zscore;

  by populationrate;

  format populationrate popfmt.;
  title1 'Life Expectancies by Birth Rate';
run;

proc print data=zscore noobs;
  title 'Standardized Life Expectancies at Birth';
  title2 'by a Country''s Birth Rate';
run;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションで、SAS ジョブの開始日時を省略するように指定します。PAGENO=オプションで、SAS の作成する出力の次ページのページ番号を指定します。LINESIZE=オプションで、線の太さを指定します。PAGESIZE=オプションで、SAS 出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=60;
```

文字列出力形式を数値に割り当てます。 PROC FORMAT で、文字値を含む出生率を識別するための出力形式 POPFMT を作成します。

```
proc format;
  value popfmt 1='Stable'
               2='Rapid';
run;
```

Lifeexp データセットを作成します。 このデータセットの各オブザベーションには、16 か国の 1950 年と 1993 年の出生時平均寿命に関する情報が含まれます。国ごとの出生率は、stable (1)または rapid (2)に分類されます。欠損データを含む国は、1950 年より後に独立した国です。データソース: Vital Signs 1994: The Trends That Are Shaping Our Future, Lester R. Brown, Hal Kane, and David Malin Roodman, eds.Copyright © 1994 by Worldwatch Institute.Reprinted by permission of W.W.Norton & Company, Inc.)によるものです。

```
data lifexp;
  input PopulationRate Country $char14. Life50 Life93 @@;
  label life50='1950 life expectancy'
       life93='1993 life expectancy';
  datalines;
2 Bangladesh     .  53 2 Brazil        51 67
2 China         41 70 2 Egypt         42 60
2 Ethiopia      33 46 1 France        67 77
1 Germany       68 75 2 India         39 59
2 Indonesia     38 59 1 Japan         64 79
2 Mozambique     . 47 2 Philippines    48 64
1 Russia         . 65 2 Turkey        44 66
1 United Kingdom 69 76 1 United States  69 75
;
```

Lifeexp データセットを並べ替えます。 PROC SORT で、オブザベーションを出生率別に並べ替えます。

```
proc sort data=lifexp;
  by populationrate;
run;
```

すべての数値変数に対し標準化されたデータを生成し、Z-score 出力データセットを作成します。 PROC STANDARD で、すべての数値変数を平均 1、標準偏差 0 に標準化します。REPLACE で欠損値を置き換えます。PRINT で統計量を印刷します。

```
proc standard data=lifexp mean=0 std=1 replace
          print out=zscore;
```

BY グループごとに標準化された値を作成します。 BY ステートメントで、出生率別に値を個別に標準化します。

```
  by populationrate;
```

変数に出力形式を割り当て、レポートのタイトルを指定します。 FORMAT ステートメントで、PopulationRate に出力形式を割り当てます。出力データセットには、フォーマットされた値が含まれます。TITLE ステートメントでタイトルを指定します。

```
  format populationrate popfmt.;
  title1 'Life Expectancies by Birth Rate';
run;
```

データセットを出力します。 PROC PRINT で、ZSCORE データセットを標準化された値で印刷します。TITLE ステートメントで、印刷する 2 つのタイトルを指定します。

```
proc print data=zscore noobs;
  title 'Standardized Life Expectancies at Birth';
  title2 'by a Country''s Birth Rate';
run;
```

出力: リスト

PROC STANDARD では、BY グループごとに標準化する各変数の変数名、平均、標準偏差、入力度数、ラベルを印刷します。バングラデシュ、モザンビーク、ロシアの平均寿命の欠損はなくなっています。欠損値は指定の平均(0)と置き換えられています。

アウトプット 58.4　出生率別の平均寿命

Life Expectancies by Birth Rate

The STANDARD Procedure

PopulationRate=Stable

Name	Mean	Standard Deviation	N	Label
Life50	67.400000	2.073644	5	1950 life expectancy
Life93	74.500000	4.888763	6	1993 life expectancy

PopulationRate=Rapid

Name	Mean	Standard Deviation	N	Label
Life50	42.000000	5.707138	8	1950 life expectancy
Life93	59.100000	8.225300	10	1993 life expectancy

Standardized Life Expectancies at Birth by a Country's Birth Rate

PopulationRate	Country	Life50	Life93
Stable	France	-0.19290	0.51138
Stable	Germany	0.28935	0.10228
Stable	Japan	-1.63963	0.92048
Stable	Russia	0.00000	-1.94323
Stable	United Kingdom	0.77159	0.30683
Stable	United States	0.77159	0.10228
Rapid	Bangladesh	0.00000	-0.74161
Rapid	Brazil	1.57697	0.96045
Rapid	China	-0.17522	1.32518
Rapid	Egypt	0.00000	0.10942
Rapid	Ethiopia	-1.57697	-1.59265
Rapid	India	-0.52566	-0.01216
Rapid	Indonesia	-0.70088	-0.01216
Rapid	Mozambique	0.00000	-1.47107
Rapid	Philippines	1.05131	0.59572
Rapid	Turkey	0.35044	0.83888

59

STREAM プロシジャ

概要: STREAM プロシジャ

STREAM プロシジャの機能

STREAM プロシジャを使用すると、SAS マクロ指定を含む可能性のある任意のテキストで構成される入力ストリームを処理できます。マクロが実行され、展開されている間、入力ストリームの他のテキストは保持されます。テキストストリームは、SAS 構文として検証されません。出力ストリームは、ファイル参照で参照され、また従来の SAS 出力先を使用するように定義できる外部ファイルに送信されます。

概念: STREAM プロシジャ

テキストおよびマクロコードの解釈

次の例では、%DOIT という名前のマクロを使用して、テーブルを含む HTML ストリームを生成します。

```
%macro doit(nrows,ncols);
<table>
%do i=1 %to &nrows;
<tr>
%do j=1 %to &ncols;
<td>&j</td>
%end;
</tr>
%end;
</table>
%mend doit;
```

このマクロが SAS コードストリーム内で実行されると、HTML 出力が生成されますが、出力は有効な SAS 構文ではないため構文上は無効になります。ただし、このマクロが PROC STREAM を介して実行されると、HTML 出力はファイルに書き込まれ、SAS 構文として検証されません。次に例を示します。

```
proc stream outfile=myfile; begin
%doit(2,3)
;;;;
```

次の出力が *myfile* に書き込まれます。

```
<table> <tr> <td>1</td> <td>2</td> <td>3</td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> </tr>
</table>
```

入力ストリームでは%INCLUDE ステートメントを使用できます。使用可能な入力ファイルの種類には、HTML ファイル、RTF ファイル、その他の種類のテキストベースファイルがあります。マクロの定義と呼び出しを除き、実際の SAS コードを含めるために%INCLUDE を使用することはありません。マクロ以外の SAS コードは実行されず、他のテキストと同様に扱われます。

%の後に続く、マクロで認識されるテキストが実行されます。%LET、%INCLUDE、%SYSFUNC などがあります。ユーザー定義マクロなど他のマクロも実行されます。

すべてのマクロ変数参照が解決されますが、マクロ名が認識されない場合は警告が発行されます。%PUT や%LIST などのマクロステートメントでは、出力がログに送

信され、トークンは生成されません。これらのステートメントを入力ストリームで使用することは可能ですが、使用しないようにする必要があります。

トークナイザーの制限

SAS トークナイザーまたはワードスキャナには、SAS 構文入力ストリーム以外の構文入力ストリームを正しく処理する際に問題となる可能性のある多くの制限があります。これらの制限について、次のリストで説明します。

- %INCLUDE ステートメントからの入力レコードは、レコードサイズのデフォルト値である 32,767 バイトを超えることはできません。
- %INCLUDE からの入力レコードが 32,767 バイトを超えると、レコードサイズに関する正確な情報がトークナイザーから PROC STREAM に提供されません。
- %INCLUDE および%LET ステートメントはステートメント境界から開始する必要があります。つまり、ステートメントの前にセミコロンを付け、ステートメントはセミコロンで終わる必要があります。マクロの呼び出しにはこの要件は適用されません。
- マクロステートメントはトークナイザーですべて 使用および実行されるため、PROC STREAM でマクロステートメントは認識されません。たとえば、%let xyz=2;はすべて 使用されます。この動作は、スコープ内でグローバルではない、%IF や%GOTO などのマクロステートメントにも当てはまります。これらのマクロステートメントにはエラーのフラグが立てられますが、PROC STREAM でそのエラーは認識されません。
- a～z (大文字または小文字)の文字、または&が前に付いているアンダースコアから始まるトークンは、解決するマクロ変数であるかどうかがマクロプロセッサによって判定されます。マクロ変数でない場合は、警告が発行されます。現在、この警告は非表示にすることができません。
- PL/I プログラミング言語のコメント(/* *comment* */)は、通常、トークナイザーによって完全に取り込まれ、PROC STREAM では表示されません。PROC STREAM ステートメントで NOABSSCMT オプションを使用すると、/*と*/およびその間にあるすべてのテキストが 表示されます。このオプションは、入力ストリームに引用符で囲まれていない/*が含まれている可能性がある場合にも必要になります。たとえば、Linux のワイルドカード指定を表す/tmp/xyz/*がこれに該当します。
- 1 つのトークンが 32,767 文字を超えることはできません。
- レコードの最後のトークンの後に、オプションのキャリッジリターンが前に付いた改行が続き、次のレコードの最初のトークンが列 1 から開始される場合、トークナイザーはこれらのトークンの間にブランクを挿入します。このブランクがあると有効な SAS 構文になります。これらのトークンは、引用符で囲まれている文字列の一部でない場合連結されません。たとえば、レコードの長さが 80 文字であるとします。前にブランクの付いている文字列 *abc* は列 78～80 に配置されます。次のレコードでは、後ろにブランクの付いた *def* が列 1～3 に表示されます。この場合、PROC STREAM は、先頭のブランクがトークンストリームに表示されていない場合でも、*abc* を受け取った後に、先頭にブランクの付いた *def* を受け取ります。そうではなく、ブランクが前に付いた*'abc* が列 77～80 に表示され、後ろにブランクの付いた *def'*が列 1～4 に表示されている場合、PROC STREAM は*'abcdef'*を受け取ります。

- 印刷不可の文字(キャリッジリターン(CR)や改行(LF)を除く)は正しくトークン化できません。
- 入力ストリームを、PDF や JPG ファイルなどのバイナリストリームにすることはできません。
- トークナイザーの通常の動作は、単一引用符または二重引用符の付いた文字列を単一のトークンとして提供することです。*don't do this* などのテキストが引用符で囲まれていない場合や、*don''t do this* のようにテキストにエスケープシーケンスが含まれている場合、これが問題になります。QUOTING=オプションを使用すると、引用符は、ハイフンやスラッシュなどの他の特殊文字と同様に処理されます。
- 1 つのマクロ変数を 64,000 文字まで展開できます。ただし、メモリやディスク領域などのリソースの制限を除き、マクロがモデルテキストとして返すことができる内容に制限はありません。

構文: STREAM プロシジャ

PROC STREAM OUTFILE=*fileref* <*options*>; BEGIN
text-1
<*text-2* ...>
;;;;

ステートメント	タスク
“PROC STREAM ステートメント”	SAS マクロ指定を含むことができる任意のテキストで構成される入力ストリームを処理します。

PROC STREAM ステートメント

SAS マクロ指定を含むことができる任意のテキストで構成される入力ストリームを処理できます。

構文

PROC STREAM OUTFILE=*fileref* <*options*>; BEGIN
text-1
<*text-2* ...>
;;;;

必須引数

OUTFILE=*fileref*
すべてのトークンが書き込まれるファイルを指定します。

ファイル参照の LRECL 指定が使用されます。LRECL を指定しない場合、グローバル LRECL=オプションのデフォルト値である 32,767 バイトが使用されます。PRESCOL オプションを使用しない場合、トークン間に適切な数のブランクが挿入されて、すべてのトークンがストリーム出力されます。レコード間でトークンは分割されません。また、ブランクがトークン内にない限り、トークンのストリームはレコード間で分割されません。たとえば、<table>X</table>はレコード間で分割されませんが、<table> X </table>はブランクが示されている場所(X の前後)で分割されます。

text
PROC STREAM で使用する SAS ステートメントまたはマクロを指定します。

オプション引数

MOD
出力ファイルが上書きされるのではなく追加されることを指定します。

NOABSSCMT
コメントが出力ストリームに書き込まれるかどうかを指定します。

PL/I プログラミング言語スタイルのコメントが表示される場合(/* *comments* */)、コメント文字(/*と*/)間のすべてのテキストが出力ストリームに表示されます。このオプションを指定しない場合、PL/I スタイルのコメントは出力ストリームに表示されません。NOABSSCMT を設定する場合は、通常単一引用符(word don't など)がコメントに表示される可能性があるため、QUOTING=も設定することを強くお勧めします。

PRESCOL
元の入力ファイルの列を保持することを示します。

マクロ代替がある場合やレコードサイズが 32,767 バイトを超える場合、このオプションを使用しても失敗します。この場合、マクロの展開が列の場所に影響する可能性があります。

PRESCOL オプションを指定すると、%INCLUDE マクロで含まれる RTF ファイルの有効性が向上します。

QUOTING=SINGLE | DOUBLE | BOTH
引用符の処理方法を指定します。

SINGLE
単一引用符を他の文字と同様に扱うことを指定します。SINGLE オプションを使用する場合で、'&hello'のように、マクロ参照が単一引用符内にある場合、マクロ参照が展開されます。

DOUBLE
二重引用符を他の文字と同様に扱うことを指定します。

BOTH
SINGLE および DOUBLE の両方のオプションを使用することを指定します。

RESETDELIM='*label*'
特殊マーカートークンを示します。

%INCLUDE や%LET ステートメントのように、ステートメントを展開する必要がある場合にこのオプションが使用されます。これらのステートメントはステートメント境界から開始する必要があります。構文でステートメント境界が許可されていない場合、指定されたラベルとその後に続くセミコロンを入力ストリームに挿入して、トークナイザー要件を満たすことができます。ラベルとセミコロンは出力ファイルに送信されません。

次の例では、%INCLUDE の前にセミコロンが付いておらず、コードは正しくありません。

x y %include *myfile*; z

ただし、RESETDELIM=オプションを使用すると、展開は予想どおりに実行されます。

resetdelim='*mylabel*'
x y *mylabel*;
%include *myfile*; z

*Mylabel;*は出力ファイルに表示されません。

Mylabel は有効な SAS 名であることが必要です。つまり、先頭は文字またはアンダースコアで、それ以降のすべての文字は文字、アンダースコア、数値である必要があります。*mylabel* の長さは 32 文字以内であることが必要です。

指定と使用法では、大文字と小文字が区別されません。RESETDELIM=にはデフォルト値がありません。

PROC STREAM は、&STREAMDELIM というマクロ変数の存在を確認します。このマクロ変数が存在する場合は、そのまま使用されます。このマクロ変数が存在しない場合、PROC STREAM は、RESETDELIM=オプションが指定されているかどうかを検証します。このオプションが指定されている場合、マクロ変数 &STREAMDELIM は、RESETDELIM=オプションの値に設定されます。RESETDELIM=オプションが指定されていない場合、PROC STREAM は、現在の日時値に基づいて&STREAMDELIM マクロ変数の一意の値を作成します。

この点を考慮して、&STREAMDELIM を PROC STREAM の入力に追加できます。このマクロ変数が表示される場合、オプションのキーワードが続くものと見なされます。終了のセミコロンが必要です。&STREAMDELIM 値からセミコロンまで(セミコロンを含む)のすべてのトークンは出力ストリームに送信されず、PROC STREAM の特別な管理情報項目になります。

入力で%INCLUDE ステートメントを指定する必要がある場合でも、その前にセミコロンを付けない場合は、%INCLUDE ステートメントの前に次のステートメントを追加する必要があります。

&STREAMDELIM;

このステートメントを指定すると、セミコロンはトークナイザーに認識されますが、PROC STREAM に取り込まれます。

&STREAMDELIM の後に次のオプションのキーワードを続けることができます。

NEWLINE
新しい行が出力ファイルに送信されることを指定します。

READFILE *filename*
指定されたファイル名が開かれ、その内容がそのまま読み込まれて出力ファイルに書き込まれることを指定します。このファイル内のマクロの展開は行われず、新しい行が保持されます。この動作は、マクロの展開が実行され、

新しい行が無視される%INCLUDE の動作と異なります。READFILE キーワードの使用例については、“READFILE キーワードの使用” (2136 ページ)を参照してください。

使用法: STREAM プロシジャ

入力ストリームでのマクロベースコードの使用

入力ストリームのマクロベースコードは、次の例のように展開されます。

```
%let abc=123;
proc stream outfile=out; begin
<title>Run &abc</title>
<table>
<tr> <td>1<td>2</tr>
<tr> <td>3<td>4</tr>
</table>
;;;;
```

出力ファイルには次の結果が含まれます。

```
<title>Run 123</title> <table> <tr> <td>1<td>2</tr> <tr> <td>3<td>4</tr> </table>
```

改行はありませんが、ブランクは保持され、改行ごとにブランクが挿入されます。

入力ストリームの%と&が引用符で囲まれておらず、エスケープシーケンスではないため、問題が発生する可能性があります。たとえば、*&*は、HTML ストリームでアンパサンドを表すエスケープシーケンスとして 表示されます。*&AMP* という名前のマクロ変数を使用すると、その値が*&amp* のかわりに使用されます。この種類のマクロ変数が存在しない場合は、警告が発行されます。これらの問題を回避するためには、次の例に示す%STR(&)のように、&をエスケープ文字として使用します。

```
<title>A %str(&)amp; B</title>
```

次の出力が出力ストリームに書き込まれます。

```
<title>A &amp; B</title>
```

%についても同様に、マクロで正しく解釈されないものは、次の例に示すように、%STR(%)として示されます。

```
<a href="www.sas.com/x%str(%20y)">the link</a>
```

次の出力が出力ストリームに書き込まれます。

```
<a href="www.sas.com/x%20y">the link</a>
```

%STR のエスケープシーケンスが単一引用符内にある場合は、QUOTING=オプションを使用するか、%STR(%')のように、エスケープシーケンスと 単一引用符を使用します。

角かっこまたはかっこを使用して入力ストリームを開始する

角かっこまたはかっこ(つまり、{、[、(で入力ストリームを開始すると、BEGIN キーワードは名前として扱われます。これは、最初の入力文字がコード内で BEGIN の後の行にある場合にも当てはまります。たとえば、次のコードでは、入力ストリームの開始を識別するキーワードとしてではなく、名前として BEGIN を扱います。

```
filename dheader "c:\temp\dheader.txt";
proc stream outfile=dheader; begin
[SAS version 9.4]
;;;;
```

これを防ぐには、&STREAMDELIM;ステートメントを使用します。&STREAMDELIM;ステートメントはいつでも GEGIN の後に使用して、入力ストリームの開始を示すことができます。

このステートメントを含めると、入力ストリームの開始点が正しく解釈されます。

```
filename dheader "c:\temp\dheader.txt"
proc stream outfile=dheader; begin &streamdelim;
[SAS version 9.4]
;;;;
```

SAS コードの実行

前のセクションで説明したように、存在するすべての SAS コードが実行されるわけではありません。トークンは他のトークンと同様にストリーム出力されます。この動作の例外は、%SYSFUNC または%SYSCALL マクロ関数が存在している場合に起こります。これらのマクロ関数は、次の例に示すように指定された関数を実行します。

```
%let abc=%sysfunc(getoption(obs));
&abc
```

このステートメントで GETOPTION 関数が呼び出され、OBS オプションの値が取得されます。GETOPTION は、値を ABC マクロ変数に配置する文字列として返します。次に、その値が出力ストリームに書き込まれます。

%SYSFUNC で DOSUB 関数を使用して、SAS コードのストリームを実行することができます。DOSUB 関数ではファイル参照が指定され、そのファイル内のすべての SAS コードが実行されます。

この例では、ファイル参照 MYCODE が、次の SAS コードを指し示しています。

```
filename myhtml "c:\temp\temp.txt";

data _null_;
```

```
  file myhtml;
  put '<table>';
  put '<tr><td>1</td></tr>';
  put '<tr><td>2</td></tr>';
  put '</table>';
run;
```

次の STREAM プロシジャを実行すると、PROC STREAM が *mycode* を%SYSFUNC 関数の DOSUB に対する引数として使用していることがわかります。

```
filename myhtml "c:\temp\temp.txt";
filename new "c:\temp\new.html";

proc stream outfile=new; begin
%let abc=%sysfunc(dosub(mycode));
%include myhtml;
;;;;
```

出力ストリームには次の結果が含まれます。

```
<table><tr><td>1</td></tr><tr><td>2</td></tr></table>
```

注: DOSUB 関数は DOSUBL 関数と似ていますが、DOSUB には、SAS コードを含むファイルのファイル参照が渡されます。DOSUBL にはテキスト文字列が渡され、その値が SAS コードとして実行されます。詳細については、"DOSUBL 関数" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

Rich Text Format (RTF)ファイル出力

Microsoft Word を使用してデータを入力し、RTF 出力を作成できます。Today is &sysdate..および My name is &name..と入力すると、Microsoft Word ではデータが Mytest.rtf という名前の RTF ファイルとして保存されます。このファイルのサイズは約 30K で、大量のマークアップデータが含まれています。そのデータの 1 つのセクションに、入力された実際のテキストが次のように含まれます。

```
\fs20\lang1033\langfe1033\cgrid\langnp1033\langfenp1033 {\rtlch\fcs1 \af0 \ltrch\fcs0 \insrsid8922096
Today is &sysdate.. }{\rtlch\fcs1 \af0 \ltrch\fcs0 \insrsid14092174
\par }{\rtlch\fcs1 \af0 \ltrch\fcs0 \insrsid8922096
\par My name is &name..
\par
```

入力されたテキストにはマクロ置換が含まれています。

この SAS プログラムを実行すると、Newrtf.rtf という名前の新しい RTF ファイルが作成されます。この出力ファイルにおける違いは、マクロ置換は行われていても、マークアップ言語が変わらないことです。

```
%let name=John;
filename oldrtf 'mytest.rtf' recfm=v lrecl=32767;
filename newrtf 'newrtf.rtf' recfm=v lrecl=32767;
proc stream outfile=newrtf quoting=both asis; begin
&streamdelim;
```

```
%include oldrtf;
;;;;
```

Newrtf.rtf ファイルが Microsoft Word で読み込まれると、次のような内容が表示されます。

```
Today is 30NOV12.

My name is John.
```

READFILE キーワードの使用

HTML では、<PRE> ...</PRE>タグが、事前にフォーマットされたテキストのストリームをカプセル化し、改行やスペースを 尊重して、そのまま表示されます。
READFILE キーワードを使用すると、ファイルの内容を読み込み、そのまま保持できます。READFILE キーワードは、&STREAMDELIM マクロ変数の後に続きます。

```
filename fixed temp;
data _null_; file fixed;
   input; put _infile_;
   datalines4;
This is the first line of fixed text.
This is another line to be fixed.
This is the last line of fixed text.
;;;;

%macro doit(nrows,ncols);
<table>
%do i=1 %to &nrows;
<tr>
%do j=1 %to &ncols;
<td>&j</td>
%end;
</tr>
%end;
%mend;

filename new temp;
proc stream outfile=new; begin
<PRE>
&streamdelim readfile fixed;
</PRE>
%doit(2,3)
;;;;

data _null_; infile new; input; put _infile_; run;
filename fixed temp;
data _null_; file fixed;
   input; put _infile_;
   datalines4;
This is the first line of fixed text.
This is another line to be fixed.
```

```
This is the last line of fixed text.
;;;;

%macro doit(nrows,ncols);
<table>
%do i=1 %to &nrows;
<tr>
%do j=1 %to &ncols;
<td>&j</td>
%end;
</tr>
%end;
%mend;

filename new temp;
proc stream outfile=new; begin
<PRE>
&streamdelim readfile fixed;
</PRE>
%doit(2,3)
;;;;

data _null_; infile new; input; put _infile_; run;
```

結果のファイルは次のようになります。

```
<PRE>
This is the first line of fixed text.
This is another line to be fixed.
This is the last line of fixed text.
</PRE> <table> <tr> <td>1</td> <td>2</td> <td>3</dt> </tr>
<tr> <td>1</td> <td>2</td> <td>3</dt> </tr> </table>
```

%INCLUDE を使用した PROC STREAM へのファイルの挿入

複雑な HTML および Javascript コードをストリーミングするには、別のファイルでコードを作成します。次に、%INCLUDE ステートメントを使用して、ファイルをプログラムに含めます。

次の例は、このアプローチを示しています。

```
filename temp1 temp;

data _null_;
  infile datalines;
  file temp1;
  input;
  l=length(_infile_);
  put @1 _infile_ $varying200. l;
  datalines4;
<table>
```

```
<tr>
<td>abc</td><td>def</td>
</tr>
<tr>
<td>ghi</td><td>jkl</td>
</tr>
</table>
;;;;

filename myfile "c:\temp\test1.html";

proc stream outfile=myfile prescol resetdelim='label'; begin
<html>
<h2>Test Example</h2>
label;
%include temp1;
</html>
;;;;
```

出力ストリームへの新しい行の挿入

新しい行を出力ストリームに挿入するには、次の例に示すように、デリミタの後ろのセミコロンの前にキーワード NEWLINE を追加します。

```
proc stream outfile=abc resetdelim='mylabel'; begin
my item here;
mylabel newline;
my next item here
;;;;
```

この例では、次の出力が生成されます。

```
my item here;
my next item here
```

他の方法では予測どおりに新しい行を挿入できません。

STREAM プロシジャの終了

PROC STREAM ステップの終了は、プロシジャの最後の行として記述される、間にブランクを含まない 4 つのセミコロンで示されます。入力ストリームの最後の項目がセミコロンで終了する場合は、最後の項目の直後に、RESETDELIM=で指定されたラベルと、それに続くセミコロンを使用します。

次の例では、here の後のセミコロンが認識されないため、正しくコード化されません。

```
proc stream outfile=abc; begin
my item here;
;;;;
```

次の例は正しくコード化されます。

```
proc stream outfile=abc resetdelim='mylabel'; begin
my item here;
mylabel;
;;;;
```

この例では、here の後のセミコロンを認識させるため、RESETDELIM=が使用されています。

60

SUMMARY プロシジャ

概要: SUMMARY プロシジャ

SUMMARY プロシジャの機能

SUMMARY プロシジャは、すべてのオブザベーションにわたる変数、またはオブザベーショングループ内の変数に対する記述統計量を計算するためのデータ要約ツールを提供します。SUMMARY プロシジャは、MEANS プロシジャと非常によく似ています。構文の詳細については、35 章, “MEANS プロシジャ,” (1221 ページ)を参照してください。 ここで説明されている相違を除くすべての PROC MEANS 情報は、PROC SUMMARY にも適用されます。

文字出力形式をプリロードする場合、ラベルを生成する代表的な未加工値が少なくとも 1 つ存在しなければなりません。範囲に未加工値がない場合、到達不能と判断され、エラーが発生します。たとえば、Other=と Low-High の両方の範囲を持つマルチラベル文字出力形式をプリロードすることはできません。Other は、すべてのラベルが定義された後に残る値または範囲を参照する特別なキーワードです。文字出力形式の場合、low 範囲に欠損値が含まれるため、その他の域を占める未加工値が残りません。PRELOADFMT オプションの詳細は、を参照してください。

PROC SUMMARY は、SAS Cloud Analytic Services(CAS)で使用できます。PROC SUMMARY を CAS アクションで実行することは、SAS 内での処理に比べていくつかの利点があります。CAS での PROCMEANS および PROCSUMMARY の実行についての情報については、35 章, “MEANS プロシジャ,” (1221 ページ) を参照してください。

VARCHAR 変数は PROC SUMMARY ではサポートされていません。VARCHAR として宣言された変数は CHAR に変換されます。

計算された統計は、オブザベーションが処理される順序に応じてわずかに異なる場合があります。このような変動は、浮動小数点演算によって導入される数値エラーによるものであり、その結果は概算であり、正確ではないと見なす必要があります。オブザベーション処理の順序は、マルチスレッド処理または並列処理の非決定論的影響を受ける可能性があります。処理の順序は、ACCESS エンジンを介してクエリ結果を提供する DBMS などのデータソースによって生成されるオブザベーションの一貫性のない、または非決定論的な並べ替えによっても影響を受ける可能性があります。詳細については、“Base SAS のスレッド処理” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

構文: SUMMARY プロシジャ

サポート: スレッド処理

ヒント: Output Delivery System をサポートします。詳細については、“Output Delivery System: 基本概念” (*SAS Output Delivery System: ユーザーガイド*)を参照してください。

ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。

構文の説明については、“構文” (1233 ページ)を参照してください。

PROC SUMMARY<*options*> <*statistic-keyword(s)*>;

ATTRIB *variable-list(s) attribute-list(s)*;

BY<DESCENDING>*variable-1*<<DESCENDING> *variable-2* ...>
<NOTSORTED>;

CLASS *variable(s)*</ *options*>;

FORMAT *variable-1*<...*variable-n*> <*format*> <DEFAULT=*default-format*>;

FREQ *variable*;

ID *variable(s)*;

LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;

OUTPUT<OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
<*id-group-specification(s)*> <*maximum-id-specification(s)*>
<*minimum-id-specification(s)*> </ *options*>;

TYPES *request(s)*;

VAR *variable(s)*</ WEIGHT=*weight-variable*>;

WAYS *list*;

WEIGHT *variable*;

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク
“PROC SUMMARY ステートメント”	すべてのオブザベーションにわたる 変数、またはオブザベーショングループ内の変数に対して記述統計量を計算します。
“BY ステートメント”	BY グループごとに別の統計量を計算します。
“CLASS ステートメント”	値が分析対象のサブグループを定義する変数を識別します。
“FREQ ステートメント”	値が各オブザベーションの度数を表す変数を識別します。
“ID ステートメント”	追加の ID 変数を出力データセットに含めます。
“OUTPUT ステートメント”	指定されている統計量と ID 変数を含む出力データを作成します。
“TYPES ステートメント”	データの分割に使用する分類変数の特定の組み合わせを識別します。
“VAR ステートメント”	分析変数と結果での順序を識別します。
“WAYS ステートメント”	分類変数の一意の組み合わせを作成するための方法の数を指定します。
“WEIGHT ステートメント”	値が統計量計算で各オブザベーションを重み付ける変数を識別します。

PROC SUMMARY ステートメント

すべてのオブザベーションにわたる 変数とオブザベーショングループ内の変数に対し記述統計量を計算します。

参照項目: 構文の詳細については、“PROC MEANS ステートメント” (1235 ページ)を参照してください。

構文

PROC SUMMARY<*options*> <*statistic-keyword(s)*>;

オプション引数

ALPHA=*value*
平均値に対する信頼限界を計算する信頼水準を指定します。信頼限界のパーセントは、(1−*value*)×100 です。例は次のとおりです(ALPHA=.05 は、95%の信頼水準になります)。

デフォルト .05

範囲 0 から 1 まで

操作 信頼限界を計算するには、*statistic-keyword* CLM、LCLM または UCLM を指定します。

参照項目: “信頼限界” (1287 ページ)

“例 7: 平均の信頼限界の計算” (1310 ページ)

CHARTYPE
出力データセットの_TYPE_変数が_TYPE_のバイナリ値の文字表現になるように指定します。変数の長さは分類変数の数と同じです。

操作 32 を超える分類変数を指定すると、_TYPE_は自動的に文字変数になります。

参照項目: “出力データセット” (1290 ページ)

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

CLASSDATA=*SAS-data-set*
出力に表示が必要な分類変数の値の組み合わせを含むデータセットを指定します。入力データセットではなく CLASSDATA=データセットで発生する分類変数の値の組み合わせが出力に表示され、度数はゼロとなります。

制限事項 CLASSDATA=データセットにすべての分類変数を含める必要があります。このデータの種類と出力形式は、入力データセットの対応する分類変数と一致する必要があります。

操作 EXCLUSIVE オプションを使用する場合、PROC MEANS は分類変数の組み合わせが CLASSDATA=データセットにない入力データセットのオブザベーションを除外します。

ヒント CLASSDATA=データセットを使用して、入力データセットをフィルタリングまたは補足します。

参照項目: “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

COMPLETETYPES
組み合わせが入力データセットで発生しない場合でも、分類変数の可能な組み合わせをすべて作成します。

操作 CLASS ステートメントの PRELOADFMT オプションにより、度数がゼロの場合でも PROC MEANS が分類変数の組み合わせに対するユーザー定義出力形式範囲または値を出力に書き込むようになります。

ヒント COMPLETETYPES を使用しても、必要メモリは増えません。

参照項目: “例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

DATA=*SAS-data-set*
入力 SAS データセットを識別します。

参照項目: “入力データセット” (24 ページ)

DESCENDTYPES
_TYPE_値の降順で出力データセットのオブザベーションを並べ替えます。

別名 DESCENDING

DESCEND

操作 NWAY を指定すると、降順は無効です。

ヒント DESCENDTYPES を使用して、全体合計(_TYPE_=0)を各 BY グループの最終オブザベーションにします。

参照項目: “出力データセット” (1290 ページ)

“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1315 ページ)

EXCLNPWGT
非正(ゼロまたは負)の重みがついたオブザベーションを分析から除外します。デフォルトでは、PROC MEANS は非正の重みが付いたオブザベーションを重みがゼロのオブザベーションのように扱い、オブザベーションの合計数にカウントします。

別名 EXCLNPWGTS

参照項目: “WEIGHT ステートメント” (2184 ページ)

VAR ステートメントの WEIGHT=オプション (1275 ページ)

EXCLUSIVE
CLASSDATA=データセットにない分類変数の組み合わせをすべて分析から除外します。

要件 CLASSDATA=データセットが指定されていない場合、このオプションは無視されます。

参照項目: “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

FW=*field-width*
統計量を印刷出力または表示出力に表示するフィールド幅を指定します。FW= は、出力データセットに保存される統計量に影響しません。

デフォルト 12

ヒント PROC MEANS が出力の列ラベルを切り捨てる場合、フィールド幅を増やします。

参照項目: “例 1: 特定の記述統計量の計算” (1292 ページ)

“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

INCAS=(YES | NO)
CAS 内での処理を許可するかどうかを指定します。

YES
CAS 内での処理を使用します。YES がデフォルトです。

NO
CAS 内での処理を使用しないでください。

IDMIN
出力データセットに ID 変数の最小値が含まれるように指定します。

操作 出力に ID 変数の値を表示する PRINTIDVARS を指定します。

参照項目: “ID ステートメント” (2168 ページ)

MAXDEC=*number*
印刷出力または表示出力に統計量を表示する小数点以下の桁の最大数を指定します。MAXDEC=は、出力データセットで保存される統計量に影響しません。

デフォルト BEST.(柱状出力形式の幅。通常は約 7)。

範囲 0-8

参照項目: “例 2: 分類変数を使用した記述統計量の計算” (1294 ページ)

“例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

MISSING
欠損値を有効値とみなし、分類変数の組み合わせを作成します。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING を省略すると、PROC MEANS は欠損分類変数値を含むオブザベーションを分析から除外します。

参照項目: *SAS プログラマガイド: エッセンシャル*で特別な意味を持つ欠損値の説明を参照してください。

“例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

NOLABEL
変数ラベルを抑制します。

例
```
proc means data=sashelp.shoes maxdec=2 nolabels;
  var sales returns;
run;
```

NONOBS
分類変数の値の一意の各組み合わせに対するオブザベーション合計数を表示する列を非表示にします。この列は、出力データセットの_FREQ_変数に対応しています。

参照項目:
“N Obs 統計量” (1290 ページ)

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

“例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

NOPRINT
すべての出力を抑制します。`“PRINT”オプションを参照してください。

ヒント OUT=出力データセットのみ作成する場合は、NOPRINT を使用します。

NOTRAP
データ処理時の浮動小数点例外(Floating Point Exception (FPE))の復旧を無効化します。デフォルトでは、PROC MEANS はこれらのエラーをトラップし、統計量を欠損に設定します。

FPE 回復のオーバーヘッドが重要な動作環境では、NOTRAP はパフォーマンスを向上させることができます。算術例外の場合に PROC MEANS が強制終了するように、通常の SAS FPE 処理は有効です。

NWAY
出力データセットに_TYPE_と_WAY_の値が最高のオブザベーションの統計量のみ含まれるように指定します。分類変数の指定時に、NWAY はすべての 分類変数の組み合わせに対応します。

操作 TYPES ステートメントまたは WAYS ステートメントを指定すると、PROC MEANS はこのオプションを無視します。

参照項目:
“出力データセット” (1290 ページ)

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
並べ替え順序を指定して、出力の分類変数の値の一意の組み合わせを作成します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 CLASS ステートメントで PRELOADFMT を使用すると、各分類変数の値の順序が、PROC FORMAT が関連するユーザー定義出力形式の値の格納に使用する順序と一致します。CLASSDATA=オプションを使用すると、PROC MEANS は CLASSDATA=データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC MEANS はまずユーザー定義出力形式を使用して、出力を並べ替えます。EXCLUSIVE を省略すると、PROC MEANS はユーザー定義出力形式と CLASSDATA=値の後に入力データセットの分類変数の一意の値を発生順に基づいて追加します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。この順序は、使用している動作環境によって異なります。

別名 FMT

EXTERNAL

FREQ
最も多くのオブザベーションを含む水準が最初に表示されるように、値を度数カウントの降順で並べ替えます。

操作 分類変数の多重組み合わせに対し、PROC MEANS は個別の分類変数度数から分類変数組み合わせの順序を決定します。

CLASS ステートメントで ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

UNFORMATTED
値をフォーマットされていない値で並べ替えます。PROC SORT と同じ順序になります。この順序は、使用している動作環境によって異なります。

別名 UNFMT

INTERNAL

デフォルト UNFORMATTED

参照項目: “分類値の並べ替え” (1227 ページ)

PRINT
PROC MEANS が統計量分析を表示するかどうかを指定します。NOPRINT は、すべての出力を行いません。

デフォルト PRINT

ヒント OUT=出力データセットのみ作成する場合は、NOPRINT を使用します。

参照項目: “例 8: 出力統計量を計算する” (1313 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

PRINTALLTYPES
印刷または表示出力の分類変数の要求された組み合わせをすべて(すべての _TYPE_値)表示します。通常、PROC MEANS は種類 NWAY のみ表示します。

別名 PRINTALL

操作 NWAY オプション、TYPES ステートメントまたは WAY ステートメントを使用すると、PROC MEANS はこのオプションを無視します。

参照項目: “例 4: CLASSDATA=データセットを分類変数とともに使用する” (1299 ページ)

PRINTIDVARS
ID 変数の値を印刷または表示出力に表示します。

別名 PRINTIDS

操作 ID 変数の最小値を表示する IDMIN を指定します。

参照項目: “ID ステートメント” (2168 ページ)

QMARKERS=*number*
P2 分位点推定方法に使用するマーカーのデフォルト数を指定します。マーカー数によって固定メモリ空間のサイズを制御します。

デフォルト デフォルト値は、要求する分位点によって異なります。中央値(P50)の場合、*number* は 7 です。分位点(P25 と P50)の場合、*number* は 25 です。分位点 P1、P5、P10、P75 P90、P95 または P99 の場合、*number* は 105 です。複数の分位点を要求すると、PROC MEANS は *number* の最大値を使用します。

範囲 3 より大きい奇数の整数

ヒント マーカー数をデフォルト設定よりも増やして推定値の正確性を向上させます。マーカー数を減らして、メモリと計算時間を節約します。

参照項目: “分位点” (1288 ページ)

QMETHOD=OS | P2
PROC MEANS が分位点の計算時に入力データの処理に使用する方法を指定します。オブザベーション数が QMARKERS=値および QNTLDEF=5 以下の場合、両方の方法による結果は同じになります。

OS
順序統計量を使用します。この方法は、PROC UNIVARIATE が使用する方法と同じです。

注: この方法は、非常に多くのメモリを必要とする可能性があります。

P2
P2 分位点推定方法を使用して、分位点を概算します。

デフォルト OS

制限事項 QMETHOD=P2 の場合、PROC MEANS では MODE または重み付き分位点は計算されません。

ヒント QMETHOD=P2 の場合、一部の分位点(P、P5、P95、P99)の信頼性のある推定は、一部のデータセットに対し可能でないことがあります。

参照項目: “分位点” (1288 ページ)

QNTLDEF=1 | 2 | 3 | 4 | 5
QMETHOD=OS の場合に PROC MEANS が分位点の計算に使用する数学的定義を指定します。QMETHOD=P2 を使用するには、QNTLDEF=5 を使用する必要があります。

別名 PCTLDEF=

デフォルト 5

参照項目: “分位数と関連統計量” (2415 ページ)

statistic-keyword(s)
計算する統計量と、それを出力に表示する順序を指定します。PROC ステートメントで使用可能なキーワードは、“統計量キーワード” (2153 ページ)にリストされています。

デフォルト N、MEAN、STD、MIN、MAX

要件 平均値の標準誤差、信頼限界、スチューデントの *t* 検定を計算するには、VARDEF=オプションのデフォルト値、DF を使用する必要があります。歪度または尖度を計算するには、VARDEF=N または VARDEF=DF を使用する必要があります。

ヒント CLM または LCLM と UCLM の両方を使用して、平均値の両側信頼限界を計算します。LCLM のみ、または UCLM を使用して、片側信頼限界を計算します。

参照項目: キーワードの定義と関連する統計量の式については、“キーワードと式” (2410 ページ)を参照してください。

“例 1: 特定の記述統計量の計算” (1292 ページ)

“例 3: BY ステートメントを分類変数とともに使用する” (1296 ページ)

STACKODSOUTPUT
データセットが印刷出力に類似している ODS 出力オブジェクトを生成します。

STACKODSOUTPUT オプションは、PROC MEANS OUTPUT ステートメントではなく、ODS OUTPUT ステートメントで作成された出力データセットに影響します。

別名 STACKODS

参照項目: “例 13: STACKODSOUTPUT オプションによるデータの制御” (1328 ページ)

SUMSIZE=*value*
分類変数の使用時にデータ要約に使用できるメモリ量を指定します。*value* は、次のうちのいずれかになる可能性があります。

n
***n*K**
***n*M**
***n*G**
利用可能なメモリ量をバイト、キロバイト、メガバイト、ギガバイトでそれぞれ指定します。*n* が 0 の場合、PROC MEANS は SAS システムオプション SUMSIZE=の値を使用します。

MAXIMUM
MAX
利用可能なメモリの最大量を指定します。

デフォルト SUMSIZE=システムオプションの値。

注 SUMSIZE=0 を指定すると、PROC MEANS は優先するグローバル REALMEMSIZE オプションを使用できるようになります。

ヒント 最高の結果を得るには、SUMSIZE=に PROC ステップで利用可能な物理メモリ量より大きい値を指定しないでください。追加容量が必要な場合、PROC MEANS はユーティリティファイルを 使用します。

参照項目: *SAS システムオプション: リファレンス*の SAS システムオプション SUMSIZE=。

THREADS | NOTHREADS
入力データセットの並列処理を有効化または無効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします。(制約を参照)。詳細については、並列処理の使用を参照してください。

デフォ SAS システムオプション THREADS | NOTHREADS の値。

ル
ト

制限事項　サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

操作　PROC MEANS は、BY ステートメントが指定されている、または SAS システムオプション CPUCOUNT の値が 2 未満の場合を除いて SAS システムオプション THREADS を有効にします。PROC MEANS ステートメントで THREADS を使用して、PROC MEANS がこれらの状況で並列処理を使用するように強制できます。

注　THREADS が指定され(SAS システムオプションとして、または PROC MEANS ステートメントで)、もう 1 つのプログラムが入力データセットを読み込み、書き込み、更新のために開いておくと、PROC MEANS が入力データセットを開けない場合があります。この場合、PROC MEANS は処理を停止し、メッセージを SAS ログに書き込みます。

VARDEF=DF | N | WDF | WEIGHT
分散と標準偏差の計算に使用する除数を指定します。次のテーブルに、*divisor* に可能な値と、関連する除数を示します。

表 60.1　VARDEF=に可能な値

値	除数	除数の式
DF	自由度	$n - 1$
N	オブザベーション数	n
WDF	重みの合計 - 1	$(\Sigma_i w_i) - 1$
WEIGHT | WGT	重みの合計	$\Sigma_i w_i$

プロシジャは分散を$CSS/divisor$として計算します。CSSは修正平方和で、$\Sigma (x_i - \bar{x})^2$と等しくなります。分析変数に重み付けをする場合、CSSは$\Sigma w_i(x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト　DF

要件　平均の標準誤差、平均の信頼限界またはスチューデントの *t*-検定を計算するには、VARDEF=のデフォルト値を使用します。

ヒント WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。*i* 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$、およびw_iは *i* 番目のオブザベーションの重みです。この方法により、単位重みを含むオブザベーションの分散の推定が生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな *n* に対し)$\sigma^2/\overline{w}$の推定になります。$\overline{w}$は平均重みです。この方法により、平均重みを含むオブザベーションの分散の漸近的な推定が生成されます。

参照項目: “キーワードと式” (2410 ページ)

“WEIGHT ステートメント” (2184 ページ)

統計量キーワード

CLM
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

CSS
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

KURTOSIS
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

別名 KURT

LCLM
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

MAX
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

MEAN
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

MIN
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

MODE
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

N
記述統計量キーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

NMISS
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

RANGE
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

SKEWNESS
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

別名 SKEW

STDDEV
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

別名 STD

STDERR
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

SUM
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

SUMWGT
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

UCLM
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

USS
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

VAR
記述統計量キーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

MEDIAN
分位点統計量のキーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

別名 P50

P1
分位点統計量のキーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

P5
分位点統計量のキーワードです。 詳細については、“*statistic-keyword(s)*” (2150 ページ)を参照してください。

P10
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

P20
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

P40
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

P70
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

Q1
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

別名 P25

Q3
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

別名 P75

P90
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

P95
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

P99
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

P30
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

P60
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

P80
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

QRANGE
分位点統計量のキーワードです。 詳細については、*“statistic-keyword(s)”* (2150 ページ)を参照してください。

PROBT
仮説検定キーワードです。 詳細については、“statistic-keyword(s)” (2150 ページ)を参照してください。

別名 PBT

T
仮説検定キーワードです。 詳細については、“statistic-keyword(s)” (2150 ページ)を参照してください。

詳細

PRINT | NOPRINT
PROC SUMMARY が記述統計量を表示するかどうかを指定します。デフォルトで、PROC SUMMARY は表示出力をしませんが、PROC MEANS は生成します。

デフォルト NOPRINT

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

BY ステートメント

出力を BY グループ順に並べ替えます。

制限事項: BY ステートメントではグループ変数を指定しないでください。BY 変数をグループ変数と同じにすることはできません。

構文

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

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

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

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

詳細

BY グループ処理

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

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

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

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

- BY ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。
- PROC SORT の詳細については、55 章, “SORT プロシジャ,” (2027 ページ)を参照してください。
- CAS での BY グループ処理の詳細については、“グループ化と順序付け” (*SAS Cloud Analytic Services: DATA ステッププログラミング*)を参照してください。
- インデックスの作成の詳細については、“INDEX CREATE ステートメント” (435 ページ)を参照してください。

BY 変数値のフォーマティング

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

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

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

2 つのデータセットで長さが異なる BY 変数

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

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

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

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

りません。フォーマットされていない文字値は、末尾のブランクを削除後にフォーマットされていない値が同一になる場合に一致します。フォーマットされていない倍精度値は、値が同一の場合に一致します。

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

- A
- A、B
- A、B、C
- A、B、C、D

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

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブランクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデータセットの'ABCD'とセカンダリデータセットの'ABCD 'は一致します。セカンダリデータセットに'ABCDEF'が含まれている場合は、一致しません。

BY ステートメントをサポートする Base SAS プロシジャ

COMPARE	SORT (必須)
CORR	STANDARD
FREQ	SUMMARY
MEANS	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE
REPORT (非ウィンドウ環境のみ)	ODSTEXT
ODSLIST	ODSTABLE

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

関連項目

"BY ステートメント" (*SAS DATA ステップステートメント: リファレンス*)

CLASS ステートメント

値が分析対象のサブグループの組み合わせを定義する変数を指定します。

別名: CLASSES

注: オプションのない CLASS ステートメントは ORDER=INTERNAL(デフォルト)、または PROC MEANS ステートメントの ORDER=オプションで指定された値を使用します。たとえば、次のコードでは、変数 c と d は ORDER=INTERNAL を使用します。PROC MEANS ステートメントで ORDER=オプションが指定されている場合、変数 c と d は PROC MEANS ステートメントの ORDER=オプションで指定された値を使用します。

```
class a b / order=data;
 class c d;
```

ヒント: 複数の CLASS ステートメントを使用できます。

一部の CLASS ステートメントオプションも PROC MEANS ステートメントで使用できます。これらは、すべての CLASS 変数に影響します。CLASS ステートメントで指定するオプションは、CLASS ステートメントの変数にのみ適用されます。

参照項目: CLASS ステートメントによるフォーマットされた値のグループ化方法については、"フォーマットされた値" (62 ページ)を参照してください。

例:
- "例 2: 分類変数を使用した記述統計量の計算" (1294 ページ)
- "例 3: BY ステートメントを分類変数とともに使用する" (1296 ページ)
- "例 4: CLASSDATA=データセットを分類変数とともに使用する" (1299 ページ)
- "例 5: 値のマルチラベル出力形式を分類変数とともに使用する" (1302 ページ)
- "例 6: 事前に読み込まれた出力形式を分類変数とともに使用する" (1307 ページ)
- "例 7: 平均の信頼限界の計算" (1310 ページ)
- "例 8: 出力統計量を計算する" (1313 ページ)
- "例 9: 複数の変数に対して、異なる出力統計量を計算する" (1315 ページ)
- "例 10: 分類変数の欠損値を使用し、出力統計量を計算する" (1318 ページ)
- "例 11: 出力統計量を使用し、極値を識別する" (1320 ページ)
- "例 12: 出力統計量を使用し、上位 3 位の極値を識別する" (1323 ページ)

構文

CLASS *variable(s)* </ *options*>;

必須引数

variable(s)
プロシジャがデータのグループ化に使用する 1 つ以上の変数を指定します。CLASS ステートメントの変数は、*分類変数*といいます。分類変数は、数値または文字です。分類変数には連続値を含めることができますが、通常は変数の水準を定義するいくつかの不連続値が含まれます。データを分類変数別に並べ替える必要はありません。

操作 TYPES ステートメントまたは WAYS ステートメントを使用して、PROC MEANS がデータのグループ化に使用する分類変数を制御します。

ヒント 分類変数水準の数を減らすには、FORMAT ステートメントを使用して、変数値をまとめます。出力形式が複数の内部値を 1 つのフォーマットされた値にまとめると、PROC MEANS は最小内部値を出力します。

参照項目: “分類変数の使用 ” (1226 ページ)

オプション引数

ASCENDING
分類変数水準を昇順に並べ替えるように指定します。

別名 ASCEND

操作 ASCENDING と DESCENDING の両方を指定すると、PROC MEANS は警告メッセージを発行し、両方のオプションを無視します。

参照項目: “例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

DESCENDING
分類変数水準を降順で並べ替えるように指定します。

別名 DESCEND

操作 ASCENDING と DESCENDING の両方を指定すると、PROC MEANS は警告メッセージを発行し、両方のオプションを無視します。

EXCLUSIVE
ユーザー定義出力形式のすでに読み込まれた範囲にない分類変数のすべての組み合わせを分析から除外します。

要件 PRELOADFMT を指定して、分類変数出力形式を事前に読み込む必要があります。

参照項目: “例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

GROUPINTERNAL
PROC MEANS が値をグループ化して分類変数の組み合わせを作成する場合は、出力形式を分類変数に適用しないように指定します。

操作 PRELOADFMT オプションを指定すると、PROC MEANS は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ORDER=FORMATTED オプションを指定すると、PROC MEANS は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ヒント このオプションは、数値分類変数に不連続値が含まれている場合にコンピューターリソースを節約します。

参照項目: “コンピューターリソース” (2166 ページ)

MISSING
欠損値を分類変数水準に有効な値とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING を省略すると、PROC MEANS は欠損分類変数値を含むオブザベーションを分析から除外します。

参照項目: 特別な意味を持つ欠損値の説明については、*SAS プログラマガイド: エッセンシャル*を参照してください。

“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

MLF
PROC MEANS により、マルチラベル出力形式が分類変数に割り当てられている場合に指定の範囲や重複する範囲の 1 番目と 2 番目の出力形式ラベルを使用して、サブグループの組み合わせを作成できるようになります。

要件 VALUE ステートメントで PROC FORMAT と MULTILABEL オプションを使用して、マルチラベル出力形式を作成する必要があります。

操作 OUTPUT ステートメントと MLF を使用すると、分類変数にはフォーマットされた値に対応する文字列が含まれます。フォーマットされた値が内部値になるため、この変数の長さは最長の出力形式ラベルの文字数です。

MLF と ORDER=FREQ を使用すると、フォーマットされた値に使用する順序が生成されない場合があります。MLF を CLASSDATA および EXCLUSIVE と使用した場合に期待する結果が得られない場合があります。MLF 処理で各 TYPE が別個に計算されるように要求されるためです。NWAY 以外の種類には、予想よりも多くの水準が含まれています。

注 フォーマットされた値が重複する場合、1 つの内部分類変数値が複数の分類変数サブグループの組み合わせにマップします。そのため、すべてのサブグループの N 統計量の合計がデータセットのオブザベーション数(N 統計量全体)を超えます。

ヒント MLF を省略すると、PROC MEANS は 1 番目の出力形式ラベルを使用します。このアクションは、最初の外部出力形式値を使用したサブグループの組み合わせの決定に一致します。

参照項目: FORMAT プロシジャの VALUE ステートメントにおける MULTILABEL オプション“オプション引数” (885 ページ)。

“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
出力の分類変数の水準をグループ化する順序を指定します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 PRELOADFMT を使用すると、各分類変数の値の順序が、PROC FORMAT が関連するユーザー定義出力形式の値の格納に使用する順序と一致します。PROC ステートメントで CLASSDATA=オプションを使用すると、PROC MEANS は CLASSDATA=データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC MEANS はまずユーザー定義出力形式を使用して、出力を並べ替えます。PROC ステートメントで EXCLUSIVE を省略すると、PROC MEANS はユーザー定義出力形式と CLASSDATA=値の後に入力データセットの分類変数の一意の値を発生順に基づいて追加します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

参照項目: “例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。この順序は、使用している動作環境によって異なります。出力形式がクラス変数に割り当てられていない場合、デフォルトの出力形式 BEST12 が使用されます。

別名 FMT

EXTERNAL

参照項目: “例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

FREQ
最も多くのオブザベーションを含む水準が最初に表示されるように、値を度数カウントの降順で並べ替えます。

操作 分類変数の多重組み合わせに対し、PROC MEANS は個別の分類変数度数から水準の順序を決定します。

ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

参照項目: “例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)

UNFORMATTED
値をフォーマットされていない値で並べ替えます。PROC SORT と同じ順序になります。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を年代順に表示する場合に特に便利です。

別名 UNFMT

INTERNAL

デフォルト UNFORMATTED

ヒント デフォルトでは、FREQ 以外のすべての順序は昇順です。降順の順序の場合は、DESCENDING オプションを使用します。

参照項目: “分類値の並べ替え” (1227 ページ)

PRELOADFMT
分類変数のすべての出力形式がすでに読み込まれていることを示します。

要件 COMPLETETYPES、EXCLUSIVE、または ORDER=DATA を指定し、出力形式を分類変数に割り当てない限り、PRELOADFMT は影響しません。

操作 PRELOADFMT と COMPLETETYPES の組み合わせは、入力データセット内に表示されない可能性のある値を生成します。場合によっては、文字クラス変数の場合、非常に低いまたは高い sentinel 値が生成されます。これらの値は、セッションエンコーディングの有効な文字を表していない可能性があります。これらの値は、生の形式で印刷するのではなく、関連する形式を使用して印刷する必要があります。出力形式定義で OTHER=値を使用すると、フォーマットを使用するときに生の sentinel 値が印刷されないようにすることができます。

PROC TABULATE 出力を入力データセットに存在するフォーマットされた分類変数値の組み合わせに制限するには、CLASS ステートメントで EXCLUSIVE オプションを使用します。

度数がゼロの場合でもユーザー定義出力形式のすべての範囲と値を出力に含めるには、PROC ステートメントで COMPLETETYPES を使用します。

注 文字出力形式をプリロードする場合、ラベルを生成する代表的な未加工値が少なくとも 1 つ存在しなければなりません。範囲に未加工値がない場合、到達不能と判断され、エラーが発生します。たとえば、Other=と Low-High の両方の範囲を持つマルチラベル文字出力形式をプリロードすることはできません。Other は、すべてのラベルが定義された後に残る値または範囲を参照する特別なキーワードです。文字出力形式の場合、low 範囲に欠損値が含まれるため、その他の域を占める未加工値が残りません。

参照 “例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

項目:

詳細

BY ステートメントと CLASS ステートメントの比較

BY ステートメントを使用することは、PROC MEANS が各 BY グループを入力データの独立したサブセットとして要約する、CLASS ステートメントと NWAY オプションを使用することと似ています。そのため、入力データの全体の要約は使用できません。ただし、CLASS ステートメントとは異なり、BY ステートメントでは BY ステートメントの変数でデータを事前に並べ替えるかインデックス付けする必要があります。

NWAY オプションを使用すると、PROC MEANS がすべての分類変数を要約するにはメモリが不足することがあります。一部の分類変数を BY ステートメントに移動できます。最大活用するには、すでに並べ替えられた、または最も多い一意の値を含む BY ステートメントに分類変数を移動します。

CLASS ステートメントと BY ステートメントを使用して、データを BY グループ内の分類変数の水準別に分析できます。“例 3: BY ステートメントを分類変数とともに使用する” (1296 ページ)を参照してください。

分類変数の欠損値の PROC MEANS による処理方法

デフォルトでは、オブザベーションに分類変数の欠損値が含まれている場合、PROC MEANS はそのオブザベーションを分析から除外します。PROC ステートメントで MISSING オプションを指定すると、プロシジャは欠損値を分類変数の組み合わせに有効な水準とみなします。

CLASS ステートメントで MISSING オプションを指定すると、個別の分類変数に対する欠損値の受け入れを制御できます。

コンピューターリソース

PROC MEANS で使用できる一意の分類変数値の合計は、使用可能なコンピューターメモリの量によって異なります。詳細については、“コンピューターリソース” (1228 ページ)を参照してください。

GROUPINTERNAL オプションは、コンピューターパフォーマンスを向上させることができます。グループ化プロセスが分類変数の内部値に基づいているためです。数値分類変数に出力形式が割り当てられておらず、GROUPINTERNAL を指定しない場合、PROC MEANS はデフォルト出力形式、BEST12.を使用して文字列として数値をフォーマットします。PROC MEANS はこれらの数値変数を文字値別にグループ化します。これにはより多くの時間とコンピューターメモリが必要になります。

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。*"出力形式の取り消し" (SAS DATA ステップステートメント: リファレンス)*を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、*"PUT ステートメント: リスト" (SAS DATA ステップステートメント: リファレンス)*を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

FREQ ステートメント

各オブザベーションの度数を含む数値変数を指定します。

別名: FREQUENCY

構文

FREQ *variable*;

必須引数

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

度数変数の合計は、オブザベーションの合計数を表します。

注: FREQ 変数は、OUTPUT ステートメントで IDGROUP 構文を使用するときの PROC MEANS による複数の極値の識別方法に影響しません。

ID ステートメント

追加の変数を出力データセットに含めます。

参照項目: Iid-group-specification の説明については、“OUTPUT ステートメント” (2171 ページ)を参照してください。

構文

ID *variable(s)*;

必須引数

variable(s)
PROC MEANS がそのオブザベーショングループの最大値を出力データセットに含める 1 つ以上の変数を入力データセットから識別します。

操作 PROC ステートメントで IDMIN を使用して、ID 変数の最小値を出力データセットに含めます。

ヒント PROC ステートメントで PRINTIDVARS オプションを使用して、ID 変数の値を表示出力に含めます。

詳細

ID 変数の値の選択

ID ステートメントで変数を 1 つだけ指定する場合、指定されているオブザベーションに対する ID 変数の値が、入力データセットの対応するオブザベーショングループで検出される最大(最小)値となります。ID ステートメントで複数の変数を指定すると、PROC MEANS は ID ステートメントの変数を表示されている順序で処理し、最大値を選択します。PROC MEANS は最初の ID 変数の値を比較して、使用するオブザベーションをすべての ID 変数から決定します。複数のオブザベーションに同一の最大(最小) ID 値が含まれている場合、PROC MEANS は 2 番目と後続の ID 変数値を“タイブレーカー”として使用します。いかなる場合でも、すべての ID 値は、指定されている BY グループまたは種類内の分類水準に対する同一のオブザベーションから取得されます。

PROC MEANS による最大値を決定するための文字値の比較方法については、“文字変数の並べ替え順序 ” (2031 ページ)を参照してください。

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=’ ’>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限 ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

事項 label N = 'Number = ';

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: “文字定数と文字変数” (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

OUTPUT ステートメント

統計量を新しい SAS データセットに書き込みます。

ヒント: 複数の OUTPUT ステートメントを使用して、複数の OUT=データセットを作成できます。

例:
“例 8: 出力統計量を計算する” (1313 ページ)
“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1315 ページ)
“例 10: 分類変数の欠損値を使用し、出力統計量を計算する” (1318 ページ)
“例 11: 出力統計量を使用し、極値を識別する” (1320 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

構文

OUTPUT <OUT=*SAS-data-set*>
<*statistic-specifications*>
<*id-group-specifications*> <*maximum-id-specifications*>
<*minimum-id-specifications*> </ *options*>;

オプション引数

OUT=*SAS-data-set*
新しい出力データセットに名前を付けます。*SAS-data-set* が存在しない場合、PROC MEANS が作成します。OUT=を省略すると、データセットの名前は DATA*n* となります。*n* は、名前を一意のものにする最小整数です。

デフォルト DATA*n*

ヒント データセットオプションを OUT=オプションと使用できます。

statistic-keyword<(***variable-list***)><=*name(s)*>
OUT=データセットに格納する統計量を指定し、その統計量を含む 1 つ以上の変数に名前を付けます。

statistic-keyword
出力データセットに格納する統計量を指定します。

使用可能な統計量キーワードは、“統計量キーワード” (2178 ページ)にリストされています。

デフォルトでは、出力データセットの統計量は分析変数の出力形式、入力形式、ラベルを自動的に継承します。ただし、N、NMISS、SUMWGT、USS、CSS、VAR、CV、T、PROBT、PRT、SKEWNESS、KURTOSIS に対して計算された統計量は、分析変数の出力形式を継承しません。この出力形式がこれらの統計量に無効な場合があるためです(ドル出力形式、日時出力形式など)。

デフォルト　N、MEAN、STD、MIN、MAX

制限事項　*variable* と *name(s)*を省略すると、AUTONAME オプションも使用しない限り、PROC MEANS は *statistic-keyword* を単一の OUTPUT ステートメントで一度だけ使用できるようにします。

参照項目:　キーワードの定義と関連する統計量の式については、“キーワードと式” (2410 ページ)を参照してください。

“例 8: 出力統計量を計算する” (1313 ページ)

“例 9: 複数の変数に対して、異なる出力統計量を計算する” (1315 ページ)

“例 11: 出力統計量を使用し、極値を識別する” (1320 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

variable-list

統計量を出力データセットに格納する 1 つ以上の数値分析変数の名前を指定します。

デフォルト　すべての 数値分析変数

names

分析変数統計量を含む出力データセットの変数に対し 1 つ以上の名前を指定します。たとえば、最初の名前には最初の分析変数に対する統計量が含まれます。2 番目の名前には 2 番目の分析変数の統計量が含まれます。

デフォルト　分析変数名。AUTONAME を指定する場合、デフォルトは分析変数名と *statistic-keyword* の組み合わせです。*output-statistic-specification* を使用せずに CLASS ステートメントと OUTPUT ステートメントを使用すると、出力データセットには分類変数の各組み合わせに対し、N、MIN、MAX、MEAN および STD の値という 5 つのオブザベーションが含まれます。WEIGHT ステートメント、または VAR ステートメントで WEIGHT オプションを使用すると、出力データセットには分類変数の各組み合わせに対する重みの合計(SUMWGT)を含むオブザベーションも含まれます。

操作　*variable-list* を指定する場合、PROC MEANS は分析変数を指定する順序を使用して、統計量を出力データセット変数に格納します。

ヒント　AUTONAME オプションを使用すると、PROC MEANS は複数の変数と統計量に対し、一意の名前を生成します。

参照項目:　“例 8: 出力統計量を計算する” (1313 ページ)

IDGROUP (<**MIN** (***variable-list-1***) | **MAX** (***variable-list-1***) <...**MIN** (***variable-list-n***) | **MAX** (***variable-list-n***) >> <**MISSING**> <**OBS**> <**LAST**> OUT <[*n*]> (*id-variable-list*)=<*names*>)

機能を組み合わせ、ID ステートメント、PROC ステートメントの IDMIN オプション、OUTPUT ステートメントの MAXID オプションと MINID オプションを拡張して、複数の極値を識別する OUT=データセットを作成します。

LAST

OUT=データセットに最後のオブザベーション(または *n* が指定されている場合は最後の *n* オブザベーション)からの値が含まれるように指定します。

LAST を指定しない場合、OUT=データセットには、最初のオブザベーション(または *n* が指定されている場合は最初の *n* オブザベーション)からの値が含まれます。OUT=データセットには、複数のオブザベーションが含まれている場合があります。これは、最後の(最初の)オブザベーションの値に加え、OUT=データセット値には分類変数値の組み合わせによって定義される各サブグループ水準の最後の(最初の)オブザベーションからの値が含まれるためです。

操作 MIN または MAX を指定し、複数のオブザベーションに同じ極値が含まれている場合、PROC MEANS はオブザベーション番号を使用して、OUT=データセットに保存するオブザベーションを解決します。LAST を指定すると、PROC MEANS は関係の解決に後のオブザベーションを使用します。LAST を使用しない場合、PROC MEANS は関係の解決に前のオブザベーションを使用します。

MAX(*variable-list*)

選択基準を指定し、*variable-list* で指定された 1 つ以上の入力データセット変数の最大極値を決定します。

複数の選択変数を指定すると、*n* 極値の選択に対するオブザベーションの並べ替えが、PROC SORT が複数の BY 変数を含むデータを並べ替えるのと同じ方法で行われます。PROC MEANS は、変数値を単一のキーにまとめます。MAX (*variable-list*)選択基準は、PROC SORT と、BY ステートメントで DESCENDING オプションを選択することに類似しています。

デフォルト MIN または MAX を指定しない場合、PROC MEANS はオブザベーション番号をオブザベーションを出力するための選択基準として使用します。

制限事項 矛盾した基準を指定すると、PROC MEANS は最初の選択基準のみ使用します。

操作 複数のオブザベーションですべての MIN 変数または MAX 変数に同じ極値が含まれている場合、PROC MEANS はオブザベーション番号を使用して、出力に書き込むオブザベーションを解決します。デフォルトでは、PROC MEANS は最初のオブザベーションを使用して、関係を解決します。ただし、LAST オプションを指定すると、PROC MEANS は最後のオブザベーションを使用して関係を解決します。

MIN(*variable-list*)

選択基準を指定し、*variable-list* で指定された 1 つ以上の入力データセット変数の最小極値を決定します。

複数の選択変数を指定すると、*n* 極値の選択に対するオブザベーションの並べ替えが、PROC SORT が複数の BY 変数を含むデータを並べ替えるのと同じ方法で行われます。PROC MEANS は、変数値を単一のキーにまとめます。

デフォルト　MIN または MAX を指定しない場合、PROC MEANS はオブザベーション番号をオブザベーションを出力するための選択基準として使用します。

制限事項　矛盾した基準を指定すると、PROC MEANS は最初の選択基準のみ使用します。

操作　複数のオブザベーションですべての MIN 変数または MAX 変数に同じ極値が含まれている場合、PROC MEANS はオブザベーション番号を使用して、出力に書き込むオブザベーションを解決します。デフォルトでは、PROC MEANS は最初のオブザベーションを使用して、関係を解決します。ただし、LAST オプションを指定すると、PROC MEANS は最後のオブザベーションを使用して関係を解決します。

MISSING
欠損値が選択基準で使用されるように指定します。

別名　MISS

OBS
極値が見つかった入力データセットにオブザベーション数を含む OUT=データセットに_OBS_変数を含めます。

操作　WHERE 処理を使用すると、_OBS_の値は、入力データセットのオブザベーションの場所に一致しないことがあります。

n]を使用して複数の極値を出力に書き込む場合、PROC MEANS は *n*_OBS_変数を作成し、接尾辞 *n* を使用して変数名を作成します。*n* は、1 から *n* までの連続した整数です。

[*n*]
OUT=データセットに含める *id-variable-list* の各変数に対する極値の数を指定します。PROC MEANS は *n* の新規変数を作成し、接尾辞_*n* を使用して変数名を作成します。*n* は、1 から *n* までの連続した整数です。

デフォルトでは、PROC MEANS は、要求された各種類の水準ごとに 1 つの極値を決定します。*n* が 1 より大きい場合、*n* 極値が各種類の水準ごとに生成されます。*n* が 1 より大きく、極値選択を要求する場合、時間計算量は $O(T * N \log_2 n)$になります。Tは要求された種類の数であり、Nは入力データセットのオブザベーションの数です。比較すると、データセット全体をグループ化するための時間計算量は $O(N \log_2 N)$になります。

デフォルト　1

範囲　1 から 100 までの整数

例　たとえば、変数ごとに 2 つの最小極値を生成するには、次を使用します。

```
idgroup(min(x) out[2](x y z)=MinX MinY MinZ);
```

OUT=データセットには、変数 MinX_1、MinX_2、MinY_1、MinY_2、MinZ_1、MinZ_2 が含まれます。

(*id-variable-list*)
PROC MEANS がその値を OUT=データセットに含める 1 つ以上の入力データセット変数を識別します。PROC MEANS は、指定する選択基準(MIN、MAX および LAST)によって生成するオブザベーションを決定します。

別名 IDGRP

要件 まず MIN | MAX 選択基準を選択し、OUT(*id-variable-list*)=をサブオプション MISSING、OBS、LAST の後に指定する必要があります。

ヒント ID ステートメントの動作を模倣するには、*id-group-specification* を使用し、また OUTPUT ステートメントで *maximum-id-specification* または *minimum-id-specification* を使用することができます。

出力データセットに極値とその他の ID 変数を含める場合、それらを *id-variable-list* に含めると個別の統計量を求めるよりも効率的です。たとえば、ステートメント output idgrp(max(x) out(x a b)=);は、ステートメント output idgrp(max(x) out(a b)=) max(x)=;より効率的です。

参照項目: “例 8: 出力統計量を計算する” (1313 ページ)

“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

names
OUT=データセットの変数に対し 1 つ以上の名前を指定します。

デフォルト *name* を省略すると、PROC MEANS は *id-variable-list* の変数名を使用します。

ヒント AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注意
IDGROUP 構文を使用して、同じ名前の出力変数を作成できます。 このアクションが発生するのは、出力データセットに最初の変数が表示される場合だけです。AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注: 指定する新しい変数名が分析変数と ID 変数の組み合わせよりも少ない場合、その他の出力変数は、PROC MEANS が新しい変数名のリストを使い果たすとすぐに ID 変数の対応する名前を使用します。

別名 IDGRP

MAXID <(***variable-1*** <(***id-variable-list-1***)> <...***variable-n*** <(***id-variable-list-n***)>>)>
= *names*
1 つ以上の ID 変数が分析変数の最大値と関連付けられるように指定します。

variable
PROC MEANS が最大値を決定する数値分析変数を識別します。PROC MEANS は、1 つの変数に対して複数の最大値を決定できます。これは、全体の最大値に加え、分類変数値の組み合わせによって定義されるサブグループ水準にも最大値が含まれているためです。

ヒント　ID ステートメントを使用し、*variable* を省略すると、PROC MEANS はすべての分析変数を使用します。

id-variable-list
値が分析変数の最大値を含むオブザベーションを識別する 1 つ以上の変数を識別します。

デフォルト　ID ステートメント変数

names
各分析変数の最大値と関連付けられている ID 変数の値を含む新しい変数の名前を指定します。

注　複数のオブザベーションに分類水準内の最大値が含まれている場合、PROC MEANS は出力データセットのこれらのオブザベーションのうち最初のものだけに対する ID 変数の値を保存します。

ヒント　ID ステートメントを使用し、*variable* と *id-variable* を省略すると、PROC MEANS はすべての ID ステートメント変数と各分類変数を関連付けます。このため、各分析変数に対し、出力データセットで作成される変数の数は ID ステートメントで指定する変数の数と同じです。

AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

参照項目:　“例 11: 出力統計量を使用し、極値を識別する” (1320 ページ)

注意
MAXID 構文を使用して、同じ名前の出力変数を作成できます。 このアクションが発生するのは、出力データセットに最初の変数が表示される場合だけです。AUTONAME オプションを使用して、名前の競合問題を自動的に解決します。

注: 指定する新しい変数名が分析変数と ID 変数の組み合わせよりも少ない場合、その他の出力変数は、PROC MEANS が新しい変数名のリストを使い果たすとすぐに ID 変数の対応する名前を使用します。

MINID<(*variable-1 <(id-variable-list-1)> <...variable-n <(id-variable-list-n)>>)> = *name(s)*
maximum-id-specification の説明を参照してください。このオプションは、PROC MEANS が最大値ではなく最小値を決定することを除けば、まったく同じように作動します。

MINID は明示的な変数リストなしで使用される場合、次のさらに詳細な IDGROUP 構文例と類似しています。

IDGRP(min(x) missing out(id_variable)=idminx) idgrp(min(y) missing out(id_variable)=idminy)

1 つ以上の分類変数に欠損値が含まれている場合、id_variable 値は、MIN 統計量に対する値を含むオブザベーションではなく、欠損値を含むオブザベーションに対応します。

スラッシュ区切り文字の後に使用するオプション

AUTOLABEL
PROC MEANS が統計量名を変数ラベルの終わりに追加することを指定します。分析変数にラベルがない場合、PROC MEANS は統計量名を分析変数名に追加してラベルを作成します。

参照項目: “例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

AUTONAME
OUTPUT ステートメントで変数名を割り当てない場合、PROC MEANS は出力統計量に対して一意の変数名を作成することを指定します。このアクションは、統計量の派生元入力変数名の終わりに *statistic-keyword* を追加して実行されます。

たとえば、ステートメント output min(x)=/autoname;では、出力データセットに x_Min 変数が生成されます。

AUTONAME は、SAS の内部メカニズムを使用して出力データセットの変数名の競合問題を自動的に解決します。変数が重複しても、エラーは発生しません。

その結果、ステートメント output min(x)= min(x)=/autoname;では、出力データセットに 2 つの変数 x_Min と x_Min2 が生成されます。

新しい変数名が 32 文字を超えると、variable-name 部分が切り捨てられます。

参照項目: “例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

KEEPLEN
出力データセットの統計量は、それらの派生に使用される分析変数の長さを継承することを指定します。

注意
分析変数の長さが原因で PROC MEANS が統計量の値を切り捨てるか丸めると、数値精度を永久に失います。ただし、統計量の精度は入力の精度と一致します。

LEVELS
出力データセットに_LEVEL_という名前の変数を含めます。この変数には、分類変数の値(_TYPE_変数の値)の一意の組み合わせを示す 1 から *n* までの値が含まれます。

参照項目: “出力データセット” (1290 ページ)

“例 8: 出力統計量を計算する” (1313 ページ)

NOINHERIT
統計量を含む出力データセットの変数は、それらを派生するときに使用される分析変数の属性(ラベルと出力形式)を継承しないことを指定します。

操作 オプションが使用されない場合(暗黙の INHERIT)、統計量は入力分析変数の属性、ラベル、出力形式を継承します。OUTPUT ステートメントで INHERIT オプションが使用されると、統計量は入力分析変数の長さ、ラベルと出力形式を継承します。

ヒント デフォルトでは、出力データセットに、各分析変数と、 N、MIN、MAX、MEAN、STDDEV を含む 5 つのオブザベーションに対する 1 つの出力変数が含まれます。NOINHERIT を指定しない場合、この変数は分析変数の出力形式を継承します。この出力形式は N 統計量に対して無効である場合があります(日時出力形式など)。

WAYS
出力データセットに_WAY_という名前の変数を含めます。この変数には、PROC MEANS が TYPE 値を作成するために組み合わせる分類変数の数を指定する、1 から分類変数の最大数までの値が 1 つ含まれます。

参照項目: “出力データセット” (1290 ページ)

“WAYS ステートメント” (2183 ページ)

“例 8: 出力統計量を計算する” (1313 ページ)

統計量キーワード

CLM
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

CSS
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

KURTOSIS
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

別名 KURT

LCLM
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

MAX
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

MEAN
記述統計量キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

MIN
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

MODE
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

N
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

NMISS
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

RANGE
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

SKEWNESS
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

別名 SKEW

STDDEV
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

別名 STD

STDERR
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

SUM
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

SUMWGT
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

UCLM
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

USS
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

VAR
記述統計量キーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

MEDIAN
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

別名 P50

P1
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P5
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P10
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P20
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P40
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P70
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

Q1
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

別名 P25

Q3
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

別名 P75

P90
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P95
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P99
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P30
分位点統計量のキーワードです。 詳細については、“*statistic-keyword<(variable-list)><=name(s)>*” (2171 ページ)を参照してください。

P60
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

P80
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

QRANGE
分位点統計量のキーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

PROBT
仮説検定キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

別名 PBT

T
仮説検定キーワードです。 詳細については、*“statistic-keyword<(variable-list)><=name(s)>”* (2171 ページ)を参照してください。

TYPES ステートメント

生成する分類変数の可能な組み合わせを決定します。

要件 CLASS ステートメント

参照項目: “出力データセット” (1290 ページ)

例: “例 2: 分類変数を使用した記述統計量の計算” (1294 ページ)
“例 5: 値のマルチラベル出力形式を分類変数とともに使用する” (1302 ページ)
“例 12: 出力統計量を使用し、上位 3 位の極値を識別する” (1323 ページ)

構文

TYPES *request(s)*;

必須引数

request(s)
PROC MEANS が種類の作成に使用する分類変数の2^kの組み合わせを指定します。このkは分類変数の数です。要求には、1 つの分類変数名、アスタリスクまたは()で区切られた複数の分類変数名が含まれます。

分類変数の組み合わせをすばやく要求するには、複数の変数をかっこで囲み、他の変数または変数の組み合わせを結合してグループ化構文を使用します。たとえば、次のステートメントはグループ化構文を示しています。

要求	等しい
types A*(B C);	types A*B A*C;
types (A B)*(C D);	types A*C A*D B*C B*D;
types (A B C)*D;	types A*D B*D C*D;

操作 CLASSDATA=オプションは、NWAY 種類に制約を設定します。PROC MEANS は、他のすべての種類を結果の NWAY 種類から派生したように生成します。

ヒント ( )を使用して全体合計を要求します(_TYPE_=0)。

出力データセットですべての種類が必要でない場合、WHERE 句をデータセットに適用するのではなく、TYPES ステートメントを使用して特定のサブタイプを指定します。これを行うことにより、時間とコンピューターメモリが節約されます。

詳細

出力における分析の順序

分析は、PROC MEANS によって計算される、_TYPE_変数の値が増える順序で出力に書き込まれます。_TYPE_変数には、分類変数の各組み合わせに対し一意の値が含まれています。値は TYPES ステートメントではなく、CLASS ステートメントの指定方法によって決定されます。そのため、

```
class A B C;
types (A B)*C;
```

を指定すると、B*C 分析(_TYPE_=3)が最初に書き込まれ、A*C 分析(_TYPE_=5)が後に続きます。ただし、

```
class B A C;
types (A B)*C;
```

を指定すると、A*C 分析が最初に実行されます。

_TYPE_変数は、出力データセットが要求されなくても計算されます。_TYPE_変数の詳細については、“出力データセット” (1290 ページ)を参照してください。

VAR ステートメント

分析変数と結果での順序を識別します。

デフォルト: VAR ステートメントを省略すると、PROC SUMMARY はオブザベーションの単純なカウントを生成します。PROC MEANS はその他のステートメントにリストされていないすべての数値変数を分析しようとします。

操作: PROC SUMMARY ステートメントで統計量を指定し、VAR ステートメントが省略された場合、または数値変数が OUTPUT ステートメントの統計量と関連付けられていない場合、PROC SUMMARY は処理を停止し、エラーメッセージが SAS ログに書き込まれます。

注: VAR ステートメントの説明については、PROC MEANS の VAR ステートメントを参照してください。

参照項目: 構文の詳細については、“VAR ステートメント” (1275 ページ)を参照してください。

WAYS ステートメント

分類変数の一意の組み合わせを作成するための方法の数を指定します。

ヒント: TYPES ステートメントを使用して、分類変数の追加の組み合わせを指定します。

例: “例 6: 事前に読み込まれた出力形式を分類変数とともに使用する” (1307 ページ)

構文

WAYS *list*;

必須引数

list
分類変数のすべての一意の組み合わせを作成するために組み合わせる分類変数の数を定義する 1 つ以上の整数を指定します。たとえば、可能なすべてのペアに対して 2 を指定し、可能なすべてのトリプルに対して 3 を指定できます。*list* は次の方法で指定できます。

- *m*
- *m1 m2 ... mn*
- *m1,m2,...,mn*
- *m* TO *n* <BY *increment*>
- *m1,m2* TO *m3* <BY *increment*>, *m4*

範囲 0 から分類変数の最大数

参照項目: WAYS オプション

例 次のコードは、分類変数 A、B、C に対する 2 次元の種類を作成する例です。この WAYS ステートメントは、TYPES ステートメントで a*b、a*c、b*c を指定する場合と同じです。

```
class A B C ; ways 2;
```

WEIGHT ステートメント

統計量計算の分析変数に対し重みを指定します。WEIGHT ステートメントと FREQ ステートメントは、両方のステートメントをサポートする任意のプロシジャの同一ステップ内で使用できます。

別名: WGT

制限事項: 重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。
歪度と尖度は、WEIGHT ステートメントと使用できません。

構文

WEIGHT *variable*;

必須引数

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

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

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

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

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

注意

単一の極値重み値により、不正確な結果が生成される可能性があります。 1 つの(唯一の)重み値がその他の重み値より桁違いに大きい場合(49 重み値が 1、1 重み値が 1×10^{14} など)、特定の統計量は可能な正しい制限内にないことがあります。影響を受ける統計量は、2 次モーメント(標準偏差、修正平方和、分散、平均値の標準誤差など)に基づいています。特定の状況下では、警告は SAS ログに書き込まれません。

プロシジャでは、w_iを WEIGHT 変数の値で置換します。これは、“キーワードと式” (2410 ページ)に記載されています。

例

WEIGHT ステートメントをサポートするプロシジャ

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

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

重み付き統計量の計算

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

WEIGHT ステートメントを使用してモーメントを計算することで、*i* 番目のオブザベーションにσ^2/w_iに等しい分散があることを想定します。VARDEF=DF (デフォルト)を指定した場合に計算される分散は、σ^2の重み付き最小二乗推定値になります。同様に、計算された標準偏差は σ の推定値になります。計算された分散は、*i* 番目のオ

ブザベーションの分散の推定値ではないことに注意してください。これは、その分散がオブザベーションの重み(オブザベーションごとに異なる)にかかわるためです。

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

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

重み付き統計量の使用

WEIGHT ステートメントの例として、30cm 幅の対象物のサイズを推定するよう 20 人に求めたとします。対象物からの距離が異なる場所にそれぞれを配置します。対象物からの距離が増加するにつれて、推定値の精度は低下します。SAS データセット SIZE には、メートル単位の各距離(Distance)におけるセンチメートル単位の推定値(ObjectSize)、および各推定値の精度(Precision)が含まれます。最大偏差(20cm の過大見積もり)が最長距離(対象物から 7.5 メートル)で発生していることに注意してください。精度(1/Distance)として、対象物に近い位置での推定値ほど重みを増やし、遠い位置での推定値ほど重みを減らします。PROC MEANS ステップでは、重みを無視して、物体サイズの平均推定値が計算されます。WEIGHT 変数がなければ、PROC MEANS では、すべてのオブザベーションに対してデフォルトの重み 1 が使用されます。したがって、すべての距離の物体サイズ推定値に、等しい重みが与えられます。物体サイズの平均推定値は、実際のサイズを 3.55cm 上回ります。

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

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

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

出力

Unweighted Analysis of the SIZE Data Set

The MEANS Procedure

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

PROC MEANS で精度のある WEIGHT ステートメントを使用

次の 2 つの PROC MEANS ステップでは、WEIGHT ステートメントで精度 (Precision)を使用し、値の異なる VARDEF=オプションの使用効果を示します。最初の PROC ステップでは、分散と標準偏差を含む出力データセットが作成されます。遠い位置での推定値の重みを減らすと、物体サイズの重み付き平均推定値が実際のサイズに近づきます。*i* 番目のオブザベーションの分散が$var(x_i) = \sigma^2/w_i$で、w_iは *i* 番目のオブザベーションの重みです。最初の PROC MEANS ステップでは、計算された分散はσ^2の推定値です。2 番目の PROC MEANS ステップでは、計算された分散は$(n - 1/n)\,\sigma^2/\bar{w}$の推定値です。このとき、$\bar{w}$は平均重みです。n が大きい場合、この値は平均重み付きのオブザベーションの分散の概算推定値です。

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

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

出力

Weighted Analysis Using Default VARDEF=DF

The MEANS Procedure

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

Weighted Analysis Using VARDEF=WEIGHT

The MEANS Procedure

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

各オブザベーション値の重み付き分散と重み付き標準偏差を使用してデータセットを作成する

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

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

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

出力

Weighted Statistics

Distance=1.5

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

Distance=3

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

Distance=4.5

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

Distance=6

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

Distance=7.5

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

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

61

TABULATE プロシジャ

概要: TABULATE プロシジャ

TABULATE プロシジャの機能

TABULATE プロシジャは、データセットの一部またはすべての変数を使用して、記述統計量を表形式で表示します。単純なものから高度にカスタマイズされたものまで、さまざまなテーブルを作成できます。

PROC TABULATE は、MEANS、FREQ、REPORT などのその他の記述統計プロシジャによって計算される統計量と同じ統計量の多くを計算します。PROC TABULATE には、次のような機能があります。

- 表レポートを作成するための簡単で強力な方法を提供する
- 変数値の分類および変数間の階層関係の構築を柔軟に行える
- 変数とプロシジャ生成統計量のラベル付けおよびフォーマットを行うための手法を提供する
- ACCESSIBLETABLE システムオプションとともに使用すると、アクセシブルな出力テーブルを作成する機能を提供します。PROC TABULATE は、テーブルにキャプションを追加する機能も提供します。

  PROC TABULATE を使用してアクセス可能なテーブルを作成する方法の詳細については、“Creating Accessible Tables” (*Creating Accessible SAS Viya Platform Output Using ODS and ODS Graphics*)を参照してください。

単純なテーブル

次の出力に、PROC TABULATE によって生成された単純なテーブルを示します。データセット“ENERGY” (2488 ページ)には、米国のリージョン、Northeast (1)、West (4)の各州における、2 種類の顧客、住宅と企業のエネルギー支出に関するデータが含まれています。テーブルでは、リージョン区分内の州に対する支出が合計されます。RTS オプションにより、列ヘッダーをハイフンを利用せずに表示するための十分なスペースが提供されます。

```
proc tabulate data=energy;
  class region division type;
  var expenditures;
  table region*division, type*expenditures /
       rts=20;
run;
```

アウトプット 61.1　PROC TABULATE によって生成される単純なテーブル

		Type	
		1	2
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
1	1	7477.00	5129.00
	2	19379.00	15078.00
4	3	5476.00	4729.00
	4	13959.00	12619.00

複合テーブル

次の出力は、アウトプット 61.301 (2195 ページ)の作成に使用されたものと同じデータセットを使用した、より複雑なテーブルです。このレポートを作成するステートメントは次のことを実施します。

- 列ヘッダーと行ヘッダーをカスタマイズする
- すべてのテーブルセルに 出力形式を適用する
- 住居顧客と企業顧客の支出を合計する
- 各区分の小計を計算する
- すべてのリージョンの 合計を計算する

このレポートを作成するプログラムの説明については、“例 6: 共通分類変数 ALL を使用した情報の要約” (2305 ページ)を参照してください。

アウトプット 61.2　PROC TABULATE によって生成される複合テーブル

Energy Expenditures for Each Region
(millions of dollars)

		Customer Base		All Customers
		Residential Customers	Business Customers	
Region	Division			
Northeast	New England	7,477	5,129	12,606
	Middle Atlantic	19,379	15,078	34,457
	Subtotal	26,856	20,207	47,063
West	Division			
	Mountain	5,476	4,729	10,205
	Pacific	13,959	12,619	26,578
	Subtotal	19,435	17,348	36,783
Total for All Regions		$46,291	$37,555	$83,846

PROC TABULATE と Output Delivery System

次の出力は、HTML5 で作成されるテーブルを示します。Output Delivery System を PROC TABULATE と使用して、カスタマイズした出力を HTML5、Rich Text Format (RTF)、Word、Portable Document Format (PDF)などの出力形式で作成できます。このテーブルを生成するプログラムの説明については、“例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)を参照してください。

アウトプット 61.3　PROC TABULATE によって生成される HTML テーブル

Energy Expenditures
(millions of dollars)

Region=Northeast

Region by Division by Type		Type		Total
		Residential Customers	Business Customers	
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063

概念: TABULATE プロシジャ

用語: TABULATE プロシジャ

次の図で、PROC TABULATE の説明で一般に使用される用語の一部について説明します。

図 61.1 PROC TABULATE テーブルの各部

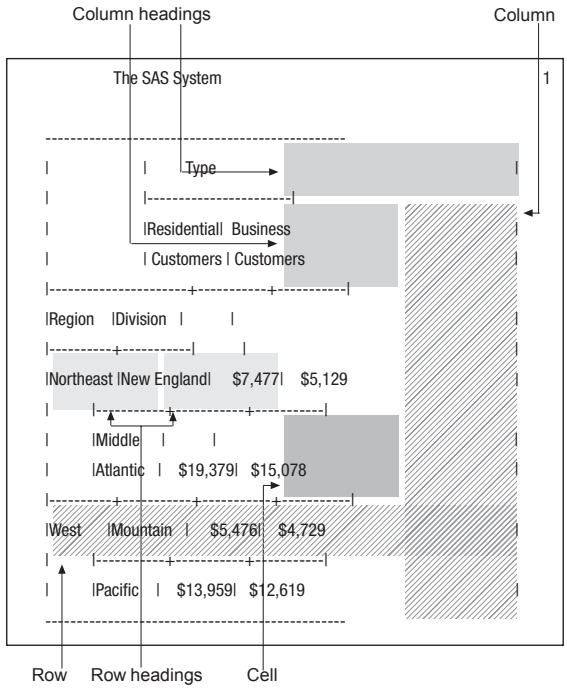
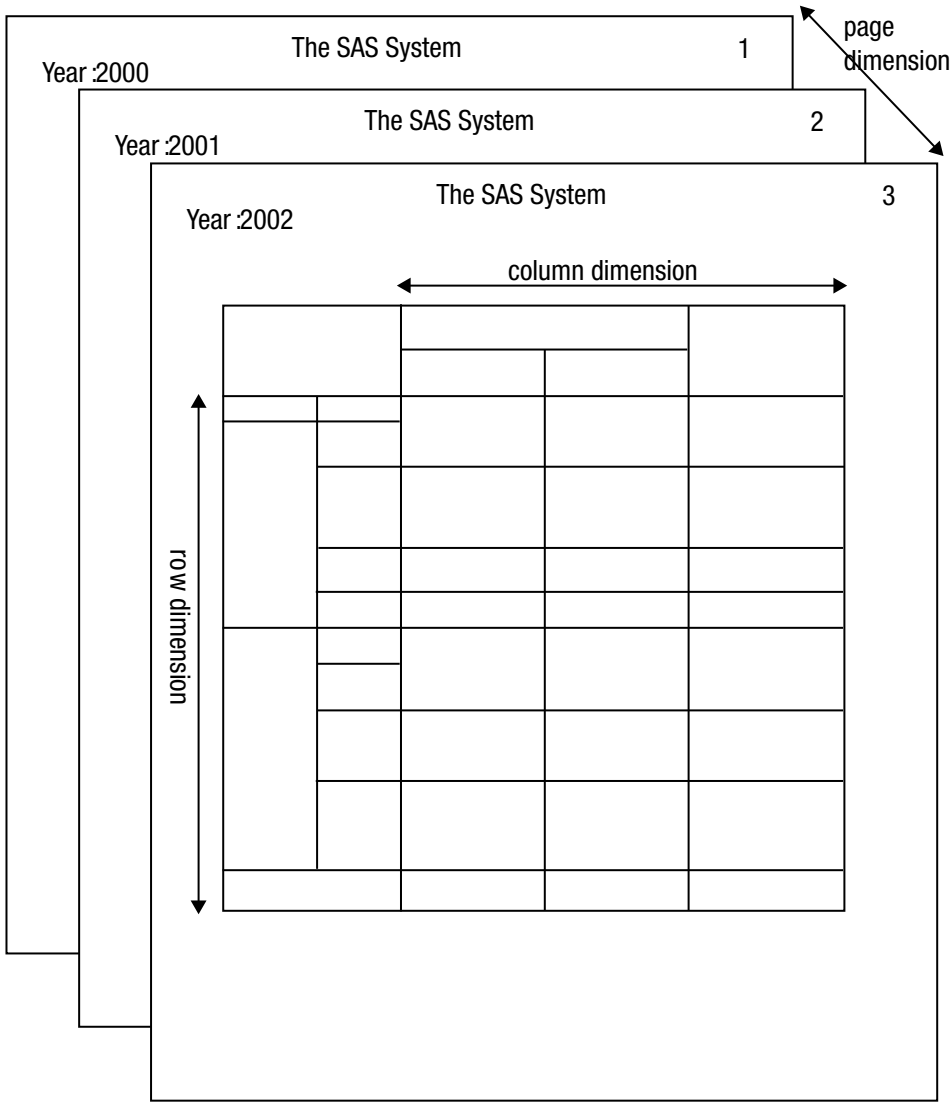


図 61.2　PROC TABULATE テーブルのディメンション



また、次の用語が PROC TABULATE の説明で頻出します。

カテゴリ
分類変数の一意の値の組み合わせ。TABULATE プロシジャは、データセットのオブザベーションに存在する値の一意の組み合わせに対してそれぞれ別のカテゴリを作成します。PROC TABULATE によって作成されるカテゴリはそれぞれ、カテゴリを説明するページ、行、列がクロスするテーブルの 1 つ以上のセルによって表されます。

図 61.59 (2197 ページ)のテーブルには、Region、Division、Type の 3 つのクラス変数が含まれています。これらの分類変数は、次のテーブルに表示される 8 つのカテゴリを形成します。(便宜上、カテゴリはフォーマットされた値について説明されます)。

表 61.1 3 つの分類変数から作成されるカテゴリ

リージョン	Division	タイプ
Northeast	New England	Residential Customers
Northeast	New England	Business Customers
Northeast	Middle Atlantic	Residential Customers
Northeast	Middle Atlantic	Business Customers
West	Mountain	Residential Customers
West	Mountain	Business Customers
West	Pacific	Residential Customers
West	Pacific	Business Customers

継続メッセージ
テーブルが複数の物理ページに及ぶ場合にテーブル下に表示されるテキスト。

ネストされた変数
値が別の変数の各値とともにテーブルに表示される変数。

図 61.59 (2197 ページ)では、Division が Region の下にネストされています。

ページ次元テキスト
テーブルにページ次元がある場合にテーブル上に表示されるテキスト。ただし、TABLE ステートメントで BOX=_PAGE_を指定すると、テーブル上に表示されるテキストがボックスに表示されます。図 61.60 (2198 ページ)では、値に続く単語**Year:**がページ次元テキストです。

ページ次元テキストには、スタイルがあります。デフォルトのスタイルは *Beforecaption* です。スタイル使用の詳細については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

サブテーブル
1 つ以上の次元に連結要素が含まれている場合に TABLE ステートメントの各次元から単一の要素をクロスすることによって生成されるセルグループ。

図 61.59 (2197 ページ)にはサブテーブルは含まれていません。複数のサブテーブルで構成されるテーブルの説明については、"例 12: 分母定義を使用し、基本的な度数カウントとパーセントを表示する" (2328 ページ)を参照してください。

入力データセットのスレッド処理

THREADED オプションにより、入力データセットの並列処理の有効化または無効化が可能になります。スレッド処理により、処理操作における一定の並列処理が達成

されます。この並列処理の目的は、所定の操作を完了するための処理時間を削減し、それによって追加 CPU リソースのコストを抑えることです。詳細については、並列処理のサポートを参照してください。

SAS システムオプション CPUCOUNT=の値はスレッド化された並べ替えのパフォーマンスに影響します。CPUCOUNT=は、スレッド化されたプロシジャで使用できるシステム CPU の数を示します。

詳細については、“THREADS システムオプション” (*SAS システムオプション: リファレンス*) および、“CPUCOUNT= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

計算された統計は、オブザベーションが処理される順序に応じてわずかに異なる場合があります。このような変動は、浮動小数点演算によって導入される数値エラーによるものであり、その結果は概算であり、正確ではないと見なす必要があります。オブザベーション処理の順序は、マルチスレッド処理または並列処理の非決定論的影響を受ける可能性があります。処理の順序は、ACCESS エンジンを介してクエリ結果を提供する DBMS などのデータソースによって 生成されるオブザベーションの一貫性のない、または非決定論的な並べ替えによっても 影響を受ける可能性があります。詳細については、“Base SAS のスレッド処理” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

構文: TABULATE プロシジャ

要件: PROC TABULATE プロシジャステップでは、最低 1 つの TABLE ステートメントが必要です。

TABLE ステートメントで使用される変数によって、CLASS ステートメント、VAR ステートメント、またはその両方が必要です。

サポート: スレッド処理

アクセシビリティの注: ACCESSIBLECHECK および ACCESSIBLETABLE システムオプションを使用して、アクセス可能なテーブルをチェックおよび作成できます。アクセス可能な PROC TABULATE テーブルの作成については、“Creating Accessible Tables” (*Creating Accessible SAS Viya Platform Output Using ODS and ODS Graphics*)を参照してください。

ヒント: ステートメント ATTRIB、FORMAT、LABEL および WHERE は、PROC TABULATE プロシジャと使用できます。

発生する In-Database 処理に対し、データは SAS In-Database 処理用に適切に構成された DBMS のサポートされているバージョン内に存在する必要があります。詳細については、“PROC TABULATE の In-Database 処理” (2278 ページ)を参照してください。

PROC TABULATE <*options*> <STYLE=*style-override(s)*>;

ATTRIB *variable-list(s) attribute-list(s)*;

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

CLASS *variable(s)* </ *options*> <STYLE=*style-override(s)* >;

CLASSLEV *variable(s)* </ STYLE=*style-override(s)*>;

FORMAT *variable-1* <...*variable-n*> <*format*> <DEFAULT=*default-format*>;

FREQ *variable*;

KEYLABEL *keyword-1*='*description-1*' <*keyword-2*='*description-2*' ...>;

KEYWORD *keyword(s)* </ STYLE=*style-override(s)*>;

LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;

TABLE <<*page-expression*,> *row-expression*,>
column-expression </ *table-options*>;

VAR *analysis-variable(s)* </ *options*> <STYLE=*style-override(s)*>;

WEIGHT *variable*;

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク	例
“PROC TABULATE ステートメント”	記述統計量を表形式で表示します	Ex. 1, Ex. 2, Ex. 3, Ex. 4, Ex. 6, Ex. 11, Ex. 13, Ex. 14
“BY ステートメント”	BY グループごとに別のテーブルを作成します	
“CLASS ステートメント”	入力データセットの変数を分類変数として識別します	Ex. 3, Ex. 4
“CLASSLEV ステートメント”	分類変数の水準値ヘッダーのスタイルを指定します	Ex. 13, Ex. 14
“FREQ ステートメント”	値が各オブザベーションの度数を表す入力データセットの変数を識別します	
“KEYLABEL ステートメント”	キーワードのラベルを指定します	
“KEYWORD ステートメント”	キーワードヘッダーのスタイルを指定します	Ex. 13
“TABLE ステートメント”	作成するテーブルを記述します	Ex. 1, Ex. 3, Ex. 4, Ex. 5, Ex. 6, Ex. 7, Ex. 8, Ex. 9, Ex. 10, Ex. 11, Ex. 12, Ex. 13, Ex. 14
“VAR ステートメント”	入力データセットの変数を分析変数として識別します	Ex. 13
“WEIGHT ステートメント”	値が統計量計算で各オブザベーションに重みを付ける入力データセットの変数を識別します	

PROC TABULATE ステートメント

記述統計量を表形式で表示します。

構文

PROC TABULATE <*options*>;

オプション引数の要約

CONTENTS=*link-text* <#BYLINE> <#BYVAL> < #BYVAR>
: 目次のエントリ用のテキストを指定します。

DATA=*SAS-data-set*
: 入力データセットを指定します。

NOTHREADS
: 入力データセットの並列処理を無効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします。

OUT=*SAS-data-set*
: 出力データセットを指定します。

THREADS
: 入力データセットの並列処理を有効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします。

TRAP
: データ処理中に通常の SAS FPE 処理によって復旧できない浮動小数点例外の復旧を可能にします。

Control the statistical analysis

ALPHA=*value*
: 信頼限界に対する信頼水準を指定します。

EXCLNPWGT
: 非正の重みが付いたオブザベーションを除外します。

QMARKERS=*number*
: P^2 分位点推定方法に使用するサンプルサイズを指定します。

QMETHOD=OS | P2
: 分位点推定方法を指定します。

QNTLDEF=1 | 2 | 3 | 4 | 5
: 分位点を計算する算術定義を指定します。

VARDEF=DF | N | WDF | WEIGHT
: 分散と標準偏差の計算に使用する分散の除数を指定します。

Customize the appearance of the table

FORMAT=*format-name*
: テーブルの各セルに対しデフォルトの出力形式を指定します。

FORMCHAR= <(*position(s)*)>='*formatting-character(s)*'
: テーブルの外枠と分割線の構成に使用する文字を定義します。

NOSEPS
: 行タイトルとテーブルの本文から水平区切り線を削除します。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
: 分類変数の値を指定した順序に従って並べ替えます。

STYLE=*style-override(s)*
: テーブルの特定の領域に適用する 1 つ以上のスタイルの上書きを指定します。

Identify categories of data that are of interest

CLASSDATA=*SAS-data-set*
: テーブルと出力データセットに含める分類変数の値の組み合わせを含む 2 次データセットを指定します。

EXCLUSIVE
: テーブルと出力データセットから、CLASSDATA=データセットにない分類変数値のすべての組み合わせを除外します。

MISSING
: 欠損値を分類変数の有効値とみなします。

オプション引数

ALPHA=*value*
: 平均値に対する信頼限界を計算する信頼水準を指定します。信頼限界のパーセントは(1–*value*)×100 です。たとえば、ALPHA=.05 の場合、信頼限界は 95%となります。

デフォルト .05

範囲 0-1

操作 信頼限界を計算するには、*statistic-keyword* LCLM または UCLM を指定します。

CLASSDATA=*SAS-data-set*
: 出力に表示が必要な分類変数の値の組み合わせを含むデータセットを指定します。クラス変数の値の任意の組み合わせが各テーブルまたは出力データセットに表示され、次の基準を満たしている場合、度数はゼロになります。

1 CLASSDATA=データセットでは発生します
2 が、入力データセットでは発生しません

制限事項 CLASSDATA=データセットにすべての分類変数を含める必要があります。このデータの種類と出力形式は、入力データセットの対応する分類変数と一致する必要があります。

操作 EXCLUSIVE オプションを使用すると、PROC TABULATE は、分類変数の値の組み合わせが CLASSDATA=データセットにない入力データセットのオブザベーションを除外します。

ヒント CLASSDATA=データセットを使用して、入力データセットをフィルタリングまたは補足します。

例 “例 2: テーブルに表示する分類変数の組み合わせを指定する” (2293 ページ)

CONTENTS=*link-text* <#BYLINE> <#BYVAL> < #BYVAR>
デフォルトで、または STYLE=オプションをサポートする ODS 出力先のオプション設定によって作成された目次のエントリのテキストを指定します。

BY ステートメントが有効な場合、次のオプションを使用できます。

#BYLINE
テキスト文字列に指定した#BYLINE を先頭と末尾にブランクを含まない BY 行全体で置き換えます。BY 行は、 *variable-name=value* の出力形式を使用します。

#BYVAL*n*
#BYVAL(*BY-variable-name*)
テキスト文字列に指定した#BYVAL を、指定した BY 変数の現在の値に置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAL2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAL(YEAR)では、BY 変数 YEAR を指定します。*variable-name* では、大文字と小文字は区別されません。

#BYVAR*n*
#BYVAR(*BY-variable-name*)
テキスト文字列に指定した#BYVAR を、BY-変数の名前またはこの変数に関連付けられているラベル(BY 行が通常表示するもの)で置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAR2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAR(SITES)では、BY 変数 SITES を指定します。*variable-name* では、大文字と小文字は区別されません。

参照項目: “テキスト文字列の BY 行値の置換” (2279 ページ)

DATA=*SAS-data-set*
入力データセットを指定します。

参照項目: “入力データセット” (24 ページ)

EXCLNPWGT
非正(ゼロまたは負)の重みがついたオブザベーションを分析から除外します。デフォルトでは、PROC TABULATE は、負の重みが付いたオブザベーションを重みが 0 のオブザベーションとして処理し、オブザベーション合計数に含めます。

別名 EXCLNPWGTS

参照項目: “WEIGHT=*weight-variable*” (2252 ページ)と“WEIGHT ステートメント” (2253 ページ)

EXCLUSIVE
テーブルと出力データセットから、CLASSDATA=データセットにない分類変数のすべての組み合わせを除外します。

要件 CLASSDATA=データセットが指定されていない場合、EXCLUSIVE オプションは無視されます。

例 “例 2: テーブルに表示する分類変数の組み合わせを指定する” (2293 ページ)

FORMAT=*format-name*
各テーブルセルの値に対しデフォルトの出力形式を指定します。SAS 出力形式またはユーザー定義の出力形式はどれでも使用できます。

別名 F=

デフォルト FORMAT=を省略すると、PROC TABULATE は BEST12.2 をデフォルトの出力形式として使用します。

操作 TABLE ステートメントで指定される出力形式は、FORMAT=で指定される出力形式に優先します。

ヒント FORMAT=オプションは、テーブルの印刷に使用される印刷位置数の制御に特に便利です。

例 “例 1: 基本的な 2 次元表の作成” (2291 ページ)

“例 6: 共通分類変数 ALL を使用した情報の要約” (2305 ページ)

FORMCHAR= <(*position(s)*)>='*formatting-character(s)*'
テーブルの外枠と分割線を構成するために使用する文字を定義します。

position(s)
SAS フォーマット文字列における 1 つ以上の文字の位置を識別します。スペースまたはカンマで位置を区切ります。

デフォルト *position(s)*の省略は、20 のすべての可能な SAS フォーマット文字を順番に指定することと同じです。

formatting-character(s)
指定位置に使用する文字をリストします。

PROC TABULATE は、*formatting-character(s)*の文字を表示されている順序で*position(s)*に割り当てます。たとえば、次のオプションではアスタリスク(*)を

3 番目のフォーマット文字に、数字記号(#)を 7 番目の文字に割り当てます。その他の文字は変更されません。

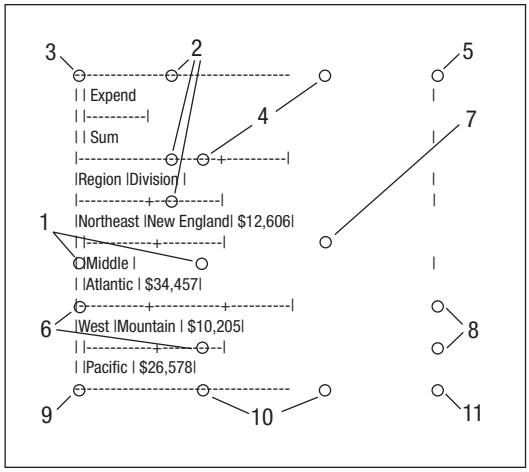
```
formchar(3,7)='*#'
```

PROC TABULATE では、SAS で提供している 20 のフォーマット文字のうち 11 が使用されます。表 61.113 (2206 ページ)には、PROC TABULATE で使用されるフォーマット文字が示されています。図 61.61 (2207 ページ)に、PROC TABULATE からの出力における各フォーマット文字の使用を示します。

表 61.2　PROC TABULATE によって使用されるフォーマット文字

位置	デフォルト	表示に使用
1	|	右枠、左枠、列間の区切り
2	-	上枠、下枠、および行間の水平区切り線
3	-	左枠の先頭文字
4	-	列を区切る文字の行の先頭文字
5	-	右枠の先頭文字
6	|	水平区切り線の行の最左文字
7	+	垂直文字の列と水平文字の行の交点
8	|	水平区切り線の行の最右文字
9	-	左枠の末尾文字
10	-	列を区切る文字の行の末尾文字
11	-	右枠の末尾文字

図 61.3 PROC TABULATE 出力のフォーマット文字



制限事項

FORMCHAR=プションは、従来の SAS モノスペース出力の出力先にのみ影響します。

操作

SAS システムオプション FORMCHAR=では、デフォルトのフォーマット文字を指定します。システムオプションは、フォーマット文字の全体の文字列を定義します。プロシジャの FORMCHAR=オプションでは、選択した文字を再定義できます。

ヒント

16 進数文字を含む *formatting-characters* の文字を使用できます。16 進数文字を使用する場合、x を終了引用符の後に付ける必要があります。たとえば、次のオプションでは、16 進文字 2-D が 3 番目のフォーマット文字に、16 進文字 7C が 7 番目の文字にそれぞれ割り当てられます。その他の文字は変わりません。

```
formchar(3,7)='2D7C'x
```

*formatting-character(s)*に対しすべてブランクを 指定すると、外枠または区切り線のないテーブルが作成されます。

```
formchar(1,2,3,4,5,6,7,8,9,10,11)       ='          '
```

(11 のブランク)

参照項目:

フォーマット出力を使用した例については、*PROC TABULATE by Example, Second Edition* を参照してください。

どの 16 進コードをどの文字に使用するかについては、ハードウェアのドキュメントを参照してください。

MISSING

欠損値を有効値とみなし、分類変数の組み合わせを作成します。数値を表すために使用される特殊欠損値(文字 A から Z、アンダースコア(_)文字)がそれぞれ個別の値としてみなされます。各欠損値に対するヘッダーがテーブルに表示されます。

デフォルト　MISSING を省略すると、PROC TABULATE は分類変数が欠損値のオブザベーションをレポートに含みません。

参照項目:　“欠損分類変数を含むオブザベーションを含める” (2284 ページ)

特殊欠損値の作成で特別な意味を持つ欠損値の説明を参照してください。

NOSEPS
行タイトルとテーブルの本文から水平区切り線を削除します。水平区切り線は、ネストされた列ヘッダーの間に残ります。

制限事項　NOSEPS オプションは、従来の SAS monospace 出力の出力先にのみ影響します。

ヒント　区切り線を削除するのではなくブランクと置き換える場合、オプション “FORMCHAR= <(*position(s)*)>='*formatting-character(s)*' ” (2205 ページ) を使用します。

NOTHREADS
入力データセットの並列処理を無効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします。

デフォルトは、SAS システムオプション THREADS | NOTHREADS の値です。

制限事項　サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

操作　PROC TABULATE は、BY ステートメントが指定されている、または SAS システムオプション CPUCOUNT の値が 2 未満の場合を除いて、SAS システムオプション THREADS の値を使用します。この場合、PROC TABULATE ステートメントで THREADS オプションを指定して、PROC TABULATE が並列処理を使用するように指定できます。並列処理でもあるマルチスレッド処理が有効な場合、オブザベーションは予想外の順序で返されることがあります。ただし、BY ステートメントが指定されている場合、オブザベーションは正しく並べ替えられます。

参照項目:　入力データセットの並列処理を有効にするには、“THREADS ” (2213 ページ)を参照してください。

詳細については、並列処理のサポートを参照してください。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
テーブルのヘッダーを形成する分類変数の値の一意の組み合わせを指定した順序に従って作成するための並べ替え順序を指定します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 CLASS ステートメントで PRELOADFMT を使用すると、各分類変数の値の順序が、PROC FORMAT が関連するユーザー定義出力形式の値の格納に使用する順序と一致します。CLASSDATA=オプションを使用する場合、PROC TABULATE は CLASSDATA= データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC TABULATE は最初にユーザー定義出力形式を使用して出力を並べ替えます。EXCLUSIVE を省略すると、PROC TABULATE はユーザー定義の出力形式と CLASSDATA=値の後に入力データセットの分類変数の一意の値を発生順に追加します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

In-Database プロシジャを ORDER=DATA オプションとともに使用している場合、結果がばらつく可能性があります。DBMS テーブルの行には固有の順序はありません。SAS プロシジャのデータベーステーブルに 書き込まれた行の順序は保持されない可能性があります。

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。出力形式が数値分類変数に割り当てられていない場合、デフォルトの出力形式 BEST12.が使用されます。この順序は、使用している動作環境によって異なります。

別名 FMT

EXTERNAL

FREQ
度数カウントの降順で値を並べ替えます。

操作 CLASS ステートメントで ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

UNFORMATTED
値をフォーマットされていない値で並べ替えます。PROC SORT と同じ順序になります。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を年代順に表示する場合に特に便利です。

別名 UNFMT

INTERNAL

デフォルト UNFORMATTED

操作 CLASS ステートメントで PRELOADFMT オプションを使用すると、PROC TABULATE はレベルをユーザー定義出力形式の値の順序で並べ替えます。

例 “ORDER=DATA を使用したヘッダーの順序について” (2289 ページ)

OUT=*SAS-data-set*
出力データセットを指定します。*SAS-data-set* は存在しない場合、PROC TABULATE が作成します。

出力データセットのオブザベーション数は、テーブルで使用されるデータのカテゴリ数と、生成されるサブテーブル数によって異なります。出力データセットには、これらの変数が次の順序で含まれています。

BY 変数　BY ステートメントで表示される変数。

分類変数　CLASS ステートメントで表示される変数。

TYPE　オブザベーションの要約統計量を生成した分類変数の組み合わせを表示する文字変数。_TYPE_の位置はそれぞれ、CLASS ステートメントの 1 つの変数を表します。その変数が統計量を生成したカテゴリ内にある場合、位置には 1 が含まれます。その他の場合、位置には 0 が含まれます。共通分類変数 ALL を使用しない単純な PROC TABULATE ステップでは、_TYPE_のすべての値には 1 のみ含まれます。これは、検討中のカテゴリのみすべての分類変数を含むためです。変数 ALL を使用すると、テーブルには一部の分類変数を含まないカテゴリに対するデータが含まれるため、_TYPE_の位置には 1 と 0 が含まれます。

PAGE　オブザベーションを含む論理ページ。

TABLE　オブザベーションを含むテーブル数。

statistics　データセットの各オブザベーションに対して計算される統計量。

例　“例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (2296 ページ)

QMARKERS=*number*
P^2 分位点推定方法に使用するマーカーのデフォルト数を指定します。マーカー数によって固定メモリ空間のサイズを制御します。

デフォルト　デフォルト値は、要求する分位点によって異なります。中央値(P50)の場合、*number* は 7 です。分位点(P25 と P75)の場合、*number* は 25 です。分位点 P1、P5、P10、P90、P95 または P99 の場合、*number* は 105 です。複数の分位点を要求すると、PROC TABULATE は *number* の最大デフォルト値を使用します。

範囲　3 より大きい奇数の整数

ヒント　デフォルト設定を超えるマーカー数を増やして推定の正確度を向上し、マーカー数を減らしてメモリと計算時間を削減します。

参照項目:　“分位点” (1288 ページ)

QMETHOD=OS | P2
PROC TABULATE が分位点の計算時に入力データの処理に使用する方法を指定します。オブザベーション数が QMARKERS=値および QNTLDEF=5 以下の場合、両方の方法による結果は同じになります。

OS
順序統計量を使用します。PROC UNIVARIATE は、この方法を使用します。

注: この方法は、非常に多くのメモリを必要とする可能性があります。

P2
P^2 方法を使用して、分位点を概算します。

別名 HIST

デフォルト OS

制限事項 QMETHOD=P2 の場合、PROC TABULATE は MODE または重み付き分位点を計算しません。

ヒント QMETHOD=P2 の場合、一部の分位点(P1、P5、P95、P99)に対して信頼できる推定は、一部の種類のデータに対して信頼できないことがあります。

参照項目: “分位点” (1288 ページ)

QNTLDEF=1 | 2 | 3 | 4 | 5
QMETHOD=OS の指定時にプロシジャが分位点の計算に使用する算術定義を指定します。QMETHOD=P2 の場合、QNTLDEF=5 を使用する必要があります。

別名 PCTLDEF=

デフォルト 5

参照項目: “分位数と関連統計量” (2415 ページ)

STYLE=*style-override(s)*
テーブルのデータセルに適用する 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、データセルの背景色が赤になるように指定します。

```
proc tabulate data=one style=[backgroundcolor=red];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。

スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override の形式は次のとおりです。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1*

<*style-attribute-name-2=style-attribute-value-2 ...*>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。

データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で中かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、"出力形式を使用してスタイル要素を割り当てる" (2275 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

別名 S=

ヒント STYLE=オプションをその他のステートメントまたは次元式で使用して、テーブルのその他の部分にスタイル要素を指定できます。

欠損値を含むデータセルにスタイル要素を指定するには、TABLE ステートメント MISSTEXT=オプションで STYLE=を使用します。

角かっこ([と])の代わりに中かっこ({と})を使用できます。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

例 “例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

THREADS
入力データセットの並列処理を有効化します。システムオプションが制限されない限り、このオプションは SAS システムオプション THREADS | NOTHREADS を無効にします。

デフォルトは、SAS システムオプション THREADS | NOTHREADS の値です。

制限事項 サイト管理者が、制限オプションテーブルを作成できます。制限オプションテーブルは、スタートアップ時に作成され、無効にできない SAS システムオプション値を指定します。THREADS | NOTHREADS システムオプションが制限オプションテーブルにリストされると、これらのシステムオプションを設定しようとしても無視され、警告メッセージが SAS ログに書き込まれます。

操作 PROC TABULATE は、BY ステートメントが指定されている、または SAS システムオプション CPUCOUNT の値が 2 未満の場合を除いて、SAS システムオプション THREADS の値を使用します。この場合、PROC TABULATE ステートメントで THREADS オプションを指定して、PROC TABULATE が並列処理を使用するように指定できます。並列処理でもあるマルチスレッド処理が有効な場合、オブザベーションは予想外の順序で返されることがあります。ただし、BY ステートメントが指定されている場合、オブザベーションは正しく並べ替えられます。

参照項目: 入力データセットの並列処理を無効にするには、“NOTHREADS” (2208 ページ)を参照してください。

詳細については、並列処理のサポートを参照してください。

TRAP
データ処理中に通常の SAS FPE 処理によって復旧できない浮動小数点例外(FPE)の復旧を可能にします。TRAP オプションを使用しない場合、算術例外の場合に PROC TABULATE が終了するように、通常の SAS FPE 処理はそのまま有効です。

VARDEF=DF | N | WDF | WEIGHT
分散と標準偏差の計算に使用する除数を指定します。次のテーブルに、*divisor* に可能な値と、関連する除数を示します。

DF
自由度

除数の式は次のとおりです。

$n - 1$

N
オブザベーション数

除数の式は次のとおりです。

n

WDF
重みの合計 - 1

除数の式は次のとおりです。

$(\Sigma_i w_i) - 1$

WEIGHT
重みの合計

除数の式は次のとおりです。

$\Sigma_i w_i$

別名 WGT

プロシジャは分散を$CSS/divisor$として計算します。CSSは修正平方和で、$\Sigma (x_i - \bar{x})^2$と等しくなります。分析変数を重み付けする場合、CSSは$\Sigma w_i(x_i - \bar{x}_w)^2$と等しくなります。$\bar{x}_w$は重み付きの平均です。

デフォルト DF

要件 平均値の標準誤差を計算するには、VARDEF=のデフォルト値を使用します。

ヒント WEIGHT ステートメントと VARDEF=DF を使用する場合、分散はσ^2の推定です。*i* 番目のオブザベーションの分散は$var(x_i) = \sigma^2/w_i$、およびw_iは *i* 番目のオブザベーションの重みです。これにより、オブザベーションの分散の推定が単位重みとともに生成されます。

WEIGHT ステートメントと VARDEF=WGT を使用する場合、計算された分散が漸近的に(大きな *n* に対し)$\sigma^2/\bar{w}$の推定になります。$\bar{w}$は、平均重みで

す。これにより、オブザベーションの分散の漸近的推定が平均重みとともに生成されます。

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント　SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント　この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント　この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

“ATTRIB ステートメント” (*SAS DATA ステップステートメント: リファレンス*).

BY ステートメント

出力を BY グループ順に並べ替えます。

制限事項: BY ステートメントではグループ変数を指定しないでください。BY 変数をグループ変数と同じにすることはできません。

構文

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

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

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

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

詳細

BY グループ処理

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

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

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

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

- BY ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。
- PROC SORT の詳細については、55 章, “SORT プロシジャ,” (2027 ページ)を参照してください。
- CAS での BY グループ処理の詳細については、“グループ化と順序付け” (*SAS Cloud Analytic Services: DATA ステッププログラミング*)を参照してください。
- インデックスの作成の詳細については、“INDEX CREATE ステートメント” (435 ページ)を参照してください。

BY 変数値のフォーマティング

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

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

このプロセスでは、非連続の内部 BY 値が同一のフォーマットされた値を共有する場合などに、予想外の結果が出る可能性があります。この場合、フォーマットされた値は、異なる BY グループに表示されます。また、異なる連続内部 BY 値が同一のフ

ォーマットされた値を共有している場合、そのオブザベーションは同一 BY グループにグループ化されます。

2 つのデータセットで長さが異なる BY 変数

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

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

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

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

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

- A
- A、B
- A、B、C
- A、B、C、D

プライマリデータセットとセカンダリデータセットの両方に同数の BY 変数が含まれ、すべての BY 変数のバイト長とフォーマット長が同じ場合、(すべての BY 変数

の)BY バッファーにあるフォーマットされていない値かフォーマットされた値のどちらかが一致する必要があります。一致しない場合、各変数が比較されます。各変数のフォーマットされた値が先に比較されます。フォーマットされた長さが一致し、さらに、フォーマットされた値が一致する必要があります。フォーマットされた長さと値が一致しない場合は、バイト長が異なっても、フォーマットされていない値が比較されます。

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブランクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデータセットの'ABCD'とセカンダリデータセットの'ABCD 'は一致します。セカンダリデータセットに'ABCDEF'が含まれている場合は、一致しません。

BY ステートメントをサポートする Base SAS プロシジャ

COMPARE	SORT (必須)
CORR	STANDARD
FREQ	SUMMARY
MEANS	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE
REPORT (非ウィンドウ環境のみ)	ODSTEXT
ODSLIST	ODSTABLE

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

関連項目

"BY ステートメント" (SAS DATA ステップステートメント: リファレンス)

CLASS ステートメント

テーブルの分類変数を識別します。分類変数によって、PROC TABULATE が統計量の計算に使用するカテゴリが決定されます。

別名: CLASSES

注: オプションがない CLASS ステートメントでは、内部デフォルト、または PROC TABULATE ステートメントのオプションにより指定された値が使用されます。たとえば次のコードでは、変数 **c** と **d** には内部デフォルトが使用されることがあります。PROC TABULATE ステートメントにおいて ORDER=オプションを指定したとすると、変数 c と d には、 PROC TABULATE ステートメントの ORDER=オプションにより指定された値が使用されます。

```
class a b / order=data;
class c d;
```

ヒント: 複数の CLASS ステートメントを使用できます。

一部の CLASS ステートメントオプションも PROC TABULATE ステートメントで使用できます。これらは、CLASS ステートメントで指定するものだけでなく、すべての CLASS 変数に影響します。

例: “例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (2296 ページ)

“例 4: マルチラベル出力形式の使用” (2300 ページ)

構文

CLASS *variable(s)* </ *options*>;

オプション引数の要約

ASCENDING
: 分類変数値を昇順で並べ替えるように指定します。

DESCENDING
: 分類変数値を降順で並べ替えるように指定します。

EXCLUSIVE
: すでに読み取られた範囲にないユーザー定義の出力形式の分類変数のすべての組み合わせを、テーブルと出力データセットから除外します。

GROUPINTERNAL
: PROC TABULATE が値をグループ化し、分類変数の組み合わせを作成する際に分類変数に出力形式を適用しないように指定します。

MISSING
: 欠損値を有効な分類変数水準とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

MLF
: マルチラベル出力形式が分類変数に割り当てられている場合に、PROC TABULATE が指定範囲または重複する範囲に出力形式ラベルを使用して、サブグループの組み合わせを作成できるようになります。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
: 出力の分類変数の水準をグループ化する順序を指定します。

PRELOADFMT

分類変数のすべての出力形式がすでに読み込まれていることを示します。

STYLE=*style-override*
ページ次元テキストと分類変数名ヘッダーに使用する 1 つ以上のスタイルの上書きを指定します。

必須引数

variable(s)
プロシジャがデータのグループ化に使用する 1 つ以上の変数を指定します。CLASS ステートメントの変数は、*分類変数*といいます。分類変数は、数値か文字です。分類変数には連続値を含めることができますが、通常は変数の分類を定義するいくつかの不連続値が含まれます。データを分類変数別に並べ替える必要はありません。

操作 変数名と統計量名が同じ場合、統計量名を単一引用符か二重引用符で囲みます。

スラッシュ区切り文字の後に使用するオプション

ASCENDING
分類変数値を昇順で並べ替えるように指定します。

別名 ASCEND

操作 ASCENDING と DESCENDING の両方を指定すると、PROC TABULATE は警告メッセージを発行して、両方のオプションを無視します。

DESCENDING
分類変数値を降順で並べ替えるように指定します。

別名 DESCEND

デフォルト ASCENDING

操作 ASCENDING と DESCENDING の両方を指定すると、PROC TABULATE は警告メッセージを発行して、両方のオプションを無視します。

EXCLUSIVE
すでに読み込まれた範囲にないユーザー定義の出力形式の分類変数のすべての組み合わせを、テーブルと出力データセットから除外します。

要件 分類変数の出力形式をあらかじめ読み込むには、CLASS ステートメントで PRELOADFMT オプションを指定する必要があります。

例 “例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (2296 ページ)

GROUPINTERNAL
PROC TABULATE が値をグループ化し、分類変数の組み合わせを作成する際に分類変数に出力形式を適用しないように指定します。

操作 CLASS ステートメントで PRELOADFMT オプションを指定すると、PROC TABULATE は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ORDER=FORMATTED オプションを指定すると、PROC TABULATE は GROUPINTERNAL オプションを無視し、フォーマットされた値を使用します。

ヒント GROUPINTERNAL オプションは、数値変数に不連続値が含まれている場合にコンピューターリソースを節約します。

MISSING
欠損値を有効な分類変数水準とみなします。数値を表すために使用される特殊欠損値(A から Z までの文字とアンダースコア(_)文字)は、それぞれ別の値とみなされます。

デフォルト MISSING オプションを省略すると、PROC TABULATE により、CLASS 変数値が欠損しているオブザベーションがテーブルと 出力データセットから除外されます。

参照項目: “例: 特別な欠損値の作成” (*SAS プログラマガイド: エッセンシャル*)で特別な意味を持つ欠損値の説明を参照してください。

MLF
マルチラベル出力形式が分類変数に割り当てられている場合に、PROC TABULATE が指定範囲または重複する範囲に出力形式ラベルを使用して、サブグループの組み合わせを作成できるようになります。

注: フォーマットされた値が重複する場合、1 つの内部分類変数値が複数の分類変数サブグループの組み合わせにマップします。そのため、すべてのサブグループの N 統計量の合計がデータセットのオブザベーション数(N 統計量全体)を超えます。

要件 VALUE ステートメントで PROC FORMAT と MULTILABEL オプションを使用して、マルチラベル出力形式を作成する必要があります。

操作 MLF と ORDER=FREQ を使用すると、フォーマットされた値に使用する順序が生成されない場合があります。

MLF を指定すると、分類変数のフォーマットされた値は内部値になります。そのため、ORDER=FORMATTED を指定すると、ORDER=UNFORMATTED を指定した場合と同じ結果になります。

ヒント MLF を省略すると、PROC TABULATE は最初の外部出力形式値に対応する 1 次出力形式ラベルを使用して、サブグループの組み合わせを決定します。

参照項目: “MULTILABEL” (909 ページ)(FORMAT プロシジャの VALUE ステートメント)。

例 “例 4: マルチラベル出力形式の使用” (2300 ページ)

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
出力の分類変数の水準をグループ化する順序を指定します。

DATA
入力データセットの順序に従って値を並べ替えます。

操作 PRELOADFMT を使用すると、各分類変数の値の順序は、PROC FORMAT が関連するユーザー定義出力形式の値の保存に使用する順序に一致します。PROC ステートメントで CLASSDATA=オプションを使用すると、PROC TABULATE は CLASSDATA=データセットの各分類変数の一意の値の順序を使用して、出力水準を並べ替えます。両方のオプションを使用すると、PROC TABULATE は最初にユーザー定義出力形式を使用して出力を並べ替えます。PROC ステートメントで EXCLUSIVE を省略すると、PROC TABULATE はユーザー定義出力形式と CLASSDATA=値の後に、入力データセットにある分類変数の一意の値を発生順序で配置します。

ヒント デフォルトでは、PROC FORMAT は出力形式定義を並べ替えられた順序で格納します。NOTSORTED オプションを使用して、ユーザー定義出力形式の値または範囲を定義順に格納します。

FORMATTED
値をフォーマットされた値の昇順で並べ替えます。この順序は、使用している動作環境によって異なります。

別名 FMT

EXTERNAL

FREQ
度数カウントの降順で値を並べ替えます。

操作 ASCENDING オプションを使用して、値を度数カウントの昇順で並べ替えます。

UNFORMATTED
値をフォーマットされていない値で並べ替えます。PROC SORT と同じ順序になります。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を年代順に表示する場合に特に便利です。

別名 UNFMT

INTERNAL

デフォルト UNFORMATTED

操作 CLASS ステートメントで PRELOADFMT オプションを使用すると、PROC TABULATE はレベルをユーザー定義出力形式の値の順序で並べ替えます。

ヒント デフォルトでは、FREQ 以外のすべての順序は昇順です。降順の順序の場合は、DESCENDING オプションを使用します。

例 “ORDER=DATA を使用したヘッダーの順序について” (2289 ページ)

PRELOADFMT
分類変数のすべての出力形式がすでに読み込まれていることを示します。

要件 PRELOADFMT は、EXCLUSIVE、ORDER=DATA または PRINTMISS を指定して、出力形式を分類変数に割り当てる場合のみ有効です。EXCLUSIVE、ORDER=DATA または PRINTMISS も指定せずに PRELOADFMT を指定すると、警告メッセージが SAS ログに書き込まれます。

操作 PROC TABULATE 出力を入力データセットに存在するフォーマットされた分類変数値の組み合わせに制限するには、CLASS ステートメントで EXCLUSIVE オプションを使用します。

ユーザー定義出力形式のすべての範囲と値を出力に含めるには、TABLE ステートメントで PRINTMISS オプションを使用します。PRELOADFMT を PRINTMISS と使用する際は注意が必要です。この機能により、フォーマットされた分類変数の可能なすべての組み合わせが作成されます。これらの組み合わせの一部が意味をなさない場合があります。

注 文字出力形式をプリロードする場合、ラベルを生成する代表的な未加工値が少なくとも 1 つ存在しなければなりません。範囲に未加工値がない場合、到達不能と判断され、エラーが発生します。たとえば、Other=と Low-High の両方の範囲を持つマルチラベル文字出力形式をプリロードすることはできません。Other は、すべてのラベルが定義された後に残る値または範囲を参照する特別なキーワードです。文字出力形式の場合、low 範囲に欠損値が含まれるため、その他の域を占める未加工値が残りません。

例 “例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (2296 ページ)

STYLE=*style-override*
ページ次元テキストと分類変数名ヘッダーに使用する 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、ページ次元テキストの背景色と分類変数名ヘッダーが薄緑色になるように指定します。

```
class region division prodtype / style=[background=lightgreen];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。

スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override の形式は次のとおりです。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1*

<*style-attribute-name-2=style-attribute-value-2* ...>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。

データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で中かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、"出力形式を使用してスタイル要素を割り当てる" (2275 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

別名 S=

ヒント CLASS ステートメントのページ次元テキストに指定されるスタイル要素を無効にする場合、TABLE ステートメントページ次元式でスタイル要素を指定できます。

CLASS ステートメントで分類変数名ヘッダーに指定されるスタイル要素を無効にする場合、関連する TABLE ステートメント次元式でスタイル要素を指定できます。

ページ次元式にネストされた要素が複数含まれている場合、Beforecaption スタイル要素がネスティングの最初の要素のスタイル要素となります。

CLASS ステートメントでの STYLE=の使用は、PROC TABULATE ステートメントでの使用とはやや異なります。CLASS ステートメントでは、継承は行と列では異なります。行の場合、親ヘッダーは現在のヘッダーの左側に置かれます。列の場合、親ヘッダーは現在のヘッダーの上に置かれます。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

例 “例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

詳細

分類変数の欠損値の PROC TABULATE による処理方法

デフォルトでは、オブザベーションに分類変数に対する欠損値が含まれている場合、PROC TABULATE は作成したすべてのテーブルからそのオブザベーションを 除外します。CLASS ステートメントは、PROC TABULATE ステップのすべての TABLE ステートメントに適用されます。そのため、変数を分類変数として定義すると、PROC TABULATE は、変数が 1 つ以上のテーブルに対し TABLE ステートメントに記述されない場合でも、変数に欠損値があるすべてのオブザベーションをすべてのテーブルから除外します。

PROC TABULATE ステートメントで MISSING オプションを指定すると、プロシジャは欠損値をすべての分類変数に対する有効水準とみなします。CLASS ステートメントで MISSING オプションを指定すると、PROC TABULATE は欠損値を CLASS ステートメントで指定される分類変数に対する有効水準とみなします。

CLASSLEV ステートメント

分類変数水準値ヘッダーにスタイル要素を指定します。

別名: CLASSLEVEL
CLASSLEVELS

例: “例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)
“例 14: スタイル優先” (2347 ページ)

構文

CLASSLEV *variable(s)* </ STYLE=*style-override(s)*>;

必須引数

variable(s)
スタイル要素を指定する CLASS ステートメントから 1 つ以上の分類変数を指定します。

スラッシュ区切り文字の後に使用するオプション

STYLE=*style-override*
分類変数水準値ヘッダーに 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、分類変数水準値ヘッダーの背景色が黄色になるように指定します。

```
classlev region division prodtype / style=[background=yellow];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。

スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override の形式は次のとおりです。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1*
<*style-attribute-name-2=style-attribute-value-2* ...>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。

データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で中かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、"出力形式を使用してスタイル要素を割り当てる" (2275 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

別名 S=

ヒント CLASSLEV ステートメントでの STYLE=の使用は、PROC TABULATE ステートメントでの使用とはやや異なります。CLASSLEV ステートメントでは、継承は行と列では異なります。行の場合、親ヘッダーは現在のヘッダーの左側に置かれます。列の場合、親ヘッダーは現在のヘッダーの上に置かれます。

CLASSLEV ステートメントで指定されるスタイル要素を無効にする場合、関連する TABLE ステートメント次元式でスタイル要素を指定できます。

角かっこ([と])の代わりに中かっこ({と})を使用できます。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

例 “例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。“出力形式の取り消し” (*SAS DATA ステッ*

プステートメント: リファレンス)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、"PUT ステートメント: リスト" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

FREQ ステートメント

各オブザベーションの度数を含む数値変数を指定します。

ヒント: FREQ ステートメントと WEIGHT ステートメントの影響は、自由度の計算時以外は似ています。

構文

FREQ *variable*;

必須引数

variable
値がオブザベーションの度数を表す数値変数を指定します。FREQ ステートメントを使用する場合、プロシジャは各オブザベーションが

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

度数変数の合計は、オブザベーションの合計数を表します。

KEYLABEL ステートメント

PROC TABULATE ステップで使用するキーワードのラベルを指定します。PROC TABULATE は、キーワードの表示にこのラベルを使用します。ラベルが指定されない場合はキーワードが表示されます。

構文

KEYLABEL *keyword-1*='*description-1*' <*keyword-2*='*description-2*' ...>;

必須引数

keyword
統計量キーワードを指定します。

keyword は、“PROC TABULATE で使用できる統計量” (2256 ページ)で述べている統計量に対するキーワードの 1 つ、または共通分類変数 ALL です。(“次元式で使用できる要素” (2246 ページ)を参照してください。)

description
キーワードラベルを指定します。

長さ　256 文字。

制限事項　各キーワードには、特定の PROC TABULATE ステップにラベルを 1 つだけ含めることができます。同じキーワードに対し複数のラベルを要求すると、PROC TABULATE はステップで指定される最後のラベルを使用します。

要件　*description* は引用符で囲む必要があります。

KEYWORD ステートメント

キーワードヘッダーにスタイル要素を指定します。

別名:　KEYWORD

例:　“例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

構文

KEYWORD *keyword(s)* </ STYLE=*style-override(s)*>;

必須引数

keyword
キーワード統計量を指定します。

keyword は、“PROC TABULATE で使用できる統計量” (2256 ページ)で述べている統計量に対するキーワードの 1 つ、または共通分類変数 ALL です。(“次元式で使用できる要素” (2246 ページ)を参照してください。)

スラッシュ区切り文字の後に使用するオプション

STYLE=*style-override*
キーワードヘッダーに 1 つ以上のスタイルの上書きを指定します。

たとえば、次のステートメントは、キーワードヘッダーの背景色が淡い黄色になるように指定します。

```
keyword all sum / style=[background=linen];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。

スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override の形式は次のとおりです。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1*

<*style-attribute-name-2=style-attribute-value-2* ...>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。

データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で中かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、“出力形式を使用してスタイル要素を割り当てる” (2275 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

別名 S=

ヒント KEYWORD ステートメントでの STYLE=の使用は、PROC TABULATE ステートメントでの使用とはやや異なります。KEYWORD ステートメントでは、継承は行と列では異なります。行の場合、親ヘッダーは現在のヘッダーの左側に置かれます。列の場合、親ヘッダーは現在のヘッダーの上に置かれます。

KEYWORD ステートメントで指定されるスタイル要素を無効にする場合、関連する TABLE ステートメント次元式でスタイル要素を指定できます。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

例 “例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=' '>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string
最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項 ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注 ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL

ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目: “文字定数と文字変数” (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

TABLE ステートメント

印刷するテーブルを記述します。

別名: TABLES

要件 TABLE ステートメントのすべての変数を VAR ステートメントまたは CLASS ステートメントに記述する必要があります。

ヒント: 複数のテーブルを作成するには、複数の TABLE ステートメントを使用します。

変数名リストショートカットの使用が、TABLE ステートメント内でサポートされています。詳細については、“変数名リストを指定するショートカット” (61 ページ)を参照してください。

構文

TABLE <<*page-expression*,> *row-expression*,>
column-expression </ *table-options*>;

オプション引数の要約

BOX={<*label*=*value*> <**STYLE**=***style-override(s)***>}
: 行タイトルの上の空のボックスにテキストとスタイルの上書きを指定します。

BOX=*value*
: 行タイトルの上の空のボックスにテキストを指定します。

CAPTION= *text*<#BYLINE> <#BYVAL><#BYVAR>
: 各テーブルの前に追加するキャプションテキスト文字列を指定します。

CONDENSE
: 1 枚の印刷ページにできるだけ多くの論理ページを印刷します。または可能な場合は、1 ページには広すぎるテーブルの複数のページを別のページに印刷するのではなく、1 ページに相互に重ねて印刷します。

CONTENTS=*link-text* <#BYLINE> <#BYVAL> <#BYVAR>
: TABLE ステートメントを使用して生成されるテーブルの ODS 出力を指す HTML 目次のリンクを指定できるようになります。

FORMAT_PRECEDENCE=PAGE | ROW | COLUMN
: ページ次元(PAGE)、行次元(ROW)、列次元(COLUMN)に対して指定される出力形式がテーブルセルのコンテンツに適用されるかどうかを指定します。

FUZZ=*number*
: 度数カウント以外のテーブルセル値が比較される数値を提供し、自明な値(FUZZ=値未満の絶対値)を印刷から削除します。

INDENT=*number-of-spaces*
: ネストされている行ヘッダーをインデントするスペース数を指定し、分類変数に対する行ヘッダーを非表示にします。

MISSTEXT='*text*'
: 欠損値を含むテーブルセルに対して、印刷される最大 256 文字のテキストを提供します。

MISSTEXT={<*label*='*text*'> <**STYLE**=***style-override(s)***>}
: 印刷される最大 256 文字のテキストを提供し、欠損値を含むテーブルセルにスタイルの上書きを指定します。

NOCELLMERGE
: データセルがテーブルのその他のデータセルと結合されないように指定します。

NOCONTINUED
: 複数ページにわたってテーブルの 下部に表示される続行メッセージ **continued** を非表示にします。

page-expression
: テーブルのページを定義します。

PRINTMISS

変数のヘッダーが印刷されるたびに分類変数に対して発生するすべての値を印刷します。これらのヘッダーが作成するセルの一部に対しデータがない場合でも同様です。

ROW=CONSTANT | FLOAT
行クロスのすべてのタイトル要素が、ブランクの場合でも割り当てられたスペースかどうかを指定します。

row-expression
テーブルの行を定義します。

RTSPACE=*number*
行ヘッダーの外枠文字の印刷に使用されるスペースを含む、行次元のすべてのヘッダーに割り当てる印刷位置の数を指定します。

STYLE_PRECEDENCE=PAGE | ROW | COLUMN
ページ次元(PAGE)、行次元(ROW)または列次元(COLUMN または COL)に対して指定されるスタイルがテーブルセルのコンテンツに適用されるかどうかを指定します。

STYLE=*style-override*
テーブルセル以外のテーブル部分に使用する 1 つ以上のスタイルの上書きを指定します。

必須引数

column-expression
テーブルの列を定義します。次元式構成の詳細については、“詳細” (2246 ページ)を参照してください。

制限事項 列次元は、TABLE ステートメントの最後の次元です。行次元または行次元とページ次元は、列次元の前に置くことができます。

オプション引数

page-expression
テーブルのページを定義します。次元式構成の詳細については、“詳細” (2246 ページ)を参照してください。

制限事項 ページ次元は、テーブルステートメントの最初の次元です。行次元と列次元はページ次元の後に置く必要があります。

ACCESSIBLETABLE=ON システムオプションが指定されていて、TABULATE テーブルにページディメンションのテキストがある 場合、そのテキストがアクセス可能なキャプションです。それ以外の場合は、CAPTION=/CONTENTS=テキストを使用します。

アクセシビリティの注 SAS 9.4M6 以降では、ACCESSIBLECHECK および ACCESSIBLETABLE システムオプションを使用して、アクセシブルテーブルをチェックおよび作成できます。アクセス可能な PROC TABULATE テーブルの作成については、“Creating Accessible Tables” (*Creating Accessible SAS Viya Platform Output Using ODS and ODS Graphics*)を参照してください。
ACCESSIBLETABLE=ON システムオプションが指定されていて、TABULATE テーブルにページディメンションのテキストがある 場合、そのテキストがアクセス可能なキャプションです。

例 “例 8: 複数ページテーブルの作成” (2310 ページ)

row-expression
テーブルの行を定義します。次元式構成の詳細については、“詳細” (2246 ページ)を参照してください。

制限事項 行次元は、テーブルステートメントの最後の次元の隣です。列次元は、行次元の後に置く必要があります。ページ次元は行次元の前に置くことができます。

テーブルオプション

BOX=*value*
行タイトルの上の空のボックスにテキストを指定します。

Value は、次のうちいずれかになります。

PAGE
ページ次元テキストをボックスに書き込みます。ボックスに合わないページ次元テキストはボックスの上のデフォルトの位置に置かれ、ボックスは空のままです。

BOX={<*label*=*value*> <**STYLE=*style-override(s)***>}
行タイトルの上の空のボックスにテキストとスタイルの上書きを指定します。

Value は、次のうちいずれかになります。

PAGE
ページ次元テキストをボックスに書き込みます。ボックスに合わないページ次元テキストはボックスの上のデフォルトの位置に置かれ、ボックスは空のままです。

'*string*'
引用符で囲まれた文字列をボックスに書き込みます。ボックスに合わない文字列は、切り捨てられます。

variable
変数の名前(または変数に含まれている場合はラベル)をボックスに書き込みます。ボックスに合わない名前またはラベルは、切り捨てられます。

STYLE=オプションの引数とその使用方法については、TABLE ステートメントの STYLE= (2243 ページ)を参照してください。

例 “例 8: 複数ページテーブルの作成” (2310 ページ)

“例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

CAPTION= *text*<#BYLINE> <#BYVAL><#BYVAR>
各テーブルの前に追加するキャプションテキスト文字列を指定します。文字列が指定されていない場合、キャプションはデフォルトで TABLE ステートメントの CONTENTS=オプションで指定されたテキストになります。

このオプションは、ACCESSIBLETABLE システムオプションが指定されている場合、テーブルのキャプションを視覚的かつアクセス可能にします。

NOACCESSIBLETABLE システムオプションが指定されている場合、キャプションは表示されませんが、キャプションテキストは引き続きアクセス可能です。

BY ステートメントが有効な場合、次のオプションを使用できます。

#BYLINE
テキスト文字列に指定した#BYLINE を先頭と末尾にブランクを含まない BY 行全体で置き換えます。BY 行は、 *variable-name=value* の出力形式を使用します。

#BYVAL*n*
#BYVAL(*BY-variable-name*)
テキスト文字列に指定した#BYVAL を、指定した BY 変数の現在の値に置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAL2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAL(YEAR)では、BY 変数 YEAR を指定します。*variable-name* では、大文字と小文字は区別されません。

#BYVAR*n*
#BYVAR(*BY-variable-name*)
テキスト文字列に指定した#BYVAR を、BY-変数の名前またはこの変数に関連付けられているラベル(BY 行が通常表示するもの)で置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAR2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAR(SITES)では、BY 変数 SITES を指定します。*variable-name* では、大文字と小文字は区別されません。

制限事項 各キャプションを有効にするには、ACCESSIBLETABLE システムオプションを指定する必要があります。

ヒント PROC DOCUMENT OBBNOTE オプションを使用して、キャプションを表示または編集できます。

参照項目: “Creating Accessible Tables” (*Creating Accessible SAS Viya Platform Output Using ODS and ODS Graphics*)

CONDENSE
logical page は、次のうちいずれかにあるすべての行と列です。

- ページ次元カテゴリ(BY グループ処理なし)
- BY グループ(ページ次元なし)
- 単一の BY グループ内のページ次元カテゴリ

制限事項 CONDENSE オプションは、BY ステートメントによって生成されるページでは無効です。BY グループの最初のテーブルは、常に新しいページで始まります。

例 “例 8: 複数ページテーブルの作成” (2310 ページ)

CONTENTS=*link-text* <#BYLINE> <#BYVAL> <#BYVAR>
TABLE ステートメントを使用して生成されるテーブルの ODS 出力を指す HTML 目次のリンクを指定できるようになります。

BY ステートメントが有効な場合、次のオプションを使用できます。

#BYLINE
テキスト文字列に指定した#BYLINE を先頭と末尾にブランクを含まない BY 行全体で置き換えます。BY 行は、 *variable-name*=*value* の出力形式を使用します。

#BYVAL*n*
#BYVAL(*BY-variable-name*)
テキスト文字列に指定した#BYVAL を、指定した BY 変数の現在の値に置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAL2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAL(YEAR)では、BY 変数 YEAR を指定します。*variable-name* では、大文字と小文字は区別されません。

#BYVAR*n*
#BYVAR(*BY-variable-name*)
テキスト文字列に指定した#BYVAR を、BY-変数の名前またはこの変数に関連付けられているラベル(BY 行が通常表示するもの)で置き換えます。

この変数を次のいずれかと一緒に指定します。

n
は、BY ステートメント内のその位置により変数を指定します。たとえば、#BYVAR2 では、BY ステートメントの 2 番目の変数を指定します。

BY-variable-name
BY ステートメントの変数をその名前で指定します。たとえば、#BYVAR(SITES)では、BY 変数 SITES を指定します。*variable-name* では、大文字と小文字は区別されません。

参照項目: “テキスト文字列の BY 行値の置換” (2279 ページ)

“Creating Accessible Tables” (*Creating Accessible SAS Viya Platform Output Using ODS and ODS Graphics*)

FORMAT_PRECEDENCE=PAGE | ROW | COLUMN
ページ次元(PAGE)、行次元(ROW)、列次元(COLUMN)に対して指定される出力形式がテーブルセルのコンテンツに適用されるかどうかを指定します。

PAGE
ページ次元に対して指定される出力形式がテーブルセルのコンテンツに適用されるかどうかを指定します。

ROW
行次元に対して指定される出力形式がテーブルセルのコンテンツに適用されるかどうかを指定します。

COLUMN
列次元に対して指定される出力形式がテーブルセルのコンテンツに適用されるかどうかを指定します。

別名 COL

デフォルト COLUMN

FUZZ=*number*
度数カウント以外のテーブルセル値が比較される数値を提供し、自明な値(FUZZ=値未満の絶対値)を印刷から削除します。絶対値が FUZZ=値未満である数値は、印刷時にゼロとして扱われます。デフォルト値は、使用しているコンピューター上の表示可能な最小浮動小数点数です。

INDENT=*number-of-spaces*
ネストされている行ヘッダーをインデントするスペース数を指定し、分類変数に対する行ヘッダーを非表示にします。

制限事項 HTML5、RTF、Printer 出力先で、INDENT=オプションは分類変数の行ヘッダーを非表示にしますが、ネストされている行ヘッダーをインデントしません。

ヒント 行次元にクロスがない場合は、インデントは実行されません。そのため、*number-of-spaces* の値は無効になります。ただしそのような場合、INDENT=は分類変数に対する行ヘッダーを非表示にします。

MISSTEXT='*text*'
欠損値を含むテーブルセルに対して、印刷される最大 256 文字のテキストを提供します。

MISSTEXT={<*label*='*text*'> <**STYLE**=*style-override(s)*>}
印刷される最大 256 文字のテキストを提供し、欠損値を含むテーブルセルにスタイルの上書きを指定します。STYLE=オプションの引数とその使用方法については、TABLE ステートメントの STYLE= (2243 ページ)を参照してください。

操作 次元式で指定されるスタイル要素は、指定セルに対し MISSTEXT=オプションで指定されるスタイル要素に優先します。

例 “欠損値を含むセルに対するテキストの提供” (2287 ページ)

“例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

NOCELLMERGE
データセルがテーブルのその他のデータセルと結合されないように指定します。

注: NOCELLMERGE オプションは、ODS 形式の出力先と機能します。これらには、ODS MARKUP グループ、ODS RTF および ODS PRINTER グループの出力先が含まれます。

制限事項 NOCELLMERGE オプションは、従来のモノスペース出力では機能しません。

操作 ROW=FLOAT または INDENT=0 を指定すると、PROC TABULATE は単一の結合されていないデータ行を生成します。NOCELLMERGE オプションは無視されます。結合が必要な行がないためです。

NOCELLMERGE オプションが有効な場合、空のデータセルのスタイルはデータセルのデフォルトのスタイルとなります。空のデータセルのスタイルは、フォーマットされたデータセルのスタイルと異なることがあります。

例 “例 15: NOCELLMERGE オプションの使用” (2351 ページ)

NOCONTINUED
複数ページにわたってテーブルの 下部に表示される続行メッセージ **continued** を非表示にします。テキストは、AFTERCAPTION スタイル要素で表示されます。

注: HTML ブラウザーはページを分割しないため、NOCONTINUED は HTML5 出力先に影響しません。

PRINTMISS
変数のヘッダーが印刷されるたびに分類変数に対して発生するすべての値を印刷します。これらのヘッダーが作成するセルの一部に対しデータがない場合でも同様です。その結果、PRINTMISS は、テーブルのすべての 論理ページに対して同じ行および列ヘッダーを単一の BY グループ内に作成します。

デフォルト RINTMISS オプションを省略すると、PROC TABULATE は、PROC TABULATE ステートメントで CLASSDATA=オプションを使用しない限り、データがない行または列を非表示にします。

制限事項 論理ページ全体に欠損値のみが含まれている場合、PRINTMISS オプションが指定されていてもそのページは 印刷されません。

参照項目: “CLASSDATA=*SAS-data-set*” (2203 ページ)

例 “すべてのカテゴリのヘッダーの提供” (2286 ページ)

ROW=CONSTANT | FLOAT
行クロスのすべてのタイトル要素が、ブランクの場合でも割り当てられたスペースかどうかを指定します。

CONSTANT
タイトルがブランクの場合でも、スペースをすべての 行タイトルに割り当てます。(N=' 'など)。

別名 CONST

FLOAT
行タイトルスペースをクロスのブランク以外の行タイトル間に均等に分配します。

デフォルト CONSTANT

例 “例 7: 行ヘッダーの削除” (2308 ページ)

RTSPACE=*number*
行ヘッダーの外枠文字の印刷に使用されるスペースを含む、行次元のすべてのヘッダーに割り当てる印刷位置の数を指定します。PROC TABULATE は、このスペースを行ヘッダーのすべてのレベル間に均等に分配します。

別名 RTS=

デフォルト SAS システムオプション LINESIZE=の値の 4 分の 1

制限事項 RTSPACE=オプションは、従来の SAS monospace 出力先にのみ影響します。

操作 デフォルトでは、PROC TABULATE はブランクの行タイトルにスペースを割り当てます。TABLE ステートメントで ROW=FLOAT を使用して、ブランク以外のタイトル間でのみスペースを分配します。

参照項目: 行タイトルのスペースを制御する例については、*PROC TABULATE by Example, Second Edition* を参照してください。

例 “例 1: 基本的な 2 次元表の作成” (2291 ページ)

STYLE=*style-override*
テーブルセル以外のテーブル部分に使用する 1 つ以上のスタイルの上書きを指定します。たとえば、次のステートメントは、欠損値の背景色が赤になり、ボックスの背景色がオレンジ色になるように指定します。

```
table (region all)*(division all),
      (prodtype all)*(actual*f=dollar10.) /
      misstext=[label='Missing' style=[background=red]]
      box=[label='Region by Division and Type'  style=[backgroundcolor=orange]];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。

スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override の形式は次のとおりです。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1*

<style-attribute-name-2=style-attribute-value-2 ...>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。

データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で中かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、"出力形式を使用してスタイル要素を割り当てる" (2275 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

別名 S=

ヒント TABLE ステートメントでオプションとして行われるスタイル要素指定を無効にする場合、STYLE=を TABLE ステートメントの次元式で指定します。

角かっこ([と])の代わりに中かっこ({と})を使用できます。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

例 “例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

STYLE_PRECEDENCE=PAGE | ROW | COLUMN
ページ次元(PAGE)、行次元(ROW)または列次元(COLUMN または COL)に対して指定されるスタイルがテーブルセルのコンテンツに適用されるかどうかを指定します。

PAGE
ページ次元に対して指定されるスタイルがテーブルセルのコンテンツに適用されるかどうかを指定します。

ROW
行次元に対して指定されるスタイルがテーブルセルのコンテンツに適用されるかどうかを指定します。

COLUMN
列次元に対して指定されるスタイルがテーブルセルのコンテンツに適用されるかどうかを指定します。

別名 COL

デフォルト COLUMN

例 “例 14: スタイル優先” (2347 ページ)

詳細

次元式について

次元式は、次元を形成する変数、変数値、統計量の組み合わせを指定することにより、次元(テーブルの列、行、ページ)のコンテンツと表示を定義します。TABLE ステートメントは、カンマで区切られた 1 つから 3 つの次元式から構成されています。オプションは、次元式の後に続きます。

3 つの次元がすべて指定される場合、最左の次元式がページを、中央の次元式が行を、最右の次元式が列をそれぞれ定義します。2 つの次元が指定される場合、左の次元式が行を、右の次元式が列をそれぞれ定義します。1 つの次元が指定される場合、次元式は列を定義します。

次元式は、1 つ以上の要素と演算子から構成されています。

次元式で使用できる要素

analysis variables
(See the VAR ステートメント (2249 ページ).)

class variables
(See the CLASS ステートメント (2219 ページ).)

the universal class variable ALL
同じ挿入グループまたは次元(変数 ALL が挿入グループに含まれていない場合)の分類変数のすべてのカテゴリを要約します。

注: 入力データセットに ALL という変数が含まれている場合、共通分類変数の名前を引用符で囲みます。

例 “例 6: 共通分類変数 ALL を使用した情報の要約” (2305 ページ)

“例 8: 複数ページテーブルの作成” (2310 ページ)

“例 12: 分母定義を使用し、基本的な度数カウントとパーセントを表示する” (2328 ページ)

keywords for statistics
使用可能な統計量のリストについては、“PROC TABULATE で使用できる統計量” (2256 ページ)を参照してください。アスタリスク(*)演算子を使用して、統計量キーワードと変数を関連付けます。N 統計量(非欠損値の数)は、変数と関連付けずに次元式で指定できます。

デフォルト　分析変数の場合、デフォルトの統計量は SUM です。その他の場合、デフォルトの統計量は N です。

制限事項　N 以外の統計量キーワードは、分析変数と関連付ける必要があります。

操作　キーワード要素が変数としてではなく統計量キーワードとして扱われるように、統計量キーワードは単一引用符または二重引用符で囲む必要が

あります。デフォルトでは、これらのキーワードは変数として扱われます。

例 n
Region*n
Sales*max

例 “例 9: 複数回答式の調査データに基づくレポート作成” (2313 ページ)

“例 12: 分母定義を使用し、基本的な度数カウントとパーセントを表示する” (2328 ページ)

format modifiers
セルの値をフォーマットする方法を定義します。アスタリスク(*)演算子を使用して、出力形式修飾子とフォーマットするセルを生成する要素(分析変数または統計量)を関連付けます。出力形式修飾子の形式は、次のとおりです。

f=*format*

ヒント 出力形式修飾子は、CLASS 変数に影響しません。

参照項目: テーブルにおける出力形式指定の詳細については、“テーブルの値のフォーマッティング” (2259 ページ)を参照してください。

例 Sales*f=dollar8.2

例 “例 6: 共通分類変数 ALL を使用した情報の要約” (2305 ページ)

labels
変数と統計量の名前を一時的に置き換えます。ラベルは、ラベルの直前の変数または統計量にのみ影響します。ラベルの形式は、次のとおりです。

statistic-keyword-or-variable-name='label-text'

ヒント PROC TABULATE はブランクの列ヘッダーのスペースをテーブルから 削除しますが、デフォルトではすべての行ヘッダーがブランクでない限り、ブランクの行ヘッダーのスペースは削除しません。ブランクの行ヘッダーのスペースを 削除するには、TABLE ステートメントで ROW=FLOAT を使用します。

例 Region='Geographical Region'
Sales*max='Largest Sale'

例 “例 5: 行と列のヘッダーのカスタマイズ” (2303 ページ)

“例 7: 行ヘッダーの削除” (2308 ページ)

style specifications
ページ次元テキスト、ヘッダー、データセルにスタイル要素とスタイル属性を指定します。詳細については、“ 次元式でのスタイル属性とスタイル要素の指定” (2248 ページ)を参照してください。

次元式で使用できる演算子

asterisk *
分類変数の値の組み合わせからカテゴリを作成し、次元に適切なヘッダーを構成します。要素のうちいずれかが分析変数である場合、分析変数の統計量が分類変数によって作成されるカテゴリに対し計算されます。このプロセスをクロスといいます。

例 Region*Division
Quarter*Sales*f=dollar8.2

例 “例 1: 基本的な 2 次元表の作成” (2291 ページ)

(blank)
各要素の出力が、先行する要素の出力の直後に置かれます。このプロセスを連結といいます。

例 n Region*Sales ALL

例 “例 6: 共通分類変数 ALL を使用した情報の要約” (2305 ページ)

parentheses ()
要素をグループ化し、演算子とグループの連結要素をそれぞれ関連付けます。

例 Division*(Sales*max Sales*min)
(Region ALL)*Sales

例 “例 6: 共通分類変数 ALL を使用した情報の要約” (2305 ページ)

angle brackets <>
分母定義を指定します。これによってパーセントの計算での分母の値が決定されます。分母定義を構成する方法については、“パーセントの計算 ” (2260 ページ)を参照してください。

例 “例 9: 複数回答式の調査データに基づくレポート作成” (2313 ページ)

“例 12: 分母定義を使用し、基本的な度数カウントとパーセントを表示する” (2328 ページ)

次元式でのスタイル属性とスタイル要素の指定

1 つ以上のスタイル要素またはスタイル属性を次元式に指定して、LISTING 以外の出力先の表示を制御できます。次の領域の表示を変更できます。

- 分析変数名のヘッダー
- 分類変数名ヘッダー
- 分類変数の水準値ヘッダー
- データセル
- キーワードのヘッダー
- ページ次元テキスト

スタイル属性またはスタイル要素を次元式に指定すると、PROC TABULATE、CLASS、CLASSLEV、KEYWORD、TABLE、または VAR ステートメントなど、別のス

テートメントに指定したスタイル属性またはスタイル要素を無効にする場合に便利です。

スタイル要素とスタイル属性を次元式に指定する構文は次のとおりです。

[STYLE<(CLASSLEV)>=<*style-element-name* | PARENT>

[*style-attribute-name-1=style-attribute-value-1<style-attribute-name-2=style-attribute-value-2 ...*>]]

これらは、次元式にあるスタイル属性の例の一部です。

-

  ```
  dept={label='Department'
      style=[color=red]}, N
  ```

- dept*[style=*MyDataStyle*], N
- dept*[format=12.2 style=*MyDataStyle*], N

注: 次元式で使用する場合、STYLE=オプションを角かっこ([と])または中かっこ({と})で囲む必要があります。

(CLASSLEV)
分類変数水準値ヘッダーにスタイル要素を割り当てます。たとえば、次の TABLE ステートメントでは、分類変数 DEPT の水準値ヘッダーの前景色が黄色になるように指定します。

```
table dept=[style(classlev)=
        [color=yellow]]*sales;
```

注: CLASSLEV オプションは、 次元式でのみ使用されます。

次元式内でスタイル要素を指定する方法を示す例については、“例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)を参照してください。PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

VAR ステートメント

分析変数として使用する数値変数を識別します。

別名: VARIABLE
VARIABLES

ヒント: 複数の VAR ステートメントを使用できます。

例: “例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

構文

VAR *analysis-variable(s)* </ *options*>;

必須引数

analysis-variable(s);
テーブルの分析変数を識別します。分析変数は、PROC TABULATE が統計量を計算する数値変数です。分析変数の値は、連続または不連続となります。

オブザベーションに分析変数に対する欠損値が含まれている場合、PROC TABULATE は N (非欠損変数値を含むオブザベーション数)と NMISS (欠損変数値を含むオブザベーション数)以外のすべての 統計量の計算からの値を省略します。たとえば、欠損値によって SUM は増加しません。欠損値は MEAN などの統計量の計算時にカウントされません。

操作 変数名と統計量名が同じ場合、統計量名を単一引用符か二重引用符で囲みます。

スラッシュ区切り文字の後に使用するオプション

STYLE=*style-override(s)*
分類変数名ヘッダーに 1 つ以上のスタイルの上書きを指定します。

たとえば、次のステートメントは、分析変数名ヘッダーの背景色が黄褐色になるように指定します。

```
var actual  / style=[background=tan];
```

style-override
レポートの特定の領域におけるデフォルトのスタイル要素と属性を無効にする 1 つ以上のスタイル属性またはスタイル要素を指定します。

スタイルの無効化は次の 3 つの方法で指定できます。

- スタイル要素を指定します。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。
- スタイル属性を指定します。スタイル属性は、出力の 1 つの領域の 1 つの動作または視覚側面を表す名前と値のペアです。これは、出力の表示を変更する最も明確な方法です。
- PARENT 値を指定します。PARENT 値では、データセルがその親ヘッダーのスタイル要素を使用することを指定します。

style-override の形式は次のとおりです。

PARENT | *style-element-name* | [*style-attribute-name-1=style-attribute-value-1*

<*style-attribute-name-2=style-attribute-value-2* ...>]

<PARENT>
データセルがその親ヘッダーのスタイル要素を使用することを指定します。

データセルの親スタイル要素は次のいずれになります。

- テーブルで行次元が指定されない場合、またはテーブルでスタイル要素が列の次元式で指定される場合の、データセルを含む列の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素が行の次元式で指定される場合の、セルを含む行の上のリーフヘッダーのスタイル要素
- テーブルでスタイル要素がページの次元式で指定される場合の、Beforecaption スタイル要素
- その他の場合は未定義

注: この使用法では、文字 PARENT を山かっこで囲む必要があります。構文で中かっこまたは角かっこを代わりに使用することはできません。

注: ヘッダーの親(PROC TABULATE ステートメントの STYLE=には適用不可)は、現在のヘッダーがネストされているヘッダーです。

style-attribute-name
変更する属性を指定します。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-attribute-value
属性に値を指定します。属性にはそれぞれ異なる有効値のセットが含まれています。SAS 出力形式を、条件付きフォーマットの属性値として使用することもできます。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

SAS 出力形式をスタイル属性値として使用する方法については、"出力形式を使用してスタイル要素を割り当てる" (2275 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、"テーブル領域のスタイル要素とスタイル属性" (2271 ページ)を参照してください。

style-element-name
ODS スタイルテンプレートの各部分を表すスタイル要素の名前。

参照項目: PROC TABULATE でのスタイルの使用については、"PROCTABULATE での ODS スタイルの使用" (2264 ページ)を参照してください。

各 ODS 出力先のデフォルトのスタイル属性およびスタイル要素のテーブルについては、“テーブル領域のスタイル要素とスタイル属性” (2271 ページ)を参照してください。

別名 S=

ヒント VAR ステートメントで指定されるスタイル要素を無効にする場合、スタイル要素を関連する TABLE ステートメント次元式で指定できます。

VAR ステートメントでの STYLE=の使用は、PROC TABULATE ステートメントでの使用とはやや異なります。VAR ステートメントでは、継承は行と列では異なります。行の場合、親ヘッダーは現在のヘッダーの左側に置かれます。列の場合、親ヘッダーは現在のヘッダーの上に置かれます。

参照項目: PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

例 “例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)

WEIGHT=*weight-variable*
その値が VAR ステートメントで指定される変数の値に重みを付ける数値変数を指定します。重み変数が整数である必要がないのは、

PROC TABULATE は、次のテーブルに従って重み値に応答します。

表 61.3 重み値に対する PROC TABULATE の応答

重み値	PROC TABULATE 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 より小さい	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

制限事項 重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。

注 バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

ヒント WEIGHT=オプションを使用する場合、適切な VARDEF=オプションの値を考慮します。“VARDEF=DF | N | WDF | WEIGHT” (2214 ページ)の説明を参照してください。

複数の VAR ステートメントで WEIGHT オプションを使用して、分析変数に対し異なる重みを指定します。

WEIGHT ステートメント

統計量計算の分析変数に対し重みを指定します。

別名: WGT

参照項目: 重み付き統計量の計算と、WEIGHT ステートメントを使用する例については、“重み付き統計量の計算” (2185 ページ)を参照してください。

構文

WEIGHT *variable*;

必須引数

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

PROC TABULATE は、次のテーブルに従って重み値に応答します。

表 61.4 重み値に対する PROC TABULATE の応答

重み値	PROC TABULATE 応答
0	オブザベーションをオブザベーションの合計数にカウントします。
0 より小さい	値をゼロに変換し、オブザベーションをオブザベーションの合計数にカウントします。
欠損値	オブザベーションを除外します

負またはゼロの重みを含むオブザベーションを分析から除外するには、EXCLNPWGT を使用します。PROC GLM などのほとんどの SAS/STAT プロシジャは、デフォルトで負とゼロの重みを除外します。

注: バージョン 7 より前の SAS では、プロシジャは重みがないオブザベーションをオブザベーションのカウントから除外しませんでした。

制限事項 重み付き分位点を計算するには、PROC ステートメントで QMETHOD=OS を使用します。

重み変数がアクティブの場合、PROC TABULATE は MODE を計算しません。代わりに、MODE の計算が必要で、重み変数がアクティブの場合は、PROC UNIVARIATE を使用しようとします。

操作 VAR ステートメントで WEIGHT=オプションを使用して重み変数を指定すると、PROC TABULATE はこの変数を代わりに使用して、これらの VAR ステートメント変数に重みを付けます。

ヒント WEIGHT ステートメントを使用する場合、VARDEF=オプションのどの値が適切かを考慮します。“VARDEF=DF | N | WDF | WEIGHT” (2214 ページ)と重み付き統計量の計算の説明については、このドキュメントの“キーワードと式” (2410 ページ)セクションを参照してください。

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE=データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD
MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

使用法: TABULATE プロシジャ

PROC TABULATE で使用できる統計量

次のキーワードを使用して、TABLE ステートメントで統計量を要求するか、KEYWORD ステートメントまたは KEYLABEL ステートメントで統計量キーワードを指定します。

注: 変数名(分類または分析)と統計量名が同じ場合、統計量名を単一引用符で囲みます。(例:'MAX')

記述統計量キーワード	
COLPCTN	PCTSUM
COLPCTSUM	RANGE
CSS	REPPCTN
CV	REPPCTSUM
KURTOSIS | KURT	ROWPCTN
LCLM	ROWPCTSUM
MAX	SKEWNESS | SKEW
MEAN	STDDEV | STD
MIN	STDERR
MODE	SUM
N	SUMWGT
NMISS	UCLM
PAGEPCTN	USS
PAGEPCTSUM	VAR
PCTN	
四分位範囲統計量キーワード	
MEDIAN | P50	Q3 | P75
P1	P70
P5	P80
P10	P90
P20	P95
P30	P99
P40	

P60	
Q1|P25	QRANGE
仮説検定キーワード	
PROBT | PRT	T

これらの統計量、その計算に使用される式、データ要件については、このドキュメントの“キーワードと式” (2410 ページ)セクションを参照してください。

平均値の標準誤差(STDERR)またはスチューデントの *t* 検定を計算するには、VARDEF=オプションのデフォルト値、DF を使用する必要があります。VARDEF=オプションは、PROC TABULATE ステートメントで指定されます。

重み付き分位点を計算するには、PROC TABULATE ステートメントで QMETHOD=OS を使用する必要があります。

LCLM と UCLM を使用して、平均の両側信頼限界を計算します。LCLM または UCLM のみを使用して、片側信頼限界を計算します。PROC TABULATE ステートメントで ALPHA=オプションを使用して、信頼水準を指定します。

分類変数のフォーマッティング

FORMAT ステートメントを使用して、出力形式を PROC TABULATE ステップの間隔に対する分類変数に割り当てます。出力形式を分類変数に割り当てると、PROC TABULATE はフォーマットされた値を使用してカテゴリを作成し、フォーマットされた値がヘッダーで使用されます。分類変数に出力形式を指定せず、変数にその他の出力形式が割り当てられていない場合、GROUPINTERNAL オプションが指定されていない限り、デフォルトの出力形式、BEST12.が使用されます。

ユーザー定義の出力形式は、値を少数のカテゴリにグループ化する場合に特に便利です。たとえば、値が 1 から 99 までの分類変数 Age がある場合、テーブルに扱いやすい数のカテゴリが含まれるように、年齢をグループ化するユーザー定義の出力形式を作成できます。次の PROC FORMAT ステップでは、年齢の可能な値をすべて 6 つの値グループに要約する出力形式を作成します。

```
proc format;
  value agefmt  0-29='Under 30'
                30-39='30-39'
                40-49='40-49'
                50-59='50-59'
                60-69='60-69'
                other='70 or over';
run;
```

ユーザー定義の出力形式作成の詳細については、26 章, “FORMAT プロシジャ,” (859 ページ)を参照してください。

デフォルトでは、PROC TABULATE は度数カウントがゼロでなく、値が欠損していない出力形式のみテーブルに含めます。すべての分類変数の欠損値を出力に含める

には、PROC TABULATE ステートメントで MISSING オプションを使用します。選択した分類変数の欠損値を含めるには、CLASS ステートメントで MISSING オプションを使用します。度数カウントがゼロの出力形式を含めるには、CLASS ステートメントで PRELOADFMT オプション、TABLE ステートメントで PRINTMISS オプション、または PROC TABULATE ステートメントで CLASSDATA=オプションを使用します。

テーブルの値のフォーマッティング

テーブルセルのデータ用出力形式には、2 つの目的があります。PROC TABULATE による値の表示方法の決定と列幅の決定です。テーブルセルの値に対するデフォルトの出力形式は、12.2 です。次を実行して、テーブルセルの値を印刷するための出力形式を変更できます。

- PROC TABULATE ステートメントの FORMAT=オプションでデフォルトの出力形式を変更する
- F=出力形式修飾子を使用して TABLE ステートメントで要素をクロスする

注: FORMAT ステートメントまたは ATTRIB ステートメントを使用して、テーブルセルの値を印刷するための出力形式を変更することはできません。これらのステートメントを使用すると、それらに含まれる分析変数の出力形式が無視されます。

PROC TABULATE は、出力形式に対する次のデフォルトの優先順位から特定のセルに使用する出力形式を決定します。

1. その他の出力形式が指定されていない場合、PROC TABULATE はデフォルトの出力形式(12.2)を使用します。
2. PROC TABULATE ステートメントの FORMAT=オプションは、デフォルトの出力形式を変更します。出力形式修飾子がセルに影響しない場合、 PROC TABULATE はこの出力形式をセルの値に使用します。
3. ページ次元の出力形式修飾子は、行次元または列次元のセルに対し別の出力形式修飾子を指定しない限り、論理ページのすべてのテーブルセルの 値に適用されます。
4. 行次元の出力形式修飾子は、列次元のセルに別の出力形式修飾子を指定しない限り、行のすべてのテーブルセルの 値に適用されます。
5. 列次元の出力形式修飾子は、列のすべてのテーブルセルの 値に適用されます。

この優先順位は、TABLE ステートメントで FORMAT_PRECEDENCE=オプションを使用して、変更できます。詳細については、“TABLE ステートメント” (2235 ページ)を参照してください。たとえば、FORMAT_PRECEDENCE=ROW を指定し、行次元で出力形式修飾子を指定すると、その出力形式はテーブルセルに指定されているその他すべての 出力形式に優先します。

パーセントの計算

単一のテーブルセルの値のパーセントを計算する

次の統計量は、セルグループの値の合計に対する単一のテーブルセルの値のパーセントを印刷します。分母定義は不要です。ただし、分析変数がパーセント合計統計量の分母定義として使用できます。

- REPPCTN 統計量と REPPCTSUM 統計量 — レポートの値の合計に対する単一のテーブルセルの値のパーセントを印刷します。
- COLPCTN 統計量と COLPCTSUM 統計量 — 列の値の合計に対する単一のテーブルセルの値のパーセントを印刷します。
- ROWPCTN 統計量と ROWPCTSUM 統計量 — 行の値の合計に対する単一のテーブルセルの値のパーセントを印刷します。
- PAGEPCTN 統計量と PAGEPCTSUM 統計量 — ページの値の合計に対する単一のテーブルセルの値のパーセントを印刷します。

これらの統計量は、通常使用されるパーセントを計算します。例については、“例 11: さまざまなパーセント表示の統計量の計算” (2324 ページ)を参照してください。

PCTN と PCTSUM の使用

PCTN 統計量と PCTSUM 統計量は、これらの同じパーセントの計算に使用できます。これらを使用して、分母を手動で定義できます。PCTN 統計量と PCTSUM 統計量は、別のテーブルセルの値(パーセントの計算の分母で使用される)、またはセルグループの値の合計に対する単一テーブルセルの値のパーセントを印刷します。デフォルトでは、PROC TABULATE はすべての N セル(PCTN 統計量の場合)またはすべての SUM セル(PCTSUM 統計量の場合)の値を要約し、要約した値を分母に使用します。PROC TABULATE が分母に使用する値を分母定義によって制御できます。

分母定義は、PCTN 統計量または PCTSUM 統計量の隣に山かっこ(<と>)で囲みます。分母定義は、分母に対して合計するカテゴリを指定します。

このセクションでは、単純なテーブルで分母定義を指定する方法について説明します。“例 12: 分母定義を使用し、基本的な度数カウントとパーセントを表示する” (2328 ページ)では、複数のサブテーブルで構成されるテーブルで分母定義を指定する方法について説明します。分母定義の例については、*PROC TABULATE by Example, Second Edition* を参照してください。

PCTN 統計量に分母を指定する

次の PROC TABULATE ステップでは、データセット“ENERGY” (2488 ページ)を使用して N 統計量と PCTN の 3 つの異なるバージョンを計算します。

```
proc tabulate data=energy;
  class division type;
  table division*
        (n='Number of customers'
         pctn<type>='% of row' 1
         pctn<division>='% of column' 2
         pctn='% of all customers'), 3
       type/rts=50;
  title 'Number of Users in Each Division';
run;
```

TABLE ステートメントは、Division の値ごとに 1 行、Type の値ごとに 1 列を作成します。各行内で、TABLE ステートメントは N と、PCTN の 3 つの異なる計算の合計 4 つの統計量をネストします。(次の図を参照。)PCTN の発生ごとに異なる分母定義が使用されます。

図 61.4 度数カウントがハイライト表示された PCTN 統計量の 3 つの異なる使用

Number of Users in Each Division

		Type	
		1	2
Division			
1	Number of customers	6.00	6.00
	% of row 1	50.00	50.00
	% of column 2	27.27	27.27
	% of all customers 3	13.64	13.64
2	Number of customers	3.00	3.00
	% of row	50.00	50.00
	% of column	13.64	13.64
	% of all customers	6.82	6.82
3	Number of customers	8.00	8.00
	% of row	50.00	50.00
	% of column	36.36	36.36
	% of all customers	18.18	18.18
4	Number of customers	5.00	5.00
	% of row	50.00	50.00
	% of column	22.73	22.73
	% of all customers	11.36	11.36

1 <type>は、Division の同じ値内の Type のすべての発生に対する度数カウントを合計します。したがって、Division=1 の場合、分母は 6 + 6、または 12 となります。

2 <division>は、Type の同じ値内の Division のすべての発生に対する度数カウントを合計します。したがって、Type=1 の場合、分母は 6 + 3 + 8 + 5、または 22 となります。

3 PCTN の 3 番目の使用には、分母定義はありません。分母定義の省略は、すべての分類変数を分母定義に含めることと同じです。したがって、すべてのセルについて、分母は 6 + 3 + 8 + 5 + 6 + 3 + 8 + 5、または 44 となります。

PCTSUM 統計量に分母を指定する

次の PROC TABULATE ステップでは、 Type と Division の各組み合わせに対する支出額を合計し、PCTSUM の 3 つの異なるバージョンを計算します。

```
proc tabulate data=energy format=8.2;
  class division type;
  var expenditures;
  table division*
        (sum='Expenditures'*f=dollar10.2
         pctsum<type>='% of row' 1
         pctsum<division>='% of column' 2
         pctsum='% of all customers'), 3
       type*expenditures/rts=40;
  title 'Expenditures in Each Division';
run;
```

TABLE ステートメントは、Division の値ごとに 1 行、Type の値ごとに 1 列を作成します。Type が Expenditures とクロスするため、各セルの値が、セルに影響するすべてのオブザベーションに対する Expenditures の値の合計となります。各行内で、TABLE ステートメントは SUM と、PCTSUM の 3 つの異なる計算の合計 4 つの統計量をネストします。(次の図を参照。)PCTSUM の発生ごとに異なる分母定義が使用されます。

図 61.5 合計がハイライト表示された PCTSUM 統計量の 3 つの異なる使用

Expenditures in Each Division

		Type	
		1	2
		Expenditu-res	Expenditu-res
Division			
1	Expenditures	$7,477.00	$5,129.00
	% of row 1	59.31	40.69
	% of column 2	16.15	13.66
	% of all customers 3	8.92	6.12
2	Expenditures	$19,379.00	$15,078.00
	% of row	56.24	43.76
	% of column	41.86	40.15
	% of all customers	23.11	17.98
3	Expenditures	$5,476.00	$4,729.00
	% of row	53.66	46.34
	% of column	11.83	12.59
	% of all customers	6.53	5.64
4	Expenditures	$13,959.00	$12,619.00
	% of row	52.52	47.48
	% of column	30.15	33.60
	% of all customers	16.65	15.05

1 <type>は、Division の同じ値内の Type のすべての発生に対し Expenditures の値を合計します。したがって、Division=1 の場合、分母は$7,477 + $5,129 となります。

2 <division>は、Type の同じ値内の Division のすべての発生に対する度数カウントを合計します。したがって、Type=1 の場合、分母は$7,477 + $19,379 + $5,476 + $13,959 となります。

3 PCTN の 3 番目の使用には、分母定義はありません。分母定義の省略は、すべての分類変数を分母定義に含めることと同じです。したがって、Type=1 の場合、分母は $7,477 + $19,379 + $5,476 + $13,959 + $5,129 + $15,078 + $4,729 + $12,619 となります。

PROCTABULATE での ODS スタイルの使用

Base SAS プロシジャでのスタイルの使用

ODS をサポートする Base SAS プロシジャの多くは、1 つ以上のテーブルテンプレートを使用して出力オブジェクトを生成します。これらのテーブルテンプレートには、テーブル要素(列、ヘッダー、フッター)に対するテンプレートが含まれています。各テーブル要素で、出力のさまざまな部分に 1 つ以上のスタイル要素を使用することを指定できます。これらのスタイル要素はプロシジャの構文内で指定することはできませんが、使用する ODS 出力先に合わせてカスタマイズされたスタイルを使用できます。テーブルとスタイルのカスタマイズの詳細については、“TEMPLATE Procedure: Creating a Style Template” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

Base SAS レポートプロシジャの PROC PRINT、PROC REPORT および PROC TABULATE を使用してデータをすばやく分析し、読みやすいテーブルに整理することができます。これらのプロシジャステートメントで STYLE=オプションを使用してレポートの表示を変更できます。STYLE=オプションを使用すると、すべての出力のデフォルトスタイルを変更せずに、出力の各セションに変更を加えることができます。プロシジャ内の特定のステートメントで STYLE=オプションを指定して、プロシジャ出力の特定のセクションをカスタマイズすることができます。

次のプログラムでは、STYLE=オプションを使用して以下の PROC TABULATE 出力の色を作成します。

```
ods _all_ close;
ods html5 file="file-path/customStyle.html";
proc sort data=sashelp.prdsale out=prdsale;
   by Country;
run;

proc tabulate data=prdsale;
  class region division prodtype / style=[background=lightgreen];
  classlev region division prodtype  / style=[background=yellow];
  var actual  / style=[background=tan];
  keyword all sum / style=[background=linen color=blue];
  keylabel all='Total';
  table (region all)*(division all),
       (prodtype all)*(actual*f=dollar10.) /
       box=[label='Region by Division and Type' style=[backgroundcolor=orange]];

title 'Actual Product Sales';
title2 '(millions of dollars)';
run;
ods _all_ close;
```

アウトプット 61.4　拡張された PROC TABULATE 出力

Actual Product Sales
(millions of dollars)

Region by Division and Type		Product type		Total
		FURNITURE	OFFICE	
		Actual Sales	Actual Sales	Actual Sales
		Sum	Sum	Sum
Region	Division			
EAST	CONSUMER	$72,570	$108,686	$181,256
	EDUCATION	$73,901	$115,104	$189,005
	Total	$146,471	$223,790	$370,261
WEST	Division			
	CONSUMER	$76,209	$105,020	$181,229
	EDUCATION	$67,945	$110,902	$178,847
	Total	$144,154	$215,922	$360,076
Total	Division			
	CONSUMER	$148,779	$213,706	$362,485
	EDUCATION	$141,846	$226,006	$367,852
	Total	$290,625	$439,712	$730,337

スタイル、スタイル要素およびスタイル属性

スタイル、スタイル要素、およびスタイル属性について

SAS 出力の外観は、ODS スタイルテンプレート(ODS スタイル)によって制御されます。ODS スタイルは、PROC TEMPLATE スタイルの構文で記述されたコンパイル済みの STYLE テンプレートから生成されます。ODS スタイルテンプレートは、SAS 出力に特定の視覚的属性を提供するスタイル要素のコレクションです。

- スタイル要素は、出力の特定の部分に適用されるスタイル属性の名前付きコレクションです。ODS 出力の各領域には、スタイル要素名が関連付けられています。スタイル要素名は、スタイル属性が適用される場所を指定します。たとえば、スタイル要素には、列見出しの表示やセル内のデータの表示に関する指示が含まれる場合があります。スタイル要素は、スタイルを使用する出力のデフォルトの色とフォントを指定する場合もあります。
- スタイル属性は、予約済みの名前と値を使用して ODS で定義される色、フォントプロパティ、線の特性などの視覚的なプロパティです。Style 属性は、スタイルテンプレート内のスタイル要素によってまとめて参照されます。各*スタイル属性*は、プレゼンテーションの 1 つの側面の値を指定します。たとえば、

BACKGROUNDCOLOR=属性は、HTML テーブルの背景色または印刷出力の色付きテーブルの色を指定します。FONTSTYLE=属性は、Roman フォントとイタリックフォントのどちらを使用するかを指定します。

注: スタイルはデータの表示を制御するため DOCUMENT または OUTPUT の出力先に送られる出力オブジェクトには影響しません。

利用可能なスタイルは SASHELP.TMPLMST アイテムストアにあります。PROC TEMPLATE で LIST ステートメントを使用して、使用可能なスタイルテンプレートのリストを表示できます。

```
proc template;
  list styles / store=sashelp.tmplmst;
run;
```

ODS スタイルの表示の詳細については、"Viewing ODS Styles Supplied by SAS" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

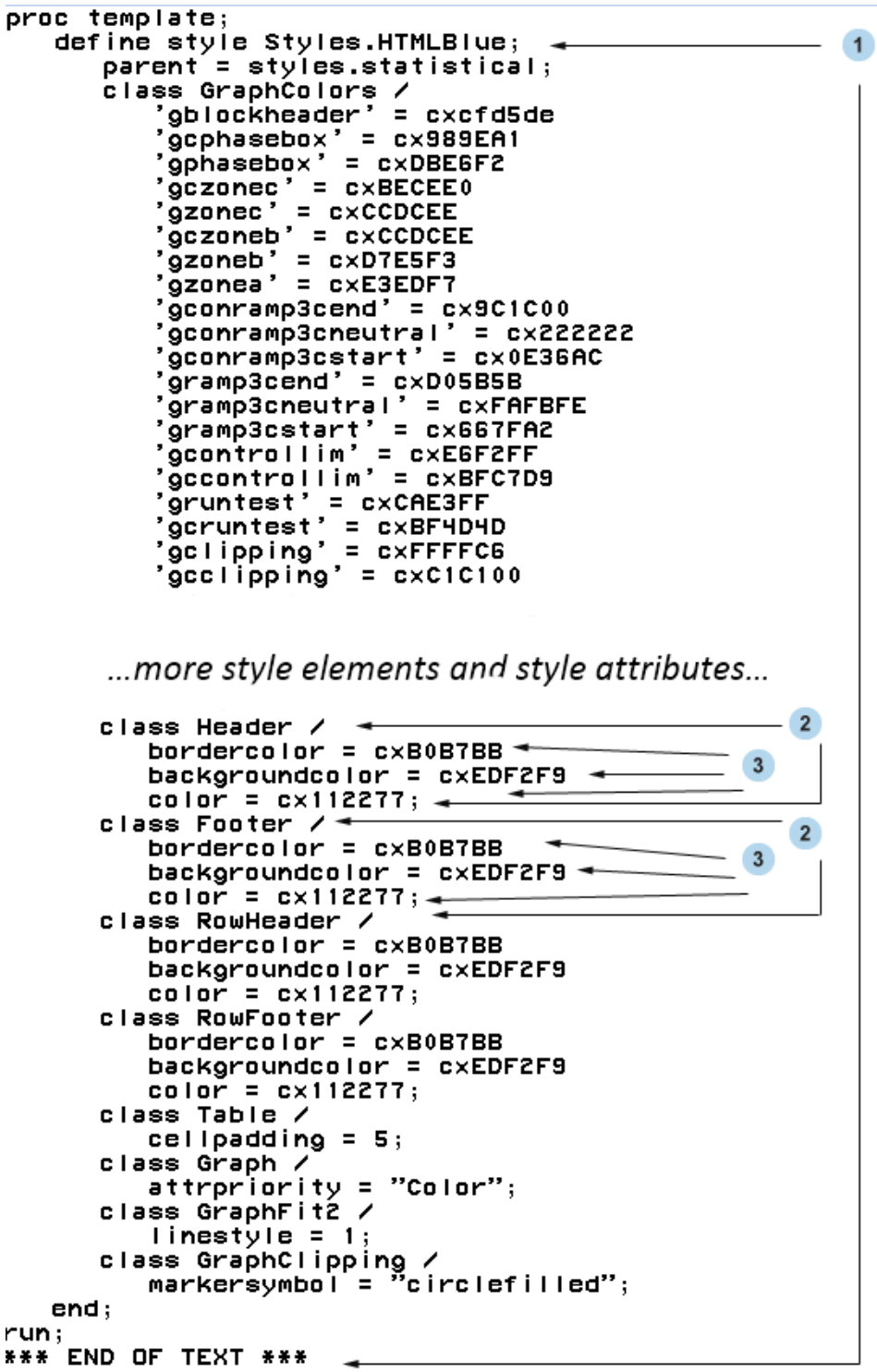
HTML5 出力は、HTMLEncore スタイルテンプレートを使用します。スタイル、スタイル要素、およびスタイル属性に慣れるために、それらの関係に注目してください。

PROC TEMPLATE で SOURCE ステートメントを使用して、スタイルテンプレートの構造を表示できます。次のコードは、HTMLBlue スタイルテンプレートの構造を SAS ログに出力します。

```
proc template;
  source styles.HTMLBlue;
run;
```

次の図は、スタイルの構造を示しています。この図は、スタイル、スタイル要素、およびスタイル属性の関係を示しています。

図 61.6 HtmlBlue スタイルの図



次のリストは、前の図の番号付きの項目に対応しています。

1 Styles.HtmlBlue は*スタイル*です。スタイルは、SAS 出力のプレゼンテーションの側面 (色、フォント、フォント サイズなど) を表示する方法を記述します。スタイルは、それを使用する ODS ドキュメントの全体的な外観を決定します。HTML 出力のデフォルトのスタイルは、HtmlBlue です。各スタイルは、スタイル要素で構成されています。

"DEFINE STYLE Statement" (*SAS Output Delivery System: Procedures Guide*)を使用して新しいスタイルを作成できます。 新しいスタイルは、独立して作成することも、既存のスタイルから作成することもできます。"PARENT= Statement" (*SAS Output Delivery System: Procedures Guide*)を使用して既存のスタイルから新しいスタイルを作成することができます。ODS スタイルの表示の詳細については、"Style Templates" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

2 ヘッダーとフッターは*スタイル要素*の例です。スタイル要素は、SAS プログラムの出力の特定の部分に適用されるスタイル属性のコレクションです。たとえば、スタイル要素には、列見出しの表示やテーブルセル内のデータの表示に関する指示が含まれる場合があります。スタイル要素は、そのスタイルを使用する出力のデフォルトの色とフォントを指定する場合もあります。スタイル要素はスタイル内に存在し、1 つ以上のスタイル属性で構成されます。スタイル要素は、ユーザーが定義するか、SAS によって提供されます。ユーザー定義のスタイル要素は、"STYLE Statement" (*SAS Output Delivery System: Procedures Guide*)で作成できます。

注: HTML およびマークアップ言語に使用されるデフォルトのスタイル要素とその継承のリストについては、"Style Elements" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

3 BORDERCOLOR=、BACKGROUNDCOLOR=、および COLOR=は、*スタイル属性*の例です。スタイル属性は、そのスタイル要素が適用される出力領域の 1 つの側面の値を指定します。たとえば、COLOR=属性は、フォントの色に値 **cx112277** を指定します。SAS が提供するスタイル属性のリストについては、"Style Attributes" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

スタイル属性は、スタイル参照で参照できます。スタイル参照の詳細については、"style-reference" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

次の表は、PROC PRINT、PROC TABULATE、および PROC REPORT で STYLE=オプションを使用して設定できる、一般的に使用されるスタイル属性を示しています。これらの属性のほとんどは、セル以外の表の部分に適用されます (たとえば、表の境界線、列と行の間の線など)。すべての出力先ですべての属性が有効であるとは限らないことに注意してください。これらのスタイル属性、その有効な値、およびその適用可能な出力先の詳細については、"Style Attributes Tables" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

表 61.5 PROC REPORT、PROC TABULATE および PROC PRINT のスタイル属性

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
ASIS=	X	X		X		X
BACKGROUNDCOLOR=	X	X	X	X	X	X
BACKGROUNDIMAGE=	X	X	X	X	X	X
BORDERBOTTOMCOLOR=	X	X		X		
BORDERBOTTOMSTYLE=	X	X	X	X		
BORDERBOTTOMWIDTH=	X	X	X	X		
BORDERLEFTCOLOR=	X	X		X		
BORDERLEFTSTYLE=	X	X	X	X		
BORDERLEFTWIDTH=	X	X	X	X		
BORDERCOLOR=	X	X		X	X	X
BORDERCOLORDARK=	X	X	X	X	X	X
BORDERCOLORLIGHT=	X	X	X	X	X	X
BORDERRIGHTCOLOR=	X	X		X		
BORDERRIGHTSTYLE=	X	X	X	X		
BORDERRIGHTWIDTH=	X	X	X	X		

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
BORDERTOPCOLOR=	X	X		X		
BORDERTOPSTYLE=	X	X	X	X		
BORDERTOPWIDTH=	X	X	X	X		
BORDERWIDTH=	X	X	X	X	X	X
CELLPADDING=	X		X		X	
CELLSPACING=	X		X		X	
CELLWIDTH=	X	X	X	X		X
CLASS=	X	X	X	X	X	X
COLOR=	X	X	X			
FLYOVER=	X	X		X		X
FONT=	X	X	X	X	X	X
FONTFAMILY=	X	X	X	X	X	X
FONTSIZE=	X	X	X	X	X	X
FONTSTYLE=	X	X	X	X	X	X
FONTWEIGHT=	X	X	X	X	X	X
FONTWIDTH=	X	X	X	X		X
FRAME=	X		X		X	
HEIGHT=	X	X		X	X	X
HREFTARGET=		X		X		X
HTMLSTYLE=	X	X	X	X	X	
NOBREAKSPACE=[2]	X	X		X		X
OUTPUTWIDTH=	X	X	X	X	X	

属性	PROC REPORT ステートメントの REPORT 領域	PROC REPORT 領域: CALLDEF, COLUMN, HEADER, LINES, SUMMARY	PROC TABULATE ステートメントのテーブル	PROC TABULATE ステートメントの VAR、CLASS、BOX、CLASSLEV キーワード	PROC PRINT のテーブルの場所	PROC PRINT: テーブル以外のすべての場所
POSTHTML=[1]	X	X	X	X	X	X
POSTIMAGE=	X	X	X	X	X	X
POSTTEXT=[1]	X	X	X	X	X	X
PREHTML=[1]	X	X	X	X	X	X
PREIMAGE=	X	X	X	X	X	X
PRETEXT=[1]	X	X	X	X	X	X
PROTECTSPECIALCHARS=		X		X	X	X
RULES=	X		X		X	
TAGATTR=	X	X	X	X	X	X
TEXTALIGN=	X	X	X	X	X	X
URL=		X		X		X
VERTICALALIGN=		X		X		X
WIDTH=	X	X	X	X	X	

1 これらの属性をこの場所で使用する場合には、属性 PRETEXT=、POSTTEXT=、PREHTML=、POSTHTML=で指定されるテキストにのみ影響します。表に表示されるテキストの前景色またはフォントを変更するには、表ではなくセルに影響する場所で対応する属性を設定する必要があります。スタイル属性とその値の詳細な説明については、"Style Attributes" (*SAS Output Delivery System: Procedures Guide*)を参照してください。

2 PROC REPORT を ODS RTF 出力先で使用する場合、長いテキスト文字列の予期せぬ折り返しを防ぐために、LINE ステートメントに影響する場所で NOBREAKSPACE=OFF を設定してください。NOBREAKSPACE=OFF 属性は、PROC REPORT コード内の LINE ステートメントまたは style(line)が指定されている PROC REPORT ステートメントのいずれかに設定する必要があります。

テーブル領域のスタイル要素とスタイル属性

次のテーブルに、PROC TABULATE テーブルのさまざまな部分に対するデフォルトのスタイル要素およびスタイル属性を示します。

このテーブルには、最もよく使用される ODS 出力先のデフォルトを示します。HTML、PDF および RTF のデフォルトがリストされています。それぞれの出力先には、出力先に書き込まれるすべての出力に適用されるデフォルトのスタイルテンプ

レートが含まれます。ODS 出力先とそのデフォルトのスタイルに関する詳細なドキュメントは、“Style Templates” (*SAS Output Delivery System: Procedures Guide*)を参照してください。

注:

SAS Studio での ODS 出力のデフォルトのスタイルは、SAS Studio がアクティブに使用しているテーマによって設定されます。テーマに応じて、デフォルトは Illuminate、Ignite、または HighContrast のいずれかです。詳細については、“Customizing SAS Studio” (*SAS Studio: User's Guide*)を参照してください。

- HTML5 出力のデフォルトスタイルは HTMLEncore です。
- PDF の出力のデフォルトスタイルは Pearl です。
- RTF 出力のデフォルトスタイルは RTF です。

表 61.6 テーブル部分のデフォルトのスタイル要素およびスタイル属性

リージョン	スタイル要素	HTML5 スタイル属性	PDF スタイル属性	RTF スタイル属性
列ヘッダーとボックス	ヘッダー	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb
ページ次元テキスト	Beforecaption	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxfafbfe	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000
行ヘッダー	Rowheader	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = bold	FONTFAMILY = "Arial, 'Albany AMT'" FONTSIZE = 8pt FONTWEIGHT = bold FONTSTYLE = roman	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 11pt

リージョン	スタイル要素	HTML5 スタイル属性	PDF スタイル属性	RTF スタイル属性
		FONTSTYLE = roman COLOR = cx112277 BACKGROUNDCOLOR = cxedf2f9	COLOR =cx000000 BACKGROUNDCOLOR = cxffffff BORDERWIDTH = NaN	FONTWEIGHT = bold FONTSTYLE = roman COLOR =cx000000 BACKGROUNDCOLOR = cxbbbbbb
データセル	Data	FONTFAMILY = "Arial, 'Albany AMT', Helvetica, Helv" FONTSIZE = 2 FONTWEIGHT = medium FONTSTYLE = roman BACKGROUNDCOLOR = cxffffff	FONTFAMILY ="'Albany AMT', Albany" FONTSIZE = 8pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000 BORDERWIDTH = NaN COLOR = cx000000	FONTFAMILY ="'Times New Roman', 'Times Roman'" FONTSIZE = 10pt FONTWEIGHT = medium FONTSTYLE = roman COLOR = cx000000

PROC TABULATE ステートメントで STYLE=オプションを使用する

PROC TABULATE スタイルの上書きは、テーブルを作成するステートメントに基づくテーブルに適用されます。STYLE=オプションを使用すると、変更する領域を制御するステートメント内に STYLE=オプションを指定することにより、出力のその領域の表示を制御するスタイル要素またはスタイル属性を変更できます。次のテーブルに、変更可能なテーブルの領域と、それらに対応するステートメントを示します。

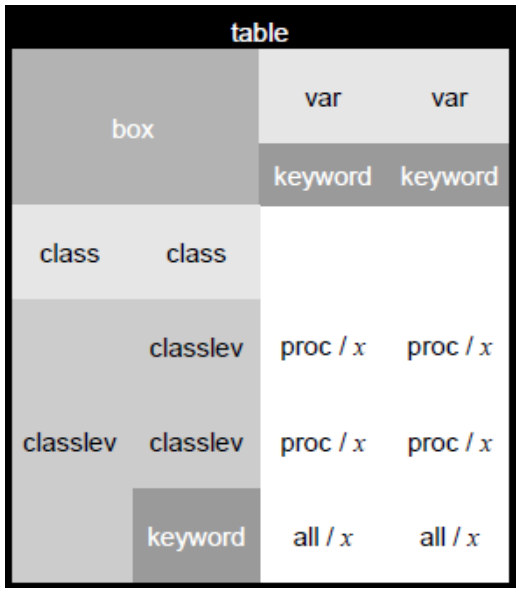
TABLE ステートメントでの指定が、PROC TABULATE、CLASS、CLASSLEV、VAR、および KEYWORD ステートメントでの同じ指定に優先します。これにより、複数の TABLE ステートメントで異なるスタイル動作を行うことができます。ただし、PROC TABULATE ステートメントで指定し、TABLE ステートメントで無効にしないスタイル属性が継承されます。たとえば、PROC TABULATE ステートメントですべてのデータセルに対し青い背景、白い前景を指定し、TABLE ステートメントで特定のクロスのデータセルに対し灰色の背景を指定すると、これらのデータセルの背景が灰色に、前景が白(PROC TABULATE ステートメントでの指定どおり)になります。

STYLE=オプションに関する詳細情報については、各ステートメントのドキュメントを参照してください。

表 61.7　PROC TABULATE で STYLE=オプションを使用する

変更する領域	使用する STYLE オプション
データセル	PROC TABULATE ステートメントまたは次元式 (p. 2202)
ページ次元テキストと分類変数名ヘッダー	“CLASS ステートメント” (p. 2219)
分類水準値ヘッダー	“CLASSLEV ステートメント” (p. 2227)
キーワードのヘッダー	“KEYWORD ステートメント” (p. 2231)
他の場所で指定されていない境界線、ルール、その他の部分	“TABLE ステートメント” (p. 2235)
ボックステキスト	TABLE ステートメントの BOX=オプション (p. 2235)
欠損値	TABLE ステートメントの MISSTEXT=オプション (p. 2235)
分析変数名ヘッダー	“VAR ステートメント” (p. 2249)

図 61.7　PROC TABULATE の領域および対応するステートメント



スタイル属性をテーブルセルに適用する

PROC TABULATE により、特定のセルに使用するスタイル属性がスタイルの次のデフォルトの優先順位から決定されます。

1 その他のスタイル属性が指定されない場合、PROC TABULATE ではデフォルトのスタイル(Data)からのデフォルトのスタイル属性が使用されます。
2 PROC TABULATE ステートメントの STYLE=オプションにより、デフォルトのスタイル属性が変更されます。その他の STYLE=オプション指定がセルに影響しない場合、PROC TABULATE ではこれらのスタイル属性がセルに使用されます。
3 ページ次元で指定される STYLE=オプションは、行次元または列次元のセルに別の STYLE=オプションを指定しない限り、論理ページのすべてのテーブルセルに適用されます。
4 行次元で指定される STYLE=オプションは、列次元のセルに別の STYLE=オプションを指定しない限り、行のすべてのテーブルセルに 適用されます。
5 列次元で指定される STYLE=オプションは、列のすべてのテーブルセルに 適用されます。

この優先順位を変更するには、TABLE ステートメント (2245 ページ)で STYLE_PRECEDENCE=オプションを使用します。たとえば、STYLE_PRECEDENCE=ROW を指定し、行次元で STYLE=オプションを指定すると、これらのスタイル属性値はテーブルセルに指定されているその他すべてに優先します。

出力形式を使用してスタイル要素を割り当てる

出力形式を使用して、コンテンツが分類変数または分析変数の値によって決定されるセルにスタイル属性値を割り当てることができます。たとえば、次のコードは、値が 10,000 未満のセルに赤い背景を割り当て、値が最低 10,000 で 20,000 未満のセルに黄色い背景を割り当て、値が最低 20,000 のセルに緑の背景を割り当てます。

```
proc format;
  value expfmt low-<10000='red'
               10000-<20000='yellow'
               20000-high='green';
run;

ods html5 file='external-HTML-file';
proc tabulate data=energy style=[backgroundcolor=expfmt.];
  class region division type;
  var expenditures;
  table (region all)*(division all),
        type*expenditures;
run;
ods html5 close;
```

次元式でのスタイル属性とスタイル要素の指定

1 つ以上のスタイル要素またはスタイル属性を次元式に指定して、LISTING 以外の出力先の表示を制御できます。次の領域の表示を変更できます。

- 分析変数名のヘッダー
- 分類変数名ヘッダー

- 分類変数の水準値ヘッダー
- データセル
- キーワードのヘッダー
- ページ次元テキスト

スタイル属性またはスタイル要素を次元式に指定すると、PROC TABULATE、CLASS、CLASSLEV、KEYWORD、TABLE、または VAR ステートメントなど、別のステートメントに指定したスタイル属性またはスタイル要素を無効にする場合に便利です。

スタイル要素とスタイル属性を次元式に指定する構文は次のとおりです。

[STYLE<(CLASSLEV)>=<*style-element-name* | PARENT>

[*style-attribute-name-1=style-attribute-value-1<style-attribute-name-2=style-attribute-value-2 ...*>]]

これらは、次元式にあるスタイル属性の例の一部です。

-
```
dept={label='Department'
    style=[color=red]}, N
```
- dept*[style=*MyDataStyle*], N
- dept*[format=12.2 style=*MyDataStyle*], N

注: 次元式で使用する場合、STYLE=オプションを角かっこ([と])または中かっこ({と})で囲む必要があります。

(CLASSLEV)
分類変数水準値ヘッダーにスタイル要素を割り当てます。たとえば、次の TABLE ステートメントでは、分類変数 DEPT の水準値ヘッダーの前景色が黄色になるように指定します。

```
table dept=[style(classlev)=
        [color=yellow]]*sales;
```

注: CLASSLEV オプションは、 次元式でのみ使用されます。

次元式内でスタイル要素を指定する方法を示す例については、“例 13: ODS 出力にスタイルの上書きを指定する” (2342 ページ)を参照してください。PROC TABULATE でのスタイルの使用については、“PROCTABULATE での ODS スタイルの使用” (2264 ページ)を参照してください。

PROC TABULATE の SAS Cloud Analytic Services 処理

あなたの入力データセットが SAS Cloud Analysis Services(CAS)からのものであれば、PROC TABULATE 分析の一部を CAS サーバーで実行できます。データの有意な

要約を必要とするレポートは、CAS 処理の恩恵を受けることができます。PROC TABULATE を CAS アクションで実行することは、SAS 内での処理に比べていくつかの利点があります。これらの利点には、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。インメモリテーブルは、比較的低速のネットワーク接続を介して転送されるのではなく、サーバー上でローカルに操作されるため、処理が高速になります。より多くの処理リソースが用意されている可能性があるので、CAS が使用されます。

DATA=入力データセットが CAS 内のインメモリテーブルまたはビューを参照する場合、TABULATE プロシジャは CAS アクションを使用してサーバー内でその作業の大部分を実行できます。インメモリテーブルまたはビューを参照するには、CAS エンジンの LIBNAME ステートメントを指定し、入力テーブル名とともに CAS エンジン libref を使用する必要があります。

デフォルトでは、PROC TABULATE は CAS エンジンの libref が入力テーブル名に指定されているときは常に CAS 処理を使用します。

次の例では、LIBNAME ステートメントは CAS セッション casauto に接続するために使用する mycas という名前の CAS エンジン libref を割り当てます。

```
option casport=5570 cashost="cloud.example.com";
cas casauto ;
libname mycas cas;
data mycas.class;
  set sashelp.class;
run;

proc tabulate data=mycas.class;
  class sex age;
  var height weight;
  table sex,age*(height weight)*(sum mean);
run;
```

次の統計量が、CAS 処理に対しサポートされています。

CSS	RANGE
CV	STDERR
LCLM	SUM
MAX	SUMWGT
MEAN	STD
MIN	UCLM
N	USS
NMISS	VAR

SAS 出力形式定義が CAS に存在する場合、分類変数のフォーマットが CAS で発生します。SAS 出力形式定義が CAS サーバーにない場合、CAS 集計が未加工値で発生し、結果のセットが PROC TABULATE 内部構造に結合される際に関連する出力形式が適用されます。SAS で作成されたユーザー定義の形式は、正常に動作するためには CAS にコピーする必要があります。SAS と CAS の間でフォーマットを一致させることをお勧めします。CAS でユーザー定義フォーマットを使用する方法の詳細なドキュメントについては、 *SAS Cloud Analytic Services: ユーザー定義出力形式* を参照してください。

CAS LIBNAME ステートメントの使用方法については、“LIBNAME ステートメント: CAS エンジン” (*SAS Cloud Analytic Services: ユーザーガイド*)を参照してください。

プロシジャが CAS 処理とどのように連携するかの詳細については、4 章, “Base プロシジャの CAS 処理” (75 ページ)を参照してください。

PROC TABULATE の In-Database 処理

In-Database 処理には、SAS 内での処理よりも優れたいくつかの利点があります。これらの利点には、セキュリティの強化、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。セキュリティの強化は、機密データをデータベース管理システム(DBMS)から抽出する必要がないため可能です。より迅速な処理は、データが比較的遅いネットワーク接続から転送される代わりに高速二次記憶装置を使用して DBMS でローカルで操作されるため、DBMS に自由に使えるより多くの処理リソースがあるため、DBMS は高度に並列かつスケーラブルな方法で実行するクエリを最適化できるため可能です。

DATA=入力データセットが DBMS にテーブルまたはビューとして保存される場合、PROC TABULATE プロシジャは In-Database 処理を使用して、データベース内の操作のほとんどを実行できます。In-Database 処理は、より迅速な処理と、データベースと SAS ソフトウェア間のデータ転送の減少という利点を提供できます。

PROC TABULATE は、SQL 暗黙パススルーを使用して、In-Database 処理を実行します。プロシジャは、TABLE ステートメントで指定する分類および統計量に基づく SQL クエリを生成します。データベースはこれらの SQL クエリを実行し、初期要約テーブルを構成します。これらは次に PROC TABULATE に送信されます。

分類変数が指定されると、プロシジャは N 次元の種類を表す SQL GROUP BY 句を作成します。DBMS では、N 次元のクラスツリーのみ生成されます。集計クエリがデータベースで実行される際に作成される結果セットが、SAS によって内部 PROC TABULATE データ構造に読み込まれます。

SAS 出力形式定義がデータベースで配置される際に、分類変数のフォーマットがデータベースで実行されます。SAS 出力形式定義がデータベースで配置されていない場合、In-Database 集計が未加工値で発生し、結果のセットが PROC TABULATE 内部構造に結合される際に関連する出力形式が適用されます。マルチラベルフォーマットは常に、データベースによって返される最初に集計された結果セットを使用して実行されます。

次の統計量が、In-Database 処理に対しサポートされています。

CSS	RANGE
CV	STDERR
LCLM	SUM
MAX	SUMWGT
MEAN	STD
MIN	UCLM
N	USS
NMISS	VAR

SQLGENERATION システムオプションまたは LIBNAME ステートメントオプションは、In-Database プロシジャがデータベース内で実行されるかどうか、およびその実行方法を制御します。デフォルトで、In-Database プロシジャは可能な場合はデータベース内で実行されます。In-Database 処理を実行しない、多くのデータセッ

トオプションがあります。完全なリストについては、*SAS/ACCESS for Relational Databases: リファレンス*の"In-Database プロシジャ"を参照してください。

PROC TABULATE には、次のような機能があります。

- Amazon Redshift
- DB2
- Greenplum
- Hadoop
- IMPALA
- Microsoft SQL Server
- MySQL
- Netezza
- Oracle
- PostgreSQL
- SAP HANA
- Snowflake
- Spark
- Teradata
- Vertica

テキスト文字列の BY 行値の置換

#BYLINE、#BYVAR、および#BYVAL の置換は、次のオプションで使用できます。

- TABLE ステートメントおよび PROC TABULATE ステートメントの CONTENTS= オプション
- TABLE ステートメントの CAPTION=オプション

#BYVAR 置換および#BYVAL 置換を使用するには、代替テキストを表示する位置のテキスト文字列にその項目を挿入します。#BYVAR と#BYVAL の両方の指定の後には、区切り文字を付ける必要があります。文字は、スペース、または引用符などの他の英数字以外の文字にすることができます。区切り文字が指定されていない場合、指定は無視され、そのテキストはそのまま残り、文字列の残りの部分とともに表示されます。#BYVAR および#BYVAL の置換のすぐ後に、区切り文字なしに、他のテキストの直後に続くようにするには、(マクロ変数と同じように)末尾にドットを使用します。解決されたテキストには、末尾のドットは表示されません。解決されたテキストの最後の文字としてピリオドを表示する場合は、＃BYVAR または#BYVAL 置換の後に 2 つのドットを使用します。

次の場合、＃BYVAR または#BYVAL の置換は行われません。

- BY ステートメントで指定されていない変数に#BYVAR または#BYVAL 指定を使用する場合。たとえば、BY 変数が 1 つしかない場合は＃BYVAL2 を使用し、ABC が存在しないか BY 変数ではない場合は#BYVAL(ABC)を使用します。

- BY ステートメントがない場合

結果: TABULATE プロシジャ

欠損値

PROC TABULATE による欠損値の処理

入力データセットの変数に対する欠損値が出力にどのように影響するかは、PROC TABULATE ステップで変数をどのように使用するかによって異なります。次のテーブルに、プロシジャが欠損値をどのように処理するかを要約します。

表 61.8　PROC TABULATE による欠損値の処理の要約

条件	PROC TABULATE デフォルト	デフォルトに優先
オブザベーションに分析変数に対する欠損値が含まれている	その特定の変数に対し統計量(N と NMISS 以外)の計算からそのオブザベーションを除外する	代替なし
オブザベーションに分類変数に対する欠損値が含まれている	テーブルからそのオブザベーションを除外する [1]	MISSING を PROC TABULATE ステートメントまたは CLASS ステートメントで使用する
カテゴリに対しデータがない	テーブルにカテゴリを表示しない	TABLE ステートメントで PRINTMISS を、PROC TABULATE ステートメントで CLASSDATA= を使用する
テーブルセルに影響するすべてのオブザベーションに分析変数に対する欠損値が含まれている	セルの統計量(N と NMISS 以外)に対する欠損値を表示する	TABLE ステートメントで MISSTEXT=を使用する
フォーマットされた値に対しデータがない	テーブルにフォーマットされた値を表示しない	CLASS ステートメントで PRELOADFMT、TABLE ステートメントで PRINTMISS を使用す

1. CLASS ステートメントは、PROC TABULATE ステップのすべての TABLE ステートメントに適用されます。そのため、変数を分類変数として定義すると、PROC TABULATE は変数に対する欠損値を含むオブザベーションを省略します。TABLE ステートメントでその変数を使用しない場合も同様です。

条件	PROC TABULATE デフォルト	デフォルトに優先
		る。または PROC TABULATE ステートメントで CLASSDATA=を使用する。あるいは各フォーマットされた値に対しデータが含まれるように入力データセットにダミーオブザベーションを追加する
FREQ 変数値が欠損値、または 1 未満である	統計量の計算にオブザベーションを使用しない	代替なし
WEIGHT 変数値が欠損値、または 0 である	値 0 を使用する	代替なし

このセクションは、一連の PROC TABULATE ステップを示し、PROC TABULATE による欠損値の処理を説明します。次のプログラムは、このセクションで使用するデータセットと出力形式を作成し、データセットを印刷します。データセット COMPREV に欠損値は含まれていません。(次の出力を参照。)

```
proc format;
  value cntryfmt 1='United States'
                 2='Japan';
  value compfmt  1='Supercomputer'
                 2='Mainframe'
                 3='Midrange'
                 4='Workstation'
                 5='Personal Computer'
                 6='Laptop';
run;

data comprev;
  input Country Computer Rev90 Rev91 Rev92;
  datalines;
1 1 788.8 877.6 944.9
1 2 12538.1 9855.6 8527.9
1 3 9815.8 6340.3 8680.3
1 4 3147.2 3474.1 3722.4
1 5 18660.9 18428.0 23531.1
2 1 469.9 495.6 448.4
2 2 5697.6 6242.4 5382.3
2 3 5392.1 5668.3 4845.9
2 4 1511.6 1875.5 1924.5
2 5 4746.0 4600.8 4363.7
;

proc print data=comprev noobs;
  format country cntryfmt. computer compfmt.;
  title 'The Data Set COMPREV';
run;
```

アウトプット 61.5　データセット COMPREV

The Data Set COMPREV

Country	Computer	Rev90	Rev91	Rev92
United States	Supercomputer	788.8	877.6	944.9
United States	Mainframe	12538.1	9855.6	8527.9
United States	Midrange	9815.8	6340.3	8680.3
United States	Workstation	3147.2	3474.1	3722.4
United States	Personal Computer	18660.9	18428.0	23531.1
Japan	Supercomputer	469.9	495.6	448.4
Japan	Mainframe	5697.6	6242.4	5382.3
Japan	Midrange	5392.1	5668.3	4845.9
Japan	Workstation	1511.6	1875.5	1924.5
Japan	Personal Computer	4746.0	4600.8	4363.7

欠損値なし

次の PROC TABULATE ステップにより、次の出力が生成されます。

```
proc tabulate data=comprev;
  class country computer;
  var rev90 rev91 rev92;
  table computer*country,rev90 rev91 rev92 /
      rts=32;
  format country cntryfmt. computer compfmt.;
  title 'Revenues from Computer Sales';
  title2 'for 1990 to 1992';
run;
```

データセットに欠損値が含まれていないため、テーブルにはすべてのオブザベーションが含まれます。すべてのヘッダーとセルには、非欠損値が含まれています。

アウトプット 61.6　コンピューター売上データ: 欠損値なし

Revenues from Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
	Japan	5392.10	5668.30	4845.90
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

欠損分類変数

次のプログラムでは COMPREV をコピーし、8 番目のオブザベーションに Computer の欠損値が含まれるようにデータを変更します。この新しいデータセットの指定を除けば、次の出力“コンピューター売上データ: Midrange, Japan 削除済み”を生成するプログラムは、前述の出力“コンピューター売上データ: 欠損値なし”を生成するプログラムと同じです。PROC TABULATE は分類変数に対する欠損値を含むオブザベーションを無視します。

```
data compmiss;
  set comprev;
  if _n_=8 then computer=.;
run;

proc tabulate data=compmiss;
  class country computer;
  var rev90 rev91 rev92;
  table computer*country,rev90 rev91 rev92 /
      rts=32;
  format country cntryfmt. computer compfmt.;
  title 'Revenues from Computer Sales';
  title2 'for 1990 to 1992';
run;
```

Computer に対する欠損値を含むオブザベーションのカテゴリは、**Midrange, Japan** でした。このカテゴリは、現在存在しません。デフォルトでは、PROC TABULATE は分類変数に対する欠損値を含むオブザベーションを無視するため、このテーブルには出力“コンピューター売上データ: 欠損値なし”よりも 1 行少ない行が含まれています。

アウトプット 61.7　コンピューター売上データ: ミッドレンジ、日本、削除

Revenues from Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

欠損分類変数を含むオブザベーションを含める

このプログラムは、MISSING オプションを前のプログラムに追加します。MISSING は、PROC TABULATE ステートメント、CLASS ステートメントで使用できます。MISSING を選択した分類変数にのみ適用する場合、選択した変数を含む別の CLASS ステートメントで MISSING を指定します。MISSING オプションにより、分類変数の欠損値を含むオブザベーションがレポートに含まれます。(次の出力を参照してください。)

```
proc tabulate data=compmiss missing;
  class country computer;
  var rev90 rev91 rev92;
  table computer*country,rev90 rev91 rev92 /
       rts=32;
  format country cntryfmt. computer compfmt.;
  title 'Revenues from Computer Sales';
  title2 'for 1990 to 1992';
run;
```

このテーブルには、Computer の欠損値を含むカテゴリが含まれます。このカテゴリは、テーブルの最初のデータ行を作成します。

アウトプット 61.8　コンピューター売上データ: コンピューターの欠損値

Revenues from Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
.	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

欠損分類変数を含むオブザベーションのヘッダーのフォーマット

デフォルトでは、出力 “コンピューター売上データ: Computer の欠損値”にあるように、PROC TABULATE は分類変数の欠損値を欠損値に対する標準の SAS 文字の 1 つで表示します(ピリオド、ブランク、アンダースコア、または A から Z までの文字のいずれか)。それ以外を表示する場合、次のプログラムのように、欠損値を含む分類変数に出力形式を割り当てる必要があります。(次の出力を参照してください。)

```
proc format;
  value misscomp 1='Supercomputer'
                 2='Mainframe'
                 3='Midrange'
                 4='Workstation'
                 5='Personal Computer'
                 6='Laptop'
                 .='No type given';
run;

proc tabulate data=compmiss missing;
  class country computer;
  var rev90 rev91 rev92;
  table computer*country,rev90 rev91 rev92 /
      rts=32;
  format country cntryfmt. computer misscomp.;
  title 'Revenues for Computer Sales';
  title2 'for 1990 to 1992';
run;
```

このテーブルでは、欠損値が MISSCOMP.出力形式によって指定されるテキストとして表示されます。

アウトプット 61.9　コンピューター売上データ: Computer の欠損値に提供されるテキスト

Revenues for Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
No type given	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

すべてのカテゴリのヘッダーの提供

デフォルトでは、PROC TABULATE は存在しないカテゴリに対し列と行を印刷および省略する各ページを評価します。たとえば、“コンピューター売上データ: Computer の欠損値に提供されるテキスト”には No type given 、United States 、Midrange、Japan の行は含まれません。これらのカテゴリにデータがないためです。可能なすべてのカテゴリをテーブルに 表示する場合、次のプログラムのように TABLE ステートメントで PRINTMISS オプションを使用します。(次の出力を参照してください。)

```
proc tabulate data=compmiss missing;
  class country computer;
  var rev90 rev91 rev92;
  table computer*country,rev90 rev91 rev92 /
       rts=32 printmiss;
  format country cntryfmt. computer misscomp.;
  title 'Revenues for Computer Sales';
  title2 'for 1990 to 1992';
run;
```

このテーブルには、カテゴリ **No type given**、**United States** とカテゴリ **Midrange, Japan** の行が含まれています。これらのカテゴリにデータがないため、統計量の値はすべて 欠損値となります。

アウトプット 61.10　コンピューター売上データ: 欠損統計量値

Revenues for Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
No type given	United States	.	.	.
	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
	Japan	.	.	.
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

欠損値を含むセルに対するテキストの提供

カテゴリの一部のオブザベーションに分析変数に対する欠損値が含まれている場合、PROC TABULATE は統計量(N と NMISS 以外)の計算にこれらのオブザベーションは使用しません。ただし、カテゴリのオブザベーションにそれぞれ欠損値が含まれている場合、PROC TABULATE は統計量の値に対する欠損値を表示します。分析変数の欠損値をテキストと置き換えるには、次のプログラムのように TABLE ステートメントで MISSTEXT=オプションを使用して、使用するテキストを指定します。(次の出力を参照してください。)

```
proc tabulate data=compmiss missing;
  class country computer;
  var rev90 rev91 rev92;
  table computer*country,rev90 rev91 rev92 /
       rts=32 printmiss misstext='NO DATA!';
  format country cntryfmt. computer misscomp.;
  title 'Revenues for Computer Sales';
  title2 'for 1990 to 1992';
run;
```

このテーブルでは、欠損値の表示に通常使用されるピリオドを MISSTEXT=オプションのテキストと置き換えます。

アウトプット 61.11　コンピューター売上データ: 欠損統計量値に対し提供されるテキスト

Revenues for Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
No type given	United States	NO DATA!	NO DATA!	NO DATA!
	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
	Japan	NO DATA!	NO DATA!	NO DATA!
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70

出力形式のすべての値に対するヘッダーの提供

PROC TABULATE は、入力データセットに表示される値に対してのみヘッダーを印刷します。たとえば、出力形式 COMPFMT.は、Computer について 6 つの可能な値を提供しています。これらの値のうち 5 つのみが、データセット COMPREV で発生します。データセットには、ラップトップコンピューターに対するデータは含まれていません。

Computer の可能なすべての値に対するヘッダーを含める場合(出力と、ラップトップに関するデータがない場合に後で作成されるテーブルとを比較しやすくするため)、そのようなテーブルの作成には 3 つの方法があります。

- CLASS ステートメントの PRELOADFMT オプションを TABLE ステートメントの PRINTMISS オプションとともに使用します。PRELOADFMT を使用する別の例については、“例 3: すでに読み込まれている出力形式を分類変数とともに使用する” (2296 ページ)を参照してください。
- PROC TABULATE ステートメントで CLASSDATA=オプションを使用します。CLASSDATA=オプションを使用する例については、“例 2: テーブルに表示する分類変数の組み合わせを指定する” (2293 ページ)を参照してください。
- 出力形式で処理される各値がデータセットに少なくとも一度表示されるように、ダミー値を入力データセットに追加します。

次のプログラムは、PRELOADFMT オプションを関連する変数を含む CLASS ステートメントに追加します。

結果は、次の出力のとおりです。

```
proc tabulate data=compmiss missing;
  class country;
  class computer / preloadfmt;
  var rev90 rev91 rev92;
  table computer*country,rev90 rev91 rev92 /
      rts=32 printmiss misstext='NO DATA!';
  format country cntryfmt. computer compfmt.;
  title 'Revenues for Computer Sales';
  title2 'for 1990 to 1992';
run;
```

このテーブルには、Computer のそれぞれの可能な値に対するヘッダーが含まれます。

アウトプット 61.12　コンピューター売上データ: 考えられるすべてのコンピューター値が含まれる

Revenues for Computer Sales
for 1990 to 1992

		Rev90	Rev91	Rev92
		Sum	Sum	Sum
Computer	Country			
.	United States	NO DATA!	NO DATA!	NO DATA!
	Japan	5392.10	5668.30	4845.90
Supercomputer	United States	788.80	877.60	944.90
	Japan	469.90	495.60	448.40
Mainframe	United States	12538.10	9855.60	8527.90
	Japan	5697.60	6242.40	5382.30
Midrange	United States	9815.80	6340.30	8680.30
	Japan	NO DATA!	NO DATA!	NO DATA!
Workstation	United States	3147.20	3474.10	3722.40
	Japan	1511.60	1875.50	1924.50
Personal Computer	United States	18660.90	18428.00	23531.10
	Japan	4746.00	4600.80	4363.70
Laptop	United States	NO DATA!	NO DATA!	NO DATA!
	Japan	NO DATA!	NO DATA!	NO DATA!

ORDER=DATA を使用したヘッダーの順序について

ORDER=オプションは、すべての分類変数に適用されます。異なる変数のヘッダーを別に並べ替えることが必要な場合があります。ヘッダーを並べ替える 1 つの方法は、データを表示によってグループ化し、ORDER=DATA を指定することです。

この方法を使用できるようにするには、最初の分類変数の最初の値が、その他すべての分類変数の可能なすべての値を含むデータで発生する必要があります。この条件に一致しない場合、ヘッダーの順序が希望と異なることがあります。

次のプログラムは、オブザベーションが最初に Animal の値、次に Food の値によって並べ替えられる単一のデータセットを作成します。PROC TABULATE ステートメントの ORDER=オプションは、分類変数のヘッダーをデータセットで表示される順序によって並べ替えます。(次の出力を参照してください。)bones は Animal=dog,のオブザベーショングループの Food の最初の値ですが、Food のその他すべての値はデータセットの bones の前に表示されます。これは、bones が Animal=cat の場合に表示されないためです。そのため、次の出力のテーブルの bones のヘッダーは、アルファベット順ではありません。

つまり、PROC TABULATE は、前のカテゴリによって構築された順序を後続のカテゴリに対して保持します。Animal の各値に対して Food の順序を再構築する場合、BY グループ処理を使用します。PROC TABULATE は BY グループに対しそれぞれ別のテーブルを作成するため、並べ替えは BY グループ間で異なる可能性があります。

```
title "Animal Food Preference";
data foodpref;
  input Animal $ Food $;
  datalines;
cat fish
cat meat
cat milk
dog bones
dog fish
dog meat
;

proc tabulate data=foodpref format=9.
            order=data;
  class animal food;
  table animal*food;
run;
```

アウトプット 61.13　分類変数のヘッダーの並べ替え

Animal Food Preference

Animal					
cat			dog		
Food			Food		
fish	meat	milk	fish	meat	bones
N	N	N	N	N	N
1	1	1	1	1	1

例: TABULATE プロシジャ

例 1: 基本的な 2 次元表の作成

要素:

CLASS ステートメント
PROC TABULATE ステートメントオプション
    DATA=
    FORMAT=
TABLE ステートメントオプション
    クロス(*)演算子
    RTS=
VAR ステートメント
DATA ステップ
FORMAT プロシジャ
FORMAT ステートメント
TITLE ステートメント

データセット: ENERGY

詳細

次のサンプルプログラムでは、次を実行します。

- 各リージョンの各区分の各種のユーザー(住居または企業)に対するカテゴリを作成する
- テーブルのすべてのセルに 同じ出力形式を適用する
- 各分類変数に出力形式を適用する
- 行ヘッダーのスペースを 拡張する

プログラム

```
proc format;
  value regfmt 1='Northeast'
               2='South'
               3='Midwest'
```

```
            4='West';
  value divfmt 1='New England'
            2='Middle Atlantic'
            3='Mountain'
            4='Pacific';
  value usetype 1='Residential Customers'
             2='Business Customers';
run;

proc tabulate data=energy format=dollar12.;

  class region division type;


  var expenditures;

  table region*division,
      type*expenditures

       / rts=25;

  format region regfmt. division divfmt. type usetype.;

  title 'Energy Expenditures for Each Region';
  title2 '(millions of dollars)';
run;
```

プログラムの説明

出力形式 REGFMT.、DIVFMT.および USETYPE.を作成します。 PROC FORMAT は、Region、Division、Type に対する出力形式を作成します。

```
proc format;
  value regfmt 1='Northeast'
            2='South'
            3='Midwest'
            4='West';
  value divfmt 1='New England'
            2='Middle Atlantic'
            3='Mountain'
            4='Pacific';
  value usetype 1='Residential Customers'
             2='Business Customers';
run;
```

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Region、Division、Type の値別に分類します。

```
  class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
  var expenditures;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Region のフォーマットされた値に対しそれぞれ行を作成します。各行内でネストされるのは、Division の各フォーマットされた値に対する行です。TABLE ステートメントは、Type のフォーマットされた値に対してもそれぞれ列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに 対する分析変数 Expenditures の合計が含まれます。

```
table region*division,
    type*expenditures
```

行タイトルスペースを 指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。

```
/ rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

出力

アウトプット 61.14　基本的な 2 次元表

Energy Expenditures for Each Region
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

例 2: テーブルに表示する分類変数の組み合わせを指定する

要素: CLASS ステートメント
PROC TABULATE ステートメントオプション

DATA=
CLASSDATA=
EXCLUSIVE
FORMAT=

TABLE ステートメントオプション
クロス(*)演算子
RTS=

VAR ステートメント

DATA ステップ

FORMAT ステートメント

TITLE ステートメント

データセット: ENERGY

詳細

この例では、次を行います。

- CLASSDATA=オプションを使用して、テーブルに表示する分類変数の組み合わせを指定する。
- EXCLUSIVE オプションを使用して、出力を CLASSDATA=データセットで指定される組み合わせにのみ制限する。EXCLUSIVE オプションを使用しない場合、出力は“例 1: 基本的な 2 次元表の作成” (2291 ページ)と同じになります。

プログラム

```
data classes;
  input region division type;
  datalines;
1 1 1
1 1 2
4 4 1
4 4 2
;

proc tabulate data=energy format=dollar12.
              classdata=classes exclusive;


  class region division type;

  var expenditures;

  table region*division,
        type*expenditures

       / rts=25;

  format region regfmt. division divfmt. type usetype.;

  title 'Energy Expenditures for Each Region';
```

```
  title2 '(millions of dollars)';
run;
```

プログラムの説明

CLASSES データセットを作成します。 CLASSES には、PROC TABULATE がテーブルの作成に使用する分類変数値の組み合わせが含まれます。

```
data classes;
  input region division type;
  datalines;
1 1 1
1 1 2
4 4 1
4 4 2
;
```

テーブルオプションを指定します。 CLASSDATA=と EXCLUSIVE は、分類変数の水準の組み合わせを CLASSES データセットで指定される組み合わせに制限します。

```
proc tabulate data=energy format=dollar12.
        classdata=classes exclusive;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Region、Division、Type の値別に分類します。

```
class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Region のフォーマットされた値に対しそれぞれ行を作成します。各行内でネストされるのは、Division の各フォーマットされた値に対する行です。TABLE ステートメントは、Type のフォーマットされた値に対してもそれぞれ列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに 対する分析変数 Expenditures の合計が含まれます。

```
table region*division,
    type*expenditures
```

行タイトルスペースを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。

```
    / rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

出力

アウトプット 61.15　リージョンごとのエネルギー支出

Energy Expenditures for Each Region
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
West	Pacific	$13,959	$12,619

例 3: すでに読み込まれている出力形式を分類変数とともに使用する

要素:

CLASS ステートメントオプション
- EXCLUSIVE
- PRELOADFMT

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=
- OUT=

TABLE ステートメントオプション
- クロス(*)演算子
- PRINTMISS
- RTS

VAR ステートメント

FORMAT ステートメント

PRINT プロシジャ

TITLE ステートメント

データセット: ENERGY

詳細

この例では、次を行います。

- フォーマットされた分類変数値の可能なすべての組み合わせを含むテーブルを作成する(PRELOADFMT と PRINTMISS) (これらの組み合わせにゼロの度数が含まれていても、これらの組み合わせが意味をなさなくても同様)
- ユーザー定義出力形式のすでに読み込まれた範囲のみを分類変数の水準として使用する(PRELOADFMT と EXCLUSIVE)
- 出力を出力データセットに書き込み、そのデータセットを印刷する

プログラム

```
proc tabulate data=energy format=dollar12.;
  class region division type / preloadfmt;
  var expenditures;
  table region*division,
      type*expenditures / rts=25 printmiss;
  format region regfmt. division divfmt. type usetype.;
  title 'Energy Expenditures for Each Region';
  title2 '(millions of dollars)';
run;
proc tabulate data=energy format=dollar12. out=tabdata;
  class region division type / preloadfmt exclusive;
  var expenditures;
  table region*division,
      type*expenditures / rts=25;
  format region regfmt. division divfmt. type usetype.;
  title 'Energy Expenditures for Each Region';
  title2 '(millions of dollars)';
run;
proc print data=tabdata;
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Region、Division、Type の値別に分類します。PRELOADFMT は、PROC TABULATE が分類変数に対するユーザー定義出力形式のすでに読み込まれた値を使用するように指定します。

```
  class region division type / preloadfmt;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;
```

テーブルの行と列を定義し、行と列のオプションを指定します。 PRINTMISS は、ユーザー定義出力形式のすべての可能な組み合わせが分類変数の水準として使用されるように指定します。

```
table region*division,
    type*expenditures / rts=25 printmiss;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

テーブルオプションと出力データセットを指定します。 OUT=オプションは、PROC TABULATE がデータを書き込む出力データセットの名前を指定します。

```
proc tabulate data=energy format=dollar12. out=tabdata;
```

分析対象となるサブグループを指定します。 EXCLUSIVE オプションは PRELOADFMT と使用されると、ユーザー定義出力形式のすでに読み込まれた範囲のみを分類変数の水準として使用します。

```
class region division type / preloadfmt exclusive;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;
```

テーブルの行と列を定義し、行と列のオプションを指定します。 この場合、PRINTMISS オプションは指定されません。指定された場合、CLASS ステートメントの EXCLUSIVE オプションに優先します。

```
table region*division,
    type*expenditures / rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

出力データセット WORK.TABDATA を印刷します。

```
proc print data=tabdata;
run;
```

出力

PRELOADFMT オプションと PRINTMISS オプションで作成されたこの出力には、分類変数値に対するすでに読み込まれたユーザー定義出力形式のすべての可能な組み合わせが含まれています。ゼロの度数を含む組み合わせや、**Northeast** と **Pacific** などの意味をなさない組み合わせが含まれます。

アウトプット 61.16　リージョンごとのエネルギー支出

Energy Expenditures for Each Region
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
	Mountain	.	.
	Pacific	.	.
South	New England	.	.
	Middle Atlantic	.	.
	Mountain	.	.
	Pacific	.	.
Midwest	New England	.	.
	Middle Atlantic	.	.
	Mountain	.	.
	Pacific	.	.
West	New England	.	.
	Middle Atlantic	.	.
	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

PRELOADFMT オプションと EXCLUSIVE オプションで作成されたこの出力には、入力データセットで表示される分類変数値に対するすでに読み込まれたユーザー定義出力形式の組み合わせのみ含まれます。この出力は、“例 1: 基本的な 2 次元表の作成” (2291 ページ)からの出力と同じです。

アウトプット 61.17　リージョンごとのエネルギー支出

Energy Expenditures for Each Region
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

この出力には、PROC TABULATE ステートメントの OUT=オプションによって作成された出力データセット TABDATA が表示されます。TABDATA には、PRELOADFMT オプションと EXCLUSIVE オプションを指定して作成されるデータが含まれます。

アウトプット 61.18　リージョンごとのエネルギー支出

Energy Expenditures for Each Region
(millions of dollars)

Obs	Region	Division	Type	_TYPE_	_PAGE_	_TABLE_
1	Northeast	New England	Residential Customers	111	1	1
2	Northeast	New England	Business Customers	111	1	1
3	Northeast	Middle Atlantic	Residential Customers	111	1	1
4	Northeast	Middle Atlantic	Business Customers	111	1	1
5	West	Mountain	Residential Customers	111	1	1
6	West	Mountain	Business Customers	111	1	1
7	West	Pacific	Residential Customers	111	1	1
8	West	Pacific	Business Customers	111	1	1

例 4: マルチラベル出力形式の使用

要素:

CLASS ステートメントオプション
　MLF
PROC TABULATE ステートメントオプション
　DATA=
　FORMAT=
TABLE ステートメント
　ALL 分類変数
　連結(ブランク)演算子

クロス(*)演算子
要素グループ化(かっこ)演算子
label
変数リスト

VAR ステートメント

DATA ステップ

FORMAT プロシジャ

FORMAT ステートメント

TITLE ステートメント

データセット: CARSURVEY

詳細

この例では、次を行います。

- PROC FORMAT の VALUE ステートメントでマルチラベル出力形式を指定する方法を示す
- MLF オプションを CLASS ステートメントと使用して、マルチラベル出力形式処理を有効化する方法を示す
- マルチラベル出力形式処理が有効化される場合の N 統計量の動作を説明する

プログラム

```
proc format;
  value agefmt (multilabel notsorted)
       15 - 29 = 'Below 30 years'
       30 - 50 = 'Between 30 and 50'
     51 - high = 'Over 50 years'
       15 - 19 = '15 to 19'
       20 - 25 = '20 to 25'
       25 - 39 = '25 to 39'
       40 - 55 = '40 to 55'
     56 - high = '56 and above';
run;

proc tabulate data=carsurvey format=10.;

  class age / mlf;

  var progressa remark jupiter dynamo;

  table age all, n all='Potential Car Names'*(progressa remark
    jupiter dynamo)*mean;


  title1 "Rating Four Potential Car Names";
  title2 "Rating Scale 0-100 (100 is the highest rating)";

  format age agefmt.;
```

```
run;
```

プログラムの説明

AGEFMT.出力形式を作成します。 FORMAT プロシジャは“MULTILABEL” (909 ページ)を使用して、年齢に対するマルチラベル出力形式を作成します。マルチラベル出力形式は、複数のラベルを同じ値に割り当てることができる出力形式です。この場合は、範囲が重複するためです。各値は発生する各範囲のテーブルに表されます。NOTSORTED オプションは、範囲を範囲が定義された順序で保存します。

```
proc format;
  value agefmt (multilabel notsorted)
      15 - 29 = 'Below 30 years'
      30 - 50 = 'Between 30 and 50'
    51 - high = 'Over 50 years'
      15 - 19 = '15 to 19'
      20 - 25 = '20 to 25'
      25 - 39 = '25 to 39'
      40 - 55 = '40 to 55'
    56 - high = '56 and above';
run;
```

テーブルオプションを指定します。 FORMAT=オプションは、最大 10 桁を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=carsurvey format=10.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは Age を分類変数として識別し、MLF オプションを使用してマルチラベル出力形式処理を有効化します。

```
class age / mlf;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が変数 Progressa、Remark、Jupiter、Dynamo の統計量を計算するように指定します。

```
var progressa remark jupiter dynamo;
```

テーブルの行と列を定義します。 TABLE ステートメントの行次元は、Age のフォーマットされた値に対しそれぞれ行を作成します。マルチラベル出力形式により、オブザベーションが複数の行または年齢のカテゴリに含まれるようになります。行次元は ALL 分類変数を使用して、すべての行に関する情報を要約します。列次元は N 統計量を使用して、各年齢グループに対するオブザベーション数を計算します。行次元の ALL 分類変数とクロスする N 統計量の結果は、行に対する N 統計量の合計ではなく、オブザベーションの合計数です。列次元はクロスの初めに ALL 分類変数を使用して、ラベル Potential Car Names を割り当てます。4 つのネストされた列が、各年齢グループに対する車名の平均評価を計算します。

```
table age all, n all='Potential Car Names'*(progressa remark
  jupiter dynamo)*mean;
```

タイトルを指定します。

```
title1 "Rating Four Potential Car Names";
title2 "Rating Scale 0-100 (100 is the highest rating)";
```

出力をフォーマットします。 FORMAT ステートメントは、ユーザー定義出力形式 AGEFMT.をこの分析用の Age に割り当てます。

```
    format age agefmt.;
run;
```

出力

アウトプット 61.19 Rating Four Potential Car Names

Rating Four Potential Car Names
Rating Scale 0-100 (100 is the highest rating)

	N	Potential Car Names			
		Progressa	Remark	Jupiter	Dynamo
		Mean	Mean	Mean	Mean
Age					
15 to 19	14	75	78	81	73
20 to 25	11	89	88	84	89
25 to 39	26	84	90	82	72
40 to 55	14	85	87	80	68
56 and above	15	84	82	81	75
Below 30 years	36	82	84	82	75
Between 30 and 50	25	86	89	81	73
Over 50 years	19	82	84	80	76
All	80	83	86	81	74

例 5: 行と列のヘッダーのカスタマイズ

要素:

CLASS ステートメント

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメントオプション
- クロス(*)演算子
- ラベル
- RTS=

VAR ステートメント

FORMAT ステートメント

TITLE ステートメント

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、行ヘッダーと列ヘッダーをカスタマイズする方法を示します。ラベルは、ヘッダー用テキストを指定します。ブランクのラベルはブランクのヘッダーを作成します。PROC TABULATE は、テーブルからブランクの列ヘッダーのスペースを削除します。

プログラム

```
proc tabulate data=energy format=dollar12.;
  class region division type;
  var expenditures;
  table region*division,
     type='Customer Base'*expenditures=' '*sum=' '
      / rts=25;
  format region regfmt. division divfmt. type usetype.;
  title 'Energy Expenditures for Each Region';
  title2 '(millions of dollars)';
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Region のフォーマットされた値に対しそれぞれ行を作成します。各行内でネストされるのは、Division の各フォーマットされた値に対する行です。TABLE ステートメントは、Type のフォーマットされた値に対してもそれぞれ列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに 対する分析変数 Expenditures の合計が含まれます。引用符内のテキストは、対応する変数または統

計量に対するヘッダーを指定します。Sum はデフォルトの統計量ですが、そのヘッダーに対しブランクを指定できるようにここで指定されます。

```
table region*division,
   type='Customer Base'*expenditures=' '*sum=' '
```

行タイトルスペースを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。

```
/ rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を Region、Division、Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

出力

Type のヘッダーには、TABLE ステートメントで指定されるテキストが含まれます。TABLE ステートメントは、Expenditures と Sum のヘッダーを削除しました。

アウトプット 61.20　リージョンごとのエネルギー支出

Energy Expenditures for Each Region
(millions of dollars)

		Customer Base	
		Residential Customers	Business Customers
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

例 6: 共通分類変数 ALL を使用した情報の要約

要素:

CLASS ステートメント

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメント
- ALL 分類変数
- 連結(ブランク演算子)

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、共通分類変数 ALL を使用して複数のカテゴリからの情報を要約する方法を示します。

プログラム

```
proc tabulate data=energy format=comma12.;
  class region division type;
  var expenditures;
  table region*(division all='Subtotal')
    all='Total for All Regions'*f=dollar12.,
    type='Customer Base'*expenditures=' '*sum=' '
      all='All Customers'*expenditures=' '*sum=' '
    / rts=25;
  format region regfmt. division divfmt. type usetype.;
  title 'Energy Expenditures for Each Region';
  title2 '(millions of dollars)';
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、COMMA12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=comma12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;

table region*(division all='Subtotal')
  all='Total for All Regions'*f=dollar12.,

  type='Customer Base'*expenditures=' '*sum=' '
    all='All Customers'*expenditures=' '*sum=' '
```

行タイトルスペースを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。

```
  / rts=25;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

出力

共通分類変数 ALL により、このテーブルの小計と合計が提供されます。

アウトプット 61.21　リージョンごとのエネルギー支出

Energy Expenditures for Each Region
(millions of dollars)

		Customer Base		All Customers
		Residential Customers	Business Customers	
Region	Division			
Northeast	New England	7,477	5,129	12,606
	Middle Atlantic	19,379	15,078	34,457
	Subtotal	26,856	20,207	47,063
West	Division			
	Mountain	5,476	4,729	10,205
	Pacific	13,959	12,619	26,578
	Subtotal	19,435	17,348	36,783
Total for All Regions		$46,291	$37,555	$83,846

例 7: 行ヘッダーの削除

要素:
- CLASS ステートメント
- PROC TABULATE ステートメントオプション
  - DATA=
  - FORMAT=
- TABLE ステートメントオプション
  - クロス(*)演算子
  - ラベル
  - ROW=FLOAT
  - RTS=
- VAR ステートメント
- FORMAT ステートメント
- TITLE ステートメント

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、ブランクの行ヘッダーをテーブルから削除する方法を示します。これを行うには、行ヘッダーに対しブランクのラベルを指定し、TABLE ステートメントで ROW=FLOAT を指定する必要があります。

プログラム

```
proc tabulate data=energy format=dollar12.;
  class region division type;
  var expenditures;
  table region*division*expenditures=' '*sum=' ',
      type='Customer Base'
      / rts=25 row=float;
  format region regfmt. division divfmt. type usetype.;
  title 'Energy Expenditures for Each Region';
  title2 '(millions of dollars)';
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;
```

テーブル行を定義します。 TABLE ステートメントの行次元は、Region のフォーマットされた値に対しそれぞれ行を作成します。これらの行内でネストされるのは、Division の各フォーマットされた値に対する行です。分析変数 Expenditures と Sum 統計量も行次元に含まれるため、PROC TABULATE はそれらの統計量の行ヘッダーも作成します。引用符内のテキストは、対応する変数または統計量に対するヘッダーを指定します。Sum はデフォルトの統計量ですが、そのヘッダーに対しブランクを指定できるようにここで指定されます。

```
table region*division*expenditures=' '*sum=' ',
```

テーブル列を定義します。 TABLE ステートメントの列次元は、Type のフォーマットされた値に対しそれぞれ列を作成します。

```
type='Customer Base'
```

行タイトルのスペースを指定し、ブランクの行ヘッダーを削除します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。ROW=FLOAT は、ブランクの行ヘッダーを削除します。

```
/ rts=25 row=float;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
title 'Energy Expenditures for Each Region';
title2 '(millions of dollars)';
run;
```

出力

このテーブルと“例 5: 行と列のヘッダーのカスタマイズ” (2303 ページ)の出力を比較します。2 つのテーブルは同じですが、このテーブルを作成するプログラムでは行次元の Expenditures と Sum が使用されます。PROC TABULATE によりブランクのヘッダーが列次元から自動的に削除されますが、ROW=FLOAT を指定して、ブランクのヘッダーを行次元から削除する必要があります。

アウトプット 61.22　リージョンごとのエネルギー支出

Energy Expenditures for Each Region
(millions of dollars)

		Customer Base	
		Residential Customers	Business Customers
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

例 8: 複数ページテーブルの作成

要素:

CLASS ステートメント

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメントオプション
- ALL 分類変数
- BOX=
- CONDENSE
- INDENT=
- ページ式
- RTS=

VAR ステートメント

FORMAT ステートメント

TITLE ステートメント

データセット: ENERGY

出力形式: REGFMT.

出力形式: DIVFMT.

出力形式: USETYPE.

詳細

この例では、各リージョンに対する個別のテーブルと、すべてのリージョンに対する 1 つのテーブルを作成します。デフォルトでは、PROC TABULATE は個別のページに各テーブルを作成しますが、CONDENSE オプションはそれらのテーブルをすべて同じページに置きます。

プログラム

```
proc tabulate data=energy format=dollar12.;
  class region division type;
  var expenditures;
  table region='Region: ' all='All Regions',
  division all='All Divisions',
      type='Customer Base'*expenditures=' '*sum=' '
      / rts=25 box=_page_ condense indent=1;
  format region regfmt. division divfmt. type usetype.;
  title 'Energy Expenditures for Each Region and All Regions';
  title2 '(millions of dollars)';
run;
```

プログラムの説明

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR12.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=energy format=dollar12.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Region、Division、Type を分類変数として識別します。

```
class region division type;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Expenditures 変数の統計量を計算するように指定します。

```
var expenditures;
```

テーブルページを定義します。 TABLE ステートメントのページ次元は、Region のフォーマットされた値に対しそれぞれ 1 つのテーブルを、すべてのリージョンに対し 1 つのテーブルを作成します。引用符内のテキストは、各ページに対しヘッダーを提供します。

```
table region='Region: ' all='All Regions',
```

テーブル行を定義します。 行次元は、Division のフォーマットされた値に対しそれぞれ 1 行、すべての区分に対し 1 行を作成します。引用符内のテキストは、行ヘッダーを提供します。

```
division all='All Divisions',
```

テーブル列を定義します。 TABLE ステートメントの列次元は、Type のフォーマットされた値に対しそれぞれ列を作成します。これらのページ、行および列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Expenditures の合計が含まれます。引用符内のテキストは、対応する変数または統計量に対するヘッダーを指定します。Sum はデフォルトの統計量ですが、そのヘッダーに対しブランクを指定できるようにここで指定されます。

```
type='Customer Base'*expenditures=' '*sum=' '
```

追加のテーブルオプションを指定します。 RTS=は、行ヘッダーに対する行ごとに 25 文字を提供します。BOX=は、ページヘッダーを行ヘッダーの上のボックス内に置きます。CONDENSE は、1 つの物理ページに可能な限り多くのテーブルを置きます。INDENT=は、Division の行ヘッダーを削除します。(行次元にはネスティングがないため、インデントは実行されません)。

```
    / rts=25 box=_page_ condense indent=1;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Region、Division および Type に割り当てます。

```
    format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
    title 'Energy Expenditures for Each Region and All Regions';
    title2 '(millions of dollars)';
run;
```

出力

アウトプット 61.23　各リージョンおよび全リージョンのエネルギー支出

Energy Expenditures for Each Region and All Regions
(millions of dollars)

Region: Northeast	Customer Base	
	Residential Customers	Business Customers
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078
All Divisions	$26,856	$20,207

Region: West	Customer Base	
	Residential Customers	Business Customers
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619
All Divisions	$19,435	$17,348

All Regions	Customer Base	
	Residential Customers	Business Customers
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619
All Divisions	$46,291	$37,555

例 9: 複数回答式の調査データに基づくレポート作成

要素:

PROC TABULATE ステートメントオプション
- DATA=

TABLE ステートメント
- 分母定義(山かっこ演算子)
- N 統計量
- PCTN 統計量
- 変数リスト

VAR ステートメント

DATA ステップ

FORMAT プロシジャ

FOOTNOTE ステートメント

OPTIONS ステートメントオプション
- FORMDLIM=
- NONUMBER

SYMPUT ルーチン

TITLE ステートメント

データセット: CUSTOMER_RESPONSE

詳細

この例の 2 つのテーブルに、次を示します。

- 製品を購入する顧客の決定に最も影響している要因
- 企業に関する顧客の情報源

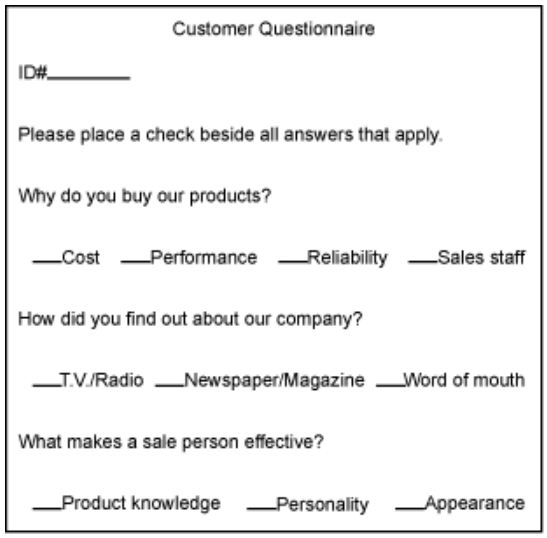
レポートは、1 つのページ番号のみ付いた 1 つの物理ページに表示されます。デフォルトでは、レポートは別々のページに表示されます。

この例では、これらのテーブルを作成する方法だけでなく、次を行う方法も示されます。

- DATA ステップを使用して、データセットのオブザベーション数をカウントする
- 値をマクロ変数に保存する
- その値に後で SAS セッションでアクセスする

次の図に、データの収集に使用される調査用紙を示します。

図 61.8　記入式調査用紙



プログラム

```
data _null_;
  if 0 then set customer_response nobs=count;
  call symput('num',left(put(count,4.)));
  stop;
run;

proc format;
  picture pctfmt low-high='009.9 %';
run;

proc tabulate data=customer_response;

  var factor1-factor4 customer;

  table factor1='Cost'
      factor2='Performance'
      factor3='Reliability'
      factor4='Sales Staff',
      (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;

  title 'Customer Survey Results: Spring 1996';
  title3 'Factors Influencing the Decision to Buy';
run;

proc tabulate
data=customer_response;

  var source1-source3 customer;

  table source1='TV/Radio'
      source2='Newspaper'
      source3='Word of Mouth',
      (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;

  title 'Source of Company Name';
  footnote "Number of Respondents: &num";
run;
```

```
options formdlim='' number;
```

プログラムの説明

オブザベーション数をマクロ変数に保存します。 SET ステートメントはコンパイル時に CUSTOMER_RESPONSE のディスクリプター部分を読み込み、オブザベーション数(回答者数)を COUNT に保存します。SYMPUT ルーチンは、COUNT の値をマクロ変数 NUM に保存します。その他のプロシジャと DATA ステップは、この変数をその他の SAS セッションに使用できます。常に false の IF 0 条件により、オブザベーションを読み込む SET ステートメントが実行されなくなります。(オブザベーションの読み込みは不要です)。STOP ステートメントにより、DATA ステップは 1 回だけ実行されるようになります。

```
data _null_;
  if 0 then set customer_response nobs=count;
  call symput('num',left(put(count,4.)));
  stop;
run;
```

PCTFMT.出力形式を作成します。 FORMAT プロシジャは、パーセント用の出力形式を作成します。PCTFMT.出力形式は、少なくとも 1 桁を含むすべての値を小数点の左に書き込み、1 桁を含むすべての値を小数点の右に書き込みます。ブランクとパーセント記号が桁の後に続きます。

```
proc format;
  picture pctfmt low-high='009.9 %';
run;
```

レポートを作成し、デフォルトのテーブルオプションを使用します。

```
proc tabulate data=customer_response;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が変数 Factor1、Factor2、Factor3、Factor4 および Customer の統計量を計算するように指定します。変数 Customer は、TABLE ステートメントで定義される Percent 列の計算に使用されるため、表示する必要があります。

```
  var factor1-factor4 customer;
```

テーブルの行と列を定義します。 TABLE ステートメントは、係数ごとに 1 行、度数カウントごとに 1 列、パーセントごとに 1 列作成します。引用符内のテキストは、対応する行または列に対するヘッダーを提供します。出力形式修飾子 F=7.と F=PCTFMT9.は関連するセルの値に対し出力形式を提供し、列幅を列ヘッダーに合うように拡張します。

```
  table factor1='Cost'
      factor2='Performance'
      factor3='Reliability'
      factor4='Sales Staff',
      (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;
```

タイトルを指定します。

```
  title 'Customer Survey Results: Spring 1996';
  title3 'Factors Influencing the Decision to Buy';
run;
```

レポートを作成し、デフォルトのテーブルオプションを使用します。

```
proc tabulate
data=customer_response;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が変数 Source1、Source2、Source3 および Customer の統計量を計算するように指定します。変数 Customer は、分母定義で表示されるため、変数リストに表示する必要があります。

```
var source1-source3 customer;
```

テーブルの行と列を定義します。 TABLE ステートメントは、会社名のソースごとに 1 行、度数カウントごとに 1 列、パーセントごとに 1 列作成します。引用符内のテキストは、対応する行または列に対するヘッダーを提供します。

```
table source1='TV/Radio'
    source2='Newspaper'
    source3='Word of Mouth',
    (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;
```

タイトルとフットノートを指定します。 マクロ変数 NUM は、回答者数に解決されます。FOOTNOTE ステートメントは、マクロ変数が解決するように単一引用符ではなく二重引用符を使用します。

```
title 'Source of Company Name';
footnote "Number of Respondents: &num";
run;
```

SAS システムオプションをリセットします。 FORMDLIM=オプションは、ページ区切りをページ排出にリセットします。NUMBER オプションは、後続ページでのページ番号の表示を再開します。

```
options formdlim='' number;
```

出力

アウトプット 61.24　顧客アンケート結果: 1996 年春

Customer Survey Results: Spring 1996

Factors Influencing the Decision to Buy

	Count	Percent
Cost	87	72.5 %
Performance	62	51.6 %
Reliability	30	25.0 %
Sales Staff	120	100.0 %

Source of Company Name

	Count	Percent
TV/Radio	92	76.6 %
Newspaper	69	57.5 %
Word of Mouth	26	21.6 %

Number of Respondents: 120

例 10: 複数選択式の調査データに基づくレポート作成

要素:

CLASS ステートメント
PROC TABULATE ステートメントオプション
　DATA=
　FORMAT=
TABLE ステートメント
　N 統計量
DATA ステップ
FORMAT プロシジャ
FORMAT ステートメント
TRANSPOSE プロシジャ
データセットオプション
　RENAME=
TITLE ステートメント

データセット: RADIO

詳細

リスナーの好みに関するこのレポートには、通常の平日の 7 つの各期間に各種の番組を選択するリスナー数を示します。データは調査によって収集され、結果は SAS データセットに保存されています。このデータセットにはこのレポートに必要なすべての情報が含まれていますが、その情報は PROC TABULATE が使用できるようには調整されていません。

時間帯とラジオ番組選択のクロス集計表を作成するには、時間帯に対する変数と番組の好みに対する変数を含むデータセットがなければなりません。PROC TRANSPOSE は、データをこれらの変数を含む新しいデータセットに再形成します。データが適切な形式になると、PROC TABULATE はレポートを作成します。

次の図に、データの収集に使用される調査用紙を示します。

図 61.9　記入式調査用紙

phone_ _ _

LISTENER SURVEY

1. ____What is your age?

2. ____What is your gender?

3. ____On the average WEEKDAY, how many hours do you listen to the radio?

4. ____On the average WEEKEND-DAY, how many hours do you listen to the radio?

Use codes 1-8 for questions 5. Use codes 0-8 for 6-19.

0 Do not listen at that time

1	Rock	5	Classical
2	Top 40	6	Easy Listening
3	Country	7	News/Information/Talk
4	Jazz	8	Other

5. ____What style of music or radio programming do you most often listen to?

On a typical WEEKDAY, what kind of radio programming do you listen to	On a typical WEEKEND-DAY, what kind of radio programming do you listen to
6. ____from 6-9 a.m.?	13. ____from 6-9 a.m.?
7. ____from 9 a.m. to noon?	14. ____from 9 a.m. to noon?
8. ____from noon to 1 p.m.?	15. ____from noon to 1 p.m.?
9. ____from 1-4 p.m.?	16. ____from 1-4 p.m.?
10. ____from 4-6 p.m.?	17. ____from 4-6 p.m.?
11. ____from 6-10 p.m.?	18. ____from 6-10 p.m.?
12. ____from 10 p.m. to 2 a.m.?	19. ____from 10 p.m. to 2 a.m.?

外部ファイル (2510 ページ)には、調査に対する生データが含まれています。そのファイルからの複数行がここに表示されます。

```
967 32 f 5 3 5
7 5 5 5 7 0 0 0 8 7 0 0 8 0
781 30 f 2 3 5
5 0 0 0 5 0 0 0 4 7 5 0 0 0
859 39 f 1 0 5
1 0 0 0 1 0 0 0 0 0 0 0 0 0

 ... more data lines ...

859 32 m .25 .25 1
1 0 0 0 0 0 0 0 1 0 0 0 0 0
```

プログラム

```
data radio;
  infile 'input-file' missover;

  input /(Time1-Time7) ($1. +1);
  listener=_n_;
run;

proc format;
  value $timefmt 'Time1'='6-9 a.m.'
                 'Time2'='9 a.m. to noon'
                 'Time3'='noon to 1 p.m.'
                 'Time4'='1-4 p.m.'
                 'Time5'='4-6 p.m.'
                 'Time6'='6-10 p.m.'
                 'Time7'='10 p.m. to 2 a.m.'
                  other='*** Data Entry Error ***';
  value $pgmfmt      '0'="Don't Listen"
                 '1','2'='Rock and Top 40'
                    '3'='Country'
               '4','5','6'='Jazz, Classical, and Easy Listening'
                    '7'='News/ Information /Talk'
                    '8'='Other'
                   other='*** Data Entry Error ***';
run;

proc transpose data=radio
               out=radio_transposed(rename=(col1=Choice))
               name=Timespan;
  by listener;
  var time1-time7;
run;

proc tabulate data=radio_transposed format=12.;

format timespan $timefmt. choice $pgmfmt.;

  class timespan choice;

  table timespan='Time of Day',
        choice='Choice of Radio Program'*n='Number of Listeners';
```

```
  title 'Listening Preferences on Weekdays';
run;
```

プログラムの説明

RADIO データセットを作成し、入力ファイルを指定します。 RADIO には、336 人のリスナーの調査からのデータが含まれています。データセットには、視聴者と、リスナーのラジオ番組の好みに関する情報が含まれています。INFILE ステートメントは、そのデータを含む外部ファイルを指定します。MISSOVER は、INPUT ステートメントに表示されるすべての変数に対する現在の行に値が見つからない場合、入力ポインターが次のレコードに移動しないようにします。

```
data radio;
  infile 'input-file' missover;

  input /(Time1-Time7) ($1. +1);
  listener=_n_;
run;
```

$TIMEFMT.出力形式と$PGMFMT.出力形式を作成します。 PROC FORMAT は、時間帯用と番組選択用の出力形式を作成します。

```
proc format;
  value $timefmt 'Time1'='6-9 a.m.'
                 'Time2'='9 a.m. to noon'
                 'Time3'='noon to 1 p.m.'
                 'Time4'='1-4 p.m.'
                 'Time5'='4-6 p.m.'
                 'Time6'='6-10 p.m.'
                 'Time7'='10 p.m. to 2 a.m.'
                   other='*** Data Entry Error ***';
  value $pgmfmt      '0'="Don't Listen"
                 '1','2'='Rock and Top 40'
                   '3'='Country'
               '4','5','6'='Jazz, Classical, and Easy Listening'
                   '7'='News/ Information /Talk'
                   '8'='Other'
                  other='*** Data Entry Error ***';
run;
```

RADIO データセットを転置して、データを再形成します。 PROC TRANSPOSE は、RADIO_TRANSPOSED を作成します。このデータセットには、元のデータセットからの変数 Listener が含まれています。また、2 つの転置された変数、Timespan と Choice も含まれています。Timespan には、出力データセットのオブザベーションを形成するために転置される入力データセットからの変数名(Time1-Time7)が含まれています。Choice には、これらの変数の値が含まれています。(PROC TRANSPOSE ステップの説明については、“詳細” (2322 ページ)を参照。)

```
proc transpose data=radio
              out=radio_transposed(rename=(col1=Choice))
              name=Timespan;
  by listener;
  var time1-time7;
run;
```

レポートを作成し、テーブルオプションを指定します。 FORMAT=オプションは、各テーブルセルの値に対するデフォルトの出力形式を指定します。

```
proc tabulate data=radio_transposed format=12.;
```

転置された変数をフォーマットします。 FORMAT ステートメントは、これらの出力形式を出力データセットの変数と常に関連付けます。

```
format timespan $timefmt. choice $pgmfmt.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Timespan と Choice を分類変数として識別します。

```
class timespan choice;
```

テーブルの行と列を定義します。 TABLE ステートメントは、Timespan のフォーマットされた値ごとに 1 行、Choice のフォーマットされた値ごとに 1 列を作成します。各列には、N 統計量の値が表示されます。引用符内のテキストは、対応する行または列に対するヘッダーを提供します。

```
table timespan='Time of Day',
    choice='Choice of Radio Program'*n='Number of Listeners';
```

タイトルを指定します。

```
title 'Listening Preferences on Weekdays';
run;
```

出力

アウトプット 61.25　平日の視聴傾向

Listening Preferences on Weekdays

	Choice of Radio Program					
	Don't Listen	Rock and Top 40	Country	Jazz, Classical, and Easy Listening	News/ Information /Talk	Other
	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners
Time of Day						
6-9 a.m.	34	143	7	39	96	17
9 a.m. to noon	214	59	5	51	3	4
noon to 1 p.m.	238	55	3	27	9	4
1-4 p.m.	216	60	5	50	2	3
4-6 p.m.	56	130	6	57	69	18
6-10 p.m.	202	54	9	44	20	7
10 p.m. to 2 a.m.	264	29	3	36	2	2

詳細

データを再形成する

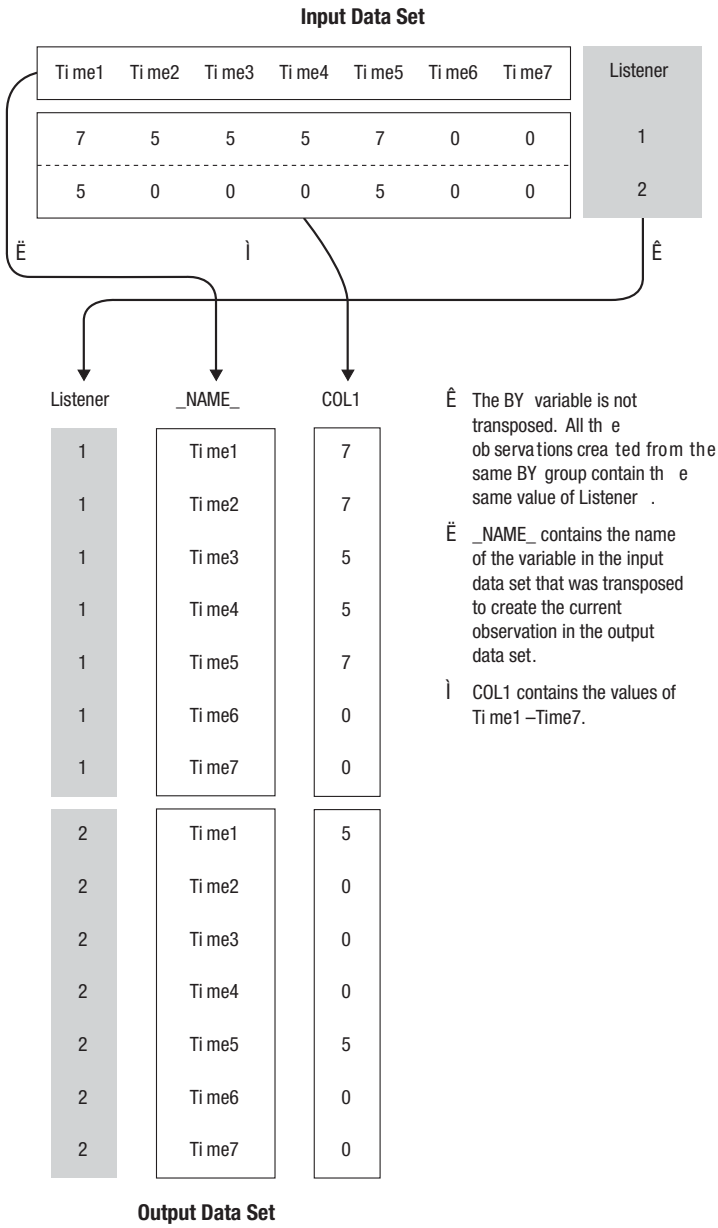
元の入力データセットには、クロス集計表の作成に必要なすべての 情報が含まれていますが、PROC TABULATE はその形式の情報を使用できません。PROC TRANSPOSE は、新しいデータセットの各オブザベーションに変数 Listener、時間帯に対する変数、番組の好みに対する変数が含まれるように、データを再配置します。次の図に、転置について説明します。PROC TABULATE はこの新しいデータセットを使用して、クロス集計表レポートを作成します。

PROC TRANSPOSE は、1 つのオブザベーションに保存された値が 1 つの変数に書き込まれるように、データを再構築します。転置する変数を指定できます。

この例のように BY 処理を使用して転置する場合、転置する変数ごとに 1 つのオブザベーションを各 BY グループから作成します。この例では、Listener が BY 変数です。入力データセットの各オブザベーションは、BY グループです。Listener の値が各オブザベーションに対して一意であるためです。

この例では、Time1 から Time7 までの 7 つの変数を転置します。そのため、出力データセットには、入力データセットの各 BY グループ(各オブザベーション)からの 7 つのオブザベーションが含まれます。

図 61.10 2 つの観測の転置



PROC TRANSPOSE ステップについて

データを再形成する PROC TRANSPOSE ステップの詳細な説明を次に示します。

```
proc transpose data=radio   1
        out=radio_transposed(rename=(col1=Choice))   2

        name=Timespan;   3
  by listener;   4
  var time1-time7;   5
  format timespan $timefmt. choice $pgmfmt.;   6
run;
```

1. DATA=オプションは、入力データセットを指定します。
2. OUT=オプションは、出力データセットを指定します。RENAME=データセットオプションは、転置された変数の名前を COL1 (デフォルト名)から Choice に変更します。
3. NAME=オプションは、現在のオブザベーションを作成するために転置中の変数の名前を含む出力データセットの変数の名前を指定します。デフォルトでは、この変数の名前は_NAME_です。
4. BY ステートメントは Listener を BY 変数として識別します。
5. VAR ステートメントは Time1 から Time7 までを転置する変数として識別します。
6. FORMAT ステートメントは出力形式を Timespan と Choice に割り当てます。レポートを作成する PROC TABULATE ステップは Timespan と Choice をフォーマットする必要はありません。出力形式はこれらの変数とともに保存されるためです。

例 11: さまざまなパーセント表示の統計量の計算

要素:

CLASS ステートメント

PROC TABULATE ステートメントオプション
- FORMAT=

TABLE ステートメントオプション
- ALL 分類変数
- COLPCTSUM 統計量
- 連結(ブランク)演算子
- クロス(*)演算子
- 出力形式修飾子
- 要素グループ化(かっこ)演算子
- ラベル
- REPPCTSUM 統計量
- ROWPCTSUM 統計量
- 変数リスト
- ROW=FLOAT
- RTS=

VAR ステートメント

DATA ステップ

FORMAT プロシジャ

TITLE ステートメント

詳細

この例では、COLPCTSUM、REPPCTSUM および ROWPCTSUM の 3 つのパーセント合計統計量の使用方法を示します。

プログラム

```
data fundrais;
  length name $ 8 classrm $ 1;
  input @1 team $ @8 classrm $ @10 name $
      @19 pencils @23 tablets;
  sales=pencils + tablets;
  datalines;
BLUE   A ANN      4   8
RED    A MARY     5  10
GREEN  A JOHN     6   4
RED    A BOB      2   3
BLUE   B FRED     6   8
GREEN  B LOUISE   12   2
BLUE   B ANNETTE   .   9
RED    B HENRY    8  10
GREEN  A ANDREW   3   5
RED    A SAMUEL   12  10
BLUE   A LINDA    7  12
GREEN  A SARA     4   .
BLUE   B MARTIN   9  13
RED    B MATTHEW  7   6
GREEN  B BETH     15  10
RED    B LAURA    4   3
;

proc format;
  picture pctfmt low-high='009 %';
run;

title "Fundraiser Sales";

proc tabulate format=7.;

  class team classrm;

  var sales;

  table (team all),

      classrm='Classroom'*sales=' '*(sum
      colpctsum*f=pctfmt9.
      rowpctsum*f=pctfmt9.
      reppctsum*f=pctfmt9.)
      all*sales*sum=' '

      /rts=20;
run;
```

プログラムの説明

FUNDRAIS データセットを作成します。 FUNDRAIS には、基金活動での学生の売上に関するデータが含まれています。DATA ステップは、データセットを作成します。

```
data fundrais;
  length name $ 8 classrm $ 1;
  input @1 team $ @8 classrm $ @10 name $
      @19 pencils @23 tablets;
  sales=pencils + tablets;
  datalines;
BLUE   A ANN      4   8
RED    A MARY     5  10
GREEN  A JOHN     6   4
RED    A BOB      2   3
BLUE   B FRED     6   8
GREEN  B LOUISE   12   2
BLUE   B ANNETTE   .   9
RED    B HENRY    8  10
GREEN  A ANDREW    3   5
RED    A SAMUEL   12  10
BLUE   A LINDA    7  12
GREEN  A SARA     4   .
BLUE   B MARTIN   9  13
RED    B MATTHEW  7   6
GREEN  B BETH     15  10
RED    B LAURA    4   3
;
```

PCTFMT.出力形式を作成します。 FORMAT プロシジャは、パーセント用の出力形式を作成します。PCTFMT.出力形式は、少なくとも 1 つの桁、ブランク、パーセント記号を含むすべての値を書き込みます。

```
proc format;
  picture pctfmt low-high='009 %';
run;
```

タイトルを指定します。

```
title "Fundraiser Sales";
```

レポートを作成し、テーブルオプションを指定します。 FORMAT=オプションは、最大 7 桁を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate format=7.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Team と Classrm を分類変数として識別します。

```
  class team classrm;
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Sales 変数の統計量を計算するように指定します。

```
  var sales;
```

テーブル行を定義します。 TABLE ステートメントの行次元は、Team のフォーマットされた値に対しそれぞれ行を作成します。レポートの最終行には、すべてのチームの売上が要約されます。

```
table (team all),
```

テーブル列を定義します。 TABLE ステートメントの列次元は、Classrm のフォーマットされた値に対しそれぞれ列を作成します。Classrm の各値内でクロスするのは、ブランクのラベルを含む分析変数(sales)です。各列内でネストされるのは、クラスの売上を要約する列です。sum とラベル付けされている最初のネストされた列は、クラスルーム行の売上合計です。ColPctSum とラベル付けされている 2 番目のネストされた列は、クラスルームのすべてのチームの 売上合計に対するクラスルーム行の売上合計です。RowPctSum とラベル付けされている 3 番目のネストされた列は、すべてのクラスルーム行の売上合計に対するクラスルーム行の売上合計のパーセントです。RepPctSum とラベル付けされている 4 番目のネストされた列は、すべてのクラスルームのすべてのチームの 売上合計に対するクラスルーム行の売上合計のパーセントです。レポートの最終列に、すべてのクラスルーム 行の売上が要約されます。

```
classrm='Classroom'*sales=' '*(sum
colpctsum*f=pctfmt9.
rowpctsum*f=pctfmt9.
reppctsum*f=pctfmt9.)
all*sales*sum=' '
```

行タイトルのスペースを 指定し、ブランクの行ヘッダーを削除します。 RTS=は、行ヘッダーに対する行ごとに 20 文字を提供します。

```
/rts=20;
run;
```

出力

アウトプット 61.26　基金活動の売上

Fundraiser Sales

	Classroom								All
	A				B				
	Sum	ColPctSum	RowPctSum	RepPctSum	Sum	ColPctSum	RowPctSum	RepPctSum	sales
team									
BLUE	31	34 %	46 %	15 %	36	31 %	53 %	17 %	67
GREEN	18	19 %	31 %	8 %	39	34 %	68 %	19 %	57
RED	42	46 %	52 %	20 %	38	33 %	47 %	18 %	80
All	91	100 %	44 %	44 %	113	100 %	55 %	55 %	204

詳細

次に、クラスルーム A の Blue チーム用の出力の生成に使用されるパーセント合計統計量計算を示します。

- COLPCTSUM=31/91*100=34%

- ROWPCTSUM=31/67*100=46%
- REPPCTSUM=31/204*100=15%

同様の計算が、その他のチームとクラスルーム用の出力の生成に使用されます。

例 12: 分母定義を使用し、基本的な度数カウントとパーセントを表示する

要素:

CLASS ステートメント
PROC TABULATE ステートメントオプション
    DATA=
    FORMAT=
TABLE ステートメントオプション
    ALL 分類変数
    分母定義(山かっこ演算子)
    N 統計量
    PCTN 統計量
    RTS=
DATA ステップ
FORMAT プロシジャ
FORMAT ステートメント
TITLE ステートメント

詳細

クロス集計表テーブル(分割表、スタブアンドバナーレポートとも呼ばれます)には、2 つ以上の変数に対して組み合わされた度数分布が表示されます。このテーブルには、4 つの職階における女性と男性に対する度数カウントがそれぞれ表示されます。度数カウントがそれぞれ次を表すパーセントも表示されます。

- 職階における女性と男性の合計(行パーセント)
- すべての 職階における性別の合計(列パーセント)
- すべての従業員の合計

プログラム

```
data jobclass;
  input Gender Occupation @@;
  datalines;
1 1  1 1  1 1  1 1  1 1  1 1  1 1
1 2  1 2  1 2  1 2  1 2  1 2  1 2
```

```
1 3  1 3  1 3  1 3  1 3  1 3  1 3
1 1  1 1  1 1  1 2  1 2  1 2  1 2
1 2  1 2  1 3  1 3  1 4  1 4  1 4
1 4  1 4  1 4  1 1  1 1  1 1  1 1
1 1  1 2  1 2  1 2  1 2  1 2  1 2
1 2  1 3  1 3  1 3  1 3  1 4  1 4
1 4  1 4  1 4  1 1  1 3  2 1  2 1
2 1  2 1  2 1  2 1  2 1  2 2  2 2
2 2  2 2  2 2  2 3  2 3  2 3  2 4
2 4  2 4  2 4  2 4  2 4  2 1  2 3
2 3  2 3  2 3  2 3  2 4  2 4  2 4
2 4  2 4  2 1  2 1  2 1  2 1  2 1
2 2  2 2  2 2  2 2  2 2  2 2  2 2
2 3  2 3  2 4  2 4  2 4  2 1  2 1
2 1  2 1  2 1  2 2  2 2  2 2  2 3
2 3  2 3  2 3  2 4
;

proc format;
  value gendfmt 1='Female'
                2='Male'
               other='*** Data Entry Error ***';
  value occupfmt 1='Technical'
                 2='Manager/Supervisor'
                 3='Clerical'
                 4='Administrative'
                other='*** Data Entry Error ***';
run;

proc tabulate data=jobclass format=8.2;

  class gender occupation;

   table (occupation='Job Class' all='All Jobs')
         *(n='Number of employees'*f=9.
         pctn<gender all>='Percent of row total'
         pctn<occupation all>='Percent of column total'
         pctn='Percent of total'),

   gender='Gender' all='All Employees'/ rts=50;

  format gender gendfmt. occupation occupfmt.;

  title 'Gender Distribution';
  title2 'within Job Classes';
run;
```

プログラムの説明

JOBCLASS データセットを作成します。 JOBCLASS には、架空の会社の従業員の性別と職階に関するエンコード情報が含まれています。

```
data jobclass;
  input Gender Occupation @@;
  datalines;
1 1  1 1  1 1  1 1  1 1  1 1  1 1
1 2  1 2  1 2  1 2  1 2  1 2  1 2
1 3  1 3  1 3  1 3  1 3  1 3  1 3
```

```
11 11 11 12 12 12 12
12 12 13 13 14 14 14
14 14 14 11 11 11 11
11 12 12 12 12 12 12
12 13 13 13 13 14 14
14 14 14 11 13 21 21
21 21 21 21 21 22 22
22 22 22 23 23 23 24
24 24 24 24 24 21 23
23 23 23 23 24 24 24
24 24 21 21 21 21 21
22 22 22 22 22 22 22
23 23 24 24 24 21 21
21 21 21 22 22 22 23
23 23 23 24
;
```

GENDFMT.出力形式と OCCUPFMT.出力形式を作成します。 PROC FORMAT は、変数 Gender と Occupation に対し出力形式を作成します。

```
proc format;
  value gendfmt 1='Female'
                2='Male'
               other='*** Data Entry Error ***';
  value occupfmt 1='Technical'
                 2='Manager/Supervisor'
                 3='Clerical'
                 4='Administrative'
                other='*** Data Entry Error ***';
run;
```

レポートを作成し、テーブルオプションを指定します。 FORMAT=オプションは、8.2 出力形式を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=jobclass format=8.2;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、Gender と Occupation を分類変数として識別します。

```
class gender occupation;
```

```
table (occupation='Job Class' all='All Jobs')
      *(n='Number of employees'*f=9.
      pctn<gender all>='Percent of row total'
      pctn<occupation all>='Percent of column total'
      pctn='Percent of total'),
```

テーブル列を定義して、行ヘッダーのスペース量を指定します。 列次元は、Gender のフォーマットされた値ごとに 1 列、全従業員に対し 1 列を作成します。引用符内のテキストは、対応する列に対してヘッダーを提供します。RTS=オプションは、行ヘッダーに対する行ごとに 50 文字を提供します。

```
gender='Gender' all='All Employees'/ rts=50;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を変数 Gender と Occupation に割り当てます。

```
format gender gendfmt. occupation occupfmt.;
```

タイトルを指定します。

```
  title 'Gender Distribution';
  title2 'within Job Classes';
run;
```

出力

アウトプット 61.27　職階内の性別分布

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

詳細

概要

テーブルの行を定義する TABLE ステートメントの部分には PCTN 統計量が使用され、3 つの異なるパーセントが計算されます。

PCTN のすべての計算では、分子は N で、テーブルの 1 つのセルに対する度数カウントです。PCTN の発生ごとの分母は、分母定義によって決定されます。分母定義は、キーワード PCTN の後に山かっこで囲まれて表示されます。これは、1 つ以上の式のリストです。リストは、分母に対して合計する度数カウントを PROC TABULATE に認識させます。

テーブル構造を分析する

テーブルの構造をよく見ると、PROC TABULATE が分母定義をどのように使用するかがわかります。次の TABLE ステートメントの簡略バージョンで、テーブルの基本構造について説明します。

```
table occupation='Job Class' all='All Jobs',
    gender='Gender' all='All Employees';
```

テーブルは、4 つのサブテーブルの組み合わせです。このレポートでは、サブテーブルはそれぞれ行次元の 1 つの分類変数と列次元の 1 つの分類変数のクロスです。クロスはそれぞれ 1 つ以上のカテゴリを作成します。カテゴリは female, technical or all, clerical などの分類変数の一意の値の組み合わせです。

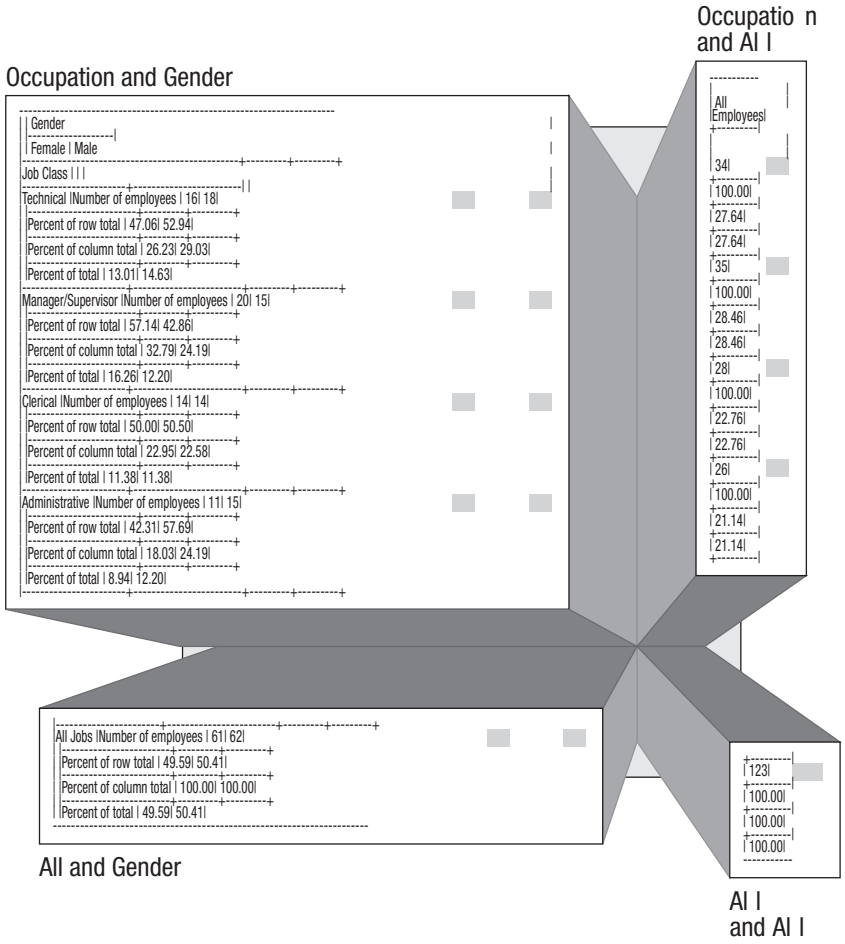
次のテーブルでは、それぞれのサブテーブルについて説明します。

表 61.9　サブテーブルのコンテンツ

サブテーブルに影響する分類変数	度数カウントの説明	カテゴリ名
Occupation と Gender	各職における女性数または各職における男性数	8
All と Gender	女性数または男性数	2
Occupation と All	各職の人数	4
All と All	すべての職の人数	1

次の図では、これらのサブテーブル、各カテゴリに対する度数カウントがハイライト表示されています。

図 61.11 4 つのサブテーブルの説明



分母定義を解釈する

TABLE ステートメントの次のフラグメントは、このレポートに対する分母定義を定義します。PCTN キーワードと分母定義がハイライト表示されています。

```
table (occupation='Job Class' all='All Jobs')
        *(n='Number of employees'*f=5.
          pctn<gender all>='Row percent'
          pctn<occupation all>='Column percent'
          pctn='Percent of total'),
```

PCTN を使用するたびに Occupation と All の各値内の統計量の行がネストされます。各分母定義は、その行の分母に対して合計する度数カウントを PROC TABULATE に認識させます。このセクションでは、これらの分母定義を PROC TABULATE がどのように解釈するかについて説明します。

行のパーセント

行のパーセントを計算し、行にラベル付けする TABLE ステートメントの一部は、次のとおりです。

```
pctn<gender all>='Row percent'
```

各サブテーブルに対するこの分母定義を PROC TABULATE がどのように解釈するかを考えます。

アウトプット 61.28　サブテーブル 1: 職業と性別

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は、分母定義の最初の要素 Gender を参照し、Gender がサブテーブルに影響するかどうかを確認します。Gender がサブテーブルに影響しているため、PROC TABULATE は Gender を分母定義として使用します。この分母定義は、PROC TABULATE に対し、Occupation の同じ値内の Gender のすべての 発生に対する度数カウントを合計するように命令します。

たとえば、カテゴリ female, technical の分母は、Occupation の値が technical であるこのサブテーブルのすべてのカテゴリに 対するすべての度数カウントの合計です。そのようなカテゴリには、female, technical と male, technical の 2 つがあります。対応する度数カウントは 16 と 18 です。そのため、このカテゴリの分母は 16+18、または 34 となります。

アウトプット 61.29　サブテーブル 2: すべてとジェンダー

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は、分母定義の最初の要素 Gender を参照し、Gender がサブテーブルに影響するかどうかを確認します。Gender がサブテーブルに影響しているため、PROC TABULATE は Gender を分母定義として使用します。この分母定義は、PROC TABULATE に対し、サブテーブルの Gender のすべての発生に対する度数カウントを合計するように命令します。

たとえば、カテゴリ all, female の分母は、all, female と all, male に対する度数カウントの合計です。対応する度数カウントは 61 と 62 です。そのため、このサブテーブルのセルの分母は、61+62 または 123 になります。

アウトプット 61.30　サブテーブル 3: 職業とすべて

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は、分母定義の最初の要素 Gender を参照し、Gender がサブテーブルに影響するかどうかを確認します。Gender がサブテーブルに影響しないため、PROC TABULATE は分母定義の次の要素、All を参照します。変数 All はこのサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。All は、カテゴリを 1 つだけ含む予約分類変数です。そのため、この分母定義は、PROC TABULATE に対し、All の度数カウントを分母として使用するように命令します。

たとえば、カテゴリ clerical, all の分母はそのカテゴリに対する度数カウント、28 となります。

注: これらのテーブルセルでは、分子と分母が同じため、このサブテーブルの行のパーセントはすべて 100 になります。

アウトプット 61.31　サブすべて 4: すべてとジェンダー

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は、分母定義の最初の要素 Gender を参照し、Gender がサブテーブルに影響するかどうかを確認します。Gender がサブテーブルに影響しないため、PROC TABULATE は分母定義の次の要素、All を参照します。変数 All はこのサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。All は、カテゴリを 1 つだけ含む予約分類変数です。そのため、この分母定義は、PROC TABULATE に対し、All の度数カウントを分母として使用するように命令します。

このサブテーブルには、カテゴリ all, all のみが含まれます。このカテゴリの分母は 123 となります。

注: このテーブルセルでは、分子と分母が同じため、このサブテーブルの行のパーセントは 100 となります。

列のパーセント

列のパーセントを計算し、行にラベル付けする TABLE ステートメントの一部は、次のとおりです。

```
pctn<occupation all>='Column percent'
```

各サブテーブルに対するこの分母定義を PROC TABULATE がどのように解釈するかを考えます。

アウトプット 61.32　サブテーブル 1: 職業と性別

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は分母定義の最初の要素 Occupation を参照し、Occupation がサブテーブルに影響するかどうかを確認します。Occupation がサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。この分母定義は、PROC TABULATE に対し、Gender の同じ値内の Occupation のすべての発生に対する度数カウントを合計するように命令します。

たとえば、カテゴリ manager/supervisor, male の分母は、Gender の値が male であるこのサブテーブルのすべてのカテゴリに 対するすべての度数カウントの合計です。そのようなカテゴリには、technical, male、manager/supervisor, male、clerical, male、administrative, male の 4 つがあります。対応する度数カウントは 18、15、14、15 です。そのため、このカテゴリの分母は、18+15+14+15 または 62 となります。

アウトプット 61.33　サブテーブル 2: すべてとジェンダー

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は分母定義の最初の要素 Occupation を参照し、Occupation がサブテーブルに影響するかどうかを確認します。Occupation がサブテーブルに影響しないため、PROC TABULATE は分母定義の次の要素、All を参照します。変数 All がこのサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。All は、カテゴリを 1 つだけ含む予約分類変数です。そのため、この分母定義は、PROC TABULATE に対し、All の度数カウントを分母として使用するように命令します。

たとえば、カテゴリ all, female の分母はそのカテゴリに対する度数カウント、61 となります。

注: これらのテーブルセルでは、分子と分母が同じため、このサブテーブルの列のパーセントはすべて 100 となります。

アウトプット 61.34　サブテーブル 3: 職業とすべて

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は分母定義の最初の要素 Occupation を参照し、Occupation がサブテーブルに影響するかどうかを確認します。Occupation がサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。この分母定義は、PROC TABULATE に対し、サブテーブルの Occupation のすべての発生に対する度数カウントを合計するように命令します。

たとえば、カテゴリ technical, all の分母は、technical, all、manager/supervisor, all、clerical, all、administrative, all に対する度数カウントの合計です。対応する度数カウントは 34、35、28、26 です。そのため、このカテゴリの分母は、34+35+28+26 または 123 となります。

アウトプット 61.35　サブテーブル 4: すべてとすべて

Gender Distribution
within Job Classes

		Gender		All Employees
		Female	Male	
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE は分母定義の最初の要素 Occupation を参照し、Occupation がサブテーブルに影響するかどうかを確認します。Occupation がサブテーブルに影響しないため、PROC TABULATE は分母定義の次の要素、All を参照します。変数 All がこのサブテーブルに影響するため、PROC TABULATE はこれを分母定義として使用します。All は、カテゴリを 1 つだけ含む予約分類変数です。そのため、この分母定義は、PROC TABULATE に対し、All の度数カウントを分母として使用するように命令します。

このサブテーブルには、カテゴリ all, all のみが含まれます。このカテゴリの分母は 123 となります。

注: この計算では、分子と分母が同じため、このサブテーブルの列のパーセントは 100 となります。

合計パーセント

合計パーセントを計算し、行にラベル付けする TABLE ステートメントの一部は、次のとおりです。

pctn='Total percent'

分母定義を指定しない場合、PROC TABULATE はサブテーブルのすべての 度数カウントを総計することによりセルに対する分母を取得します。次のテーブルに、この例のすべてのサブテーブルに 対するプロセスを要約します。

表 61.10　合計パーセントの分母

サブテーブルに影響する分類変数	度数カウント	合計
Occupant と Gender	16, 18, 20, 15 14, 14, 11, 15	123
Occupant と All	34, 35, 28, 26	123
Gender と All	61, 62	123
All と All	123	123

結果として、合計パーセントの合計の分母は常に 123 になります。

例 13: ODS 出力にスタイルの上書きを指定する

要素:

CLASS ステートメントオプション
    STYLE=
CLASSLEV ステートメントオプション
    STYLE=
KEYLABEL ステートメント
KEYWORD ステートメントオプション
    STYLE=
PROC TABULATE ステートメントオプション
    DATA=
    STYLE=
TABLE ステートメントオプション
    STYLE=
    MISSTEXT=
    BOX=
ODS HTML5 ステートメント

ODS PDF ステートメント
ODS WORD ステートメント
OPTIONS ステートメント
TITLE ステートメント

データセット: ENERGY
出力形式: REGFMT.
出力形式: DIVFMT.
出力形式: USETYPE.

詳細

この例では、HTML5、Word、および PDF ファイルを作成して、さまざまなテーブル部分に対しスタイルの上書きを指定します。

プログラム

```
options nodate pageno=1;

proc sort data=energy;
  by region;
run;

ods _all_ close;
ods html5 path='file-path' file='CustomEnergy.htm';
ods pdf file='file-path/CustomEnergy.pdf' contents=yes;
ods word file='file-path/CustomEnergy.docx';

proc tabulate data=energy style=[fontweight=bold];

by region;

  class region division type / style=[textalign=center];

  classlev region division type / style=[textalign=left];

  var expenditures / style=[fontsize=3];

  keyword all sum / style=[fontwidth=wide];
  keylabel all="Total";

  table (region all)*(division all*[style=[backgroundcolor=yellow]]),
      (type all)*(expenditures*f=dollar10.) /
      style=[bordercolor=blue]

      misstext=[label="Missing" style=[fontweight=light]]

      box=[label="Region by Division by Type"
        style=[fontstyle=italic]];

  format region regfmt. division divfmt. type usetype.;

  title 'Energy Expenditures';
```

```
  title2 '(millions of dollars)';
run;

ods _all_ close;
```

プログラムの説明

SAS システムオプションを設定します。 NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。

```
options nodate pageno=1;
```

データセットを並べ替えます。

```
proc sort data=energy;
  by region;
run;
```

ODS 出力ファイル名を指定します。 複数の ODS 出力先を開くことによって、一度の実行で複数の出力ファイルを作成できます。ODS HTML5 ステートメントは、HTML 5.0 で書き込まれる出力を生成します。ODS PDF ステートメントは出力を Portable Document Format (PDF)で作成します。ODS WORD ステートメントは、Microsoft Word の出力を生成します。PROC TABULATE からの出力は、これらのファイルにそれぞれ移動します。

ODS PDF ステートメントでは、CONTENTS=オプションによって目次が作成されます。

```
ods _all_ close;
ods html5 path='file-path' file='CustomEnergy.htm';
ods pdf file='file-path/CustomEnergy.pdf' contents=yes;
ods word file='file-path/CustomEnergy.docx';
```

データセルをカスタマイズします。 PROC TABULATE ステートメントの STYLE=オプションは、テーブルのデータセルに対しスタイルの上書きを指定します。

```
proc tabulate data=energy style=[fontweight=bold];
```

BY グループを指定します。 BY ステートメントを指定すると、BY グループテーブルのラベルが目次に表示されます。ラベルは BY 変数の値に基づいています。

```
by region;
```

分類変数名ヘッダーをカスタマイズします。 CLASS ステートメントの STYLE=オプションは、分類変数名ヘッダーに対しスタイルの上書きを指定します。

```
  class region division type / style=[textalign=center];
```

分類変数値のヘッダーをカスタマイズします。 CLASSLEV ステートメントの STYLE=オプションは、分類変数水準値ヘッダーに対しスタイルの上書きを指定します。

```
  classlev region division type / style=[textalign=left];
```

分析変数名ヘッダーをカスタマイズします。 VAR ステートメントの STYLE=オプションは、変数名ヘッダーに対しスタイル要素を指定します。

```
  var expenditures / style=[fontsize=3];
```

キーワードに対しスタイル属性を指定し、“all”キーワードにラベル付けします。
KEYWORD ステートメントの STYLE=オプションは、キーワードに対しスタイル要素を指定します。KEYLABEL ステートメントは、ラベルをキーワードに割り当てます。

```
keyword all sum / style=[fontwidth=wide];
keylabel all="Total";
```

テーブルの行と列を定義します。 次元式の STYLE=オプションは、テーブルセルの優先を指定する PROC TABULATE の他のすべての STYLE=指定に優先します。スラッシュ(/)の後の STYLE=オプションは、テーブルセル以外のテーブルの部分に対しスタイルの上書きを指定します。

```
table (region all)*(division all*[style=[backgroundcolor=yellow]]),
    (type all)*(expenditures*f=dollar10.) /
    style=[bordercolor=blue]
```

欠損値をカスタマイズします。 TABLE ステートメントの MISSTEX オプションの STYLE=オプションは、欠損値を含むテーブルセルのテキストに使用するスタイル要素を指定します。

```
misstext=[label="Missing" style=[fontweight=light]]
```

行タイトルの上にあるボックスをカスタマイズします。 TABLE ステートメントの BOX オプションの中の STYLE=オプションは、行タイトルの上のボックスのテキストに使用するスタイルの上書きを指定します。

```
box=[label="Region by Division by Type"
  style=[fontstyle=italic]];
```

分類変数値をフォーマットします。

```
format region regfmt. division divfmt. type usetype.;
```

タイトルを指定します。

```
  title 'Energy Expenditures';
  title2 '(millions of dollars)';
run;
```

ODS 出力先をすべて閉じます。

```
ods _all_ close;
```

出力

アウトプット 61.36　スタイルのオーバーライドを示す HTML 出力

Energy Expenditures
(millions of dollars)

Region=Northeast

Region by Division by Type		Type		Total
		Residential Customers	Business Customers	
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063

アウトプット 61.37　スタイルのオーバーライドを示す PDF 出力

Energy Expenditures
(millions of dollars)

Region=Northeast

Region by Division by Type		Type		Total
		Residential Customers	Business Customers	
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063

アウトプット 61.38　スタイルのオーバーライドを示す Word 出力

Energy Expenditures
(millions of dollars)

Region=Northeast

Region by Division by Type		Type		
		Residential Customers	Business Customers	Total
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063
Total	Division			
	New England	$7,477	$5,129	$12,606
	Middle Atlantic	$19,379	$15,078	$34,457
	Total	$26,856	$20,207	$47,063

例 14: スタイル優先

要素:

CLASS statement0

CLASSLEV ステートメントオプション
- STYLE=

KEYLABEL ステートメント

LABEL ステートメント

PROC TABULATE ステートメントオプション
- DATA=
- FORMAT=

TABLE ステートメント
- クロス(*)演算子
- STYLE=
- STYLE_PRECEDENCE=オプション

VAR ステートメント

ODS HTML5 ステートメント

FORMAT プロシジャ

FORMAT ステートメント

TITLE ステートメント

データセット: SALES

詳細

この例では、次を行います。

- リージョン別に、販売の種類(小売または卸売)ごとにカテゴリを作成する
- ドル出力形式をテーブルのすべてのセルに 適用する
- 各リージョンと販売の種類に対し、イタリックのフォントスタイルを適用する
- STYLE_PRECEDENCE=オプションに基づいて、スタイル(背景=#edf8b1 (黄色-緑色)またはオレンジ色)を適用する
- ODS HTML5 出力を生成する

プログラム

```
proc format;
    value $saletypefmt 'R'='Retail'
                       'W'='WholeSale';
run;

ods _all_ close;
ods html5 file="file-path/stylePrecedence.html";

title "Style Precedence";
title2 "First Table: no precedence, Orange";
title3 "Second Table: style_precedence=page, Yellow";

proc tabulate data=sales format=dollar10.;

class product region saletype;


classlev region saletype / style={font_style=italic};


var netsales;

label netsales="Net Sales";


keylabel all="Total";


table product *{style={background=#edf8b1}},
region*{style={background=yellow}},
saletype*{style={background=orange}};

table product *{style={background=#edf8b1}},
region*{style={background=yellow}},
saletype*{style={background=orange}} / style_precedence=page;

format saletype $saletypefmt.;


run;
```

プログラムの説明

SALETYPEFMT.出力形式を作成します。 PROC FORMAT は、SALETYPE に対し出力形式を作成します。

```
proc format;
    value $saletypefmt 'R'='Retail'
                       'W'='WholeSale';
run;
```

ODS 出力ファイル名を指定します。 ODS HTML5 ステートメントは、HTML5 で書き込まれる出力を生成します。

```
ods _all_ close;
ods html5 file="file-path/stylePrecedence.html";
```

生成するテーブルのタイトルを指定します。 2 つのテーブルが生成されます。First Table にはスタイル優先は表示されません。Second Table には、優先する色が STYLE_PRECEDENCE オプションによって指定される色に基づいていることが表示されます。

```
title "Style Precedence";
title2 "First Table: no precedence, Orange";
title3 "Second Table: style_precedence=page, Yellow";
```

テーブルオプションを指定します。 FORMAT=オプションは、DOLLAR10.を各テーブルセルの値に対するデフォルトの出力形式として指定します。

```
proc tabulate data=sales format=dollar10.;
```

分析対象となるサブグループを指定します。 CLASS ステートメントは、分析を Product、Region、SaleType の値別に分類します。

```
class product region saletype;
```

サブグループにスタイルを指定します。 CLASSLEV ステートメントは、Region 要素と Saletype 要素にスタイルを指定します。

```
classlev region saletype / style={font_style=italic};
```

分析変数を指定します。 VAR ステートメントは、PROC TABULATE が Netsales 変数の統計量を計算するように指定します。

```
var netsales;
```

ラベルを指定します。 LABEL ステートメントは、Netsales 変数の名前を Net Sales に変更します。

```
label netsales="Net Sales";
```

キーラベルを指定します。 KEYLABEL ステートメントは、共通分類変数 ALL を Total にラベル付けします。

```
keylabel all="Total";
```

テーブルの行と列を定義します。 TABLE ステートメントは、ページごとに製品別のテーブルを作成します。この例では、1 つの製品、A100 があります。また、TABLE ステートメントは、Region のフォーマットされた値ごとに 1 行、SaleType のフォーマットされた値ごとに 1 列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Net Sales の合計が含まれます。次元式の STYLE=オプションは、テーブルセルに属性を指定する PROC TABULATE でのその他の STYLE=指定に優先します。この最初のテーブルでは、列次元はデフォルトで、列と関連付けられているスタイルが優先します。そのため、背景色としてオレンジがデフォルトで使用されます。

```
table product *{style={background=#edf8b1}},
region*{style={background=yellow}},
saletype*{style={background=orange}};
```

STYLE_PRECEDENCE オプションを使用して、テーブル行とテーブル列を定義します。 TABLE ステートメントはページごとに製品別のテーブル、A100 を作成します。また、TABLE ステートメントは、Region のフォーマットされた値ごとに 1 行、SaleType のフォーマットされた値ごとに 1 列を作成します。これらの行と列によって作成される各セルには、セルに影響するすべてのオブザベーションに対する分析変数 Net Sales の合計が含まれます。次元式の STYLE=オプションは、テーブルセルに属性を指定する PROC TABULATE でのその他の STYLE=指定に優先します。2 番目のテーブルでは、STYLE_PRECEDENCE オプションがページ式で指定されます。そのため、背景に適用されるスタイルは#edf8b1(黄緑色)です。

```
table product *{style={background=#edf8b1}},
region*{style={background=yellow}},
saletype*{style={background=orange}} / style_precedence=page;
```

出力をフォーマットします。 FORMAT ステートメントは、出力形式を SaleType 変数に割り当てます。

```
format saletype $saletypefmt.;
```

プログラムを実行します。

```
run;
```

出力

アウトプット 61.39　スタイルの優先順位がないテーブル

Style Precedence
First Table: no precedence, orange
Second Table: style_precedence=page, Yellow

Product A100

	SaleType	
	Retail	*WholeSale*
	N	N
Region		
NC	$3	$3
TX	.	$2

アウトプット 61.40　テーブルスタイルの優先順位

Style Precedence
First Table: no precedence, orange
Second Table: style_precedence=page, Yellow

Product A100

	SaleType	
	Retail	*WholeSale*
	N	N
Region		
NC	$3	$3
TX	.	$2

例 15: NOCELLMERGE オプションの使用

要素:

CLASS ステートメント

PROC TABULATE ステートメントオプション
- DATA=
- STYLE=

CLASS ステートメント

TABLE ステートメント
- クロス(*)演算子
- STYLE=オプション
- NOCELLMERGE=オプション

ODS HTML5 ステートメント
TITLE ステートメント

詳細

この例では、次を行います。

- セルスタイル動作が結合されたテーブルを作成する
- セルが結合されていない 2 つ目のテーブルを作成する
- 空のデータセルとフォーマットデータセルで異なるスタイルが使用されている場合にセルスタイルがどのように影響を受けるかを示す

プログラム

```
ods html5 file="file-path/tabstyle.html";
proc tabulate data=sashelp.class style={background=#edf8b1};
class sex age;

table sex*{style={background=#7fcdbb}} all, age;
title 'Data Cell Styles in Merged Cells';
run;

proc tabulate data=sashelp.class style={background=#edf8b1};
class sex age;

table sex*{style={background=#7fcdbb}} all, age/nocellmerge;
title1 'Data Cell Styles with NOCELLMERGE Option';
run;

ODS html5 close;
```

プログラムの説明

ODS 出力ファイル名を指定します。 ODS HTML5 ステートメントは、HTML で書き込まれる出力を生成します。

```
ods html5 file="file-path/tabstyle.html";
```

PROC TABULATE オプションを指定します。 STYLE=オプションは、テーブルのセルの背景色を#edf8b1 (黄緑色)に設定します。

```
proc tabulate data=sashelp.class style={background=#edf8b1};
```

サブグループを指定します。 CLASS ステートメントは、データを性別および年齢別に分類します。

```
class sex age;
```

テーブルの行と列を定義します。 TABLE ステートメントはテーブルを作成します。次元式の STYLE=オプションは、テーブルセル属性の PROC TABULATE ステートメントからの STYLE=設定に優先し、色を#7fcdbb (緑-シアン)に変更します。

```
table sex*{style={background=#7fcdbb}} all, age;
```

生成するテーブルのタイトルを指定します。 このテーブルでは、スタイル色の変更が結合されたセルにどのように影響するかを示します。

```
title 'Data Cell Styles in Merged Cells';
```

プログラムを実行します。

```
run;
```

PROC TABULATE オプションを指定します。 STYLE=オプションは、テーブルのセルの背景色を#edf8b1 (黄緑色)に設定します。

```
proc tabulate data=sashelp.class style={background=#edf8b1};
```

サブグループを指定します。 CLASS ステートメントは、データを性別および年齢別に分類します。

```
class sex age;
```

テーブルの行と列を定義します。 TABLE ステートメントはテーブルを作成します。次元式の STYLE=オプションは、PROC TABULATE ステートメントからの STYLE=設定に優先しますが、フォーマットされたデータセルに限ります。

```
table sex*{style={background=#7fcdbb}} all, age/nocellmerge;
```

生成するテーブルのタイトルを指定します。 このテーブルでは、スタイル色の変更が結合されていないフォーマットセルにどのように影響するかを示します。

```
title1 'Data Cell Styles with NOCELLMERGE Option';
```

プログラムを実行します。

```
run;
```

ODS HTML5 出力先をクローズします。

```
ODS html5 close;
```

出力

アウトプット 61.41　結合されたセル

Data Cell Styles in Merged Cells

	Age					
	11	12	13	14	15	16
	N	N	N	N	N	N
Sex						
F	1	2	2	2	2	.
M	1	3	1	2	2	1
All	2	5	3	4	4	1

アウトプット 61.42　結合されていないセル

Data Cell Styles with NOCELLMERGE Option

	Age					
	11	12	13	14	15	16
	N	N	N	N	N	N
Sex						
F	1	2	2	2	2	.
M	1	3	1	2	2	1
All	2	5	3	4	4	1

参考文献

Jain, Raj and Imrich Chlamtac. 1985. “The P² Algorithm for Dynamic Calculation of Quantiles and Histograms without Storing Observations.” *Communications of the Association of Computing Machinery* 28 (10): 1076-1085.

62

TRANSPOSE プロシジャ

概要: TRANSPOSE プロシジャ

TRANSPOSE プロシジャの動作について

TRANSPOSE プロシジャは、SAS データセットの値を再構築し、選択した変数をオブザベーションに転置して、出力データセットを作成します。多くの場合、TRANSPOSE プロシジャを使用すると、長い DATA ステップを書かずに同じ結果を得ることができます。さらに、出力データセットは、分析、レポート作成、その他のデータ操作のための後続の DATA または PROC ステップでも使用できます。

PROC TRANSPOSE では、印刷出力は作成されません。PROC TRANSPOSE ステップから出力データセットを印刷するには、PROC PRINT、PROC REPORT または別の SAS レポートツールを使用します。

転置された変数を作成するために、このプロシジャで、入力データセットのオブザベーション値が出力データセットの変数値に転置されます。

PROC TRANSPOSE が実行可能な転置の種類について

単純な転置

次の例では、単純な転置について説明します。入力データセットでは、各変数が 1 つのテスター のスコアを表します。出力データセットでは、各オブザベーションが 1 つのテスターのスコアを表します。_NAME_の各値は、プロシジャで転置された入力データセットの変数名です。したがって、_NAME_の値によって、出力データセットの各オブザベーションのソースが識別されます。たとえば、出力データセットの最初のオブザベーションの値は、入力データセットの変数 Tester1 の値から生成されています。出力を生成するステートメントは次のとおりです。

```
proc print data=proclib.product noobs;
  title 'The Input Data Set';
run;

proc transpose data=proclib.product
               out=proclib.product_transposed;
run;
```

```
proc print data=proclib.product_transposed noobs;
  title 'The Output Data Set';
run;
```

アウトプット 62.1　単純な転置

The Input Data Set 1

Tester1	Tester2	Tester3	Tester4
22	25	21	21
15	19	18	17
17	19	19	19
20	19	16	19
14	15	13	13
15	17	18	19
10	11	9	10
22	24	23	21

The Output Data Set 2

NAME	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8
Tester1	22	15	17	20	14	15	10	22
Tester2	25	19	19	19	15	17	11	24
Tester3	21	18	19	16	13	18	9	23
Tester4	21	17	19	19	13	19	10	21

BY グループを使用する複合転置

BY グループを使用する次の例は、より複合的です。入力データセットは、2 つの湖の魚の重量と体長の測定値を表します。出力データセットを作成するステートメントでは、次が行われます。

- 体長の測定値を含む変数のみを転置します。
- 湖と日付の組み合わせごとに 1 つずつ、6 つの BY グループを作成します。
- データセットオプションを使用して、転置された変数の名前を指定します。

アウトプット 62.2　BY グループによる転置

Input Data Set 1

Location	Date	Length1	Width1	Length2	Width2	Length3	Width3	Length4	Width4
Cole Pond	02JUN95	31	0.25	32	0.30	32	0.25	33	0.30
Cole Pond	03JUL95	33	0.32	34	0.41	37	0.48	32	0.28
Cole Pond	04AUG95	29	0.23	30	0.25	34	0.47	32	0.30
Eagle Lake	02JUN95	32	0.35	32	0.25	33	0.30	.	.
Eagle Lake	03JUL95	30	0.20	36	0.45	.	.	.	.
Eagle Lake	04AUG95	33	0.30	33	0.28	34	0.42	.	

Fish Length Data for Each Location and Date 2

Location	Date	_NAME_	Measurement
Cole Pond	02JUN95	Length1	31
Cole Pond	02JUN95	Length2	32
Cole Pond	02JUN95	Length3	32
Cole Pond	02JUN95	Length4	33
Cole Pond	03JUL95	Length1	33
Cole Pond	03JUL95	Length2	34
Cole Pond	03JUL95	Length3	37
Cole Pond	03JUL95	Length4	32
Cole Pond	04AUG95	Length1	29
Cole Pond	04AUG95	Length2	30
Cole Pond	04AUG95	Length3	34
Cole Pond	04AUG95	Length4	32
Eagle Lake	02JUN95	Length1	32
Eagle Lake	02JUN95	Length2	32
Eagle Lake	02JUN95	Length3	33
Eagle Lake	02JUN95	Length4	.
Eagle Lake	03JUL95	Length1	30
Eagle Lake	03JUL95	Length2	36
Eagle Lake	03JUL95	Length3	.
Eagle Lake	03JUL95	Length4	.
Eagle Lake	04AUG95	Length1	33
Eagle Lake	04AUG95	Length2	33
Eagle Lake	04AUG95	Length3	34
Eagle Lake	04AUG95	Length4	.

これらの結果を生成する SAS プログラムの詳細な説明については、“例 4: BY グループの転置” (2384 ページ)を参照してください。

PROC TRANSPOSE の CAS 処理

入力が SAS Cloud Analytic Services (CAS)から発信され、出力が CAS サーバーに戻される場合、その転置は完全にサーバー上で実行される可能性があります。PROC TRANSPOSE を CAS アクションで実行することは、SAS 内での処理に比べていくつかの利点があります。これらの利点には、ネットワークトラフィックの減少、より迅速な処理の可能性が含まれます。インメモリテーブルは、比較的低速のネットワーク接続を介して転送されるのではなく、サーバー上でローカルに操作されるため、処理が高速になります。より多くの処理リソースが用意されている可能性があるので、CAS が使用されます。

CAS で実行される Transpose アクションについての情報は、*"転置アクションセット" (SAS Viya プラットフォーム: システムプログラミングガイド)*を参照してください。

DATA=入力データセットが CAS 内のインメモリテーブルまたはビューを参照する場合、TRANSPOSE プロシジャは CAS アクションを使用してサーバー内ですべての作業を実行できます。

CAS LIBNAME ステートメントオプションは、CAS プロシジャが CAS 内で実行されるかどうか、およびその実行方法を制御します。デフォルトで、CAS プロシジャは可能な場合は CAS 内で実行されます。CAS 処理を実行しない、多くのデータセットオプションがあります。

CAS は、SAS Viya プラットフォームの分析サーバーと関連するクラウドサービスです。CAS LIBNAME エンジンは、CAS セッション名または CAS セッション UUID を使用して、SAS 9.4 セッションを既存の SAS Cloud Analytic Services セッションに接続できます。ライブラリ参照名は、SAS から特定のセッションと通信するためのハンドルになります。PROC TRANSPOSE と例の詳細については、The Future of Transpose を参照してください。

CAS で実行されるプロシジャの概念的な情報については、4 章, "Base プロシジャの CAS 処理" (75 ページ)を参照してください。

構文: TRANSPOSE プロシジャ

制限事項: DATA=オプションと OUT=オプションが CAS を参照する場合、CAS TRANSPOSE アクションを呼び出して、CAS 内で転置が実行されます。

ヒント: Output Delivery System はサポートされていません。

ATTRIB ステートメント、FORMAT ステートメント、LABEL ステートメント、WHERE ステートメントを使用できます。グローバルステートメントを使用することもできます。リストについては、*"SAS グローバルステートメントのディクショナリ" (SAS グローバルステートメント: リファレンス)*を参照してください。

PROC TRANSPOSE<DATA=*input-data-set*> <DELIMITER=*delimiter*> <LABEL=*label*>

<LET> <NAME=*name*> <OUT=*output-data-set*> <PREFIX=*prefix*> <SUFFIX=*suffix*>;

ATTRIB *variable-list(s) attribute-list(s)*;

BY<DESCENDING>*variable-1*
<<DESCENDING> *variable-2 ...*>
<NOTSORTED>;

COPY *variable(s)*;

FORMAT *variable-1*<...*variable-n*> <*format*> <DEFAULT=*default-format*>;

ID *variable*;

IDLABEL *variable*;

LABEL *variable-1*=*label-1*<...*variable-n*=*label-n*>;

VAR *variable(s)*;

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

ステートメント	タスク	例
“PROC TRANSPOSE ステートメント”	SAS データセットの値を再構築し、選択した変数をオブザベーションに転置して、出力データセットを作成します。	Ex. 1, Ex. 2, Ex. 3, Ex. 5
“BY ステートメント”	各 BY グループを転置します。	Ex. 4
“COPY ステートメント”	変数を転置せずに直接コピーします。	Ex. 6
“ID ステートメント”	転置された変数の命名元の値を含む変数を指定します。	Ex. 2
“IDLABEL ステートメント”	転置された変数のラベルを作成します。	Ex. 3
“VAR ステートメント”	転置する変数をリスト表示します。	Ex. 4, Ex. 6

PROC TRANSPOSE ステートメント

SAS データセットの値を再構築し、選択した変数をオブザベーションに転置して、出力データセットを作成します。

ヒント: DATA=および OUT=オプションではデータセットオプションを使用できます。グローバルステートメントを使用することもできます。リストについては、“SAS グローバルステートメントのディクショナリ” (*SAS グローバルステートメント: リファレンス*)を参照してください。

例: “例 1: 単純な転置を実行” (2379 ページ)
“例 2: 転置された変数の名前の指定” (2380 ページ)
“例 3: 転置された変数のラベル作成” (2382 ページ)
“例 4: BY グループの転置” (2384 ページ)

“例 5: ID 変数の値が重複する場合の転置された変数名の指定” (2387 ページ)
“例 6: 統計分析用データの転置” (2389 ページ)

構文

PROC TRANSPOSE <DATA=*input-data-set*> <DELIMITER=*delimiter*> <LABEL=*label*> <LET> <NAME=*name*> <OUT=*output-data-set*> <PREFIX=*prefix*> <SUFFIX=*suffix*>;

オプション引数

DATA= *input-data-set*
転置する SAS データセットの名前を指定します。

デフォルト　直前に作成された SAS データセット

DELIMITER= *delimiter*
出力データセットの転置された変数の名前の作成に使用するデリミタを指定します。指定した場合、ID ステートメントで変数を複数指定すると、変数値の間にデリミタが挿入されます。

別名　DELIM=

ヒント　DELIMITER の値には名前リテラル(n-literals)を使用できます。名前リテラルは、特に VALIDVARNAME=ANY の場合、印刷用文字や外国語の文字を指定するときに役立ちます。

参照項目: “ID ステートメント” (2370 ページ)

LABEL= *label*
現在のオブザベーションを作成するために転置されている変数のラベルを含む、出力データセットの変数の名前を指定します。

デフォルト　_LABEL_

ヒント　LABEL の値には名前リテラル(n-literals)を使用できます。名前リテラルは、特に VALIDVARNAME=ANY の場合、印刷用文字や外国語の文字を指定するときに役立ちます。

LET
ID 変数の重複する値を許可します。PROC TRANSPOSE では、データセットまたは BY グループ内で最後に発生した特定の ID 値を含むオブザベーションが転置されます。

参照項目: “例 5: ID 変数の値が重複する場合の転置された変数名の指定” (2387 ページ)

NAME= *name*
現在のオブザベーションを作成するために転置されている変数の名前を含む、出力データセットの変数の名前を指定します。

デフォルト _NAME_

参照項目: “例 2: 転置された変数の名前の指定” (2380 ページ)

OUT= *output-data-set*
出力データセットを指定します。*output-data-set* が存在しない場合は、PROC TRANSPOSE で、DATA*n* 命名規則を使用して作成されます。

デフォルト DATA*n*

参照項目: “例 1: 単純な転置を実行” (2379 ページ)

PREFIX= *prefix*
出力データセットの転置された変数の名前の作成に使用する接頭辞を指定します。たとえば、PREFIX=VAR の場合、変数の名前は VAR1、VAR2、 ...、VAR*n* となります。

操作 PREFIX=を ID ステートメントとともに使用すると、変数名は接頭辞の値で始まり、その後に ID 値が続きます。

ヒント PREFIX の値には名前リテラル(n-literals)を使用できます。名前リテラルは、特に VALIDVARNAME=ANY の場合、印刷用文字や外国語の文字を指定するときに役立ちます。

参照項目: “例 2: 転置された変数の名前の指定” (2380 ページ)

SUFFIX= *suffix*
出力データセットの転置された変数の名前の作成に使用する接尾辞を指定します。

操作 SUFFIX=を ID ステートメントとともに使用すると、値が ID 値に追加されます。

ヒント SUFFIX の値には名前リテラル(n-literals)を使用できます。名前リテラルは、特に VALIDVARNAME=ANY の場合、印刷用文字や外国語の文字を指定するときに役立ちます。

ATTRIB ステートメント

1 つまたは複数の変数に出力形式、入力形式、ラベル、長さを設定します。

構文

ATTRIB *variable-list(s) attribute-list(s)*;

引数

variable-list(s)
属性を設定する変数を指定します。

> ヒント SAS で使用可能な任意の形式で変数をリストします。

attribute-list(s)
variable-list に割り当てる 1 つまたは複数の属性を指定します。ATTRIB ステートメントでは、次の属性を 1 つ以上指定します。

FORMAT=*format*
variable-list の変数に出力形式を割り当てます。

ヒント この出力形式は、標準の SAS 出力形式でも FORMAT プロシジャで定義した出力形式でもかまいません。

INFORMAT=*informat*
variable-list の変数に入力形式を割り当てます。

ヒント この入力形式は、標準の SAS 入力形式でも FORMAT プロシジャで定義した入力形式でもかまいません。

LABEL='*label*'
variable-list の変数にラベルを割り当てます。

関連項目

"ATTRIB ステートメント" (*SAS DATA ステップステートメント: リファレンス*).

BY ステートメント

出力を BY グループ順に並べ替えます。

制限事項:
別のユーザーが同時にデータセットを更新している場合、PROC TRANSPOSE を BY ステートメントや ID ステートメントとともに使用しないでください。
BY ステートメントではグループ変数を指定しないでください。BY 変数をグループ変数と同じにすることはできません。

構文

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

必須引数

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

オプション引数

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

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

BY 変数の値によるオブザベーションの順序またはインデックスの要件は、NOTSORTED オプションの使用時は BY グループ処理に向けて保留されます。実際、NOTSORTED を指定する場合、プロシジャはインデックスを 使用しません。プロシジャは、すべての BY 変数に対して同じ値を持つ一連の連続したオブザベーションとして BY グループを定義します。BY 変数の値が同じオブザベーションが連続していない場合、プロシジャは連続セットをそれぞれ個別の BY グループとして処理します。

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

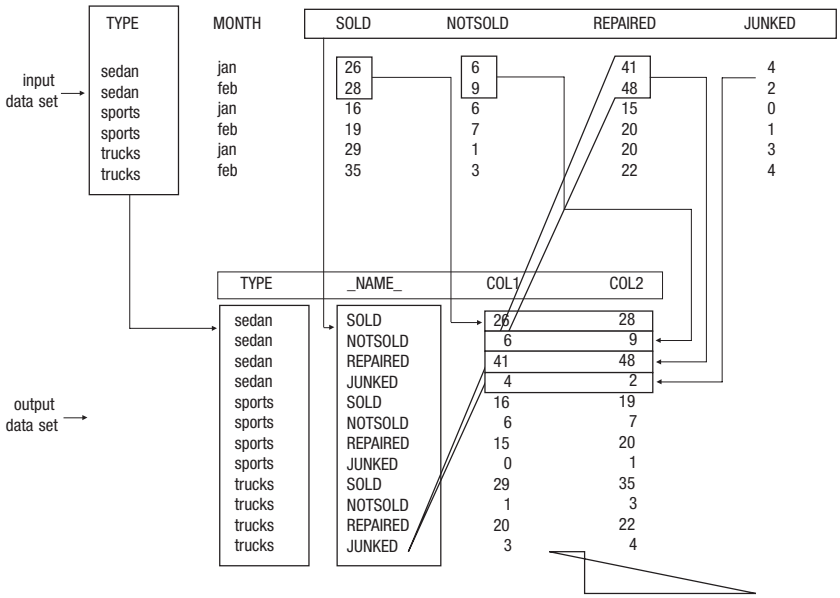
注: GROUPFORMAT オプション(DATA ステップの BY ステートメントで使用可能)は、PROC ステップの BY ステートメントでは使用できません。

詳細

BY グループ処理

次の図は、BY グループを使用してデータセットを転置した結果を示しています。TYPE が BY 変数で、SOLD、NOTSOLD、REPAIRED、JUNKED が転置する変数です。

図 62.1 BY グループによる転置



- 出力データセットのオブザベーション数(12)は、BY グループ数(3)に転置する変数の数(4)を乗算した結果です。
- BY 変数は転置されません。
- _NAME_には、出力データセットの現在のオブザベーションを作成するために転置された入力データセットの変数の名前が含まれます。_NAME_変数に別の名前を指定するには、NAME=オプションを使用します。
- 入力データセットのどの BY グループでもオブザベーションの最大数は 2 です。したがって、出力データセットには、COL1 と COL2 という 2 つの変数が含まれます。COL1 と COL2 には、SOLD、NOTSOLD、REPAIRED、JUNKED の値が含まれます。

注: 入力データセットに他の BY グループよりもオブザベーションが多い BY グループがある場合、PROC TRANSPOSE は、対応する入力オブザベーションのない変数に、出力データセットで欠損値を割り当てます。

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

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

各 PROC ステップでは 1 つの BY ステートメントしか使用できません。BY ステートメントを使用すると、プロシジャでは、BY 順に並べ替えたか、適切なインデックスが付いた入力データセットが求められます。入力データセットがこれらの基準を

満たしていなければ、エラーが発生します。SORT プロシジャで並べ替えるか、BY 変数の適切なインデックスを作成します。

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

- BY ステートメントの詳細については、*SAS DATA ステップステートメント: リファレンス*を参照してください。
- PROC SORT の詳細については、55 章, "SORT プロシジャ," (2027 ページ)を参照してください。
- CAS での BY グループ処理の詳細については、"グループ化と順序付け" (*SAS Cloud Analytic Services: DATA ステッププログラミング*)を参照してください。
- インデックスの作成の詳細については、"INDEX CREATE ステートメント" (435 ページ)を参照してください。

BY 変数値のフォーマティング

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

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

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

2 つのデータセットで長さが異なる BY 変数

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

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

- プロシジャで、BY ステートメントが常にセカンダリデータセットに適用される場合があります。この場合、BY ステートメントの変数が 1 つ以上、セカンダリデータセットに出現している必要があります。

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

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

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

- A
- A、B
- A、B、C
- A、B、C、D

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

対応する文字変数の長さが異なる場合、長い方の文字変数の余分な文字は末尾のブランクのみという可能性もあります。文字変数の長さが異なる場合は、末尾のブランクを削除すると同一になるのであれば、その値は一致します。たとえば、プライマリデータセットの'ABCD'とセカンダリデータセットの'ABCD 'は一致します。セカンダリデータセットに'ABCDEF'が含まれている場合は、一致しません。

BY ステートメントをサポートする Base SAS プロシジャ

COMPARE	SORT (必須)

CORR	STANDARD
FREQ	SUMMARY
MEANS	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE
REPORT (非ウィンドウ環境のみ)	ODSTEXT
ODSLIST	ODSTABLE

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

関連項目

"BY ステートメント" (*SAS DATA ステップステートメント: リファレンス*)

COPY ステートメント

変数を転置せずに、入力データセットから出力データセットに直接コピーします。

例: "例 6: 統計分析用データの転置" (2389 ページ)

構文

COPY *variable(s)*;

必須引数

variable(s)
COPY ステートメントが転置せずに入力データセットから出力データセットに直接コピーする変数の名前を 1 つ以上指定します。

詳細

COPY ステートメントでは、変数が直接出力データセットにコピーされるので、出力データセットのオブザベーション数と入力データセットのオブザベーション数が等しくなります。

このプロシジャでは、入力データセットのオブザベーションの数と、転置される変数の数が等しくない場合、出力データセットに欠損値が埋め込まれます。

FORMAT ステートメント

変数に出力形式を関連付けます。

構文

FORMAT *variable-1* <...*variable-n*> *format variable-1* <...*variable-n*> *format*;

引数

variable
1 つまたは複数の変数に出力形式を関連付けます。少なくとも 1 つの *variable* を指定する必要があります。

ヒント 変数から出力形式の関連付けを取り消すには、DATA ステップまたは PROC DATASETS で出力形式を指定せず、FORMAT ステートメントでこの変数を使用します。DATA ステップでは、この FORMAT ステートメントを SET ステートメントの後に配置します。"出力形式の取り消し" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。PROC DATASETS を使うこともできます。

format
変数の値を出力するときに適用する出力形式を指定します。

ヒント FORMAT ステートメントを使用して変数に関連付けられた出力形式は、コロン修飾子で使用される出力形式と同じように、後続の PUT ステートメントで動作します。コロン修飾子の使用方法の詳細については、"PUT ステートメント: リスト" (*SAS DATA ステップステートメント: リファレンス*)を参照してください。

参照項目: *SAS 出力形式と入力形式: リファレンス*

ID ステートメント

出力データセットの転置された変数の名前になるフォーマットされた非欠損値を含む、入力データセットの変数を 1 つ以上指定します。転置後の(出力)データセットで変数名が生成されるときに、リストされたすべての ID 変数のフォーマットされた値が、ID ステートメントの変数リストと同じ順序で連結されます。PREFIX=、DELIMITER=および SUFFIX=オプションを使用すると、生成された変数名を変更できます。PREFIX=オプションでは、生成された変数名の先頭にくる共通の文字または文字列を指定します。DELIMITER=オプションでは、ID 変数の値間に挿入される共通の文字または文字列を指定します。SUFFIX=オプションでは、生成された各変数名の末尾に追加される共通の文字または文字列を指定します。

制限事項: 別のユーザーが同時にデータセットを更新している場合、同時アクセス権をサポートするエンジンで PROC TRANSPOSE を ID ステートメントや BY ステートメントとともに使用することはできません。

ヒント: いずれかの ID 変数の値が欠損している場合、PROC TRANSPOSE により、警告メッセージがログに書き込まれます。このプロシジャでは、いずれかの ID 変数が欠損値のオブザベーションは転置されません。

例: “例 2: 転置された変数の名前の指定” (2380 ページ)

構文

ID *variable(s)*;

必須引数

variable(s)
出力データセットの変数の名前の生成に使用するフォーマットされた値を含む変数の名前を 1 つ以上指定します。

詳細

重複する出力データセット変数名

PREFIX、DELIMITER および SUFFIX オプション値を組み合わせて、入力データセットの ID 変数値から生成される変数名は出力データセット内で重複しないようにします。出力データセット変数名が 2 回以上出現する場合は、入力データセットの 2 つ以上のオブザベーションが出力データセットの 1 つの変数に転置され、その結果、データ損失が生じたことを示します。ID 変数が 1 つしかない場合に、入力データセット内または(BY ステートメントを使用する場合)BY グループ内に重複するフォーマットされた値があると、このような状況が生じます。同様に、ID 変数が複数ある場合、ID 変数のフォーマットされた値の組み合わせが、入力データセット内または BY グループ内で 2 回以上発生すると、このような状況が発生します。PROC TRANSPOSE では、重複する出力データセット変数名が生成された場合、データ損

失を防ぐために、重複する ID 値についてエラーメッセージが出力され、処理が停止されます。ただし、PROC TRANSPOSE ステートメントで LET オプションを指定した場合は、プロシジャで警告メッセージが出力され、処理が続行されて、重複するフォーマット変数値のうち最後に発生した値を含むオブザベーションが転置されます。

注: ID 変数から生成された変数名が VALIDVARNAME オプション設定に基づいた有効な SAS 変数名であるために必要な文字の置き換えや切り捨てによって、ID 変数または ID 変数の組み合わせのフォーマットされた値が入力データセット内で重複しなくとも、出力データセットの変数名が重複する可能性があります。

数値からの変数名の作成

数値変数を ID 変数として使用すると、PROC TRANSPOSE により、フォーマットされた ID 値が有効な SAS 名に変更されます。

VALIDVARNAME=ANY を設定しない限り、SAS 変数名の先頭は数字にすることはできません。フォーマットされた値の最初の文字が数値の場合、プロシジャによりその値にアンダースコアの接頭辞が付けられます。このアクションによって、32 文字値の最後の文字が切り捨てられます。その他の無効な文字はアンダースコアに置き換えられます。32 文字よりも長い ID 値を使用して転置される変数に名前を付けると、その ID 値はすべて 32 文字まで切り捨てられます。

フォーマットされた値が数値定数のように見える場合、文字+、–、.が、それぞれ P、N、D に変更されます。フォーマットされた値に数値ではない文字が含まれている場合、文字+、–、.がアンダースコアに変更されます。

注: VALIDVARNAME システムオプションの値が V6 の場合、転置された変数名が 8 文字まで切り捨てられます。

複数の ID 変数からの変数名の作成

ID 変数を 1 つだけ指定した場合は、出力データセット変数名を生成するときに、変数のフォーマットされた値は VALIDVARNAME オプションによって課された SAS 変数命名規則に従って作成されます。また、ID 変数値を PREFIX および SUFFIX オプションと組み合わせて生成する名前も、SAS 変数命名規則に従って作成されます。フォーマットされた ID 変数値でも、その変数値と PREFIX および SUFFIX オプションとの組み合わせでも、無効な文字はアンダースコアに置き換えられます。または名前が数値定数であるように見える場合は、アンダースコアが接頭辞として使用され、文字+、-、.が P、N、D に変更されます。その結果生成された名前は、VALIDVARNAME オプション設定で許可される名前の最大長に合わせて切り捨てられます。複数の ID 変数を指定する場合、各 ID 変数のフォーマットされた値を使用する変数名のコンポーネントも、ID 変数値と PREFIX、DELIMITER および SUFFIX オプションからなる名前も、SAS 変数の命名規則に従う必要があります。その結果生成された名前は、VALIDVARNAME オプション設定に適した長さに切り捨てられます。

IDLABEL ステートメント

転置された変数のラベルを作成します。

制限事項: ID ステートメントの後に記述する必要があります。

例: “例 3: 転置された変数のラベル作成” (2382 ページ)

構文

IDLABEL *variable*;

必須引数

variable
ID ステートメントで名前を指定された変数にラベルを付けるためにプロシジャで使用される値を含む変数の名前を指定します。 *variable* には文字または数値を指定できます。

注: IDLABEL ステートメントの結果を確認するには、PRINT プロシジャで LABEL オプションを使用して出力データセットを出力します。また、DATASETS プロシジャで CONTENTS ステートメントを使用して、出力データセットのコンテンツを印刷することもできます。

LABEL ステートメント

説明を示すラベルを変数に割り当てます。

構文

LABEL *variable-1* = '*text-string*' <...*variable-n* = '*text-string*'>;

LABEL *variable-1* = *' '* < ...*variable-n*=’ ‘>; | *variable-1* = <...*variable-n*= >;

引数

variable
ラベルを付ける変数を指定します。複数の変数のラベルは、1 つの LABEL ステートメントで指定できます。

```
data test;
```

```
  L = 5;
  W = 10;
  label L = 'Length' W = 'Width';
run;
```

text-string

最大 256 バイトのラベルを指定します。

```
data test;
  L = 5;
  label L = 'Length';
run;
```

制限事項

ラベルにセミコロン(;)や等号(=)が含まれている場合は、ラベルのテキストを単一引用符または二重引用符で囲む必要があります。

```
label N = 'Number = ';
```

ラベルに単一引用符(')が含まれている場合は、ラベルのテキストを二重引用符で囲む必要があります。

```
label Name = "Owner's Last Name";
```

JAVAIMG のデバイスでは、変数名の置き換えを一部サポートしています。変数に指定したラベルのテキストが許可されたスペースを超える場合、軸や凡例のタイトルをラベリングするプロシジャのかわりに、変数名が使用されます。SAS/GRAPH GPLOT プロシジャを除き、JAVAIMG デバイスが指定されていない場合、変数名は使用されません。

注

ラベルにセミコロン、引用符、等号が含まれていない限り、引用符は必要ありません。

LABEL ステートメントは、最初の変数の後に等号(=)があるものと想定します。それ以外の文字が見つかった場合、エラーが発生します。LABEL ステートメントは、最初の変数の直後に続く等号の後から、等号が直後に続く別の変数に達するまでの区間に存在するすべての文字を、引用符の有無にかかわらず、ラベルの一部と見なします。次の例では、変数 x のラベルは、**this is x y 4 =this is y** になります。なぜなら、等号が直後に続く変数は z であるためです。

```
data test;
  x = 1;
  y = 1;
  z = 1;
  label x = 'this is x' y 4 = 'this is y' z = 'This is z';
run;
```

次のような LABEL ステートメントでも、変数 x のラベルは前述の例と同じになります。

```
label x = this is x y 4 = this is y  z = This is z;
```

参照項目:

“文字定数と文字変数” (*SAS プログラマガイド: エッセンシャル*).

' '

変数からラベルを削除します。ブランク 1 つを引用符で囲むと、指定済みのラベルが取り消されます。1 つの LABEL ステートメントで複数の変数からラベルを削除できます。

```
data test;
  L=5; W=10;
  label L = ' '  W = ' ';
run;
```

また、等号の後に何も指定しないでラベルを削除することもできます。

```
data test;
  L=5; W=10;
  label L =  W = ;
run;
```

VAR ステートメント

転置する変数をリストします。

注: 特殊な SAS 変数リスト名_CHARACTER_を VAR ステートメントに追加して、すべての文字変数を転置に含めます。特殊な SAS 名リストを参照してください。

例: “例 4: BY グループの転置” (2384 ページ)
“例 6: 統計分析用データの転置” (2389 ページ)

構文

VAR *variable(s)*;

必須引数

variable(s)
転置する変数の名前を 1 つ以上指定します。

詳細

- VAR ステートメントを省略すると、TRANSPOSE プロシジャでは、別のステートメントにリストされていない入力データセットのすべての数値変数が転置されます。
- 文字変数を転置する場合は、VAR ステートメントでそれらの文字変数をリストする必要があります。

注: プロシジャで文字変数を転置すると、転置された変数はすべて文字変数になります。

WHERE ステートメント

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

構文

WHERE *where-expression-1*
<*logical-operator where-expression-n*>;

引数

where-expression
オペランドや演算子で構成される算術式または論理式を指定します。

ヒント 次のいくつかのセクションで説明されるオペランドや演算子は WHERE= データセットオプションでも使用できます。

where-expression を複数指定することができます。

logical-operator
AND、AND NOT、OR、OR NOT を指定できます。

例

WHERE ステートメントをサポートするプロシジャ

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

COMPARE	REPORT
CORR	SORT
DATASETS (APPEND ステートメント)	SQL
FREQ	STANDARD

MEANS または SUMMARY	TABULATE
PRINT	TRANSPOSE
RANK	UNIVARIATE

詳細

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

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

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

出力形式と変数の関連付けを一時的に解除する

この例では、PROC PRINT で、WHERE 式の条件に合うオブザベーションのみが印刷されます。

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

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

出力

Team Members with a GPA Greater than 3.5

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

結果: TRANSPOSE プロシジャ

出力データセット

TRANSPOSE プロシジャでは、PROC TRANSPOSE ステートメントで OUT=オプションを指定するかどうかに関係なく、常に出力データセットが生成されます。PROC TRANSPOSE では、出力データセットは印刷されません。PROC PRINT、PROC REPORT または別の SAS レポートツールを使用して、出力データセットを印刷します。

出力データセット変数

出力データセットには次の変数が含まれます。

- 各変数の値をオブザベーションに転置した結果として生じる変数。
- PROC TRANSPOSE で、出力データセットの各オブザベーションの値のソースを識別するために作成される変数。この変数は文字変数で、その値は入力データセットから転置された変数の名前です。PROC TRANSPOSE では、この変数の名前はデフォルトで_NAME_になります。デフォルト名を無効にするには、NAME=オ

プションを使用します。_NAME_変数のラベルは **NAME OF FORMER VARIABLE** です。

- BY または COPY ステートメントの使用時に、PROC TRANSPOSE で入力データセットからコピーされる変数。これらの変数の名前と値は、入力データセットと同じです。これらの変数は、属性も同じです(例:種類、長さ、ラベル、入力形式、出力形式)。
- (プロシジャで転置される変数のいずれかにラベルが含まれる場合に)転置される変数の変数ラベルを値として含む文字変数。LABEL=オプションを使用して変数名を指定します。デフォルトは_LABEL_です。

注: LABEL=オプションまたは NAME=オプションの値が、BY または COPY ステートメントに記述された変数と同じ場合、出力データセットには、転置された変数の名前またはラベルが値となる変数は含まれません。

転置された変数の属性

- 転置された変数はすべて、種類と長さが同じです。
- プロシジャで転置されている変数がすべて数値の場合、転置された変数は数値になります。したがって、数値変数にフォーマットされた値として文字列が含まれていると、未フォーマットの数値が転置されます。
- プロシジャで転置されている変数が文字の場合、転置された変数はすべて文字になります。フォーマットされた値として文字列が含まれている数値変数を転置すると、フォーマットされた値が転置されます。
- 転置された変数の長さは、転置する最長の変数の長さと等しくなります。

転置された変数の名前

PROC TRANSPOSE では、転置された変数の命名に次のルールが使用されます。

1. ID ステートメントで入力データセットの変数を 1 つまたは複数指定すると、そのフォーマットされた値が転置された変数の名前になります。複数の ID 変数を指定した場合、転置された変数の名前は、ID 変数の値の連結になります。DELMITER=オプションを指定した場合は、転置された変数の名前を生成するときに、ID 変数のフォーマットされた値の間にその値が挿入されます。
2. PREFIX=オプションで、転置された変数の名前の作成に使用する接頭辞を指定します。また、SUFFIX=オプションで、転置された変数の名前に追加する接尾辞も指定します。
3. ID ステートメント、PREFIX=オプションまたは SUFFIX=オプションを使用しない場合、PROC TRANSPOSE では、_NAME_という名前の入力変数が検索され、転置された変数の名前が取得されます。

4 ID ステートメントも PREFIX=オプションも使用せず、入力データセットに _NAME_という変数が含まれていない場合、PROC TRANSPOSE では、転置された変数に名前 COL1、COL2、 ...、COL*n* が割り当てられます。

例: TRANSPOSE プロシジャ

例 1: 単純な転置を実行

要素: PROC TRANSPOSE ステートメントオプション
OUT=

この例では、デフォルトの転置を実行し、従属ステートメントは使用しません。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=40;

data score;
  input Student $9. +1 StudentID $ Section $ Test1 Test2 Final;
  datalines;
Capalleti 0545 1  94 91 87
Dubose    1252 2  51 65 91
Engles    1167 1  95 97 97
Grant     1230 2  63 75 80
Krupski   2527 2  80 76 71
Lundsford 4860 1  92 40 86
McBane    0674 1  75 78 72
;

proc transpose data=score out=score_transposed;
run;

proc print data=score_transposed noobs;
   title 'Student Test Scores in Variables';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

SCORE データセットを作成します。 データセット SCORE には、学生の名前、ID 番号、および 2 つのテストと期末試験の成績が含まれます。

```
data score;
  input Student $9. +1 StudentID $ Section $ Test1 Test2 Final;
  datalines;
Capalleti 0545 1  94 91 87
Dubose    1252 2  51 65 91
Engles    1167 1  95 97 97
Grant     1230 2  63 75 80
Krupski   2527 2  80 76 71
Lundsford 4860 1  92 40 86
McBane    0674 1  75 78 72
;
```

データセットを転置します。 PROC TRANSPOSE では、数値変数 Test1、Test2 および Final のみが転置されます。これは、VAR ステートメントが記述されておらず、別のステートメントにも数値変数が記述されていないためです。OUT=では、転置結果がデータセット SCORE_TRANSPOSED に入力されます。

```
proc transpose data=score out=score_transposed;
run;
```

SCORE_TRANSPOSED データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=score_transposed noobs;
   title 'Student Test Scores in Variables';
run;
```

出力

出力データセット SCORE_TRANSPOSED の変数 COL1～COL7 に、学生の個々のスコアが含まれます。各オブザベーションに、1 つのテストのスコアがすべて含まれます。変数_NAME_には、転置された入力データセットの変数名が含まれます。

アウトプット 62.3　変数に含まれる学生のテストのスコア

Student Test Scores in Variables

NAME	COL1	COL2	COL3	COL4	COL5	COL6	COL7
Test1	94	51	95	63	80	92	75
Test2	91	65	97	75	76	40	78
Final	87	91	97	80	71	86	72

例 2: 転置された変数の名前の指定

要素:　PROC TRANSPOSE ステートメントオプション

NAME=
PREFIX=
ID ステートメント

データセット: SCORE

この例では、変数値とユーザー指定の値を使用して、転置された変数に名前を付けます。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=40;

proc transpose data=score out=idnumber name=Test
    prefix=sn;
   id studentid;
run;

proc print data=idnumber noobs;
  title 'Student Test Scores';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

データセットを転置します。 PROC TRANSPOSE では、VAR ステートメントが記述されていないので、数値変数 Test1、Test2 および Final のみが転置されます。OUT=では、転置結果が IDNUMBER データセットに入力されます。NAME=では、プロシジャが転置する入力データセットの変数名を含む変数の名前として Test が指定されます。プロシジャでは、PREFIX=の値 sn と ID 変数 StudentID の値を使用して、転置された変数に名前が付けられます。

```
proc transpose data=score out=idnumber name=Test
    prefix=sn;
   id studentid;
run;
```

IDNUMBER データセットを出力します。 NOOBS オプションはオブザベーション番号の印刷を行いません。

```
proc print data=idnumber noobs;
  title 'Student Test Scores';
run;
```

出力

次のデータセットは、出力データセット IDNUMBER です。

アウトプット 62.4　学生のテストのスコア

Student Test Scores

Test	sn0545	sn1252	sn1167	sn1230	sn2527	sn4860	sn0674
Test1	94	51	95	63	80	92	75
Test2	91	65	97	75	76	40	78
Final	87	91	97	80	71	86	72

例 3: 転置された変数のラベル作成

要素: PROC TRANSPOSE ステートメントオプション
PREFIX=
IDLABEL ステートメント

データセット: SCORE

この例では、IDLABEL ステートメントの変数値を使用して、転置された変数にラベルを付けます。

プログラム 1

```
options nodate pageno=1 linesize=80 pagesize=40;

proc transpose data=score out=idlabel name=Test
    prefix=sn;
  id studentid;

    idlabel student;
run;

 proc print data=idlabel label noobs;
    title 'Student Test Scores';
 run;

proc contents data=idlabel;
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

データセットを転置します。 PROC TRANSPOSE では、VAR ステートメントが記述されていないので、数値変数 Test1、Test2 および Final のみが転置されます。OUT=では、転置結果が IDLABEL データセットに入力されます。NAME=では、プロシジャが転置する入力データセットの変数名を含む変数の名前として Test が指定されます。プロシジャでは、PREFIX=の値 sn と ID 変数 StudentID の値を使用して、転置された変数に名前が付けられます。

```
proc transpose data=score out=idlabel name=Test
   prefix=sn;
  id studentid;
```

出力変数にラベルを割り当てます。 PROC TRANSPOSE では、変数 Student の値を使用して、転置された変数にラベルが付けられます。プロシジャでは、_NAME_変数のラベルとして NAME OF FORMER VARIABLE が提供されます。

```
    idlabel student;
run;
```

IDLABEL データセットを出力します。 LABEL オプションを使用すると、PROC PRINT で、列ヘッダーの変数ラベルが印刷されます。NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=idlabel label noobs;
  title 'Student Test Scores';
run;
```

IDLABEL 変数の名前とラベルを表示します。 PROC CONTENTS では、変数の名前とラベルが表示されます。

```
proc contents data=idlabel;
run;
```

出力 1

このデータセットは、出力データセット IDLABEL です。

アウトプット 62.5　学生のテストのスコア(IDLABEL)

Student Test Scores

NAME OF FORMER VARIABLE	Capalleti	Dubose	Engles	Grant	Krupski	Lundsford	McBane
Test1	94	51	95	63	80	92	75
Test2	91	65	97	75	76	40	78
Final	87	91	97	80	71	86	72

プログラム 2

変数とラベルの名前を表示します。 PROC CONTENTS では、最初のプログラムで使用した変数名とラベルが表示されます。

```
proc contents data=idlabel;
run;
```

出力 2

次の出力では、PROC CONTENTS によって変数とラベルが表示されます。

アウトプット 62.6　CONTENTS プロシジャ

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Label
1	Test	Char	8	NAME OF FORMER VARIABLE
2	sn0545	Num	8	Capalleti
8	sn0674	Num	8	McBane
4	sn1167	Num	8	Engles
5	sn1230	Num	8	Grant
3	sn1252	Num	8	Dubose
6	sn2527	Num	8	Krupski
7	sn4860	Num	8	Lundsford

例 4: BY グループの転置

要素:

BY ステートメント
VAR ステートメント
データセットオプション
　RENAME=

この例では、BY グループの転置と転置する変数の選択方法を示します。

プログラム

```
options nodate pageno=1 linesize=80 pagesize=40;
data fishdata;
  infile datalines missover;
  input Location & $10. Date date7.
      Length1 Weight1 Length2 Weight2 Length3 Weight3
```

```
      Length4 Weight4;
  format date date7.;
  datalines;
Cole Pond   2JUN95 31 .25 32 .3  32 .25 33 .3
Cole Pond   3JUL95 33 .32 34 .41 37 .48 32 .28
Cole Pond   4AUG95 29 .23 30 .25 34 .47 32 .3
Eagle Lake  2JUN95 32 .35 32 .25 33 .30
Eagle Lake  3JUL95 30 .20 36 .45
Eagle Lake  4AUG95 33 .30 33 .28 34 .42
;

proc transpose data=fishdata
    out=fishlength(rename=(col1=Measurement));

  var length1-length4;

  by location date;
run;

proc print data=fishlength noobs;
  title 'Fish Length Data for Each Location and Date';
run;
```

プログラムの説明

FISHDATA データセットを作成し、SAS システムオプションを設定します。
FISHDATA のデータは、3 日間に 2 つの池で釣れた魚の体長と重量の測定値を日ごとに表したものです。生データが Location および Date 順に並べ替えられます。NODATE オプションは、出力の日付と時間の表示を非表示にします。PAGENO=では開始ページ番号を指定します。LINESIZE=で出力行長を指定し、PAGESIZE=で出力ページの行数を指定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
data fishdata;
  infile datalines missover;
  input Location & $10. Date date7.
      Length1 Weight1 Length2 Weight2 Length3 Weight3
      Length4 Weight4;
  format date date7.;
  datalines;
Cole Pond   2JUN95 31 .25 32 .3  32 .25 33 .3
Cole Pond   3JUL95 33 .32 34 .41 37 .48 32 .28
Cole Pond   4AUG95 29 .23 30 .25 34 .47 32 .3
Eagle Lake  2JUN95 32 .35 32 .25 33 .30
Eagle Lake  3JUL95 30 .20 36 .45
Eagle Lake  4AUG95 33 .30 33 .28 34 .42
;
```

データセットを転置します。 OUT=では、転置結果が FISHLENGTH データセットに入力されます。RENAME=では、出力データセットの名前 COL1 が Measurement に変更されます。

```
proc transpose data=fishdata
    out=fishlength(rename=(col1=Measurement));
```

転置する変数を指定します。 VAR ステートメントでは、PROC TRANSPOSE で転置される変数が制限されます。

```
  var length1-length4;
```

出力データセットを BY グループに編成します。 BY ステートメントでは、Location と Date の値の一意な組み合わせごとに BY グループが作成されます。プロシジャでは、BY 変数は転置されません。

```
  by location date;
run;
```

FISHLENGTH データセットを出力します。 NOOBS オプションはオブザベーション番号の印刷を行いません。

```
proc print data=fishlength noobs;
  title 'Fish Length Data for Each Location and Date';
run;
```

出力

次のデータセットは、出力データセット FISHLENGTH です。PROC TRANSPOSE では、元のデータセットの各 BY グループに対して、転置する変数ごとに 1 つずつ、4 つのオブザベーションが作成されます。その BY グループに対して転置する変数の値が入力データセットにない場合、変数 Measurement(COL1 から名前を変更)に欠損値が表示されます。複数のオブザベーションで、Measurement に欠損値があります。たとえば、最後のオブザベーションでは、入力データに Eagle Lake での 04AUG95 の Length4 に対する値が含まれていないため、欠損値が表示されます。

アウトプット 62.7　魚の体長のデータ

Fish Length Data for Each Location and Date

Location	Date	_NAME_	Measurement
Cole Pond	02JUN95	Length1	31
Cole Pond	02JUN95	Length2	32
Cole Pond	02JUN95	Length3	32
Cole Pond	02JUN95	Length4	33
Cole Pond	03JUL95	Length1	33
Cole Pond	03JUL95	Length2	34
Cole Pond	03JUL95	Length3	37
Cole Pond	03JUL95	Length4	32
Cole Pond	04AUG95	Length1	29
Cole Pond	04AUG95	Length2	30
Cole Pond	04AUG95	Length3	34
Cole Pond	04AUG95	Length4	32
Eagle Lake	02JUN95	Length1	32
Eagle Lake	02JUN95	Length2	32
Eagle Lake	02JUN95	Length3	33
Eagle Lake	02JUN95	Length4	.
Eagle Lake	03JUL95	Length1	30
Eagle Lake	03JUL95	Length2	36
Eagle Lake	03JUL95	Length3	.
Eagle Lake	03JUL95	Length4	.
Eagle Lake	04AUG95	Length1	33
Eagle Lake	04AUG95	Length2	33
Eagle Lake	04AUG95	Length3	34
Eagle Lake	04AUG95	Length4	.

例 5: ID 変数の値が重複する場合の転置された変数名の指定

要素:　PROC TRANSPOSE ステートメントオプション
　　LET

この例では、変数(ID)の値を使用して、ID 変数の値が重複していても、転置された変数の名前を指定する方法を示します。

プログラム

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

data stocks;
   input Company $14. Date $ Time $ Price;
   datalines;
Horizon Kites jun11 opening 29
Horizon Kites jun11 noon    27
Horizon Kites jun11 closing 27
Horizon Kites jun12 opening 27
Horizon Kites jun12 noon    28
Horizon Kites jun12 closing 30
SkyHi Kites   jun11 opening 43
SkyHi Kites   jun11 noon    43
SkyHi Kites   jun11 closing 44
SkyHi Kites   jun12 opening 44
SkyHi Kites   jun12 noon    45
SkyHi Kites   jun12 closing 45
;

proc transpose data=stocks out=close let;

  by company;

  id date;
 run;

proc print data=close noobs;
  title 'Closing Prices for Horizon Kites and SkyHi Kites';
run;
```

プログラムの説明

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

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

STOCKS データセットを作成します。 STOCKS には、競合する 2 つの凧製造会社の株価が含まれます。価格は、2 日間にわたり 1 日 3 回(開始時、正午、終了時)記録されます。入力データセットでは、Date 変数に重複する値が含まれることに注意してください。

```
data stocks;
   input Company $14. Date $ Time $ Price;
   datalines;
Horizon Kites jun11 opening 29
Horizon Kites jun11 noon    27
Horizon Kites jun11 closing 27
Horizon Kites jun12 opening 27
Horizon Kites jun12 noon    28
Horizon Kites jun12 closing 30
SkyHi Kites   jun11 opening 43
SkyHi Kites   jun11 noon    43
```

```
SkyHi Kites   jun11 closing 44
SkyHi Kites   jun12 opening 44
SkyHi Kites   jun12 noon    45
SkyHi Kites   jun12 closing 45
;
```

データセットを転置します。 LET では、各 BY グループの最後のオブザベーションのみが転置されます。PROC TRANSPOSE では、Price 変数のみが転置されます。OUT=では、転置結果が CLOSE データセットに入力されます。

```
proc transpose data=stocks out=close let;
```

出力データセットを BY グループに編成します。 BY ステートメントでは、会社ごとに 1 つずつ、2 つの BY グループが作成されます。

```
  by company;
```

転置された変数の名前を指定します。 Date の値が、転置された変数の名前として使用されます。

```
  id date;
run;
```

CLOSE データセットを印刷します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。

```
proc print data=close noobs;
  title 'Closing Prices for Horizon Kites and SkyHi Kites';
run;
```

出力

次のデータセットは、出力データセット CLOSE です。

アウトプット 62.8　終値

Closing Prices for Horizon Kites and SkyHi Kites

Company	_NAME_	jun11	jun12
Horizon Kites	Price	27	30
SkyHi Kites	Price	44	45

例 6: 統計分析用データの転置

要素:
COPY ステートメント
VAR ステートメント

この例では、多変量または単変量の反復測定分析に適したものになるようにデータを配置します。

データ元は、SAS System for Linear Models, Third Edition の第 8 章“Repeated-Measures Analysis of Variance”です。

プログラム 1

```
options nodate pageno=1 linesize=80 pagesize=40;

data weights;
  input Program $ s1-s7;
  datalines;
CONT  85 85 86 85 87 86 87
CONT  80 79 79 78 78 79 78
CONT  78 77 77 77 76 76 77
CONT  84 84 85 84 83 84 85
CONT  80 81 80 80 79 79 80
RI    79 79 79 80 80 78 80
RI    83 83 85 85 86 87 87
RI    81 83 82 82 83 83 82
RI    81 81 81 82 82 83 81
RI    80 81 82 82 82 84 86
WI    84 85 84 83 83 83 84
WI    74 75 75 76 75 76 76
WI    83 84 82 81 83 83 82
WI    86 87 87 87 87 87 86
WI    82 83 84 85 84 85 86
;

data split;
  set weights;
  array s{7} s1-s7;
  Subject + 1;
  do Time=1 to 7;
    Strength=s{time};
    output;
  end;
  drop s1-s7;
run;

proc print data=split(obs=15) noobs;
  title 'SPLIT Data Set';
  title2 'First 15 Observations Only';
run;
```

プログラムの説明

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

```
options nodate pageno=1 linesize=80 pagesize=40;
```

WEIGHTS データセットを作成します。 WEIGHTS のデータは、3 つの重量挙げプログラムの運動療法研究の結果を表しています。ここでは、CONT は対照群、RI は反復回数を増やすプログラム、WI は重量を増やすプログラムです。

```
data weights;
  input Program $ s1-s7;
  datalines;
CONT  85 85 86 85 87 86 87
CONT  80 79 79 78 78 79 78
CONT  78 77 77 77 76 76 77
CONT  84 84 85 84 83 84 85
CONT  80 81 80 80 79 79 80
RI    79 79 79 80 80 78 80
RI    83 83 85 85 86 87 87
RI    81 83 82 82 83 83 82
RI    81 81 81 82 82 83 81
RI    80 81 82 82 82 84 86
WI    84 85 84 83 83 83 84
WI    74 75 75 76 75 76 76
WI    83 84 82 81 83 83 82
WI    86 87 87 87 87 87 86
WI    82 83 84 85 84 85 86
;
```

SPLIT データセットを作成します。 この DATA ステップでは、WEIGHT が再配置され、データセット SPLIT が作成されます。DATA ステップでは、強さの値が転置され、2 つの新しい変数 Time と Subject が作成されます。SPLIT には、反復測定値ごとに 1 つずつオブザベーションが含まれます。PROC GLM ステップで SPLIT を使用すると、単変量反復測定分析を行うことができます。

```
data split;
  set weights;
  array s{7} s1-s7;
  Subject + 1;
  do Time=1 to 7;
    Strength=s{time};
    output;
  end;
  drop s1-s7;
run;
```

SPLIT データセットを出力します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。OBS=データセットオプションは、出力を最初の 15 のオブザベーションのみに制限します。SPLIT には 105 のオブザベーションがあります。

```
proc print data=split(obs=15) noobs;
  title 'SPLIT Data Set';
  title2 'First 15 Observations Only';
run;
```

出力 1

アウトプット 62.9　SPLIT データセット

SPLIT Data Set
First 15 Observations Only

Program	Subject	Time	Strength
CONT	1	1	85
CONT	1	2	85
CONT	1	3	86
CONT	1	4	85
CONT	1	5	87
CONT	1	6	86
CONT	1	7	87
CONT	2	1	80
CONT	2	2	79
CONT	2	3	79
CONT	2	4	78
CONT	2	5	78
CONT	2	6	79
CONT	2	7	78
CONT	3	1	78

プログラム 2

```
options nodate pageno=1 linesize=80 pagesize=40;

proc transpose data=split out=totsplit prefix=Str;

  by program subject;
  copy time strength;

  var strength;
run;

proc print data=totsplit(obs=15) noobs;
  title  'TOTSPLIT Data Set';
  title2 'First 15 Observations Only';
run;
```

プログラムの説明

SAS システムオプションを設定します。

```
options nodate pageno=1 linesize=80 pagesize=40;
```

SPLIT データセットを転置します。 PROC TRANSPOSE では、SPLIT を転置して TOTSPLIT を作成します。TOTSPLIT データセットには、SPLIT と同じ変数に加えて、強さの測定値それぞれに対する変数(Str1-Str7)が含まれます。TOTSPLIT は、多変量反復測定分析または単変量反復測定分析に使用できます。

```
proc transpose data=split out=totsplit prefix=Str;
```

出力データセットを BY グループに編成し、各 BY グループに未転置の値を入力します。 BY および COPY ステートメントの変数は転置されません。TOTSPLIT には、SPLIT と同じ値の変数 Program、Subject、Time、Strength が含まれます。BY ステートメントでは、各 BY グループの最初のオブザベーションが作成されます。これには転置された Strength の値が含まれます。COPY ステートメントでは、転置はせずに Time と Strength の値がコピーされ、各 BY グループのその他のオブザベーションが作成されます。

```
  by program subject;
  copy time strength;
```

転置する変数を指定します。 VAR ステートメントでは、転置する唯一の変数として Strength 変数が指定されます。

```
  var strength;
run;
```

TOTSPLIT データセットを出力します。 NOOBS オプションを指定すると、オブザベーション番号は印刷されません。OBS=データセットオプションは、出力を最初の 15 のオブザベーションのみに制限します。SPLIT には 105 のオブザベーションがあります。

```
proc print data=totsplit(obs=15) noobs;
  title  'TOTSPLIT Data Set';
  title2 'First 15 Observations Only';
run;
```

出力 2

次の出力では、値の欠損している TOTSPLIT の変数は、多変量反復測定分析でのみ使用されます。PROC GLM の MODEL ステートメントでは欠損値のあるオブザベーションは無視されるので、欠損値があることでこのデータセットが反復測定分析で使用されなくなることはありません。

アウトプット 62.10　TOTSPLIT データセット

TOTSPLIT Data Set
First 15 Observations Only

Program	Subject	Time	Strength	_NAME_	Str1	Str2	Str3	Str4	Str5	Str6	Str7
CONT	1	1	85	Strength	85	85	86	85	87	86	87
CONT	1	2	85		.	.	.	.	.	.	.
CONT	1	3	86		.	.	.	.	.	.	.
CONT	1	4	85		.	.	.	.	.	.	.
CONT	1	5	87		.	.	.	.	.	.	.
CONT	1	6	86		.	.	.	.	.	.	.
CONT	1	7	87		.	.	.	.	.	.	.
CONT	2	1	80	Strength	80	79	79	78	78	79	78
CONT	2	2	79		.	.	.	.	.	.	.
CONT	2	3	79		.	.	.	.	.	.	.
CONT	2	4	78		.	.	.	.	.	.	.
CONT	2	5	78		.	.	.	.	.	.	.
CONT	2	6	79		.	.	.	.	.	.	.
CONT	2	7	78		.	.	.	.	.	.	.
CONT	3	1	78	Strength	78	77	77	77	76	76	77

63

XSL プロシジャ

概要: XSL プロシジャ

Extensible Style Sheet Language (XSL)プロシジャの動作について

XSL プロシジャは、XML ドキュメントを、HTML、テキスト、または別の種類の XML ドキュメントなどの別の出力形式に変換します。PROC XSL は入力 XML ドキュメントを読み込み、XSL スタイルシートを使用して変換して、出力ファイルを書き出します。

PROC XSL は、XML ドキュメントを変換するために、Saxonica の Saxon-EE version 9.3 ソフトウェアアプリケーションを使用します。このアプリケーションは、XML ドキュメント処理用ツールを集めたものです。XSLT プロセッサには、XSLT 2.0 基準が導入されています。Saxon の詳細については、Web サイト About Saxon を参照してください。

XSL について

XSL は、XML でエンコードされるファイルの変換方法の説明を可能にする一群の変換言語です。言語には、次が含まれます。

- XML ドキュメントを変換するための XSL Transformations (XSLT)
- XSLT によって使用される、XML ドキュメントの一部を選択するための XML Path Language (XPath)

XSLT 基準の詳細については、Web サイト XSL Transformations (XSLT) Version 2.0 を参照してください。

構文: XSL プロシジャ

制限事項:

2024.12 以降、SAS がロックダウンされている場合、XSL プロシジャはデフォルトで無効になります。SAS Viya でプロシジャを使用できるようにするには、管理者が LOCKDOWN ステートメントで ENABLE_AMS=XSL を指定する必要があります。詳細については、“LOCKDOWN” (*SAS プログラマガイド: エッセンシャル*)および “Example 6: Enabling PROC XSL While in Lockdown Mode” (*SAS Viya Platform: Programming Run-Time Servers*)を参照してください。

PROC XSL IN=*fileref* | '*external-file*'OUT=*fileref* | '*external-file*'XSL=*fileref* | '*external-file*';

PARAMETER '*parameter*'='*value*';

ステートメント	タスク	例
“PROC XSL ステートメント”	XML ドキュメントを変換する	Ex. 1, Ex. 2, Ex. 3
“PARAMETER ステートメント”	パラメーターを XSL スタイルシートに渡して値を設定する	Ex. 2, Ex. 3

PROC XSL ステートメント

XML ドキュメントを変換します。

注:

XSL プロシジャは SAS LOCKDOWN 制限をサポートします。詳細については、“LOCKDOWN” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

例:

“例 1: XML ドキュメントの別の XML ドキュメントへの変換” (2398 ページ)

構文

PROC XSL IN=*fileref* | '*external-file*'OUT=*fileref* | '*external-file*'XSL=*fileref* | '*external-file*';

必須引数

IN=*fileref* | '*external-file*'
入力ファイルを指定します。ファイルは、正しい形式の XML ドキュメントである必要があります。

fileref
入力 XML ドキュメントに割り当てられている SAS ファイル参照名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'***external-file***'
入力 XML ドキュメントの物理的な場所です。完全なパス名とファイル名を含める必要があります。物理名は単一引用符または二重引用符で囲みます。最大長は 200 文字です。

制限事項 LOCKDOWN オプションまたは NOXCMD オプションがセッションに設定されている場合、XSL ファイルに指定されているオペレーティングシステムのコマンドは ERROR で失敗します。

例 “例 1: XML ドキュメントの別の XML ドキュメントへの変換” (2398 ページ)

OUT=*fileref* | '*external-file*'
出力ファイルを指定します。

fileref
出力ファイルに割り当てられる SAS ファイル参照名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'***external-file***'
出力ファイルの物理的な場所です。完全なパス名とファイル名を含める必要があります。物理名は単一引用符または二重引用符で囲みます。最大長は 200 文字です。

例 “例 1: XML ドキュメントの別の XML ドキュメントへの変換” (2398 ページ)

XSL=*fileref* | '*external-file*'
XML ドキュメントを変換する XSL スタイルシートを指定します。XSL スタイルシートは、XSLT 言語を使用した XML ドキュメントの変換方法を説明するファイルです。正しい形式の XML ドキュメントである必要があります。

fileref
XSL スタイルシートに割り当てられる SAS ファイル参照名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'***external-file***'
XSL スタイルシートの物理的な場所です。完全なパス名とファイル名を含める必要があります。物理名は単一引用符または二重引用符で囲みます。最大長は 200 文字です。

別名 XSLT=

例 “例 1: XML ドキュメントの別の XML ドキュメントへの変換” (2398 ページ)

PARAMETER ステートメント

XSL スタイルのパラメーターのインスタンスを指定した値に変更します。

例: “例 2: 文字列パラメーター値の XSL スタイルシートへの受け渡し” (2400 ページ)
“例 3: 数値パラメーター値の XSL スタイルシートへの受け渡し” (2403 ページ)

構文

PARAMETER '*parameter*'='*value*';

必須引数

'*parameter*'='*value*'
XSL スタイルシートに渡されるパラメーター名と値を指定します。PROC XSL は、スタイルシートのパラメーターのインスタンスを、指定した値に変更するものです。指定したパラメーターが XSL スタイルシート内に存在する必要があります。パラメーター名を単一引用符または二重引用符で囲みます。指定した値は文字列または数値になります。値が文字列である場合は、値を単一引用符か二重引用符で囲みます。

例: XSL プロシジャ

例 1: XML ドキュメントの別の XML ドキュメントへの変換

要素: PROC XSL ステートメント

詳細

次の PROC XSL 例では、XML ドキュメントを別の XML ドキュメントに変換しています。

これは、車に関するデータを含む XMLInput.xml という名前の入力 XML ドキュメントです。第 2 レベルの繰り返し要素は、それぞれ特定の車を、そのモデルと年度に関する情報を含むネスト要素を使用して記述しています。製造メーカー情報は、第 2 レベルの繰り返し要素の属性です。

```
<?xml version="1.0" ?>
<vehicles>
 <car make="Ford">
   <model>Mustang</model>
   <year>1965</year>
 </car>
 <car make="Chevrolet">
   <model>Nova</model>
   <year>1967</year>
 </car>
</vehicles>
```

これは、XML の変換方法を説明する XSLTransform.xsl という名前の XSL スタイルシートです。変換は<root>をルートのエンクロージング要素として、<model>を第 2 レベルの繰り返し要素として作成します。出力 XML ドキュメントの各<model>要素は、<car>要素からの値と、入力 XML ドキュメントからの make=属性を含みます。

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/vehicles">
     <root> <xsl:apply-templates select="car"/> </root>
</xsl:template>

<xsl:template match="car">
   <model make="{@make}">
     <xsl:value-of select="model" />
   </model>
</xsl:template>

</xsl:stylesheet>
```

プログラム

PROC XSL ステートメントを抽出します。 PROC XSL ステートメントは、入力 XML ドキュメント、XSL スタイルシート、出力 XML ドキュメントを指定します。

```
proc xsl
  in='file-path/XmlInput.xml'
  xsl='file-path/XslTransform.xsl'
  out='file-path/XmlOutput.xml';
run;
```

出力: XML ドキュメントの別の XML ドキュメントへの変換

アウトプット 63.1　変換された XML ドキュメントの PROC XSL 出力

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<model make="Ford">Mustang</model>
<model make="Chevrolet">Nova</model>
</root>
```

例 2: 文字列パラメーター値の XSL スタイルシートへの受け渡し

要素:
PROC XSL ステートメント
PARAMETER ステートメント

詳細

この例では、PROC XSL を使用して、文字列値を XSL スタイルシートのパラメーターに渡す方法を示します。パラメーターは、値を設定できるスタイルシート内にある名前がついている変数であり、生成した出力を簡単にカスタマイズできる方法です。

これは Format.xsl という名前の XSL スタイルシートです。XSL スタイルシートは、入力 XML ドキュメントから要素と属性を抽出して HTML 出力を生成します。このスタイルシートには、宣言パラメーター DateVar が含まれます。

```
<?xml version="1.0" encoding="UTF-8"?>
 <xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="DateVar"/>
  <xsl:template match="TABLE">
   <html>
   <head/>
```

```
  <body>
  <h2><xsl:value-of select="$DateVar"/></h2>
  <table border="1" rules="all">
  <tr>
    <td>Name</td>
    <td>Sex</td>
    <td>Age</td>
    <td>Height</td>
    <td>Weight</td>
  </tr>
  <xsl:apply-templates/>
  </table>
  </body>
 </html>
 </xsl:template>
 <xsl:template match="CLASS">
  <tr>
     <td><xsl:value-of select="Name"/></td>
     <td><xsl:value-of select="Sex"/></td>
     <td><xsl:value-of select="Age"/></td>
     <td><xsl:value-of select="Height"/></td>
     <td><xsl:value-of select="Weight"/></td>
  </tr>
 </xsl:template>
</xsl:stylesheet>
```

Class.xml という名前の入力 XML ドキュメントには、クラスルームデータが含まれます。

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
  <CLASS>
    <Name> Alfred </Name>
    <Sex> M </Sex>
    <Age> 14 </Age>
    <Height> 69 </Height>
    <Weight> 112.5 </Weight>
  </CLASS>
  <CLASS>
    <Name> Alice </Name>
    <Sex> F </Sex>
    <Age> 13 </Age>
    <Height> 56.5 </Height>
    <Weight> 84 </Weight>
  </CLASS>
  <CLASS>
    <Name> Barbara </Name>
    <Sex> F </Sex>
    <Age> 13 </Age>
    <Height> 65.3 </Height>
    <Weight> 98 </Weight>
  </CLASS>
  .
  .
  .
  <CLASS>
    <Name> Thomas </Name>
```

```
    <Sex> M </Sex>
    <Age> 11 </Age>
    <Height> 57.5 </Height>
    <Weight> 85 </Weight>
  </CLASS>
  <CLASS>
    <Name> William </Name>
    <Sex> M </Sex>
    <Age> 15 </Age>
    <Height> 66.5 </Height>
    <Weight> 112 </Weight>
  </CLASS>
</TABLE>
```

プログラム

```
proc xsl
  in='file-path/Class.xml'
  xsl='file-path/Format.xsl'
  out='file-path/Class.html';

  parameter 'DateVar'="Report Date: &sysdate";
run;
```

プログラムの説明

PROC XSL ステートメントを抽出します。 PROC XSL ステートメントは、入力 XML ドキュメント、XSL スタイルシート、出力 HTML ファイルを指定します。

```
proc xsl
  in='file-path/Class.xml'
  xsl='file-path/Format.xsl'
  out='file-path/Class.html';
```

パラメーター値を XSL スタイルシートに渡します。 PARAMETER ステートメントは、パラメーター DateVar の文字列値を XSL スタイルシートに渡します。この値はマクロ変数参照を含んでいるため、必ず二重引用符で囲んでください。

```
  parameter 'DateVar'="Report Date: &sysdate";
run;
```

出力: 文字列パラメーター値の XSL スタイルシートへの受け渡し

アウトプット 63.2 PROC XSL Output Class.html

Report Date: 15NOV12

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

例 3: 数値パラメーター値の XSL スタイルシートへの受け渡し

要素: PROC XSL ステートメント
PARAMETER ステートメント

詳細

この例では、PROC XSL を使用して、数値を XSL スタイルシートのパラメーターに渡す方法を示します。

これは、Discount.xsl という名前の XSL スタイルシートです。XSL スタイルシートは、入力 XML ドキュメントから要素と属性を抽出して XML 出力を生成します。このスタイルシートには、宣言パラメーター DiscountPct が含まれます。PROC XSL は数値の状態でパラメーターへの受け渡しを行い、割引額が算出されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" indent="yes" />
 <xsl:param name="DiscountPct" />

 <xsl:template match="table">
  <xsl:copy>
   <xsl:apply-templates select="product" />
  </xsl:copy>
 </xsl:template>

 <xsl:template match="product">
  <xsl:copy>
   <xsl:copy-of select="category|type|size|gender|price" />
   <discount>
    <xsl:value-of select="$DiscountPct" />
   </discount>
   <discountAmount>
    <xsl:value-of select="(price * $DiscountPct)" />
   </discountAmount>
   <xsl:copy-of select="in_stock|ship_type" />
  </xsl:copy>
 </xsl:template>

</xsl:stylesheet>
```

Product.xml という名前の入力 XML ドキュメントには、製品データが含まれます。

```
<?xml version="1.0"?>
<table>
 <product>
  <category>shoe</category>
  <type>Nike Air</type>
  <size>12</size>
  <gender>male</gender>
  <price>99</price>
  <in_stock>yes</in_stock>
  <ship_type>overnight</ship_type>
 </product>
</table>
```

プログラム

```
proc xsl
  in='file-path/Product.xml'
  xsl='file-path/Discount.xsl'
  out='file-path/Calculated.xml';

  parameter 'DiscountPct'=.20;
run;
```

プログラムの説明

PROC XSL ステートメントを抽出します。 PROC XSL ステートメントは、入力 XML ドキュメント、XSL スタイルシート、出力 XML ドキュメントを指定します。

```
proc xsl
  in='file-path/Product.xml'
  xsl='file-path/Discount.xsl'
  out='file-path/Calculated.xml';
```

パラメーター値を XSL スタイルシートに渡します。 PARAMETER ステートメントは、パラメーター DiscountPct の数値を XSL スタイルシートに渡します。出力 XML ドキュメントに、割引値と算出された割引額が含まれるようになります。

```
  parameter 'DiscountPct'=.20;
run;
```

出力: 数値パラメーター値の XSL スタイルシートへの受け渡し

アウトプット 63.3　PROC XSL Output Calculated.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<table>
   <product>
      <category>shoe</category>
      <type>Nike Air</type>
      <size>12</size>
      <gender>male</gender>
      <price>99</price>
      <discount>0.2</discount>
      <discountAmount>19.8</discountAmount>
      <in_stock>yes</in_stock>
      <ship_type>overnight</ship_type>
   </product>
</table>
```

3 部

付録

付録 1

基本的な SAS 統計プロシジャ

基本的な SAS 統計プロシジャの概要

この付録では、基本的な統計に対する Base SAS プロシジャの出力の解釈に必要な一部の統計的概念について簡単に説明しています。また、この付録には統計表記、式、Base SAS プロシジャで一般的な統計に使用される標準キーワードがリストされています。簡単な例で、統計的概念を示します。

表 A1.1 (2411 ページ)に、最も一般的な統計量と、それらを計算するプロシジャを表示します。

キーワードと式

単純統計量

Base SAS プロシジャでは、標準化された一連のキーワードを使用して統計量を表します。これらのキーワードを SAS ステートメントで指定して、統計量が表示されるように、または出力データセットに保存されるように要求します。

次の表記では、合計が分析済変数の非欠損値を含むオブザベーションを超え、表示されている部分以外では、非欠損の重みと度数を超えています。

x_i
は、オブザベーション *i* に対して分析された変数の非欠損値です。

f_i
は、FREQ ステートメントを使用する場合、x_iに関連付けられる度数です。FREQ ステートメントを省略する場合、$f_i = 1$ (すべての *i* に対し)となります。

w_i
は、WEIGHT ステートメントを使用する場合、x_iに関連付けられている重みです。基本プロシジャは欠損した重みを含むx_iの値を分析から自動的に除外します。

デフォルトで、基本プロシジャは負の重みをゼロとして処理します。ただし、PROC ステートメントで EXCLNPWGT オプションを使用する場合、プロシジャは非正の重みを含むx_iの値も除外します。PROC TTEST や PROC GLM などの SAS/STAT プロシジャのほとんどが、デフォルトで非正の重みを持つ値を除外します。

WEIGHT ステートメントを省略する場合、$w_i = 1$ (すべての *i* に対し)となります。

n
は、x_i、Σf_iの非欠損値の数です。EXCLNPWGT オプションと WEIGHT ステートメントを使用する場合、nは、正の重みを含む非欠損値の数です。

$\overline{x}$
平均です

$$\Sigma w_i x_i / \Sigma w_i$$

s^2
分散です

$$\frac{1}{d} \Sigma w_i (x_i - \overline{x})^2$$

dは、PROC ステートメントで指定する分散の除数(VARDEF=オプション)です。有効な値は次のとおりです。

When VARDEF=	d等しい
N	n
DF	$n - 1$
WEIGHT	$\Sigma\, w_i$
WDF	$\Sigma\, w_i - 1$

デフォルトは DF です。

z_i

は標準化された変数です

$(x_i - \overline{x})/s$

統計量ごとの標準キーワードおよび式が後に続きます。一部の式ではキーワードを使用して、対応する統計量を指定します。

表 A57.1 最も一般的な基本統計量

統計量	PROC MEANS および SUMMARY	PROC UNIVARIATE	PROC TABULATE	PROC REPORT	PROC CORR	PROC SQL
欠損値の数	X	X	X	X		X
非欠損値の数	X	X	X	X	X	X
オブザベーション数	X	X				X
重みの合計	X	X	X	X	X	X
Mean	X	X	X	X	X	X
合計	X	X	X	X	X	X
極値	X	X				
最小	X	X	X	X	X	X
最大	X	X	X	X	X	X
範囲	X	X	X	X		X
無修正平方和	X	X	X	X	X	X
修正済み平方和	X	X	X	X	X	X
分散	X	X	X	X	X	X

統計量	PROC MEANS および SUMMARY	PROC UNIVARIATE	PROC TABULATE	PROC REPORT	PROC CORR	PROC SQL
共分散					X	
標準偏差	X	X	X	X	X	X
平均の標準誤差	X	X	X	X		X
変動係数	X	X	X	X		X
歪度	X	X	X			
尖度	X	X	X			
信頼限界						
(平均)	X	X	X			
(分散)		X				
(分位点)		X				
中央値	X	X	X	X	X	
モード	X	X	X	X		
パーセント点/十分位点/分位点	X	X	X	X		
t 検定						
(平均=0)	X	X	X	X		X
(平均= μ_0)		X				
位置のノンパラメトリック検定		X				
正規性の検定		X				
相関係数					X	
Cronbach のアルファ統計量					X	

記述統計量

記述統計量のキーワードは、次のとおりです。

CSS
修正済平方和です。次のように計算されます

$$\Sigma\ w_i(x_i - \overline{x})^2$$

CV
は単位がパーセントの変動係数です。次のように計算されます。

$$(100s)/\overline{x}$$

KURTOSIS | KURT
は尖度、度数の重みの尺度です。VARDEF=DF の場合、尖度は次のように計算されます。

$$c_{4_n}\ \Sigma\ z_i^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

ここで、c_{4_n}は、$\frac{n(n+1)}{(n-1)(n-2)(n-3)}$です。重み付き歪度は、次のように計算されます

$$= c_{4_n}\ \Sigma\ ((x_i - \overline{x})/\widehat{\sigma}_i)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

$$= c_{4_n}\ \Sigma\ w_i^2((x_i - \overline{x})/\widehat{\sigma})^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

VARDEF=N の場合、尖度は次のように計算されます。

$$= \frac{1}{n}\ \Sigma\ z_i^4 - 3$$

また、重みの付いた尖度は、次のように計算されます

$$= \frac{1}{n}\ \Sigma\ ((x_i - \overline{x})/\widehat{\sigma}_i)^4 - 3$$

$$= \frac{1}{n}\ \Sigma\ w_i^2((x_i - \overline{x})/\widehat{\sigma})^4 - 3$$

ここで、σ_i^2は、σ^2/w_iです。式は、$w_i^* = zw_i,\ z > 0$変換で変わりません。
VARDEF=WDF または VARDEF=WEIGHT を使用する場合、歪度は欠損に設定されます。

注: PROC MEANS および PROC TABULATE は、重みの付いた尖度を計算しません。

MAX
は、x_iの最大値です。

MEAN
は、算術平均$\overline{x}$です。

MIN
は、x_iの最小値です。

MODE
は、x_iの最頻値です。

注: QMETHOD=P2 の場合、PROC REPORT、PROC MEANS および PROC TABULATE は MODE を計算しません。

N
は、欠損していないx_i値の数です。1 未満のf_i、欠損値に等しいw_i、または$w_i \leq 0$(EXCLNPWGT オプションの使用時)のオブザベーションは分析から除外され、N の計算に含まれません。

NMISS
は、欠損しているx_i値の数です。1 未満のf_i、欠損値に等しいw_i、または$w_i \leq 0$(EXCLNPWGT オプションの使用時)のオブザベーションは分析から除外され、NMISS の計算に含まれません。

NOBS
オブザベーションの総計で、N と NMISS の合計として計算されます。ただし、WEIGHT ステートメントを使用する場合、NOBS は N、NMISS、および欠損している重みまたは非正の重みのために除外されたオブザベーションの数の合計として計算されます。

RANGE
範囲で、最大値と最小値の間の差として計算されます。

SKEWNESS | SKEW
歪度です。1 つの方向でより大きくなる偏差の傾向を測定します。VARDEF=DF の場合、歪度は次のように計算されます

$$c_{3_n} \Sigma z_i^3$$

ここで、c_{3_n}は、$\frac{n}{(n-1)(n-2)}$です。重み付けられた歪度は、次のように計算されます。

$$= c_{3_n} \Sigma \left((x_i - \overline{x})/\widehat{\sigma_j}\right)^3$$
$$= c_{3_n} \Sigma w_i^{3/2}((x_i - \overline{x})/\widehat{\sigma})^3$$

VARDEF=N の場合、歪度は次のように計算されます

$$= \frac{1}{n} \Sigma z_i^3$$

また、重み付けられた歪度は、次のように計算されます。

$$= \frac{1}{n} \Sigma \left((x_i - \overline{x})/\widehat{\sigma_j}\right)^3$$
$$= \frac{1}{n} \Sigma w_i^{3/2}((x_i - \overline{x})/\widehat{\sigma})^3$$

式は、$w_i^* = zw_i,\ z > 0$変換で変わりません。VARDEF=WDF または VARDEF=WEIGHT を使用する場合、歪度は欠損に設定されます。

注: PROC MEANS および PROC TABULATE は、重み付けられた歪度を計算しません。

STDDEV | STD
is the standard deviation *s* and is computed as the square root of the variance, s^2 .

STDERR | STDMEAN
は平均の標準誤差です。VARDEF=DF (デフォルト)の場合、次のように計算されます。

$$s/\sqrt{\Sigma\, w_i}$$

デフォルトは VARDEF=DF です。その他の場合、STDERR は欠損に設定されます。

SUM
合計です。次のように計算されます。

$$\Sigma\, w_i x_i$$

SUMWGT
は、重みの合計、Wです。次のように計算されます。

$$\Sigma\, w_i$$

USS
無修正平方和です。次のように計算されます

$$\Sigma\, w_i x_i^2$$

VAR
は、分散s^2です。

分位数と関連統計量

分位点のキーワードおよび関連する統計量は、次のとおりです。

MEDIAN
中央値です。

P1
最初(1^{st})のパーセント点です。

P5
5 番目(5^{th})のパーセント点です。

P10
10 番目(5^{th})のパーセント点です。

P90
90 番目(5^{th})のパーセント点です。

P95
95 番目(5th)のパーセント点です。

P99
99 番目(5th)のパーセント点です。

Q1
下位の四分位(25 番目(25th)のパーセント点)です。

Q3
上位の四分位(75 番目(75th)のパーセント点)です。

QRANGE
四分位範囲です。 次のように計算されます。

$$Q_3 - Q_1$$

QNTLDEF=オプション(PROC UNIVARIATE の PCTLDEF=)を使用して、プロシジャがパーセント点の計算に使用する方法を指定します。nを変数の非欠損値数にし、$x_1, x_2, \ldots, x_n$で、x_1が最小値、x_2が次の最小値、x_nが最大値になるように、変数の順序付けされた値を表します。*T 番目のパーセント点(0 から 1 まで)の場合*、番目のパーセント点(0 から 1 まで)の場合、$p = t/100$にします。次に、jをnpの整数部分として、gをnpまたは$(n+1)p$の小数部分として、次のようになるように指定します。

$$np = j + g \qquad \text{when QNTLDEF} = 1, 2, 3, \text{or } 5$$
$$(n+1)p = j + g \qquad \text{when QNTLDEF} = 4$$

ここで QNTLDEF=は、後に続く表のように、プロシジャが *t* 番目のパーセント点の計算に使用する方法を指定します。

WEIGHT ステートメントを使用する場合、*t* 番目のパーセント点が次のように計算されます。

$$y = \begin{cases} \frac{1}{2}(x_i + x_{i+1}) & \text{if } \sum_{j=1}^{i} w_j = pW \\ x_{i+1} & \text{if } \sum_{j=1}^{i} w_j < pW < \sum_{j=1}^{i+1} w_j \end{cases}$$

w_jはx_iに関連付けられた重みであり、$W = \sum_{i=1}^{n} w_i$は重みの合計です。オブザベーションの重みが同じ場合、重み付けられているパーセント点は、QNTLDEF=5 の重み付けされていないパーセント点と同じです。

表 A57.2 四分位範囲統計量の計算方法

QNTLDEF=	説明	式
1	x_{np}での重み付き平均	$y = (1-g)x_j + gx_{j+1}$
		x_oは次のように解釈されます。x_1
2	npに最も近い番号のオブザベーション	$y = x_i$ $g \neq \frac{1}{2}$の場合

QNTLDEF=	説明	式	
		$y = x_j$	$g = \frac{1}{2}$でjが偶数の場合
		$y = x_{j+1}$	$g = \frac{1}{2}$でjが奇数の場合
		ここで i は$np + \frac{1}{2}$の小数部分です。	
3	経験分布関数	$y = x_j$	$g = 0$の場合
		$y = x_{j+1}$	$g > 0$の場合
4	$x_{(n+1)p}$を目的とした重み付き平均	$y = (1-g)x_j + gx_{j+1}$	
		x_{n+1}は次のように解釈されます。x_n	
5	平均化を含む経験分布関数	$y = \frac{1}{2}(x_j + x_{j+1})$	$g = 0$の場合
		$y = x_{j+1}$	$g > 0$の場合

仮説検定統計量

仮説検定統計量のキーワードは、次のとおりです。

T
スチューデントの t 統計量です。母平均がμ_0と等しい帰無仮説を検定します。

$$\frac{\bar{x} - \mu_0}{s/\sqrt{\Sigma w_i}}$$

デフォルトで、μ_0はゼロと等しくなります。PROC UNIVARIATE ステートメントで MU0=オプションを使用して、μ_0を指定できます。デフォルトの分散の除数となる VARDEF=DF を使用する必要があります。その他の場合、T は欠損に設定されます。

デフォルトで、WEIGHT ステートメントの使用時は、プロシジャは自由度に非正の重みを含むx_i値をカウントします。PROC ステートメントで EXCLNPWGT オプションを使用して、非正の重みを含む値を除外します。PROC TTEST、PROC GLM などの SAS/STAT プロシジャのほとんどが非正の重みを含む値を自動的に除外します。

PROBT | PRT
両側 p 値(スチューデントの t 統計量 T に対する)であり、$n-1$自由度が含まれます。この値は、このサンプルで観測されるよりも多くの極値 T を取得する帰無仮説での確率です。

平均の信頼限界

信頼限界のキーワードは、次のとおりです。

CLM
平均の両側信頼限界です。平均に対する両側$100(1-\alpha)$パーセント信頼区間には、上限と下限があります。

$$\overline{x} \pm t_{(1-\alpha/2;n-1)}\frac{s}{\sqrt{\Sigma w_i}}$$

ここで、sは、$\sqrt{\frac{1}{n-1}\Sigma(x_i-\overline{x})^2}$であり、$t_{(1-\alpha/2;n-1)}$は自由度が$1-\alpha/2$であるスチューデントの t 統計量の$(n-1)$限界値です。αは ALPHA=オプションの値で、デフォルトで 0.05 になります。デフォルトの分散の除数となる VARDEF=DF を使用する場合を除いて、CLM は欠損に設定されます。

LCLM
平均値より下の片側信頼限界です。平均の片側$100(1-\alpha)$パーセント信頼区間には、下限があります。

$$\overline{x} - t_{(1-\alpha;n-1)}\frac{s}{\sqrt{\Sigma w_i}}$$

デフォルトの分散の除数となる VARDEF=DF を使用する場合を除いて、LCLM は欠損に設定されます。

UCLM
平均より上の片側信頼限界です。平均の片側$100(1-\alpha)$パーセント信頼区間には、上限があります。

$$\overline{x} + t_{(1-\alpha;n-1)}\frac{s}{\sqrt{\Sigma w_i}}$$

デフォルトの分散の除数となる VARDEF=DF を使用する場合を除いて、UCLM は欠損に設定されます。

重みの使用

WEIGHT ステートメントの例として、30cm 幅の対象物のサイズを推定するよう 20 人に求めたとします。対象物からの距離が異なる場所にそれぞれを配置します。対象物からの距離が増加するにつれて、推定値の精度は低下します。SAS データセット SIZE には、メートル単位の各距離(Distance)におけるセンチメートル単位の推定値(ObjectSize)、および各推定値の精度(Precision)が含まれます。最大偏差(20cm の

過大見積もり)が最長距離(対象物から 7.5 メートル)で発生していることに注意してください。精度(1/Distance)として、対象物に近い位置での推定値ほど重みを増やし、遠い位置での推定値ほど重みを減らします。PROC MEANS ステップでは、重みを無視して、物体サイズの平均推定値が計算されます。WEIGHT 変数がなければ、PROC MEANS では、すべてのオブザベーションに対してデフォルトの重み 1 が使用されます。したがって、すべての距離の物体サイズ推定値に、等しい重みが与えられます。物体サイズの平均推定値は、実際のサイズを 3.55cm 上回ります。

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

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

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

Unweighted Analysis of the SIZE Data Set

The MEANS Procedure

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

要約プロシジャのデータの必要条件

次に、重みなしの統計量を計算するための最小データ要件を示します。推奨されるサンプルサイズは記載されていません。統計量は、VARDEF=DF (デフォルト)で、次の要件が満たされていない場合は欠損としてレポートされます。

- N および NMISS は、欠損オブザベーション数、または非欠損オブザベーション数に関係なく計算されます。
- SUM、MEAN、MAX、MIN、RANGE、USS、CSS には、非欠損オブザベーションが少なくとも 1 つ必要です。
- VAR、STD、STDERR、CV、T、PRT、PROBT には、非欠損オブザベーションが少なくとも 2 つ必要です。
- SKEWNESS には、非欠損オブザベーションが少なくとも 3 つ必要です。

- KURTOSIS には、非欠損オブザベーションが少なくとも 4 つ必要です。
- SKEWNESS、KURTOSIS、T、PROBT、PRT では、STD がゼロより大きい必要があります。
- CV では、MEAN がゼロに等しくない必要があります。
- CLM、LCLM、UCLM、STDERR、T、PRT、PROBT では、VARDEF=DF である必要があります。

統計的手法の知識

母集団とパラメーター

通常、対象となる要素は明確に定義されています。この一連の要素は*ユニバース*と呼ばれ、これらの要素と関連する一連の値は値の*母集団*と呼ばれます。統計用語の*母集団*は、人々とは関係ありません。統計的な母集団は、人々の集まりではなく、値の集まりです。たとえば、ユニバースは特定の学校の全生徒で、対象となる 2 つの母集団、1 つは身長の値、もう 1 つは体重の値が存在する可能性があります。または、ユニバースは特定の会社で製造された全ウィジェットで、値の母集団は各ウィジェットが故障するまで使用される時間の長さである可能性があります。

値の母集団は、その*累積分布関数*の観点から表すことができます。可能な値にそれぞれ満たない、または等しい母集団の割合が示されます。個別の母集団は、*確率関数*によっても 表すことができます。 可能な値にそれぞれ等しい母集団の割合が示されます。連続した母集団は、*密度関数*によって表すことができます。これは累積分布関数の導関数です。密度関数は、それぞれの一連の値内の母集団の割合を示すヒストグラムによって近似できます。確率密度関数は、無数の無限に小さな間隔を含むヒストグラムに似ています。

技術文献では、用語*分布*が条件なしで使用される場合、通常、累積分布関数を指します。一般的な文章では、*分布*は密度関数を表すことがあります。用語*分布*は、多くの場合、具体的な母集団ではなく、値の抽象的な母集団を指すために使用されます。したがって、統計的な文献では、正規分布、指数分布、Cauchy 分布など、多種の抽象的な分布を指します。*正規分布*などの言葉が使用される場合、累積分布関数または密度関数を指しているのどうかかは多くの場合関係ありません。

分布の関連機能を要約する指標の観点から母集団を説明することは適切です。母集団値から計算されるそのような指標は、*パラメーター*と呼ばれます。多くの異なるパラメーターを指定して、分布のさまざまな側面を測定できます。

最も一般的に使用されているパラメーターは、(算術)*平均*です。母集団に有限数の値が含まれている場合、母平均は母集団の要素数によって除算される母集団のすべての値の合計として計算されます。無限の母集団の場合、平均の概念は似ていますが、より複雑な計算が必要になります。

E(*x*)は、*x* で表される、高さなどの値の母平均を示します。この場合、E は*期待値*を表します。元の値の導関数の期待値を考慮することもできます。たとえば、*x* が高さ

を表す場合、$\mathrm{E}(x^2)$は、高さの期待値の 2 乗、つまり、高さの母集団のすべての値を 2 乗することによって得られる母平均値です。

サンプルと統計量

母集団のすべての値を測定することはほとんど不可能です。測定された値の集合は、*サンプル*と呼ばれます。値のサンプルの数学関数は、*統計量*と呼ばれます。パラメーターが母集団に対応しているように、統計量はサンプルに対応しています。従来、統計量はローマ文字、パラメーターはギリシャ文字で表します。たとえば、多くの場合、母平均は μ、サンプル平均は$\bar{x}$と表されます。*統計量*の分野は、サンプル統計量の動作の調査と大きく関連しています。

サンプルは、さまざまな方法で選択可能です。ほとんどの SAS プロシジャは、データが*単純無作為抽出*を構成しているとみなします。 つまり、サンプルはすべての可能なサンプルが均等に選択されるような方法で選択されました。

サンプルからの統計量は、母集団のパラメーターに関して推定または合理的な推測を行うために使用できます。たとえば、高校から 30 名の生徒を無作為抽出する場合、これらの 30 名の生徒の平均身長が高校の全生徒の平均身長の合理的な推定、または*推測*となります。標準誤差などのその他の統計量では、推定がどれだけ正確になるかに関する情報を提供可能です。

母集団のパラメーターの場合、複数の統計量で推定可能です。ただし、多くの場合、指定パラメーターの推定に通常使用される特定の統計量が 1 つあります。たとえば、サンプル平均は母平均の通常の推定量です。平均の場合、パラメーターの式と統計量の式は同じです。その他の場合、パラメーターの式は最も一般的に使用される推定量の式とは異なることがあります。最も一般的に使用される推定量は、必ずしもすべてのアプリケーションでの 最適な推定量ではありません。

位置の統計量

位置の統計量の概要

位置の統計量には、平均、中央値、モードが含まれます。これらの統計量は、分布の中央を表します。後に続く定義では、一定量を各オブザベーションに追加することによりサンプル全体が変わる場合、これらの位置の統計量は同じ一定量によって変わります。

平均値

母平均$\mu = \mathrm{E}(x)$は、通常、サンプル平均$\bar{x}$によって推定されます。

中央値

母集団の中央値は、母集団値の半分の上下にある中央値です。サンプル中央値は、データが昇順または降順に調整されるときの中間値です。偶数のオブザベーションの場合、2 つの中間値の中間点は通常、中央値としてレポートされます。

モード

モードは、母集団の密度が最大の値です。一部の密度には複数のローカル最大値(ピーク)があり、*多峰分布*と呼ばれます。サンプルモードは、サンプルで最も頻繁に発生する値です。デフォルトで、最も頻繁に発生するサンプル値のタイがある場合、PROC UNIVARIATE はそのような最低値をレポートします。PROC ステートメントで MODES オプションを指定する場合、PROC UNIVARIATE はすべての可能なモードをリストします。母集団が連続している場合、すべてのサンプル値が一度に発生し、サンプルモードはほとんど使用されません。

パーセント点

パーセント点(分位点、四分位点、中央値を含む)は、分布の詳細調査に使用されます。統計量はその大きさで並べられます。*p*th パーセント点には、その点の下に *p* パーセント、その点より上に(100−*p*)パーセントの数の統計量が存在します。中央値は 50 パーセント点です。希望どおりのパーセント点が取得できるようにデータを分割できないことがあるため、UNIVARIATE プロシジャはより正確な定義を使用します。

分布の上位の四分位は、統計量の 75% (75 番目のパーセント点)が下回る値です。統計量の 25 パーセントが下位の四分位値を下回ります。

分位点

次の例では、SAS で乱数関数を含むデータを作為的に生成します。UNIVARIATE プロシジャはさまざまな分位点と位置の統計量を計算し、値を SAS データセットに出力します。次に DATA ステップは SYMPUT ルーチンを使用して、統計量の値をマクロ変数に割り当てます。マクロ%FORMGEN はこれらのマクロ変数を使用して、FORMAT プロシジャに対し値ラベルを作成します。PROC CHART は結果の出力形式を使用して、統計量の値をヒストグラムに表示します。

```
options nodate pageno=1 linesize=80 pagesize=52;

title 'Example of Quantiles and Measures of Location';

data random;
  drop n;
```

```
  do n=1 to 1000;
    X=floor(exp(rannor(314159)*.8+1.8));
    output;
  end;
run;

proc univariate data=random nextrobs=0;
  var x;
  output out=location
       mean=Mean mode=Mode median=Median
       q1=Q1 q3=Q3 p5=P5 p10=P10 p90=P90 p95=P95
       max=Max;
run;

proc print data=location noobs;
run;

data _null_;
  set location;
  call symput('MEAN',round(mean,1));
  call symput('MODE',mode);
  call symput('MEDIAN',round(median,1));
  call symput('Q1',round(q1,1));
  call symput('Q3',round(q3,1));
  call symput('P5',round(p5,1));
  call symput('P10',round(p10,1));
  call symput('P90',round(p90,1));
  call symput('P95',round(p95,1));
  call symput('MAX',min(50,max));
run;

%macro formgen;
%do i=1 %to &max;
  %let value=&i;
  %if &i=&p5     %then %let value=&value  P5;
  %if &i=&p10    %then %let value=&value  P10;
  %if &i=&q1     %then %let value=&value  Q1;
  %if &i=&mode   %then %let value=&value  Mode;
  %if &i=&median %then %let value=&value  Median;
  %if &i=&mean   %then %let value=&value  Mean;
  %if &i=&q3     %then %let value=&value  Q3;
  %if &i=&p90    %then %let value=&value  P90;
  %if &i=&p95    %then %let value=&value  P95;
  %if &i=&max    %then %let value=>=&value;
  &i="&value"
%end;
%mend;

proc format print;
  value stat %formgen;
run;
options pagesize=42 linesize=80;

proc chart data=random;
  vbar x / midpoints=1 to &max by 1;
  format x stat.;
  footnote  'P5  =  5TH PERCENTILE';
```

```
  footnote2 'P10 = 10TH PERCENTILE';
  footnote3 'P90 = 90TH PERCENTILE';
  footnote4 'P95 = 95TH PERCENTILE';
  footnote5 'Q1  =  1ST QUARTILE  ';
  footnote6 'Q3  =  3RD QUARTILE  ';
run;
```

Example of Quantiles and Measures of Location

The UNIVARIATE Procedure
Variable: X

Moments			
N	1000	Sum Weights	1000
Mean	7.605	Sum Observations	7605
Std Deviation	7.38169794	Variance	54.4894645
Skewness	2.73038523	Kurtosis	11.1870588
Uncorrected SS	112271	Corrected SS	54434.975
Coeff Variation	97.0637467	Std Error Mean	0.23342978

Basic Statistical Measures			
Location		Variability	
Mean	7.605000	Std Deviation	7.38170
Median	5.000000	Variance	54.48946
Mode	3.000000	Range	62.00000
		Interquartile Range	6.00000

Tests for Location: Mu0=0				
Test	Statistic		p Value	
Student's t	t	32.57939	Pr > |t|	<.0001
Sign	M	494.5	Pr >= |M|	<.0001
Signed Rank	S	244777.5	Pr >= |S|	<.0001

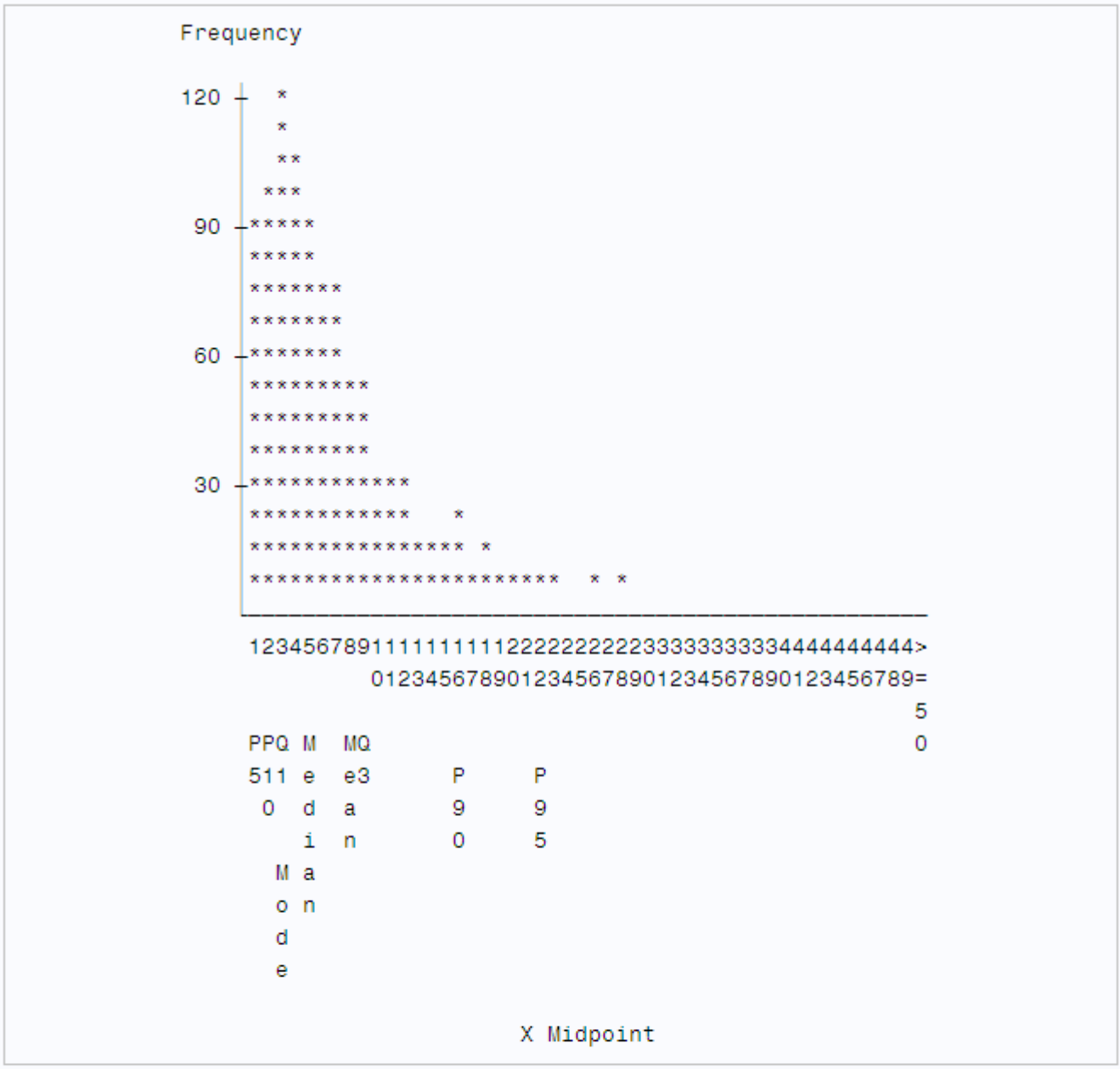
Tests for Location: Mu0=0				
Test	Statistic		p Value	
Student's t	t	32.57939	Pr > |t|	<.0001
Sign	M	494.5	Pr >= |M|	<.0001
Signed Rank	S	244777.5	Pr >= |S|	<.0001

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	62.0
99%	37.5
95%	21.5
90%	16.0
75% Q3	9.0
50% Median	5.0
25% Q1	3.0
10%	2.0
5%	1.0
1%	0.0
0% Min	0.0

Example of Quantiles and Measures of Location

Mean	Max	P95	P90	Q3	Median	Q1	P10	P5	Mode
7.605	62	21.5	16	9	5	3	2	1	3

Example of Quantiles and Measures of Location



P5 = 5TH PERCENTILE
P10 = 10TH PERCENTILE
P90 = 90TH PERCENTILE
P95 = 95TH PERCENTILE
Q1 = 1ST QUARTILE
Q3 = 3RD QUARTILE

ばらつきの統計量

ばらつきの統計量の概要

別の統計量グループが母集団の分布の調査で重要です。これらの統計量は、*拡がり*とも呼ばれる、値のばらつきを測定します。後に続くセクションで指定される定義では、サンプル全体が各オブザベーションへの一定量の追加によって変更される場合、これらの統計量の値は変わらないことに注意してください。ただし、サンプルの各オブザベーションに定数が乗算されると、これらの統計量の値は適切に調整されます。

範囲

サンプル範囲は、サンプルの最大値と最小値の間の差です。多くの母集団の場合、少なくとも統計的な理論では、範囲は無限です。このため、サンプル範囲には母集団が示されないことがあります。サンプル範囲は、サンプルサイズが大きくなるごとに増大する傾向があります。すべてのサンプル値に定数が乗算されると、サンプル範囲に同じ定数が乗算されます。

四分位範囲

四分位範囲は、上位の四分位と下位の四分位の間の差です。すべてのサンプル値に定数が乗算されると、サンブルの四分位範囲に同じ定数が乗算されます。

分散

通常、σ^2で表される母分散は、母平均からの値の 2 乗された差の期待値です。

$$\sigma^2 = \mathrm{E}(x - \mu)^2$$

サンプル分散は、次のように表されます。 s^2値と平均の間の差は、*平均からの偏差*と呼ばれます。そのため、分散により 2 乗された偏差の平均が近似されます。

すべての値が平均に近い場合、分散は小さくなりますが、ゼロ未満にはなりません。値がさらに散在すると、分散はより大きくなります。すべてのサンプル値に定数が乗算されると、サンプル分散に定数の 2 乗が乗算されます。

$n - 1$以外の値が分母で使用される場合があります。VARDEF=オプションは、プロシジャが使用する除数を制御します。

標準偏差

標準偏差は母集団またはサンプルのいずれかで分散の平方根、または平均の標準偏差となります。通常、母集団には σ、 サンプルには *s* のシンボルがそれぞれ使用されます。標準分散は、2 乗単位ではなく、オブザベーションと同じ単位で表されます。すべてのサンプル値に定数が乗算されると、サンプル標準分散に同じ定数が乗算されます。

変動係数

変動係数は、相対的なばらつきの単位を持たない指標です。これは、標準偏差と平均の比をパーセントで表示したものです。変動係数は、変数が比尺度で測定された場合にのみ意味を持ちます。すべてのサンプル値に定数が乗算されると、サンプルの変動係数は変わりません。

形状の統計量

歪度

分散は、平均からの偏差の全体サイズの指標です。分散の式では偏差を 2 乗するため、正と負の偏差は同じように分散に影響します。分布の多くでは、正の偏差は負の偏差より大きさが大きい、またはその逆となる傾向があります。*歪度*は、1 つの方向でより大きくなる偏差の傾向の指標です。たとえば、最後の例のデータは、右に偏っています。

母集団の歪度は、次のように定義されます。

$$\mathrm{E}(x-\mu)^3/\sigma^3$$

偏差は 2 乗ではなく 3 乗されるため、偏差の符号は保持されます。偏差の 3 乗により、大きな偏差の影響が強調されます。式にはスケールの影響を取り除くためのσ^3の除数が含まれているため、すべての値に定数を乗算しても、歪度は変わりません。そのため歪度は、母集団の裾が片方よりも重い傾向として解釈されます。歪度は、正または負である可能性があり、無制限です。

尖度

分布の裾の重さは、多くの統計量の動作に影響します。そのため、裾の重さの指標を使用すると便利です。そのような指標の 1 つは*尖度*です。母集団の尖度は、通常、次のように定義されます。

$$\frac{\mathrm{E}(x-\mu)^4}{\sigma^4} - 3$$

注: 一部の統計量では、3 の減算が省略されます。

偏差が 4 累乗されるため、正と負の偏差の分布は同じで、大きな偏差は強く強調されます。除数σ^4により、各値に定数を除算しても、尖度には影響しません。

母集団の尖度の範囲は、-2と$+\infty$の間です(両端の値を含める)。M_3が母集団の歪度を表し、M_4が母集団の尖度を表す場合、次のようになります。

$$M_4 > (M_3)^2 - 2$$

統計の文献では、尖度で密度の*集中具合*が測定されることが報告される場合があります。ただし、重い裾は、平均に近い分布の形状よりも尖度に大きく影響します (Kaplansky 1945; Ali 1974; Johnson, et al.1980).

サンプルの歪度および尖度は、小さなサンプルの対応するパラメーターの信頼性の低い推定量です。サンプルが非常に大きい場合は、より信頼できる推定量となります。ただし、歪度または尖度の値が大きい場合は、小さいサンプルででも注目に値することがあります。 そのような値は、正規性の仮定に基づく統計手法が不適切である場合があることを示すためです。

正規分布

特に重要な理論分布は、*正規*分布または *Gaussian* 分布です。正規分布は滑らかな対称機能で、"つり鐘型"とも言います。歪度と尖度はゼロです。正規分布は、2 つのパラメーター、平均と標準偏差によってのみ完全に指定できます。正規母集団の値の約 68%が母平均の 1 つの標準偏差内、値の約 95%が平均の 2 つの標準偏差内、約 99.7%が 3 つの標準偏差内にあります。特定の分布の種類を表す用語*正規*を使用しても、その他の分布の種類が必ずしも正規ではない、または異常であるということではありません。

多くの統計手法は、抽出中の母集団が正規に分布されるという仮定の下で設計されます。それにも関らず、実際の母集団の多くには、正規分布は含まれていません。正規性の仮定に基づいて統計手法を使用する前に統計に関する文献を参照し、その手法が非正規性に対しどれだけ敏感かを確認し、必要に応じてそのサンプルを非正規性の証拠に関してチェックする必要があります。

次の例では、平均が 50、標準偏差が 10 の正規分布からのサンプルを生成します。UNIVARIATE プロシジャは位置と正規性について検定を実行します。データは正規分布からのものであるため、正規性の検定からの *p* 値はすべて 0.15 よりも大きくなります。CHART プロシジャは、オブザベーションのヒストグラムを表示します。ヒストグラムの形状は、つり鐘型の正規密度です。

```
options nodate pageno=1 linesize=80 pagesize=52;

title '10000 Obs Sample from a Normal Distribution';
title2 'with Mean=50 and Standard Deviation=10';

data normaldat;
```

```
  drop n;
  do n=1 to 10000;
    X=10*rannor(53124)+50;
    output;
  end;
run;

proc univariate data=normaldat nextrobs=0 normal
                    mu0=50 loccount;
  var x;
run;

proc format;
  picture msd
    20='20 3*Std' (noedit)
    30='30 2*Std' (noedit)
    40='40 1*Std' (noedit)
    50='50 Mean ' (noedit)
    60='60 1*Std' (noedit)
    70='70 2*Std' (noedit)
    80='80 3*Std' (noedit)
  other=' ';
run;
options linesize=80 pagesize=42;

proc chart;
  vbar x / midpoints=20 to 80 by 2;
  format x msd.;
run;
```

10000 Obs Sample from a Normal Distribution with Mean=50 and Standard Deviation=10

The UNIVARIATE Procedure
Variable: X

Moments			
N	10000	Sum Weights	10000
Mean	50.0323744	Sum Observations	500323.744
Std Deviation	9.92013874	Variance	98.4091525
Skewness	-0.019929	Kurtosis	-0.0163755
Uncorrected SS	26016378	Corrected SS	983993.116
Coeff Variation	19.8274395	Std Error Mean	0.09920139

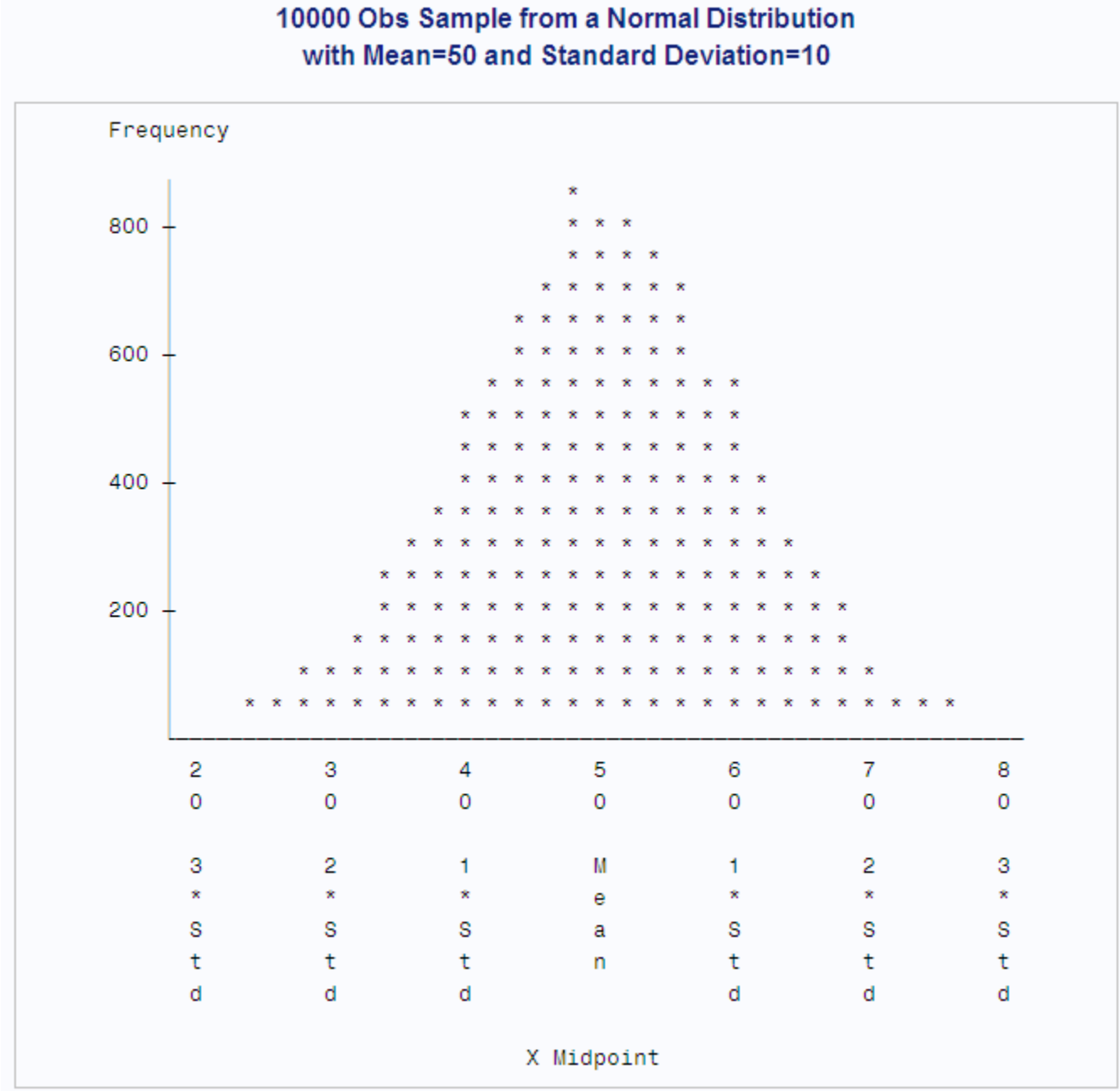
Basic Statistical Measures			
Location		Variability	
Mean	50.03237	Std Deviation	9.92014
Median	50.06492	Variance	98.40915
Mode	.	Range	76.51343
		Interquartile Range	13.28179

Tests for Location: Mu0=50				
Test	Statistic		p Value	
Student's t	t	0.32635	Pr > |t|	0.7442
Sign	M	26	Pr >= |M|	0.6101
Signed Rank	S	174063	Pr >= |S|	0.5466

Location Counts: Mu0=50.00	
Count	Value
Num Obs > Mu0	5026
Num Obs ^= Mu0	10000
Num Obs < Mu0	4974

Tests for Normality				
Test	Statistic		p Value	
Kolmogorov-Smirnov	D	0.006595	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.049963	Pr > W-Sq	>0.2500
Anderson-Darling	A-Sq	0.371151	Pr > A-Sq	>0.2500

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	90.2105
99%	72.6780
95%	66.2221
90%	62.6678
75% Q3	56.7280
50% Median	50.0649
25% Q1	43.4462
10%	37.1139
5%	33.5454
1%	26.9189
0% Min	13.6971



平均の標本分布

母集団からのサイズ *n* のサンプルを繰り返し取り出して、各サンプルの平均を計算する場合、サンプルはそれ自体に分布が含まれていることを意味します。元の母集団から取り出される可能性のあるすべてのサンプルの 平均で構成される新しい母集団について考えます。この新しい母集団の分布は、*標本分布*と呼ばれます。

元の母集団に平均 μ、標準偏差 σ が含まれている場合、平均の標本分布にも平均 μ が含まれていますが、その標準偏差は$\sigma/\sqrt{n}$ということが数学的に実証可能です。平

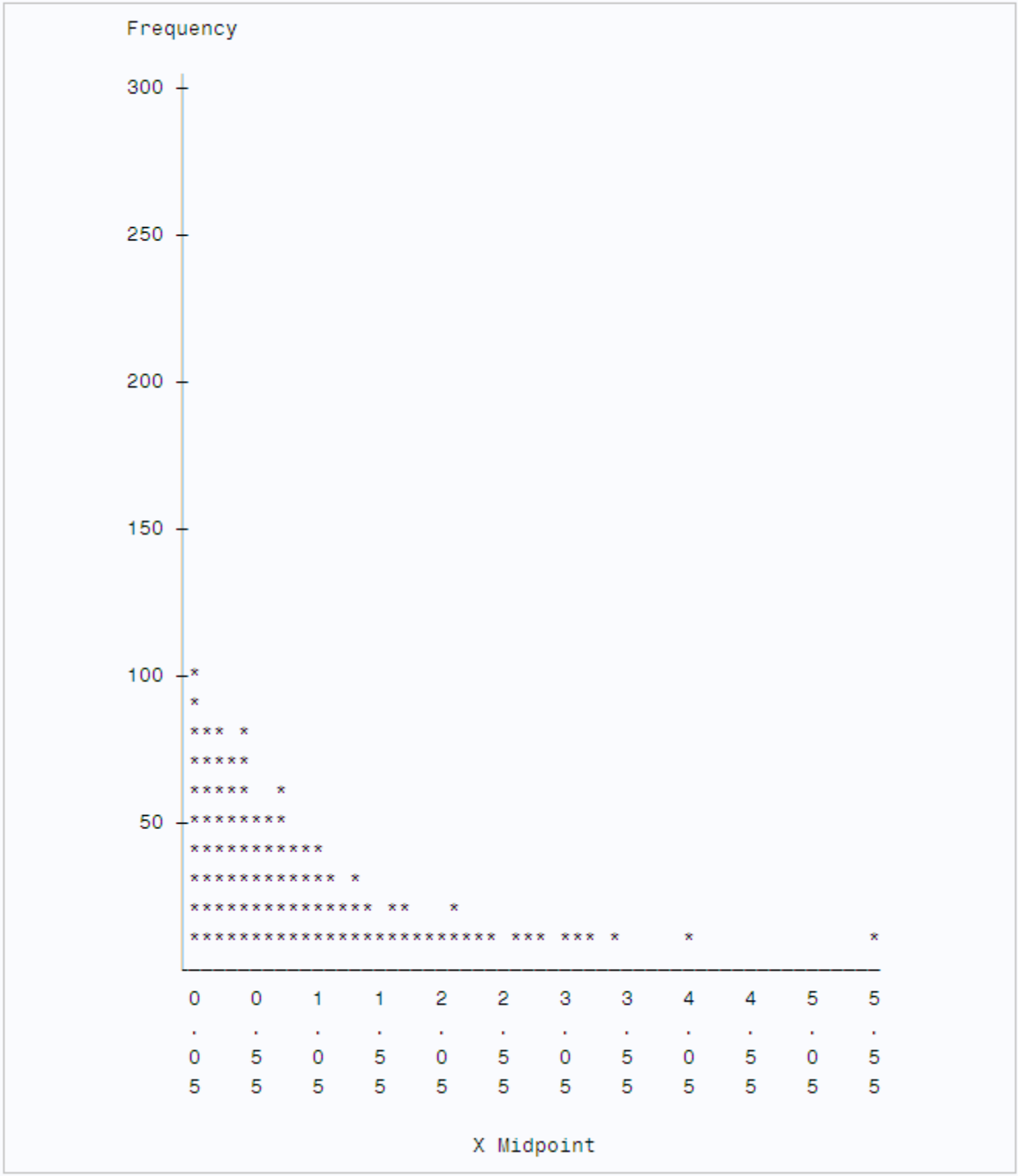
均の標本分布の標準偏差は、*平均値の標準誤差*と呼ばれます。平均の標準誤差により、母平均の推定量としてのサンプル平均の正確性が示されます。

元の母集団に正規分布が含まれている場合、平均の標本分布も正規です。元の分布が正規でなく、裾が過剰に長くない場合、平均の標本分布は大きなサンプルサイズの正規分布によって近似可能です。

次の例は、平均の標本分布がサンプルサイズが大きくなるにつれて正規分布によってどのように近似可能であるかを示す、3 つの別個のプログラムから構成されています。最初の DATA ステップでは RANEXP 関数を使用して、指数分布からの 1000 オブザベーションのサンプルを作成します。理論母平均は 1.00、サンプル平均は 1.01 (小数点以下 2 桁まで)です。母集団の標準偏差は 1.00 です。 サンプル標準差異は 1.04 です。

次の例は、非正規分布の例です。母集団の歪度は 2.00 で、これはサンプル歪度 1.97 に近いです。母集団の尖度は 6.00 ですが、サンプル尖度は 4.80 です。

```
options nodate pageno=1 linesize=80 pagesize=42;

title '1000 Observation Sample';
title2 'from an Exponential Distribution';

data expodat;
  drop n;
  do n=1 to 1000;
    X=ranexp(18746363);
    output;
  end;
run;
proc format;
   value axisfmt
    .05='0.05'
    .55='0.55'
   1.05='1.05'
   1.55='1.55'
   2.05='2.05'
   2.55='2.55'
   3.05='3.05'
   3.55='3.55'
   4.05='4.05'
   4.55='4.55'
   5.05='5.05'
   5.55='5.55'
   other=' ';
run;

proc chart data=expodat ;
  vbar x / axis=300
         midpoints=0.05 to 5.55 by .1;
  format x axisfmt.;
run;

options pagesize=64;

proc univariate data=expodat noextrobs=0 normal
              mu0=1;
  var x;
run;
```


1000 Observation Sample
from an Exponential Distribution
Frequency
300
250
200
150
100
50
0 0 1 1 2 2 3 3 4 4 5 5
. . . .
0 5 0 5 0 5 0 5 0 5 0 5
5 5 5 5 5 5 5 5 5 5 5 5
X Midpoint

1000 Observation Sample from an Exponential Distribution

The UNIVARIATE Procedure
Variable: X

Moments			
N	1000	Sum Weights	1000
Mean	1.01176214	Sum Observations	1011.76214
Std Deviation	1.04371187	Variance	1.08933447
Skewness	1.96963112	Kurtosis	4.80150594
Uncorrected SS	2111.90777	Corrected SS	1088.24514
Coeff Variation	103.15783	Std Error Mean	0.03300507

Basic Statistical Measures			
Location		Variability	
Mean	1.011762	Std Deviation	1.04371
Median	0.689502	Variance	1.08933
Mode	.	Range	6.63851
		Interquartile Range	1.06252

Tests for Location: Mu0=1				
Test	Statistic		p Value	
Student's t	t	0.356374	Pr > |t|	0.7216
Sign	M	-140	Pr >= |M|	<.0001
Signed Rank	S	-50781	Pr >= |S|	<.0001

Tests for Normality				
Test	Statistic		p Value	
Shapiro-Wilk	W	0.801498	Pr < W	<0.0001
Kolmogorov-Smirnov	D	0.166308	Pr > D	<0.0100
Cramer-von Mises	W-Sq	9.507975	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	54.5478	Pr > A-Sq	<0.0050

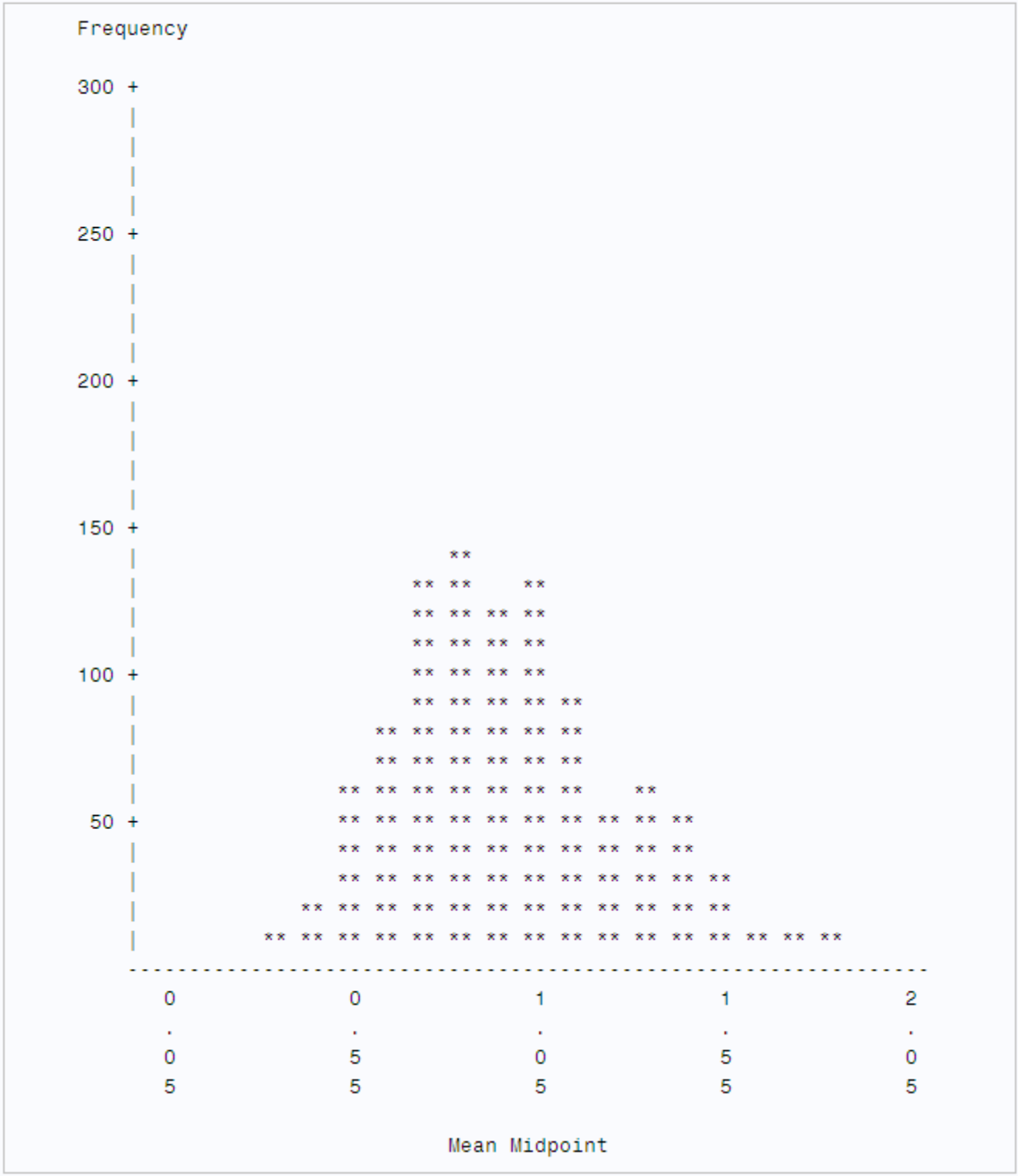
Quantiles (Definition 5)	
Quantile	Estimate
100% Max	6.63906758
99%	5.04491651
95%	3.13482318
90%	2.37803632
75% Q3	1.35733401
50% Median	0.68950221
25% Q1	0.29481436
10%	0.10219011
5%	0.05192799
1%	0.01195590
0% Min	0.00055441

次の DATA ステップは、同じ指数分布から 1000 の異なるサンプルを生成します。サンプルにはそれぞれ 10 のオブザベーションが含まれています。MEANS プロシジャは、各サンプルの平均を計算します。PROC MEANS によって作成されるデータセットで、オブザベーションはそれぞれ指数分布からの 10 のオブザベーションのサンプルの平均を表します。そのため、データセットは、指数母集団に対する平均の標本分布からのサンプルです。

PROC UNIVARIATE は、この平均値のサンプルに対する統計量を表示します。平均のサンプルの平均は.99 で、元の母平均とほぼ同じです。理論的に、標本分布の標準偏差は$\sigma/\sqrt{n} = 1.00/\sqrt{10} = .32$です。標本分布からのこのサンプルの標準偏差は.30 です。歪度(.55)と尖度(-.006)は、指数分布からの元のサンプルのものより標本分布からのサンプルのものの方がゼロに近くなります。これは、標本分布が元の指数分布よりも正規分布に近いためです。CHART プロシジャは、1000 のサンプル平均のヒストグラムを表示します。ヒストグラムの形状はつり鐘型に近い正規密度ですが、きわめて不均等です。

```
options nodate pageno=1 linesize=80 pagesize=48;

title '1000 Sample Means with 10 Obs per Sample';
title2 'Drawn from an Exponential Distribution';

data samp10;
  drop n;
  do Sample=1 to 1000;
    do n=1 to 10;
      X=ranexp(433879);
      output;
    end;
  end;

proc means data=samp10 noprint;
  output out=mean10 mean=Mean;
```

```
  var x;
  by sample;
run;

proc format;
    value axisfmt
     .05='0.05'
     .55='0.55'
    1.05='1.05'
    1.55='1.55'
    2.05='2.05'
    other=' ';
 run;

proc chart data=mean10;
  vbar mean/axis=300
          midpoints=0.05 to 2.05 by .1;
  format mean axisfmt.;
run;

options pagesize=64;
proc univariate data=mean10 nextrobs=0 normal
             mu0=1;
  var mean;
run;
```


1000 Sample Means with 10 Obs per Sample
Drawn from an Exponential Distribution
Frequency
300
250
200
150
100
50
0
0
1
1
2
0
5
0
5
0
5
5
5
5
Mean Midpoint

1000 Sample Means with 10 Obs per Sample Drawn from an Exponential Distribution

The UNIVARIATE Procedure
Variable: Mean

Moments			
N	1000	Sum Weights	1000
Mean	0.9906857	Sum Observations	990.685697
Std Deviation	0.30732649	Variance	0.09444957
Skewness	0.54575615	Kurtosis	-0.0060892
Uncorrected SS	1075.81327	Corrected SS	94.3551193
Coeff Variation	31.0215931	Std Error Mean	0.00971852

Basic Statistical Measures			
Location		Variability	
Mean	0.990686	Std Deviation	0.30733
Median	0.956152	Variance	0.09445
Mode	.	Range	1.79783
		Interquartile Range	0.41703

Tests for Location: Mu0=1				
Test	Statistic		p Value	
Student's t	t	-0.95841	Pr > |t|	0.3381
Sign	M	-53	Pr >= |M|	0.0009
Signed Rank	S	-22687	Pr >= |S|	0.0129

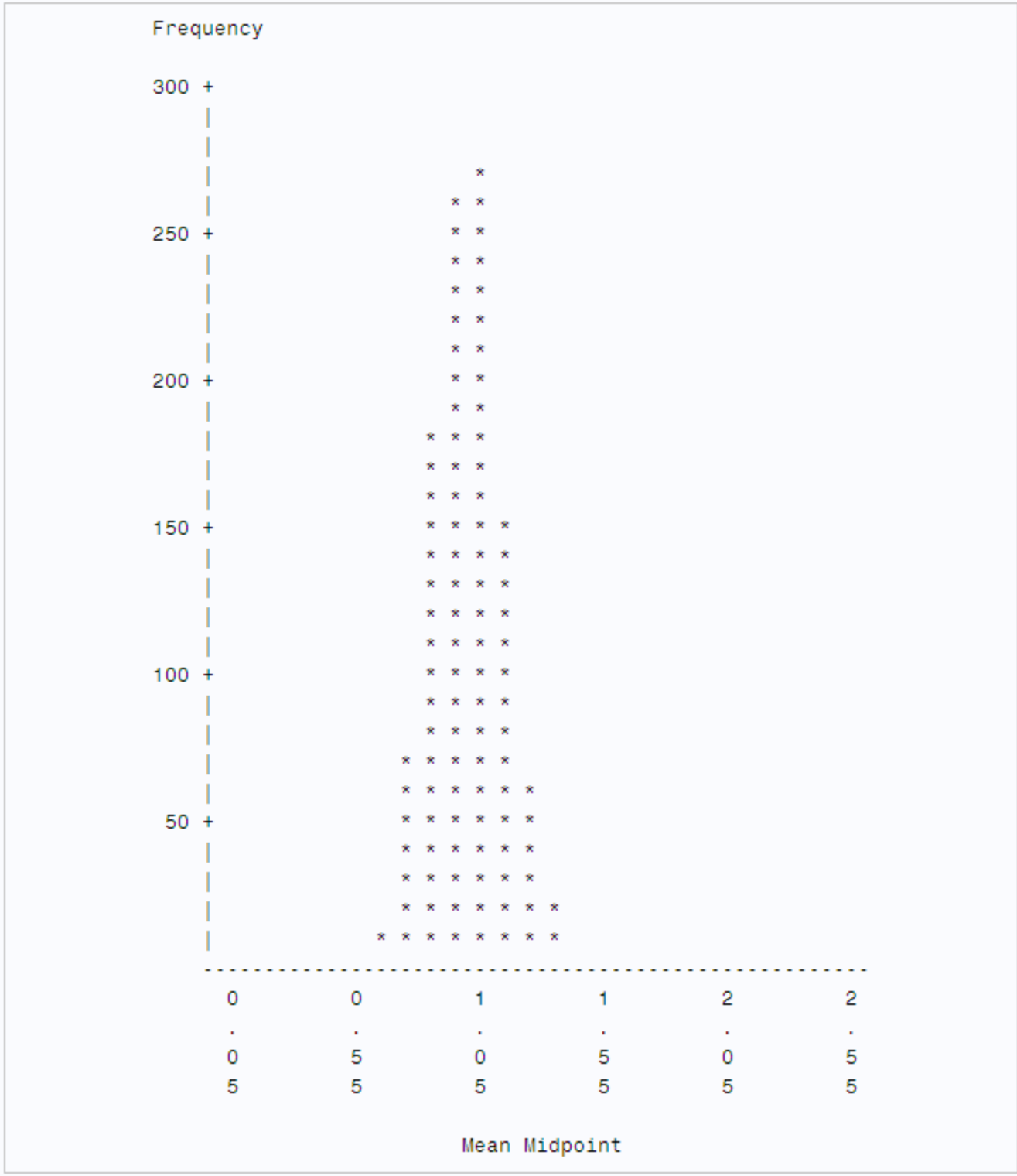
Tests for Normality				
Test	Statistic		p Value	
Shapiro-Wilk	W	0.9779	Pr < W	<0.0001
Kolmogorov-Smirnov	D	0.055498	Pr > D	<0.0100
Cramer-von Mises	W-Sq	0.953926	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	5.945023	Pr > A-Sq	<0.0050

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	2.053899
99%	1.827503
95%	1.557175
90%	1.416611
75% Q3	1.181006
50% Median	0.956152
25% Q1	0.763973
10%	0.621787
5%	0.553568
1%	0.433820
0% Min	0.256069

次の DATA ステップでは、指数分布からの各サンプルサイズが 50 に増えます。標本分布の標準偏差は前の例よりも小さくなります。これは、各サンプルサイズがより大きいためです。また、ヒストグラムと歪度から見て、標本分布は正規分布に近くなります。

```
options nodate pageno=1 linesize=80 pagesize=48;

title '1000 Sample Means with 50 Obs per Sample';
title2 'Drawn from an Exponential Distribution';

data samp50;
  drop n;
  do sample=1 to 1000;
    do n=1 to 50;
      X=ranexp(72437213);
      output;
    end;
  end;

proc means data=samp50 noprint;
  output out=mean50 mean=Mean;
  var x;
  by sample;
run;

proc format;
  value axisfmt
     .05='0.05'
     .55='0.55'
    1.05='1.05'
    1.55='1.55'
    2.05='2.05'
    2.55='2.55'
```

```
    other=' ';
run;

proc chart data=mean50;
  vbar mean / axis=300
          midpoints=0.05 to 2.55 by .1;
  format mean axisfmt.;
run;

options pagesize=64;

proc univariate data=mean50 nextrobs=0 normal
           mu0=1;
  var mean;
run;
```


1000 Sample Means with 50 Obs per Sample
Drawn from an Exponential Distribution
Frequency
300
250
200
150
100
50
0 0 1 1 2 2
. . . . . .
0 5 0 5 0 5
5 5 5 5 5 5
Mean Midpoint

1000 Sample Means with 50 Obs per Sample Drawn from an Exponential Distribution

The UNIVARIATE Procedure
Variable: Mean

Moments			
N	1000	Sum Weights	1000
Mean	0.99679697	Sum Observations	996.796973
Std Deviation	0.13815404	Variance	0.01908654
Skewness	0.19062633	Kurtosis	-0.1438604
Uncorrected SS	1012.67166	Corrected SS	19.067451
Coeff Variation	13.8597969	Std Error Mean	0.00436881

Basic Statistical Measures			
Location		Variability	
Mean	0.996797	Std Deviation	0.13815
Median	0.996023	Variance	0.01909
Mode	.	Range	0.87040
		Interquartile Range	0.18956

Tests for Location: Mu0=1				
Test	Statistic		p Value	
Student's t	t	-0.73316	Pr > |t|	0.4636
Sign	M	-13	Pr >= |M|	0.4292
Signed Rank	S	-10767	Pr >= |S|	0.2388

Tests for Normality				
Test	Statistic		p Value	
Shapiro-Wilk	W	0.996493	Pr < W	0.0247
Kolmogorov-Smirnov	D	0.023687	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.084468	Pr > W-Sq	0.1882
Anderson-Darling	A-Sq	0.66039	Pr > A-Sq	0.0877

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	1.454957
99%	1.337016
95%	1.231508
90%	1.179223
75% Q3	1.086515
50% Median	0.996023
25% Q1	0.896953
10%	0.814906
5%	0.780783
1%	0.706588
0% Min	0.584558

仮説検定

仮説の定義

これまで述べてきた統計手法の目的は、標本統計量によって母集団のパラメーターを推定することです。別のクラスの統計手法が、母集団パラメーターに関する仮説の検定、または仮説に対する証拠量の測定に使用されます。

大学の生徒のユニバースについて考えます。変数 X を、生徒の体重が同じ性別、身長、体型の人の理想体重から外れるポンド数とします。生徒の母集団が平均、低体重、または過体重であるかどうかを見つけます。そのために、9 人の生徒から無作為抽出の X 値を取り出します。結果は次の DATA ステップにあるとおりです。

```
title 'Deviations from Normal Weight';

data x;
  input X @@;
  datalines;
-7 -2 1 3 6 10 15 21 30
;
```

関心のある複数の仮説を定義できます。1 つの仮説は、生徒は平均でまさに理想の体重というものです。μ が X 値の母平均を表す場合、*帰無*仮説と呼ばれるこの仮説をH_0: $\mu = 0$と表すことができます。*対立*仮説と呼ばれるその他 2 つの仮説は、生徒が平均で低体重であり、H_1: $\mu < 0$、学生が平均で過体重である、H_2: $\mu > 0$。

帰無仮説がそう呼ばれるのは、多くの場合、“影響なし”または“相違なし”という仮定に対応しているためです。ただし、この解釈はすべての検定問題に適しているわけではありません。帰無仮説は、統計的な証拠によって倒される可能性があるかかしに似ています。かかしが倒れる方法に従って、対立仮説のどちらかを決めます。

この問題にアプローチするためのシンプルな方法は、サンプル平均$\bar{x}$を見て、次のルールに従って 3 つの仮説から決めることです。

- $\bar{x} < 0$の場合、$H_1: \mu < 0$で決めます。
- $\bar{x} = 0$の場合、$H_0: \mu = 0$で決めます。
- $\bar{x} > 0$の場合、$H_2: \mu > 0$で決めます。

このアプローチに関する問題は、正しくない決定を行う可能性が高いことです。H_0 が true の場合、ほぼ確実に誤った決定を行います。これは、$\bar{x}$がゼロになる可能性がほとんどないためです。μ がわずかにゼロ未満で、H_1 が true の場合、$\bar{x}$が繰り返されるサンプリングでゼロより大きい可能性は約 50%です。そのため H_2 を誤って選択する可能性も約 50%になります。したがって、$\bar{x}$がゼロに近い場合は、選択を誤る可能性が高くなります。そのような場合は自信のある決定を行うために十分な証拠がないため、最善の対応としてはさらに多くの証拠が得られるまで判断を留保することです。

問題は、自信を持って決定できるように、$\bar{x}$がゼロからどのくらい離れているかということです。答えは、$\bar{x}$の標本分布について考えるとわかります。X の分布がほぼ正規分布である場合、$\bar{x}$の分布はほとんど正規標本分布です。$\bar{x}$の標本分布の平均は μ です。X の標準偏差が 12 であるとします。9 つのオブザベーションのサンプルに対する$\bar{x}$の標準誤差は、$\sigma/\sqrt{n} = 12/\sqrt{9} = 4$です。

正規分布からの値の約 95%が平均の 2 つの標準偏差内にあることがわかっています。このため、9 つの X 値の可能なサンプルの約 95%に、$\bar{x}$から$0 - 2(4)$まで$0 + 2(4)$、または−8 から 8 までのサンプル平均が含まれています。次の決定ルールで間違いをする可能性について考えます。

- $\bar{x} < -8$の場合、$H_1: \mu < 0$で決めます。
- $-8 \le \bar{x} \le 8$の場合、判断を留保します。
- $\bar{x} > 8$の場合、$H_2: \mu > 0$で決めます。

H_0 が true の場合、可能なサンプルの約 95%は$\bar{x}$が*臨界値* -8から 8 までになるため、判断を留保します。この場合、統計的証拠は、かかしを倒すほど強くありません。サンプルの別の 5%で間違いをします。サンプルの 2.5%で誤って H_1 を選択し、2.5%で誤って H_2 を選択します。

間違いをする可能性を管理するために支払う代償は、帰無仮説の却下に十分な統計的証拠がない場合に判断を留保する必要性です。

有意性と検定力

帰無仮説を却下する可能性は、true の場合は統計的検定の*第 1 種の過誤率*と呼ばれ、通常はαと表されます。この例では、-8未満、または 8 より大きい$\bar{x}$値は、5%水準で*統計的に有意*と呼ばれます。異なる臨界値を選択して、第 1 種の過誤率をニーズに

従って調整できます。たとえば、臨界値が−4 および 4 の場合は有意水準は約 32%となり、−12 および 12 の場合は、第 1 種の過誤率は約 0.3%となります。

決定ルールは*両側検定*です。これは、対立仮説では帰無仮説で指定された値よりも小さい、または大きい母平均が可能なためです。生徒が平均で過体重である可能性については、片側検定を使用できます。

- $\bar{x} \leq 8$の場合、判断を留保します。
- $\bar{x} > 8$の場合、$H_2: \mu > 0$で決めます。

この片側検定では、第 1 種の過誤率は 2.5%で、両側検定の半分です。

帰無仮説を却下する可能性は、false の場合は*検定力*と呼ばれ、通常は$1 - \beta$と表されます。βは*第 2 種の過誤率*と呼ばれ、false の帰無仮説を却下しない可能性です。検定力は、パラメーターの true 値によって異なります。例では、母平均を 4 とします。H_2 を検出するための検定力は、8 よりも大きいサンプル平均を取得する確率です。臨界値 8 は、母平均 4 よりも高い 1 つの標準誤差です。正規分布からの平均より大きい標準偏差を少なくとも 1 つ取得する可能性は、約 16%です。そのため、対立仮説 H_2 を検出するための検定力は約 16%です。母平均が 8 の場合、H_2 の検定力が 50%になります。 母平均 12 により検定力が約 84%になります。

第 1 種の過誤率が低くなると、誤った決定を行う可能性が低くなりますが、判断を留保する必要がある可能性は高くなります。第 1 種の過誤率を選択する場合、対象となるさまざまな代替に対する結果の検定力を考慮する必要があります。

スチューデントの t 分布

実際には、σ に基づいた臨界値を使用する決定ルールを通常は使用できません。σ の値が不明なためです。ただし、*s* を σ の推定として使用できます。次の統計量について考えます。

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

この *t* 統計量は、サンプル平均と、平均の推定標準誤差で割られる仮説平均μ_0の差です。

帰無仮説が true で、母集団が正規分布されている場合、*t* 統計量には*スチューデントの t 分布*($n - 1$自由度を含む)と呼ばれるものが存在します。この分布は正規分布と非常に似ていますが、スチューデントの *t* 分布の裾はさらに重くなっています。サンプルサイズが大きくなるにつれて、サンプルの標準偏差は母集団の標準偏差のより適した推定量となり、*t* 分布は正規分布に近づきます。

決定ルールは *t* 統計量を基準にすることができます。

- $t < -2.3$の場合、$H_1: \mu < 0$で決めます。
- $-2.3 \leq t \leq 2.3$の場合、判断を留保します。
- $t > 2.3$の場合、$H_0: \mu > 0$で決めます。

値 2.3 は、スチューデントの *t* 分布の表から取得され、8 (つまり$9 - 1 = 8$)自由度に対し第 1 種の過誤率は 5%となります。一般的な統計テキストのほとんどには、スチ

ューデントの *t* 分布の表が含まれています。統計テキストがない場合は、DATA ステップと TINV 関数を使用して、*t* 分布からの値を印刷できます。

デフォルトで、PROC UNIVARIATE は *t* 統計量を$\mu_0 = 0$という帰無仮説に対し関連する統計量とともに計算します。PROC ステートメントで MU0=オプションを使用して、帰無仮説に対し別の値を指定します。

この例では、9 つのオブザベーションから構成される標準体重からの偏差のデータが使用されます。まず、PROC MEANS は *t* 統計量を$\mu = 0$という帰無仮説に対し計算します。次に、DATA ステップの TINV 関数は両側検定(有意性水準 5%、自由度 8)に対するスチューデントの *t* 分布の値を計算します。

```
data devnorm;
  title 'Deviations from Normal Weight';
  input X @@;
  datalines;
-7 -2 1 3 6 10 15 21 30
;

proc means data=devnorm maxdec=3 n mean
        std stderr t probt;
run;

title 'Student''s t Critical Value';

data _null_;
  file print;
  t=tinv(.975,8);
  put t 5.3;
run;
```

Deviations from Normal Weight

The MEANS Procedure

Analysis Variable : X					
N	Mean	Std Dev	Std Error	t Value	Pr > |t|
9	8.556	11.759	3.920	2.18	0.0606

Student's t Critical Value

```
2.306
```

```
        Deviations from Normal Weight              1
                The MEANS Procedure

                Analysis Variable : X

N        Mean      Std Dev    Std Error  t Value  Pr > |t|
-----------------------------------------------------------
9       8.556       11.759        3.920     2.18    0.0606
-----------------------------------------------------------
```

```
              Student's t Critical Value              2
2.306
```

現在の例では、*t* 統計量の値は 2.18 です、これは、臨界 *t* 値 2.3 (有意水準 5%、自由度 8)よりも小さいです。そのため、有意水準 5%では、判断を留保する必要があります。有意水準 10%の使用を選択した場合、*t* 分布の臨界値は 1.86 となり、帰無仮説を却下できます。ただし、サンプルサイズが非常に小さいため、結論の妥当性は、母集団の分布がどれだけ正規分布に近いかということによって大きく異なります。

確率値(p 値)

統計的検定の結果を報告する別の方法としては、*確率値*または *p 値*の計算があります。*p* 値は、繰り返されるサンプリングで、対立仮説によって指定される方向の実際に観測される値まで統計量を取得する確率を示します。*t* 統計量に対する両側の *p* 値は、絶対 *t* 値(観測された絶対 *t* 値よりも大きい)を取得する確率です。*T 統計量に対する片側 p 値(代替仮説*統計量に対する片側 *p* 値(代替仮説$\mu > \mu_0$)は、観測された *t* 値よりも大きい *t* 値を取得する確率です。*p* 値が計算されると、*p* 値と必要な有意水準を比較して、仮説検定を実行できます。*p* 値が検定の第 1 種の過誤率に満たない、または等しい場合、帰無仮説は却下できます。両側 *p* 値(PROC MEANS 出力で Pr > |t| とラベル付け)が.0606 であるため、帰無仮説は有意水準が 10%の場合は却下できますが、有意水準が 5%の場合は却下できません。

p 値は、帰無仮説に対する証拠の強度の指標です。*p* 値が小さくなると、帰無仮説を却下するための証拠の強度が増します。

注: 詳細については、統計の入門書(Mendenhall and Beaver (1998)、Ott and Mendenhall (1994)、Snedecor and Cochran (1989)など)を参照してください。

参考文献

Ali, M. M. 1974. “Stochastic Ordering and Kurtosis Measure.” *Journal of the American Statistical Association* 69: 543–545.

Johnson, M. E., G. L. Tietjen, and R. J. Beckman. 1980. “A New Family of Probability Distributions with Applications to Monte Carlo Studies.” *Journal of the American Statistical Association* 75: 276-279.

Kaplansky, I. 1945. “A Common Error Concerning Kurtosis.” *Journal of the American Statistical Association,* 40: 259-263.

Mendenhall, W. and R. Beaver. 1998. *Introduction to Probability and Statistics*. 10 番目版 Belmont, CA: Wadsworth Publishing Company.

Ott, R. and W. Mendenhall. 1994. *Understanding Statistics*. 6 番目版 North Scituate, MA: Duxbury Press.

Schlotzhauer, S. D. and R. C. Littell. 1997. *SAS System for Elementary Statistical Analysis*. Second 版 Cary, NC: SAS Institute Inc.

Snedecor, G. W. and W. C. Cochran. 1989. *Statistical Methods*. 8 番目版 Ames, IA: Iowa State University Press.

付録 2

Base SAS プロシジャの生データと DATA ステップ

Base SAS プロシジャの生データと DATA ステップの概要

次の生データ例と DATA ステップ例は、Base SAS プロシジャでの使用を示しています。

このドキュメント内のプログラム例では、通常、使用するデータセットの作成方法が説明されています。一部のデータのみが示されている例もあります。これらの例については、付録に完全なデータが示されています。

CARSURVEY

```
data carsurvey;
  input Rater Age Progressa Remark Jupiter Dynamo;
  datalines;
1  38  94  98  84  80
2  49  96  84  80  77
3  16  64  78  76  73
4  27  89  73  90  92
5  50  93  79  84  34
6  57  92  89  75  89
7  21  88  90  89  91
8  39  88  87  76  64
9  26  77  94  93  47
10 17  68  72  85  79
11 38  94  93  84  70
12 29  78  97  74  33
13 41  89  83  75  82
14 37  54  98  70  83
15 52  92  85  88  78
16 23  85  89  89  95
17 61  92  88  77  85
18 24  87  88  88  87
19 18  54  50  62  74
20 62  90  91  90  86
21 49  94  98  84  80
22 16  96  84  80  77
23 27  64  78  76  73
24 50  89  73  90  92
25 57  93  79  84  34
26 21  92  86  75  93
27 39  88  97  89  91
28 26  88  87  76  64
29 17  77  94  93  47
30 38  68  72  85  79
31 29  94  93  84  70
32 41  78  97  74  33
33 37  89  83  75  82
34 52  54  98  70  83
35 23  92  85  88  78
36 61  85  93  89  66
37 24  92  88  77  85
38 18  87  88  88  87
39 62  54  50  62  74
40 38  90  91  90  86
41 57  94  98  84  80
42 16  96  84  80  77
43 19  64  78  76  73
```

```
44 59 89 73 90 92
45 57 93 79 84 34
46 21 92 86 75 90
47 39 88 97 89 91
48 26 88 87 76 64
49 17 77 94 93 47
50 56 68 72 85 79
51 29 94 93 84 70
52 41 78 97 74 33
53 37 89 83 75 82
54 52 54 98 70 83
55 17 92 85 88 78
56 61 85 93 89 66
57 24 92 85 77 85
58 18 87 88 88 87
59 62 54 50 62 74
60 38 90 91 90 86
61 38 94 98 84 80
62 49 96 84 80 77
63 16 64 78 76 73
64 27 89 73 90 92
65 50 93 79 84 34
66 57 92 89 75 89
67 21 88 93 89 91
68 39 88 87 76 64
69 26 77 94 93 47
70 17 68 72 85 79
71 38 94 93 84 70
72 29 78 97 74 33
73 41 89 83 75 82
74 37 54 98 70 83
75 52 92 85 88 78
76 23 85 93 89 88
77 61 92 88 77 85
78 24 87 88 88 91
79 18 54 50 62 74
80 62 90 91 90 86
;
```

CENSUS

```
data census;
  input Density CrimeRate State $ 14-27 PostalCode $ 29-30;
  datalines;
263.3 4575.3 Ohio          OH
62.1 7017.1  Washington    WA
103.4 5161.9 South Carolina SC
53.4 3438.6  Mississippi   MS
180.0 8503.2 Florida       FL
80.8 2190.7  West Virginia WV
428.7 5477.6 Maryland      MD
```

```
71.2 4707.5  Missouri       MO
43.9 4245.2  Arkansas       AR
7.3 6371.4   Nevada         NV
264.3 3163.2 Pennsylvania   PA
11.5 4156.3  Idaho          ID
44.1 6025.6  Oklahoma       OK
51.2 4615.8  Minnesota      MN
55.2 4271.2  Vermont        VT
27.4 6969.9  Oregon         OR
205.3 5416.5 Illinois       IL
94.1 5792.0  Georgia        GA
9.1 2678.0   South Dakota   SD
9.4 2833.0   North Dakota   ND
102.4 3371.7 New Hampshire  NH
54.3 7722.4  Texas          TX
76.6 4451.4  Alabama        AL
307.6 4938.8 Delaware       DE
151.4 6506.4 California     CA
111.6 4665.6 Tennessee      TN
120.4 4649.9 North Carolina NC
;
```

CHARITY

```
data Charity;
  input School $ 1-7 Year 9-12 Name $ 14-20 MoneyRaised 22-26
      HoursVolunteered 28-29;
  datalines;
Monroe  2016 Allison 31.65 19
Monroe  2016 Barry   23.76 16
Monroe  2016 Candace 21.11  5
Monroe  2016 Danny    6.89 23
Monroe  2016 Edward  53.76 31
Monroe  2016 Fiona   48.55 13
Monroe  2016 Gert    24.00 16
Monroe  2016 Harold  27.55 17
Monroe  2016 Ima     15.98  9
Monroe  2016 Jack    20.00 23
Monroe  2016 Katie   22.11  2
Monroe  2016 Lisa    18.34 17
Monroe  2016 Tonya   55.16 40
Monroe  2016 Max     26.77 34
Monroe  2016 Ned     28.43 22
Monroe  2016 Opal    32.66 14
Monroe  2017 Patsy   18.33 18
Monroe  2017 Quentin 16.89 15
Monroe  2017 Randall 12.98 17
Monroe  2017 Sam     15.88  5
Monroe  2017 Tyra    21.88 23
Monroe  2017 Myrtle  47.33 26
Monroe  2017 Frank   41.11 22
```

Monroe 2017 Cameron 65.44 14
Monroe 2017 Vern 17.89 11
Monroe 2017 Wendell 23.00 10
Monroe 2017 Bob 26.88 6
Monroe 2017 Leah 28.99 23
Monroe 2018 Becky 30.33 26
Monroe 2018 Sally 35.75 27
Monroe 2018 Edgar 27.11 12
Monroe 2018 Dawson 17.24 16
Monroe 2018 Lou 5.12 16
Monroe 2018 Damien 18.74 17
Monroe 2018 Mona 27.43 7
Monroe 2018 Della 56.78 15
Monroe 2018 Monique 29.88 19
Monroe 2018 Carl 31.12 25
Monroe 2018 Reba 35.16 22
Monroe 2018 Dax 27.65 23
Monroe 2018 Gary 23.11 15
Monroe 2018 Suzie 26.65 11
Monroe 2018 Benito 47.44 18
Monroe 2018 Thomas 21.99 23
Monroe 2018 Annie 24.99 27
Monroe 2018 Paul 27.98 22
Monroe 2018 Alex 24.00 16
Monroe 2018 Lauren 15.00 17
Monroe 2018 Julia 12.98 15
Monroe 2018 Keith 11.89 19
Monroe 2018 Jackie 26.88 22
Monroe 2018 Pablo 13.98 28
Monroe 2018 L.T. 56.87 33
Monroe 2018 Willard 78.65 24
Monroe 2018 Kathy 32.88 11
Monroe 2018 Abby 35.88 10
Kennedy 2016 Arturo 34.98 14
Kennedy 2016 Grace 27.55 25
Kennedy 2016 Winston 23.88 22
Kennedy 2016 Vince 12.88 21
Kennedy 2016 Claude 15.62 5
Kennedy 2016 Mary 28.99 34
Kennedy 2016 Abner 25.89 22
Kennedy 2016 Jay 35.89 35
Kennedy 2016 Alicia 28.77 26
Kennedy 2016 Freddy 29.00 27
Kennedy 2016 Eloise 31.67 25
Kennedy 2016 Jenny 43.89 22
Kennedy 2016 Thelma 52.63 21
Kennedy 2016 Tina 19.67 21
Kennedy 2016 Eric 24.89 12
Kennedy 2016 Bubba 37.88 12
Kennedy 2016 G.L. 25.89 21
Kennedy 2016 Bert 28.89 21
Kennedy 2017 Clay 26.44 21
Kennedy 2017 Leeann 27.17 17
Kennedy 2017 Georgia 38.90 11
Kennedy 2017 Bill 42.23 25
Kennedy 2017 Holly 18.67 27

```
Kennedy 2017 Benny   19.09 25
Kennedy 2017 Cammie  28.77 28
Kennedy 2017 Amy     27.08 31
Kennedy 2017 Doris   22.22 24
Kennedy 2017 Robbie  19.80 24
Kennedy 2017 Ted     27.07 25
Kennedy 2017 Sarah   24.44 12
Kennedy 2017 Megan   28.89 11
Kennedy 2017 Jeff    31.11 12
Kennedy 2017 Taz     30.55 11
Kennedy 2017 George  27.56 11
Kennedy 2017 Heather 38.67 15
Kennedy 2018 Nancy   29.90 26
Kennedy 2018 Rusty   30.55 28
Kennedy 2018 Mimi    37.67 22
Kennedy 2018 J.C.    23.33 27
Kennedy 2018 Clark   27.90 25
Kennedy 2018 Rudy    27.78 23
Kennedy 2018 Samuel  34.44 18
Kennedy 2018 Forrest 28.89 26
Kennedy 2018 Luther  72.22 24
Kennedy 2018 Trey     6.78 18
Kennedy 2018 Albert  23.33 19
Kennedy 2018 Che-Min 26.66 33
Kennedy 2018 Preston 32.22 23
Kennedy 2018 Larry   40.00 26
Kennedy 2018 Anton   35.99 28
Kennedy 2018 Sid     27.45 25
Kennedy 2018 Will    28.88 21
Kennedy 2018 Morty   34.44 25
;
```

CONTROL ライブラリ

CONTROL ライブラリの内容

DATASETS プロシジャセクションで使用する CONTROL ライブラリの内容は次のとおりです。

```
                Directory

Libref         CONTROL
Engine         V9
Physical Name  \myfiles\control
File Name      \myfiles\control


     Member   Obs, Entries                File
```

```
 # Name     Type     or Indexes   Vars  Label            Size  Last Modified

 1 A1       CATALOG      23                        62464 04Jan18:14:20:12
 2 A2       CATALOG       1                        17408 04Jan18:14:20:12
 3 ALL      DATA         23       17                13312 04Jan18:14:20:12
 4 BODYFAT  DATA          1        2                 5120 04Jan18:14:20:12
 5 CONFOUND DATA          8        4                 5120 04Jan18:14:20:12
 6 CORONARY DATA         39        4                 5120 04Jan18:14:20:12
 7 DRUG1    DATA          6        2  JAN18 Data     5120 04Jan18:14:20:12
 8 DRUG2    DATA         13        2  MAY18 Data     5120 04Jan18:14:20:12
 9 DRUG3    DATA         11        2  JUL17 Data     5120 04Jan18:14:20:12
10 DRUG4    DATA          7        2  JAN17 Data     5120 04Jan18:14:20:12
11 DRUG5    DATA          1        2  JUL17 Data     5120 04Jan18:14:20:12
12 ETEST1   CATALOG       1                        17408 04Jan18:14:20:16
13 ETEST2   CATALOG       1                        17408 04Jan18:14:20:16
14 ETEST3   CATALOG       1                        17408 04Jan18:14:20:16
15 ETEST4   CATALOG       1                        17408 04Jan18:14:20:16
16 ETEST5   CATALOG       1                        17408 04Jan18:14:20:16
17 ETESTS   CATALOG       1                        17408 04Jan18:14:20:16
18 FORMATS  CATALOG       6                        17408 04Jan18:14:20:16
19 GROUP    DATA        148       11                25600 04Jan18:14:20:16
20 MLSCL    DATA         32        4  Multiple       5120 04Jan18:14:20:16
                                    Sclerosis Data
21 NAMES    DATA          7        4                 5120 04Jan11:14:20:16
22 OXYGEN   DATA         31        7                 9216 04Jan18:14:20:16
23 PERSONL  DATA        148       11                25600 04Jan18:14:20:16
24 PHARM    DATA          6        3  Sugar Study    5120 04Jan18:14:20:16
25 POINTS   DATA          6        6                 5120 04Jan18:14:20:16
26 PRENAT   DATA        149        6                17408 04Jan18:14:20:16
27 RESULTS  DATA         10        5                 5120 04Jan18:14:20:16
28 SLEEP    DATA        108        6                 9216 04Jan18:14:20:16
29 SYNDROME DATA         46        8                 9216 04Jan18:14:20:16
30 TENSION  DATA          4        3                 5120 04Jan18:14:20:16
31 TEST2    DATA         15        5                 5120 04Jan18:14:20:16
32 TRAIN    DATA          7        2                 5120 04Jan18:14:20:16
33 VISION   DATA         16        3                 5120 04Jan18:14:20:16
34 WEIGHT   DATA         83       13  California     13312 04Jan18:14:20:16
                                    Results
35 WGHT     DATA         83       13  California     13312 04Jan18:14:20:16
                                    Results
16
```

CONTROL.ALL

CONTROL ライブラリのすべてのデータファイルの生データと DATA ステップは次のとおりです。

```
data control.all;
  input FMTNAME $8. START $9. END $8. LABEL $20. MIN best4. MAX best4.
      DEFAULT best4. LENGTH best4. FUZZ best8. PREFIX $2. MULT best8.
      FILL $1. NOEDIT best4.  TYPE $2. SEXCL $2. EEXCL $2. HLO $7. ;
  label FMTNAME='Format name'
       START='Starting value for format'
```

```
        END='Ending value for format'
        LABEL='Format value label'
        MIN='Minimum length'
        MAX='Maximum length'
        DEFAULT='Default length'
        LENGTH='Format length'
        FUZZ='Fuzz value'
        PREFIX='Prefix characters'
        MULT='Multiplier'
        FILL='Fill character'
        NOEDIT='Is picture string noedit?'
        TYPE='Type of format'
        SEXCL='Start exclusion'
        EEXCL='End exclusion'
        HLO='Additional information';
    datalines;
BENEFIT      LOW    7304        WORDDATE20.   1 40  20  20   1E-12     0.00   0 N N N    LF
BENEFIT     7305    HIGH  ** Not Eligible **   1 40  20  20   1E-12     0.00   0 N N N
DOLLARS      LOW    HIGH          000,000   1 40   7   7   1E-12 $    1.96   0 P N N    LH
NOZEROS      LOW    0.01            999   1 40   5   5   1E-12 . 1000.00   0 P N Y    L
NOZEROS     0.01    0.1              99   1 40   5   5   1E-12 .  100.00   0 P N Y
NOZEROS      0.1     1             0.000   1 40   5   5   1E-12 . 1000.00   0 P N Y
NOZEROS        1    HIGH           0.000   1 40   5   5   1E-12   1000.00   0 P N N    H
BRIT         BR1     BR1         Birmingham   1 40  14  14      0     0.00   0 C N N
BRIT         BR2     BR2          Plymouth   1 40  14  14      0     0.00   0 C N N
BRIT         BR3     BR3             York   1 40  14  14      0     0.00   0 C N N
BRIT     *OTHER****OTHER*     INCORRECT CODE   1 40  14  14      0     0.00   0 C N N    0
SKILL         A      D~           Test A   1 40   6   6      0     0.00   0 C N N
SKILL         E      M~           Test B   1 40   6   6      0     0.00   0 C N N
SKILL         N      Z~           Test C   1 40   6   6      0     0.00   0 C N N
SKILL         a      d~           Test A   1 40   6   6      0     0.00   0 C N N
SKILL         e      m~           Test B   1 40   6   6      0     0.00   0 C N N
SKILL         n      z~           Test C   1 40   6   6      0     0.00   0 C N N
EVAL          0      4           _SAME_   1 40   1   1      0     0.00   0 I N N    I
EVAL          C      C              1   1 40   1   1      0     0.00   0 I N N
EVAL          E      E              2   1 40   1   1      0     0.00   0 I N N
EVAL          N      N              0   1 40   1   1      0     0.00   0 I N N
EVAL          O      O              4   1 40   1   1      0     0.00   0 I N N
EVAL          S      S              3   1 40   1   1      0     0.00   0 I N N
;
run;
```

CONTROL.BODYFAT

```
data control.bodyfat;
   input NAME $  AGE $;
   datalines;
jeff    44
;
run;
```

CONTROL.CONFOUND

```
data control.confound;
   input SMOKING $8.  STATUS $8.  CANCER $8.  WT;
   datalines;
Yes     Single  Yes     34
Yes     Single  No      120
Yes     Married Yes     7
Yes     Married No      30
Yes     Single  Yes     2
Yes     Single  No      30
Yes     Married Yes     6
Yes     Married No      145
;
run;
```

CONTROL.CORONARY

```
data control.coronary;
   input SEX  ECG AGE CA;
   datalines;
0      0      28     0
0      0      34     0
0      0      38     0
0      0      41     0
0      0      44     0
0      0      45     1
0      0      46     0
0      0      47     0
0      0      50     0
0      0      51     0
0      0      51     0
0      0      53     0
0      0      55     1
0      0      59     0
0      0      60     1
0      0      32     1
0      0      33     0
0      0      35     0
0      0      39     0
0      0      40     0
0      0      46     0
0      0      48     1
0      0      49     0
0      0      49     0
0      0      52     0
0      0      53     1
0      0      54     1
```

```
0     0     55    0
0     0     57    1
0     0     46    1
0     0     48    0
0     0     57    1
0     0     60    1
0     0     30    0
0     0     34    0
0     0     36    1
0     0     38    1
0     0     39    0
0     0     42    0
;
run;
```

CONTROL.DRUG1

```
data control.drug1 (label='JAN2018 DATA');
   input CHAR $8.  NUM;
   datalines;
junk   0
junk   0
junk   0
junk   0
junk   0
junk   0
;
run;
```

CONTROL.DRUG2

```
 data control.drug2 (label='MAY2018 DATA');
   input CHAR $8.  NUM;
   datalines;
junk   0
junk   0
junk   0
junk   0
junk   0
junk   0
junk   0
junk   0
junk   0
junk   0
junk   0
junk   0
;
run;
```

CONTROL.DRUG3

```
data control.drug3 (label='JUL2017 DATA');
   input CHAR $8.  NUM;
   datalines;
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
;
run;
```

CONTROL.DRUG4

```
data control.drug4 (label='JAN2017 DATA');
   input CHAR $8.  NUM;
   datalines;
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
junk    0
;
run;
```

CONTROL.DRUG5

```
data control.drug5 (label='JUL2017 DATA');
   input CHAR $8.  NUM;
   datalines;
junk    0
;
run;
```

CONTROL.GROUP

```
data control.group;
   input IDNUM $ 1-4 LNAME $ 5-18 FNAME $ 19-32 CITY $ 33-47 STATE $
         48-50 SEX $ 52-53 JOBCODE $ 54-57 SALARY comma8. BIRTH
         HIRED date9. HPHONE $ 86-97;
   format salary comma8.;
   format hired date9.;
   informat hired date9.;
   datalines;
1919 ADAMS       GERALD      STAMFORD      CT M TA2   34,377    11212 04JUN2007 203/781-1255
1653 AHMAD       AZEEM       BRIDGEPORT    CT F ME2   35,109     9054 09AUG2010 203/675-7715
1400 ALVAREZ     GLORIA      NEW YORK      NY M ME1   29,770    10170 16OCT2010 212/586-0808
1350 ARTHUR      BARBARA     NEW YORK      NY F FA3   32,887     9374 29JUL2010 718/383-1549
1401 AVERY       JERRY       PATERSON      NJ M TA3   38,823     3999 17NOV2005 201/732-8787
1499 BAREFOOT    JOSEPH      PRINCETON     NJ M ME3   43,026     5229 07JUN2010 201/812-5665
1101 BASQUEZ     RICHARDO    NEW YORK      NY M SCP   18,724     8192 01OCT2010 212/586-8060
1333 BEAULIEU    ARMANDO     NEW YORK      NY M TA2   32,616     7759 10FEB2011 718/384-2849
1402 BLALOCK     RALPH       NEW YORK      NY M TA2   32,615     8417 02DEC2010 718/384-2849
1479 BOSTIC      MARIE       NEW YORK      NY F TA3   38,786    10583 05OCT2007 718/384-8816
1403 BOWDEN      EARL        BRIDGEPORT    CT M ME1   28,073    10620 21DEC2016 203/675-3434
1739 BOYCE       JONATHAN    NEW YORK      NY M PT1   66,518     9125 27JAN2011 212/587-1247
1658 BRADLEY     JEREMY      NEW YORK      NY M SCP   17,944     9959 29FEB2012 212/587-3622
1428 BRADY       CHRISTINE   STAMFORD      CT F PT1   68,768     7399 16NOV2011 203/781-1212
1782 BROWN       JASON       STAMFORD      CT M ME2   35,346     7643 22FEB2012 203/781-0019
1244 BRYANT      LEONARD     NEW YORK      NY M ME2   36,926     8643 17JAN2008 718/383-3334
1383 BURNETTE    THOMAS      NEW YORK      NY M BCK   25,824    10251 20OCT2009 718/384-3569
1574 CAHILL      MARSHALL    NEW YORK      NY M FA2   28,573     7422 20DEC2012 718/383-2338
1789 CANALES     VIVIANA     NEW YORK      NY M SCP   18,327     6232 11APR2008 212/587-9000
1404 CARTER      DONALD      NEW YORK      NY M PT2   91,377     4803 01JAN2010 718/384-2946
1437 CARTER      DOROTHY     BRIDGEPORT    CT F FA3   33,105     7568 31AUG2004 203/675-4117
1639 CARTER      KAREN       STAMFORD      CT F TA3   40,261     6386 28JAN2014 203/781-8839
1269 CASTON      FRANKLIN    STAMFORD      CT M NA1   41,691     4506 28NOV2012 203/781-3335
1065 CHAPMAN     NEIL        NEW YORK      NY M ME2   35,091     1486 07JAN2007 718/384-5618
1876 CHIN        JACK        NEW YORK      NY M TA3   39,676     6714 27APR2015 212/588-5634
1037 CHOW        JANE        STAMFORD      CT F TA1   28,559     8866 13SEP2012 203/781-8868
1129 COOK        BRENDA      NEW YORK      NY F ME2   34,930     8012 17AUG2011 718/383-2313
1988 COOPER      ANTHONY     NEW YORK      NY M FA3   32,218     7273 18SEP2004 212/587-1228
1405 DAVIDSON    JASON       PATERSON      NJ M SCP   18,057     9560 26JAN2012 201/732-2323
1430 DEAN        SANDRA      BRIDGEPORT    CT F TA2   32,926     8094 27APR2017 203/675-1647
1983 DEAN        SHARON      NEW YORK      NY F FA3   33,420      789 30APR2005 718/384-1647
1134 DELGADO     MARIA       STAMFORD      CT F TA2   33,463    10656 21DEC2016 203/781-1528
1118 DENNIS      ROGER       NEW YORK      NY M PT3  111,380     1476 18DEC2000 718/383-1122
1438 DESAI       AAKASH      STAMFORD      CT F TA3   39,224     9205 18NOV2007 203/781-2229
1125 DUNLAP      DONNA       NEW YORK      NY F FA2   28,889    10539 11DEC2007 718/383-2094
1475 EATON       ALICIA      NEW YORK      NY F FA2   27,788     8019 13JUL2010 718/383-2828
1117 EDGERTON    JOSHUA      NEW YORK      NY M TA3   39,772     8556 13AUG2012 212/588-1239
1935 FERNANDEZ   KATRINA     BRIDGEPORT    CT F NA2   51,082     5200 16OCT2011 203/675-2962
1124 FIELDS      DIANA       WHITE PLAINS  NY F FA1   23,178     6765 01OCT2010 914/455-2998
1422 FLETCHER    MARIE       PRINCETON     NJ F FA1   22,455     8921 06APR2011 201/812-0902
1616 FLOWERS     ANNETTE     NEW YORK      NY F TA2   34,138    11017 04JUN2013 718/384-3329
1406 FOSTER      GERALD      BRIDGEPORT    CT M ME2   35,186     7737 17FEB2007 203/675-6363
```

1120 GARCIA JACK NEW YORK NY M ME1 28,620 11942 07OCT2013 718/384-4930
1094 GOMEZ ALAN BRIDGEPORT CT M FA1 22,269 11049 17APR2011 203/675-7181
1389 GORDON LEVI NEW YORK NY M BCK 25,029 7135 18AUG2010 718/384-9326
1905 GRAHAM ALVIN NEW YORK NY M PT1 65,112 11794 29MAY2012 212/586-8815
1407 GRANT DANIEL MT. VERNON NY M PT1 68,097 10674 18MAR2010 914/468-1616
1114 GREEN JANICE NEW YORK NY F TA2 32,929 10853 27JUN2007 212/588-1092
1410 HARRIS CHARLES STAMFORD CT M PT2 84,686 9984 07NOV2006 203/781-0937
1439 HARRISON FELICIA BRIDGEPORT CT F PT1 70,737 8831 10SEP2010 203/675-4987
1409 HARTFORD RAYMOND STAMFORD CT M ME3 41,552 3761 22OCT2001 203/781-9697
1408 HENDERSON WILLIAM PRINCETON NJ M TA2 34,139 7393 14OCT2007 201/812-4789
1121 HERNANDEZ MICHAEL NEW YORK NY M ME1 29,113 7939 07DEC2011 718/384-3313
1991 HOLMES GABRIEL BRIDGEPORT CT F TA1 27,646 8162 12DEC2012 203/675-0007
1102 HOLMES SHANE WHITE PLAINS NY M TA2 34,543 7213 15APR2011 914/455-0976
1356 HOLMES SHAWN NEW YORK NY M ME2 36,870 6478 22FEB2003 212/586-8411
1545 HUNTER CLYDE STAMFORD CT M PT1 66,131 7163 29MAY2010 203/781-1119
1292 HUNTER HELEN BRIDGEPORT CT F ME2 36,692 9067 02JUL2009 203/675-4830
1440 JACKSON LAURA STAMFORD CT F ME2 35,758 8305 09APR2011 203/781-0088
1368 JEPSEN RONALD STAMFORD CT M FA2 27,809 7832 03NOV2014 203/781-8413
1369 JOHNSON ANTHONY NEW YORK NY M TA2 33,706 8032 13MAR2017 212/587-5385
1411 JOHNSON JACKSON PATERSON NJ M FA2 27,266 7817 01DEC2017 201/732-3678
1113 JONES LESLIE NEW YORK NY F FA1 22,368 10241 17OCT2011 718/383-3003
1704 JOSHI ABHAY NEW YORK NY M BCK 25,466 9738 28JUN2007 718/384-0049
1900 KING WILLIAM NEW YORK NY M ME2 35,106 8180 27OCT2007 718/383-3698
1126 KOSTECKA NICHOLAS NEW YORK NY F TA3 40,900 8548 21NOV2010 212/586-1229
1677 KRAMER JACKSON BRIDGEPORT CT M BCK 26,008 8709 27MAR2009 203/675-7432
1441 LAWRENCE KATHY PRINCETON NJ F FA2 27,159 10915 23MAR2011 201/812-3337
1421 LEE RUSSELL MT. VERNON NY M TA2 33,156 6947 28FEB2010 914/468-9143
1119 LI JEFF NEW YORK NY M TA1 26,925 8206 06SEP2008 212/586-2344
1834 LONG RUSSELL NEW YORK NY M BCK 26,897 8074 02JUL2012 718/384-0040
1777 LUFKIN ROY NEW YORK NY M PT3 109,631 4283 21JUN2001 718/383-4413
1663 MAHANNAHS SHANTHA NEW YORK NY M BCK 26,453 9872 11AUG2011 212/587-7742
1106 MARSHBURN JASPER STAMFORD CT M PT2 89,633 6519 16AUG2004 203/781-1457
1103 MCDANIEL RONDA NEW YORK NY F FA1 23,739 10273 23JUL2012 212/586-0013
1477 MEYERS PRESTON BRIDGEPORT CT M FA2 28,567 8846 07MAR2008 203/675-8125
1476 MONROE JOYCE STAMFORD CT F TA2 34,804 7890 17MAR2017 203/781-2837
1379 MORGAN ALFRED STAMFORD CT M ME3 42,265 8515 10JUN2004 203/781-2216
1104 MORGAN CHRISTOPHER NEW YORK NY M SCP 17,947 8515 10JUN2011 718/383-9740
1009 MORGAN GEORGE NEW YORK NY M TA1 28,881 7000 26MAR2012 212/586-7753
1412 MURPHEY JOHN PRINCETON NJ M ME1 27,800 6013 05DEC2011 201/812-4414
1115 MURPHY ALICE NEW YORK NY F FA3 32,700 7539 28FEB2010 718/384-1982
1128 NELSON FELICIA BRIDGEPORT CT F TA2 32,778 9274 20OCT2010 203/675-1166
1442 NEWKIRK SANDRA PRINCETON NJ F PT2 84,537 9744 12APR2008 201/812-3331
1417 NEWKIRK WILLIAM PATERSON NJ M NA2 52,271 8944 07MAR2009 201/732-6611
1478 NEWTON JAMES NEW YORK NY M PT2 84,204 7160 24OCT2010 212/587-5549
1673 NICHOLLS HENRY STAMFORD CT M BCK 25,478 7362 15JUL2011 203/781-7770
1839 NORRIS DIANE NEW YORK NY F NA1 43,434 7638 03JUL2013 718/384-1767
1347 O'NEAL BRYAN NEW YORK NY M TA3 40,080 10125 06SEP2014 718/384-0230
1423 OSWALD LESLIE MT. VERNON NY F ME2 35,774 10361 19AUG2010 914/468-9171
1200 OVERMAN MICHELLE STAMFORD CT F ME1 27,817 7680 14AUG2012 203/781-1835
1970 PAPI PAOLO NEW YORK NY F FA1 22,616 9034 12MAR2011 718/383-3895
1521 PAPIA ISMAEL NEW YORK NY M ME3 41,527 8502 13JUL2008 212/587-7603
1354 PAPIA FRANCISCO WHITE PLAINS NY F SCP 18,336 7819 16JUN2012 914/455-2337
1424 PATTERSON RENEE NEW YORK NY F FA2 28,979 7155 11DEC2009 212/587-8991
1132 PEARCE CAROL NEW YORK NY F FA1 22,414 4533 22OCT2003 718/384-1986
1845 PEARSON JAMES NEW YORK NY M BCK 25,997 7263 22MAR2000 718/384-2311
1556 PENNINGTON MICHAEL NEW YORK NY M PT1 71,350 8939 11DEC2011 718/383-5681

```
1413 PETERS       RANDALL      PRINCETON     NJ M FA2   27,436    5737 02JAN2010 201/812-2478
1123 PETERSON     SUZANNE      NEW YORK      NY F TA1   28,407    8339 05DEC2012 718/383-0077
1907 PHELPS       WILLIAM      STAMFORD      CT M TA2   33,330    7624 06JUL2007 203/781-1118
1436 PORTER       SUSAN        NEW YORK      NY F TA2   34,476    8928 12MAR2007 718/383-5777
1385 RAYNOR       MILTON       BRIDGEPORT    CT M ME3   43,901    8051 01APR2016 203/675-2846
1432 REED         MARILYN      MT. VERNON    NY F ME2   35,328    7977 10FEB2015 914/468-5454
1111 RHODES       JEREMY       PRINCETON     NJ M NA1   40,587    7977 31OCT2012 201/812-1837
1116 RICHARDS     CASEY        NEW YORK      NY F FA1   22,863    7210 21MAR2011 212/587-1224
1352 RIVERS       SIMON        NEW YORK      NY M NA2   53,799    7641 16OCT2006 718/383-3345
1555 RODRIGUEZ    JULIA        BRIDGEPORT    CT F FA2   27,500    6649 04JUL2012 203/675-2401
1038 RODRIGUEZ    MARIA        BRIDGEPORT    CT F TA1   26,534   10905 23NOV2011 203/675-2048
1420 ROUSE        JEREMY       PATERSON      NJ M ME3   43,072    9181 22JUL2017 201/732-9834
1561 SANDERS      RAYMOND      NEW YORK      NY M TA2   34,515    8734 07OCT2007 212/588-6615
1434 SANDERSON    EDITH        STAMFORD      CT F FA2   28,623    8227 28OCT2010 203/781-1333
1414 SARKAR       ABHEEK       BRIDGEPORT    CT M FA1   23,645    8118 12APR2012 203/675-1715
1112 SAYERS       RANDY        NEW YORK      NY M TA1   26,906    9099 07DEC2012 718/384-4895
1390 SMART        JONATHAN     NEW YORK      NY M FA2   27,762    9181 23JUN2011 718/383-1141
1332 STEPHENSON   ADAM         BRIDGEPORT    CT M NA1   42,179    7565 04JUN2011 203/675-1497
1890 STEPHENSON   ROBERT       NEW YORK      NY M PT2   85,897    7871 25NOV2017 718/384-9874
1429 THOMPSON     ALICE        STAMFORD      CT F TA1   27,940    7363 07AUG2012 203/781-3857
1107 THOMPSON     WAYNE        NEW YORK      NY M PT2   89,978    5273 10FEB2009 718/384-3785
1908 TRENTON      MELISSA      NEW YORK      NY F TA2   32,996    7283 23APR2010 212/586-6262
1830 TRIPP        KATHY        BRIDGEPORT    CT F PT2   84,472    6356 29JAN2013 203/675-2479
1882 TUCKER       ALAN         NEW YORK      NY M ME3   41,539    6400 21NOV2008 718/384-0216
1050 TUTTLE       THOMAS       WHITE PLAINS  NY M ME2   35,168    8595 24AUG2016 914/455-2119
1425 UNDERWOOD    JENNY        STAMFORD      CT F FA1   23,980    8032 28FEB2013 203/781-0978
1928 UPCHURCH     LARRY        WHITE PLAINS  NY M PT2   89,859    5372 13JUL2010 914/455-5009
1480 UPDIKE       THERESA      NEW YORK      NY F TA3   39,584    6455 25MAR2001 212/587-8729
1100 VANDEUSEN    RICHARD      NEW YORK      NY M BCK   25,005    7640 07MAY2008 212/586-2531
1995 VARNER       ELIZABETH    NEW YORK      NY F ME1   28,811    8636 19SEP2013 718/384-7113
1135 VEGA         ANNA         NEW YORK      NY F FA2   27,322    7568 31MAR2010 718/384-5913
1415 VEGA         FRANKLIN     NEW YORK      NY M FA2   28,279    6642 12FEB2008 718/384-2823
1076 VENTER       RANDALL      NEW YORK      NY M PT1   66,559    5765 03OCT2011 718/383-2321
1426 VICK         THERESA      PRINCETON     NJ F TA2   32,992    9835 25JUN2010 201/812-2424
1564 WALTERS      ANNE         NEW YORK      NY F SCP   18,834    8137 01JUL2012 212/587-3257
1221 WALTERS      DIANE        NEW YORK      NY F FA2   27,897   10126 04OCT2011 718/384-1918
1133 WANG         CHIN         NEW YORK      NY M TA1   27,702    9690 12FEB2012 212/587-1956
1435 WARD         ELAINE       NEW YORK      NY F TA3   38,809    7071 08FEB2010 718/383-4987
1418 WATSON       BERNARD      NEW YORK      NY M ME1   28,006    6297 06JAN2012 718/383-1298
1017 WELCH        DARIUS       NEW YORK      NY M TA3   40,859    6571 16OCT2011 212/586-5535
1443 WELLS        AGNES        STAMFORD      CT F NA1   42,275   10548 29AUG2011 203/781-5546
1131 WELLS        NADINE       NEW YORK      NY F TA2   32,576    8030 19APR2011 718/383-1045
1427 WHALEY       CAROLYN      MT. VERNON    NY F TA2   34,047   11261 30JAN2010 914/468-4528
1036 WONG         LESLIE       NEW YORK      NY F TA3   39,393    9270 23OCT2004 212/587-2570
1130 WOOD         DEBORAH      NEW YORK      NY F FA1   23,917    7806 05JUN2012 212/587-0013
1127 WOOD         SANDRA       NEW YORK      NY F TA2   33,012    9079 07DEC2006 212/587-2881
1433 YANCEY       ROBIN        PRINCETON     NJ F FA3   32,983    9685 17JAN2007 201/812-1874
1431 YOUNG        DEBORAH      STAMFORD      CT F FA3   33,231    8926 05APR2008 203/781-2987
1122 YOUNG        JOANN        NEW YORK      NY F FA2   27,957    8521 27NOV2008 718/384-2021
1105 YOUNG        LAWRENCE     NEW YORK      NY M ME2   34,806    8095 13AUG2010 718/384-0008
;
run;
```

CONTROL.MLSCL

```
data control.mlscl (label='Multiple Sclerosis Data');
   input GROUP OBS1 OBS2 WT;
   datalines;
    1     4     4    10
    1     4     1     3
    1     4     2     7
    1     4     3     3
    1     1     4     1
    1     1     1    38
    1     1     2     5
    1     1     3     0
    1     2     4     0
    1     2     1    33
    1     2     2    11
    1     2     3     3
    1     3     4     6
    1     3     1    10
    1     3     2    14
    1     3     3     5
    2     4     1     1
    2     4     2     2
    2     4     3     4
    2     4     4    14
    2     3     1     2
    2     3     2    13
    2     3     3     3
    2     3     4     4
    2     2     1     3
    2     2     2    11
    2     2     3     4
    2     2     4     0
    2     1     1     5
    2     1     2     3
    2     1     3     0
    2     1     4     0
;
run;
```

CONTROL.NAMES

```
data control.names;
   input LABEL $ 1-16 START $ 17-24 FMTNAME $ 31-35 TYPE $ 41-41;
   datalines;
Capalleti, Jimmy   2355     bonus    C
Chen, Len          5889     bonus    C
Davis, Brad        3878     bonus    C
```

```
Leung, Brenda      4409     bonus    C
Patel, Mary        2398     bonus    C
Smith, Robert      5862     bonus    C
Zook, Carla        7385     bonus    C
;
run;
```

CONTROL.OXYGEN

```
data control.oxygen;
   input AGE WEIGHT RUNTIME RSTPULSE RUNPULSE MAXPULSE OXYGEN;
   datalines;
44    89.47    11.37     62       178       182     44.609
40    75.07    10.07     62       185       185     45.313
44    85.84     8.65     45       156       168     54.297
42    68.15     8.17     40       166       172     59.571
38    89.02     9.22     55       178       180     49.874
47    77.45    11.63     58       176       176     44.811
40    75.98    11.95     70       176       180     45.681
43    81.19    10.85     64       162       170     49.091
44    81.42    13.08     63       174       176     39.442
38    81.87     8.63     48       170       186     60.055
44    73.03    10.13     45       168       168     50.541
45    87.66    14.03     56       186       192     37.388
45    66.45    11.12     51       176       176     44.754
47    79.15    10.60     47       162       164     47.273
54    83.12    10.33     50       166       170     51.855
49    81.42     8.95     44       180       185     49.156
51    69.63    10.95     57       168       172     40.836
51    77.91    10.00     48       162       168     46.672
48    91.63    10.25     48       162       164     46.774
49    73.37    10.08     76       168       168     50.388
57    73.37    12.63     58       174       176     39.407
54    79.38    11.17     62       156       165     46.080
52    76.32     9.63     48       164       166     45.441
50    70.87     8.92     48       146       155     54.625
51    67.25    11.08     48       172       172     45.118
54    91.63    12.88     44       168       172     39.203
51    73.71    10.47     59       186       188     45.790
57    59.08     9.93     49       148       155     50.545
49    76.32     9.40     56       186       188     48.673
48    61.24    11.50     52       170       176     47.920
52    82.78    10.50     53       170       172     47.467
;
run;
```

CONTROL.PERSONL

```
data control.personl;
```

```
   input IDNUM $ 1-4 LNAME $ 4-18 FNAME $ 19-33 CITY $ 32-46 STATE $ 47-48
       SEX $ 50-50 JOBCODE $ 52-54 SALARY BIRTH date. @63
       HIRED date9. @73 HPHONE $ 84-99;
   format birth date9.;
   informat birth date.;
   format hired date9.;
   informat hired date.;
   datalines;
1919 ADAMS       GERALD       STAMFORD      CT M TA2  34376 12SEP1990 04JUN2007  203/781-1255
1653 AHMAD       AZEEM        BRIDGEPORT    CT F ME2  35108 15OCT1984 09AUG2010  203/675-7715
1400 ALVAREZ     GLORIA       NEW YORK      NY M ME1  29769 05NOV1987 16OCT2010  212/586-0808
1350 ARTHUR      BARBARA      NEW YORK      NY F FA3  32886 31AUG1985 29JUL2010  718/383-1549
1401 AVERY       JERRY        PATERSON      NJ M TA3  38822 13DEC1970 17NOV2005  201/732-8787
1499 BAREFOOT    JOSEPH       PRINCETON     NJ M ME3  43025 26APR1974 07JUN2010  201/812-5665
1101 BASQUEZ     RICHARDO     NEW YORK      NY M SCP  18723 06JUN1982 01OCT2010  212/586-8060
1333 BEAULIEU    ARMANDO      STAMFORD      CT M PT2  88606 30MAR1981 10FEB2011  203/781-1777
1402 BLALOCK     RALPH        NEW YORK      NY M TA2  32615 17JAN1983 02DEC2010  718/384-2849
1479 BOSTIC      MARIE        NEW YORK      NY F TA3  38785 22DEC1988 05OCT2007  718/384-8816
1403 BOWDEN      EARL         BRIDGEPORT    CT M ME1  28072 28JAN1989 21DEC2016  203/675-3434
1739 BOYCE       JONATHAN     NEW YORK      NY M PT1  66517 25DEC1984 27JAN2011  212/587-1247
1658 BRADLEY     JEREMY       NEW YORK      NY M SCP  17943 08APR1987 29FEB2012  212/587-3622
1428 BRADY       CHRISTINE    STAMFORD      CT F PT1  68767 04APR1980 16NOV2011  203/781-1212
1782 BROWN       JASON        STAMFORD      CT M ME2  35345 04DEC1980 22FEB2012  203/781-0019
1244 BRYANT      LEONARD      NEW YORK      NY M ME2  36925 31AUG1983 17JAN2008  718/383-3334
1383 BURNETTE    THOMAS       NEW YORK      NY M BCK  25824 25JAN1985 20OCT2009  718/384-3569
1574 CAHILL      MARSHALL     NEW YORK      NY M FA2  28572 27APR1980 20DEC2012  718/383-2338
1789 CANALES     VIVIANA      NEW YORK      NY M SCP  18326 23JAN1977 11APR2008  212/587-9000
1404 CARTER      DONALD       NEW YORK      NY M PT2  91376 24FEB1973 01JAN2010  718/384-2946
1437 CARTER      DOROTHY      BRIDGEPORT    CT F FA3  33104 20SEP1980 31AUG2004  203/675-4117
1639 CARTER      KAREN        STAMFORD      CT F TA3  40260 26JUN1977 28JAN2014  203/781-8839
1269 CASTON      FRANKLIN     STAMFORD      CT M NA1  41690 03MAY1972 28NOV2012  203/781-3335
1065 CHAPMAN     NEIL         NEW YORK      NY M ME2  35090 26JAN1977 07JAN2007  718/384-5618
1876 CHIN        JACK         NEW YORK      NY M TA3  39675 20MAY1978 27APR2015  212/588-5634
1037 CHOW        JANE         STAMFORD      CT F TA1  28558 10APR1984 13SEP2012  203/781-8868
1129 COOK        BRENDA       NEW YORK      NY F ME2  34929 08DEC1981 17AUG2011  718/383-2313
1988 COOPER      ANTHONY      NEW YORK      NY M FA3  32217 30NOV1979 18SEP2004  212/587-1228
1405 DAVIDSON    JASON        PATERSON      NJ M SCP  18056 05MAR1986 26JAN2012  201/732-2323
1430 DEAN        SANDRA       BRIDGEPORT    CT F TA2  32925 28FEB1982 27APR2017  203/675-1647
1983 DEAN        SHARON       NEW YORK      NY F FA3  33419 28FEB1962 30APR2005  718/384-1647
1134 DELGADO     MARIA        STAMFORD      CT F TA2  33462 05MAR1989 21DEC2016  203/781-1528
1118 DENNIS      ROGER        NEW YORK      NY M PT3 111379 16JAN1964 18DEC2000  718/383-1122
1438 DESAI       AAKASH       STAMFORD      CT F TA3  39223 15MAR1985 18NOV2007  203/781-2229
1125 DUNLAP      DONNA        NEW YORK      NY F FA2  28888 08NOV1988 11DEC2007  718/383-2094
1475 EATON       ALICIA       NEW YORK      NY F FA2  27787 15DEC1981 13JUL2010  718/383-2828
1117 EDGERTON    JOSHUA       NEW YORK      NY M TA3  39771 05JUN1983 13AUG2012  212/588-1239
1935 FERNANDEZ   KATRINA      BRIDGEPORT    CT F NA2  51081 28MAR1974 16OCT2011  203/675-2962
1124 FIELDS      DIANA        WHITE PLAINS  NY F FA1  23177 10JUL1978 01OCT2010  914/455-2998
1422 FLETCHER    MARIE        PRINCETON     NJ F FA1  22454 04JUN1984 06APR2011  201/812-0902
1616 FLOWERS     ANNETTE      NEW YORK      NY F TA2  34137 01MAR1990 04JUN2013  718/384-3329
1406 FOSTER      GERALD       BRIDGEPORT    CT M ME2  35185 08MAR1981 17FEB2007  203/675-6363
1120 GARCIA      JACK         NEW YORK      NY M ME1  28619 11SEP1992 07OCT2013  718/384-4930
1094 GOMEZ       ALAN         BRIDGEPORT    CT M FA1  22268 02APR1990 17APR2011  203/675-7181
1389 GORDON      LEVI         NEW YORK      NY M BCK  25028 15JUL1979 18AUG2010  718/384-9326
1905 GRAHAM      ALVIN        NEW YORK      NY M PT1  65111 16APR1992 29MAY2012  212/586-8815
1407 GRANT       DANIEL       MT. VERNON    NY M PT1  68096 23MAR1989 18MAR2010  914/468-1616
1114 GREEN       JANICE       NEW YORK      NY F TA2  32928 18SEP1989 27JUN2007  212/588-1092
```

1410 HARRIS CHARLES STAMFORD CT M PT2 84685 03MAY1987 07NOV2006 203/781-0937
1439 HARRISON FELICIA BRIDGEPORT CT F PT1 70736 06MAR1984 10SEP2010 203/675-4987
1409 HARTFORD RAYMOND STAMFORD CT M ME3 41551 19APR1970 22OCT2001 203/781-9697
1408 HENDERSON WILLIAM PRINCETON NJ M TA2 34138 29MAR1980 14OCT2007 201/812-4789
1121 HERNANDEZ MICHAEL NEW YORK NY M ME1 29112 26SEP1981 07DEC2011 718/384-3313
1991 HOLMES GABRIEL BRIDGEPORT CT F TA1 27645 07MAY1982 12DEC2012 203/675-0007
1102 HOLMES SHANE WHITE PLAINS NY M TA2 34542 01OCT1979 15APR2011 914/455-0976
1356 HOLMES SHAWN NEW YORK NY M ME2 36869 26SEP1977 22FEB2003 212/586-8411
1545 HUNTER CLYDE STAMFORD CT M PT1 66130 12AUG1979 29MAY2010 203/781-1119
1292 HUNTER HELEN BRIDGEPORT CT F ME2 36691 28OCT1984 02JUL2009 203/675-4830
1440 JACKSON LAURA STAMFORD CT F ME2 35757 27SEP1982 09APR2011 203/781-0088
1368 JEPSEN RONALD STAMFORD CT M FA2 27808 11JUN1981 03NOV2014 203/781-8413
1369 JOHNSON ANTHONY NEW YORK NY M TA2 33705 28DEC1981 13MAR2017 212/587-5385
1411 JOHNSON JACKSON PATERSON NJ M FA2 27265 27MAY1981 01DEC2017 201/732-3678
1113 JONES LESLIE NEW YORK NY F FA1 22367 15JAN1988 17OCT2011 718/383-3003
1704 JOSHI ABHAY NEW YORK NY M BCK 25465 30AUG1986 28JUN2007 718/384-0049
1900 KING WILLIAM NEW YORK NY M ME2 35105 25MAY1982 27OCT2007 718/383-3698
1126 KOSTECKA NICHOLAS NEW YORK NY F TA3 40899 28MAY1983 21NOV2010 212/586-1229
1677 KRAMER JACKSON BRIDGEPORT CT M BCK 26007 05NOV1983 27MAR2009 203/675-7432
1441 LAWRENCE KATHY PRINCETON NJ F FA2 27158 19NOV1989 23MAR2011 201/812-3337
1421 LEE RUSSELL MT. VERNON NY M TA2 33155 08JAN1979 28FEB2010 914/468-9143
1119 LI JEFF NEW YORK NY M TA1 26924 20JUN1982 06SEP2008 212/586-2344
1834 LONG RUSSELL NEW YORK NY M BCK 26896 08FEB1982 02JUL2012 718/384-0040
1777 LUFKIN ROY NEW YORK NY M PT3 109630 23SEP1971 21JUN2001 718/383-4413
1663 MAHANNAHS SHANTHA NEW YORK NY M BCK 26452 11JAN1987 11AUG2011 212/587-7742
1106 MARSHBURN JASPER STAMFORD CT M PT2 89632 06NOV1977 16AUG2004 203/781-1457
1103 MCDANIEL RONDA NEW YORK NY F FA1 23738 16FEB1988 23JUL2012 212/586-0013
1477 MEYERS PRESTON BRIDGEPORT CT M FA2 28566 21MAR1984 07MAR2008 203/675-8125
1476 MONROE JOYCE STAMFORD CT F TA2 34803 30MAY1986 17MAR2017 203/781-2837
1379 MORGAN ALFRED STAMFORD CT M ME3 42264 08AUG1981 10JUN2004 203/781-2216
1104 MORGAN CHRISTOPHER NEW YORK NY M SCP 17946 25APR1983 10JUN2011 718/383-9740
1009 MORGAN GEORGE NEW YORK NY M TA1 28880 02MAR1979 26MAR2012 212/586-7753
1412 MURPHEY JOHN PRINCETON NJ M ME1 27799 18JUN1976 05DEC2011 201/812-4414
1115 MURPHY ALICE NEW YORK NY F FA3 32699 22AUG1980 28FEB2010 718/384-1982
1128 NELSON FELICIA BRIDGEPORT CT F TA2 32777 23MAY1985 20OCT2010 203/675-1166
1442 NEWKIRK SANDRA PRINCETON NJ F PT2 84536 05SEP1986 12APR2008 201/812-3331
1417 NEWKIRK WILLIAM PATERSON NJ M NA2 52270 27JUN1984 07MAR2009 201/732-6611
1478 NEWTON JAMES NEW YORK NY M PT2 84203 09AUG1979 24OCT2010 212/587-5549
1673 NICHOLLS HENRY STAMFORD CT M BCK 25477 27FEB1980 15JUL2011 203/781-7770
1839 NORRIS DIANE NEW YORK NY F NA1 44433 29NOV1980 03JUL2013 718/384-1767
1347 O'NEAL BRYAN NEW YORK NY M TA3 40079 21SEP1987 06SEP2014 718/384-0230
1423 OSWALD LESLIE MT. VERNON NY F ME2 35773 14MAY1988 19AUG2010 914/468-9171
1200 OVERMAN MICHELLE STAMFORD CT F ME1 27816 10JAN1981 14AUG2012 203/781-1835
1970 PAPI PAOLO NEW YORK NY F FA1 22615 25SEP1984 12MAR2011 718/383-3895
1521 PAPIA ISMAEL NEW YORK NY M ME3 41526 12APR1983 13JUL2008 212/587-7603
1354 PAPIA FRANCISCO WHITE PLAINS NY F SCP 18335 29MAY1981 16JUN2012 914/455-2337
1424 PATTERSON RENEE NEW YORK NY F FA2 28978 04AUG1979 11DEC2009 212/587-8991
1132 PEARCE CAROL NEW YORK NY F FA1 22413 30MAY1972 22OCT2003 718/384-1986
1845 PEARSON JAMES NEW YORK NY M BCK 25996 20NOV1979 22MAR2000 718/384-2311
1556 PENNINGTON MICHAEL NEW YORK NY M PT1 71349 22JUN1984 11DEC2011 718/383-5681
1413 PETERS RANDALL PRINCETON NJ M FA2 27435 16SEP1975 02JAN2010 201/812-2478
1123 PETERSON SUZANNE NEW YORK NY F TA1 28407 31OCT1982 05DEC2012 718/383-0077
1907 PHELPS WILLIAM STAMFORD CT M TA2 33329 15NOV1980 06JUL2007 203/781-1118
1436 PORTER SUSAN NEW YORK NY F TA2 34475 11JUN1984 12MAR2007 718/383-5777
1385 RAYNOR MILTON BRIDGEPORT CT M ME3 43900 16JAN1982 01APR2016 203/675-2846
1432 REED MARILYN MT. VERNON NY F ME2 35327 03NOV1981 10FEB2015 914/468-5454

```
1111 RHODES       JEREMY        PRINCETON      NJ M NA1  40586 14JUL1983 31OCT2012  201/812-1837
1116 RICHARDS     CASEY         NEW YORK       NY F FA1  22862 28SEP1979 21MAR2011  212/587-1224
1352 RIVERS       SIMON         NEW YORK       NY M NA2  53798 02DEC1980 16OCT2006  718/383-3345
1555 RODRIGUEZ    JULIA         BRIDGEPORT     CT F FA2  27499 16MAR1978 04JUL2012  203/675-2401
1038 RODRIGUEZ    MARIA         BRIDGEPORT     CT F TA1  26533 09NOV1989 23NOV2011  203/675-2048
1420 ROUSE        JEREMY        PATERSON       NJ M ME3  43071 19FEB1985 22JUL2017  201/732-9834
1561 SANDERS      RAYMOND       NEW YORK       NY M TA2  34514 30NOV1983 07OCT2007  212/588-6615
1434 SANDERSON    EDITH         STAMFORD       CT F FA2  28622 11JUL1982 28OCT2010  203/781-1333
1414 SARKAR       ABHEEK        BRIDGEPORT     CT M FA1  23644 24MAR1982 12APR2012  203/675-1715
1112 SAYERS       RANDY         NEW YORK       NY M TA1  26905 29NOV1984 07DEC2012  718/384-4895
1390 SMART        JONATHAN      NEW YORK       NY M FA2  27761 19FEB1985 23JUN2011  718/383-1141
1332 STEPHENSON   ADAM          BRIDGEPORT     CT M NA1  42178 17SEP1980 04JUN2011  203/675-1497
1890 STEPHENSON   ROBERT        NEW YORK       NY M PT2  85896 20JUL1981 25NOV2017  718/384-9874
1429 THOMPSON     ALICE         STAMFORD       CT F TA1  27939 28FEB1980 07AUG2012  203/781-3857
1107 THOMPSON     WAYNE         NEW YORK       NY M PT2  89977 09JUN1974 10FEB2009  718/384-3785
1908 TRENTON      MELISSA       NEW YORK       NY F TA2  32995 10DEC1979 23APR2010  212/586-6262
1830 TRIPP        KATHY         BRIDGEPORT     CT F PT2  84471 27MAY1977 29JAN2013  203/675-2479
1882 TUCKER       ALAN          NEW YORK       NY M ME3  41538 10JUL1977 21NOV2008  718/384-0216
1050 TUTTLE       THOMAS        WHITE PLAINS   NY M ME2  35167 14JUL1983 24AUG2016  914/455-2119
1425 UNDERWOOD    JENNY         STAMFORD       CT F FA1  23979 28DEC1981 28FEB2013  203/781-0978
1928 UPCHURCH     LARRY         WHITE PLAINS   NY M PT2  89858 16SEP1974 13JUL2010  914/455-5009
1480 UPDIKE       THERESA       NEW YORK       NY F TA3  39583 03SEP1977 25MAR2001  212/587-8729
1100 VANDEUSEN    RICHARD       NEW YORK       NY M BCK  25004 01DEC1980 07MAY2008  212/586-2531
1995 VARNER       ELIZABETH     NEW YORK       NY F ME1  28810 24AUG1983 19SEP2013  718/384-7113
1135 VEGA         ANNA          NEW YORK       NY F FA2  27321 20SEP1980 31MAR2010  718/384-5913
1415 VEGA         FRANKLIN      NEW YORK       NY M FA2  28278 09MAR1978 12FEB2008  718/384-2823
1076 VENTER       RANDALL       NEW YORK       NY M PT1  66558 14OCT1975 03OCT2011  718/383-2321
1426 VICK         THERESA       PRINCETON      NJ F TA2  32991 05DEC1986 25JUN2010  201/812-2424
1564 WALTERS      ANNE          NEW YORK       NY F SCP  18833 12APR1982 01JUL2012  212/587-3257
1221 WALTERS      DIANE         NEW YORK       NY F FA2  27896 22SEP1987 04OCT2011  718/384-1918
1133 WANG         CHIN          NEW YORK       NY M TA1  27701 13JUL1986 12FEB2012  212/587-1956
1435 WARD         ELAINE        NEW YORK       NY F TA3  38808 12MAY1979 08FEB2010  718/383-4987
1418 WATSON       BERNARD       NEW YORK       NY M ME1  28005 29MAR1977 06JAN2012  718/383-1298
1017 WELCH        DARIUS        NEW YORK       NY M TA3  40858 28DEC1977 16OCT2011  212/586-5535
1443 WELLS        AGNES         STAMFORD       CT F NA1  42274 17NOV1988 29AUG2011  203/781-5546
1131 WELLS        NADINE        NEW YORK       NY F TA2  32575 26DEC1981 19APR2011  718/383-1045
1427 WHALEY       CAROLYN       MT. VERNON     NY F TA2  34046 31OCT1990 30JAN2010  914/468-4528
1036 WONG         LESLIE        NEW YORK       NY F TA3  39392 19MAY1985 23OCT2004  212/587-2570
1130 WOOD         DEBORAH       NEW YORK       NY F FA1  23916 16MAY1981 05JUN2012  212/587-0013
1127 WOOD         SANDRA        NEW YORK       NY F TA2  33011 09NOV1984 07DEC2006  212/587-2881
1433 YANCEY       ROBIN         PRINCETON      NJ F FA3  32982 08JUL1986 17JAN2007  201/812-1874
1431 YOUNG        DEBORAH       STAMFORD       CT F FA3  33230 09JUN1984 05APR2008  203/781-2987
1122 YOUNG        JOANN         NEW YORK       NY F FA2  27956 01MAY1983 27NOV2008  718/384-2021
1105 YOUNG        LAWRENCE      NEW YORK       NY M ME2  34805 01MAR1982 13AUG2010  718/384-0008
;
run;
```

CONTROL.PHARM

```
data control.pharm (label='Sugar Study');
   input DRUG $8. RESPONSE $8. WT ;
   datalines;
    A cured   14
```

```
    A uncured 22
    B cured   24
    B uncured 19
    C cured   17
    C uncured 13
;
run;
```

CONTROL.POINTS

```
data control.points;
   input EMPID $8. Q1 Q2 Q3 Q4 TOTPTS;
   datalines;
2355        3    4    4    3    14
5889        2    2    2    2    8
3878        1    2    2    2    7
4409        0    1    1    1    3
2398        2    2    1    1    6
5862        1    1    1    2    5
;
run;
```

CONTROL.PRENAT

```
data control.prenat;
   input IDNUM $ 1-4 LNAME $ 6-20  FNAME $ 22-36 CITY $ 39-53
       STATE $ 55-56 HPHONE $ 58-69;
   datalines;
1919 ADAMS          GERALD         STAMFORD       CT 203/781-1255
1653 ALIBRANDI      MARIA          BRIDGEPORT     CT 203/675-7715
1400 ALHERTANI      ABDULLAH       NEW YORK       NY 212/586-0808
1350 ALVAREZ        MERCEDES       NEW YORK       NY 718/383-1549
1401 ALVAREZ        CARLOS         PATERSON       NJ 201/732-8787
1499 BAREFOOT       JOSEPH         PRINCETON      NJ 201/812-5665
1101 BAUCOM         WALTER         NEW YORK       NY 212/586-8060
1333 BANADYGA       JUSTIN         STAMFORD       CT 203/781-1777
1402 BLALOCK        RALPH          NEW YORK       NY 718/384-2849
1479 BALLETTI       MARIE          NEW YORK       NY 718/384-8816
1403 BOWDEN         EARL           BRIDGEPORT     CT 203/675-3434
1739 BRANCACCIO     JOSEPH         NEW YORK       NY 212/587-1247
1658 BREUHAUS       JEREMY         NEW YORK       NY 212/587-3622
1428 BRADY          CHRISTINE      STAMFORD       CT 203/781-1212
1782 BREWCZAK       JAKOB          STAMFORD       CT 203/781-0019
1244 BUCCI          ANTHONY        NEW YORK       NY 718/383-3334
1383 BURNETTE       THOMAS         NEW YORK       NY 718/384-3569
1574 CAHILL         MARSHALL       NEW YORK       NY 718/383-2338
1789 CARAWAY        DAVIS          NEW YORK       NY 212/587-9000
1404 COHEN          LEE            NEW YORK       NY 718/384-2946
1437 CARTER         DOROTHY        BRIDGEPORT     CT 203/675-4117
```

1639 CARTER-COHEN KAREN STAMFORD CT 203/781-8839
1269 CASTON FRANKLIN STAMFORD CT 203/781-3335
1065 COPAS FREDERICO NEW YORK NY 718/384-5618
1876 CHIN JACK NEW YORK NY 212/588-5634
1037 CHOW JANE STAMFORD CT 203/781-8868
1129 COUNIHAN BRENDA NEW YORK NY 718/383-2313
1988 COOPER ANTHONY NEW YORK NY 212/587-1228
1405 DACKO JASON PATERSON NJ 201/732-2323
1430 DABROWSKI SANDRA BRIDGEPORT CT 203/675-1647
1983 DEAN SHARON NEW YORK NY 718/384-1647
1134 DELGADO MARIA STAMFORD CT 203/781-1528
1118 DENNIS ROGER NEW YORK NY 718/383-1122
1438 DABBOUSSI KAMILLA STAMFORD CT 203/781-2229
1125 DUNLAP DONNA NEW YORK NY 718/383-2094
1475 ELGES MARGARETE NEW YORK NY 718/383-2828
1117 EDGERTON JOSHUA NEW YORK NY 212/588-1239
1935 FERNANDEZ KATRINA BRIDGEPORT CT 203/675-2962
1124 FIELDS DIANA WHITE PLAINS NY 914/455-2998
1422 FUJIHARA KYOKO PRINCETON NJ 201/812-0902
1616 FUENTAS CARLA NEW YORK NY 718/384-3329
1406 FOSTER GERALD BRIDGEPORT CT 203/675-6363
1120 GARCIA JACK NEW YORK NY 718/384-4930
1094 GOMEZ ALAN BRIDGEPORT CT 203/675-7181
1389 GOLDSTEIN LEVI NEW YORK NY 718/384-9326
1905 GRAHAM ALVIN NEW YORK NY 212/586-8815
1407 GREGORSKI DANIEL MT. VERNON NY 914/468-1616
1114 GREENWALD JANICE NEW YORK NY 212/588-1092
1410 HARRIS CHARLES STAMFORD CT 203/781-0937
1439 HASENHAUER CHRISTINA BRIDGEPORT CT 203/675-4987
1409 HAVELKA RAYMOND STAMFORD CT 203/781-9697
1408 HENDERSON WILLIAM PRINCETON NJ 201/812-4789
1121 HERNANDEZ ROBERTO NEW YORK NY 718/384-3313
1991 HOWARD GRETCHEN BRIDGEPORT CT 203/675-0007
1102 HERMANN JOACHIM WHITE PLAINS NY 914/455-0976
1356 HOWARD MICHAEL NEW YORK NY 212/586-8411
1545 HERRERO CLYDE STAMFORD CT 203/781-1119
1292 HUNTER HELEN BRIDGEPORT CT 203/675-4830
1440 JACKSON LAURA STAMFORD CT 203/781-0088
1368 JEPSEN RONALD STAMFORD CT 203/781-8413
1369 JONSON ANTHONY NEW YORK NY 212/587-5385
1411 JOHNSEN JACK PATERSON NJ 201/732-3678
1113 JOHNSON LESLIE NEW YORK NY 718/383-3003
1704 JONES NATHAN NEW YORK NY 718/384-0049
1900 KING WILLIAM NEW YORK NY 718/383-3698
1126 KIMANI ANNE NEW YORK NY 212/586-1229
1677 KRAMER JACKSON BRIDGEPORT CT 203/675-7432
1441 LAWRENCE KATHY PRINCETON NJ 201/812-3337
1421 LEE RUSSELL MT. VERNON NY 914/468-9143
1119 LI JEFF NEW YORK NY 212/586-2344
1834 LEBLANC RUSSELL NEW YORK NY 718/384-0040
1777 LUFKIN ROY NEW YORK NY 718/383-4413
1663 MARKS JOHN NEW YORK NY 212/587-7742
1106 MARSHBURN JASPER STAMFORD CT 203/781-1457
1103 MCDANIEL RONDA NEW YORK NY 212/586-0013
1477 MEYERS PRESTON BRIDGEPORT CT 203/675-8125
1476 MONROE JOYCE STAMFORD CT 203/781-2837

1379 MORGAN ALFRED STAMFORD CT 203/781-2216
1104 MORGAN CHRISTOPHER NEW YORK NY 718/383-9740
1009 MORGAN GEORGE NEW YORK NY 212/586-7753
1412 MURPHEY JOHN PRINCETON NJ 201/812-4414
1115 MURPHY ALICE NEW YORK NY 718/384-1982
1128 NELSON FELICIA BRIDGEPORT CT 203/675-1166
1442 NEWKIRK SANDRA PRINCETON NJ 201/812-3331
1417 NEWKIRK WILLIAM PATERSON NJ 201/732-6611
1478 NEWTON JAMES NEW YORK NY 212/587-5549
1673 NICHOLLS HENRY STAMFORD CT 203/781-7770
1839 NORRIS DIANE NEW YORK NY 718/384-1767
1347 O'NEAL BRYAN NEW YORK NY 718/384-0230
1423 OSWALD LESLIE MT. VERNON NY 914/468-9171
1200 OVERMAN MICHELLE STAMFORD CT 203/781-1835
1970 PARKER ANNE NEW YORK NY 718/383-3895
1521 PARKER JAY NEW YORK NY 212/587-7603
1354 PARKER MARY WHITE PLAINS NY 914/455-2337
1424 PATTERSON RENEE NEW YORK NY 212/587-8991
1132 PEARCE CAROL NEW YORK NY 718/384-1986
1845 PEARSON JAMES NEW YORK NY 718/384-2311
1556 PENNINGTON MICHAEL NEW YORK NY 718/383-5681
1413 PETERS RANDALL PRINCETON NJ 201/812-2478
1123 PETERSON SUZANNE NEW YORK NY 718/383-0077
1907 PHELPS WILLIAM STAMFORD CT 203/781-1118
1436 PORTER SUSAN NEW YORK NY 718/383-5777
1385 RAYNOR MILTON BRIDGEPORT CT 203/675-2846
1432 REED MARILYN MT. VERNON NY 914/468-5454
1111 RHODES JEREMY PRINCETON NJ 201/812-1837
1116 RICHARDS CASEY NEW YORK NY 212/587-1224
1352 RIVERS SIMON NEW YORK NY 718/383-3345
1555 RODRIGUEZ JULIA BRIDGEPORT CT 203/675-2401
1038 RODRIGUEZ MARIA BRIDGEPORT CT 203/675-2048
1420 ROUSE JEREMY PATERSON NJ 201/732-9834
1561 SANDERS RAYMOND NEW YORK NY 212/588-6615
1434 SANDERSON EDITH STAMFORD CT 203/781-1333
1414 SANDERSON NATHAN BRIDGEPORT CT 203/675-1715
1112 SANYERS RANDY NEW YORK NY 718/384-4895
1390 SMART JONATHAN NEW YORK NY 718/383-1141
1332 STEPHENSON ADAM BRIDGEPORT CT 203/675-1497
1890 STEPHENSON ROBERT NEW YORK NY 718/384-9874
1429 THOMPSON ALICE STAMFORD CT 203/781-3857
1107 THOMPSON WAYNE NEW YORK NY 718/384-3785
1908 TRENTON MELISSA NEW YORK NY 212/586-6262
1830 TRIPP KATHY BRIDGEPORT CT 203/675-2479
1882 TUCKER ALAN NEW YORK NY 718/384-0216
1050 TUTTLE THOMAS WHITE PLAINS NY 914/455-2119
1425 UNDERWOOD JENNY STAMFORD CT 203/781-0978
1928 UPCHURCH LARRY WHITE PLAINS NY 914/455-5009
1480 UPDIKE THERESA NEW YORK NY 212/587-8729
1100 VANDEUSEN RICHARD NEW YORK NY 212/586-2531
1995 VARNER ELIZABETH NEW YORK NY 718/384-7113
1135 VEGA ANNA NEW YORK NY 718/384-5913
1415 VEGA FRANKLIN NEW YORK NY 718/384-2823
1076 VENTER RANDALL NEW YORK NY 718/383-2321
1426 VICK THERESA PRINCETON NJ 201/812-2424
1564 WALTERS ANNE NEW YORK NY 212/587-3257

```
1221 WALTERS       DIANE          NEW YORK       NY 718/384-1918
1133 WANG          CHIN           NEW YORK       NY 212/587-1956
1435 WARD          ELAINE         NEW YORK       NY 718/383-4987
1418 WATSON        BERNARD        NEW YORK       NY 718/383-1298
1017 WELCH         DARIUS         NEW YORK       NY 212/586-5535
1443 WELLS         AGNES          STAMFORD       CT 203/781-5546
1131 WELLS         NADINE         NEW YORK       NY 718/383-1045
1427 WHALEY        CAROLYN        MT. VERNON     NY 914/468-4528
1036 WONG          LESLIE         NEW YORK       NY 212/587-2570
1130 WOOD          DEBORAH        NEW YORK       NY 212/587-0013
1127 WOOD          SANDRA         NEW YORK       NY 212/587-2881
1433 YANCEY        ROBIN          PRINCETON      NJ 201/812-1874
1431 YOUNG         DEBORAH        STAMFORD       CT 203/781-2987
1122 YOUNG         JOANN          NEW YORK       NY 718/384-2021
1105 YOUNG         LAWRENCE       NEW YORK       NY 718/384-0008
;
run;
```

CONTROL.RESULTS

```
data control.results;
   input ID    TREAT $8.   INITWT    WT3MOS    AGE;
   datalines;
     1   Other  166.28  146.98    35
     2   Other  214.42  210.22    54
     3   Other  172.46  159.42    33
     5   Other  175.41  160.66    37
     6   Other  173.13  169.40    20
     7   Other  181.25  170.94    30
    10   Other  239.83  214.48    48
    11   Other  175.32  162.66    51
    12   Other  227.01  211.06    29
    13   Other  274.82  251.82    31
;
run;
```

CONTROL.SLEEP

```
data control.sleep;
   input  GROUP TIME  SOL  WASO FNA TST;
   datalines;
   1.00   1.00  38.69  48.43    0     0
   2.36 424.50   0.00   0.00    0     0
   1.00   2.00  15.83   9.67    0     0
   2.16 500.30   0.00   0.00    0     0
   1.00   1.00  93.04  87.10    0     0
   1.86 302.40   0.00   0.00    0     0
   1.00   2.00  65.00  16.67    0     0
   1.50 305.00   0.00   0.00    0     0
```

1.00 1.00 19.82 74.38 0 0
2.00 359.20 0.00 0.00 0 0
1.00 2.00 13.75 13.90 0 0
1.40 378.13 0.00 0.00 0 0
1.00 1.00 47.35 60.52 0 0
2.89 248.10 0.00 0.00 0 0
1.00 2.00 72.14 86.07 0 0
4.14 308.50 0.00 0.00 0 0
1.00 1.00 99.65 151.79 0 0
3.86 263.93 0.00 0.00 0 0
1.00 2.00 65.35 118.33 0 0
2.71 374.17 0.00 0.00 0 0
1.00 1.00 47.15 105.36 0 0
2.50 254.60 0.00 0.00 0 0
1.00 2.00 66.00 92.60 0 0
2.20 297.40 0.00 0.00 0 0
1.00 1.00 30.84 84.97 0 0
3.24 242.90 0.00 0.00 0 0
1.00 2.00 7.86 23.86 0 0
1.28 340.70 0.00 0.00 0 0
1.00 1.00 117.41 36.93 0 0
1.00 204.20 0.00 0.00 0 0
1.00 2.00 50.42 24.86 0 0
1.85 231.00 0.00 0.00 0 0
1.00 1.00 6.50 41.15 0 0
2.67 308.00 0.00 0.00 0 0
1.00 2.00 2.00 0.00 0 0
0.00 454.40 0.00 0.00 0 0
2.00 1.00 54.29 84.93 0 0
3.43 394.70 0.00 0.00 0 0
2.00 2.00 13.57 48.00 0 0
1.00 405.00 0.00 0.00 0 0
2.00 1.00 4.43 58.43 0 0
2.89 375.70 0.00 0.00 0 0
2.00 2.00 5.00 49.28 0 0
3.57 423.60 0.00 0.00 0 0
2.00 1.00 32.50 38.43 0 0
4.57 357.20 0.00 0.00 0 0
2.00 2.00 17.14 33.14 0 0
3.57 315.70 0.00 0.00 0 0
2.00 1.00 33.57 83.43 0 0
4.36 282.50 0.00 0.00 0 0
2.00 2.00 22.85 37.86 0 0
2.42 288.57 0.00 0.00 0 0
2.00 1.00 53.21 120.00 0 0
3.50 366.80 0.00 0.00 0 0
2.00 2.00 29.00 97.40 0 0
2.80 388.00 0.00 0.00 0 0
2.00 1.00 56.79 67.73 0 0
3.19 326.00 0.00 0.00 0 0
2.00 2.00 52.50 64.16 0 0
4.00 45.80 0.00 0.00 0 0
2.00 1.00 23.50 35.19 0 0
6.60 416.80 0.00 0.00 0 0
2.00 2.00 25.71 38.43 0 0
8.50 446.30 0.00 0.00 0 0

```
   2.00   1.00   43.00   95.59    0     0
   1.75  288.90    0.00    0.00    0     0
   2.00   2.00   45.00   85.71    0     0
   1.43  272.10    0.00    0.00    0     0
   3.00   1.00   24.70   67.17    0     0
   3.90  399.90    0.00    0.00    0     0
   3.00   2.00    3.86   22.50    0     0
   3.00  425.14    0.00    0.00    0     0
   3.00   1.00   70.72   80.43    0     0
   3.00  286.30    0.00    0.00    0     0
   3.00   2.00   22.50  139.33    0     0
   3.50  283.50    0.00    0.00    0     0
   3.00   1.00   48.23   40.57    0     0
   1.65  407.20    0.00    0.00    0     0
   3.00   2.00   30.00   38.00    0     0
   1.57  372.90    0.00    0.00    0     0
   3.00   1.00   43.93   34.57    0     0
   1.29  393.40    0.00    0.00    0     0
   3.00   2.00   30.71   58.43    0     0
   2.28  348.60    0.00    0.00    0     0
   3.00   1.00   95.00   35.22    0     0
   2.06  409.90    0.00    0.00    0     0
   3.00   2.00   66.43   68.57    0     0
   2.14  345.60    0.00    0.00    0     0
   3.00   1.00   18.93   86.43    0     0
   5.36  349.50    0.00    0.00    0     0
   3.00   2.00   17.50  116.50    0     0
   5.00  337.70    0.00    0.00    0     0
   3.00   1.00  125.00   69.15    0     0
   2.00  242.50    0.00    0.00    0     0
   3.00   2.00   60.00   81.43    0     0
   1.57  304.30    0.00    0.00    0     0
   3.00   1.00   38.57   68.61    0     0
   3.64  394.50    0.00    0.00    0     0
   3.00   2.00   18.33   19.83    0     0
   2.00  448.83    0.00    0.00    0     0
   3.00   1.00   43.00  121.72    0     0
   2.63  247.80    0.00    0.00    0     0
   3.00   2.00   40.00   98.83    0     0
   1.86  199.67    0.00    0.00    0     0
   3.00   1.00   11.61   70.36    0     0
   2.86  348.40    0.00    0.00    0     0
   3.00   2.00   20.43   70.57    0     0
   3.14  335.40    0.00    0.00    0     0
;
run;
```

CONTROL.SYNDROME

```
data control.syndrome;
   input FLIGHT $ 1-3 @10 DATE DATE7.  @22 DEPART TIME5. ORIG $ 31-33
       DEST $ 39-41 MILES  BOARDED  CAPACITY;
   format date DATE7.;
```

```
   format depart TIME5.;
   informat date DATE7.;
   informat depart TIME5.;
   datalines;
114    01MAR18   7:10   LGA   LAX   2475   172    210
202    01MAR18   10:43  LGA   ORD    740   151    210
219    01MAR18   9:31   LGA   LON   3442   198    250
622    01MAR18   12:19  LGA   FRA   3857   207    250
132    01MAR18   15:35  LGA   YYZ    366   115    178
271    01MAR18   13:17  LGA   PAR   3635   138    250
302    01MAR18   20:22  LGA   WAS    229   105    180
114    02MAR18   7:10   LGA   LAX   2475   119    210
202    02MAR18   10:43  LGA   ORD    740   120    210
219    02MAR18   9:31   LGA   LON   3442   147    250
622    02MAR18   12:19  LGA   FRA   3857   176    250
132    02MAR18   15:35  LGA   YYZ    366   106    178
302    02MAR18   20:22  LGA   WAS    229    78    180
271    02MAR18   13:17  LGA   PAR   3635   104    250
114    03MAR18   7:10   LGA   LAX   2475   197    210
202    03MAR18   10:43  LGA   ORD    740   118    210
219    03MAR18   9:31   LGA   LON   3442   197    250
622    03MAR18   12:19  LGA   FRA   3857   180    250
132    03MAR18   15:35  LGA   YYZ    366    75    178
271    03MAR18   13:17  LGA   PAR   3635   147    250
302    03MAR18   20:22  LGA   WAS    229   123    180
114    04MAR18   7:10   LGA   LAX   2475   178    210
202    04MAR18   10:43  LGA   ORD    740   148    210
219    04MAR18   9:31   LGA   LON   3442   232    250
622    04MAR18   12:19  LGA   FRA   3857   137    250
132    04MAR18   15:35  LGA   YYZ    366   117    178
271    04MAR18   13:17  LGA   PAR   3635   146    250
302    04MAR18   20:22  LGA   WAS    229   115    180
114    05MAR18   7:10   LGA   LAX   2475   117    210
202    05MAR18   10:43  LGA   ORD    740   104    210
219    05MAR18   9:31   LGA   LON   3442   160    250
622    05MAR18   12:19  LGA   FRA   3857   185    250
132    05MAR18   15:35  LGA   YYZ    366   157    178
271    05MAR18   13:17  LGA   PAR   3635   177    250
114    06MAR18   7:10   LGA   LAX   2475   128    210
202    06MAR18   10:43  LGA   ORD    740   115    210
219    06MAR18   9:31   LGA   LON   3442   163    250
132    06MAR18   15:35  LGA   YYZ    366   150    178
302    06MAR18   20:22  LGA   WAS    229    66    180
114    07MAR18   7:10   LGA   LAX   2475   160    210
202    07MAR18   10:43  LGA   ORD    740   175    210
219    07MAR18   9:31   LGA   LON   3442   241    250
622    07MAR18   12:19  LGA   FRA   3857   210    250
132    07MAR18   15:35  LGA   YYZ    366   164    178
271    07MAR18   13:17  LGA   PAR   3635   155    250
302    07MAR18   20:22  LGA   WAS    229   135    180
;
run;
```

CONTROL.TENSION

```
data control.tension;
   input TENSION $8.  CHD $8.  COUNT ;
   datalines;
yes     yes      97
yes     no      307
no      yes     200
no      no     1409
;
run;
```

CONTROL.TEST2

```
data control.test2;
   input STD1 $ TEST1 $  STD2 $   TEST2 $    WT;
   datalines ;
neg     neg    neg     neg    509
neg     neg    neg     pos     4
neg     neg    pos     neg    17
neg     neg    pos     pos     3
neg     pos    neg     neg    13
neg     pos    neg     pos     8
neg     pos    pos     pos     8
pos     neg    neg     neg    14
pos     neg    neg     pos     1
pos     neg    pos     neg    17
pos     neg    pos     pos     9
pos     pos    neg     neg     7
pos     pos    neg     pos     4
pos     pos    pos     neg     9
pos     pos    pos     pos   170
;
run;
```

CONTROL.TRAIN

```
data control.train;
   input NAME $ 1-16 IDNUM $ 17-24;
   datalines;
Capalleti, Jimmy    2355
Chen, Len           5889
Davis, Brad         3878
```

```
Leung, Brenda      4409
Patel, Mary        2398
Smith, Robert      5862
Zook, Carla        7385
;
run;
```

CONTROL.VISION

```
data control.vision;
   input RIGHT LEFT COUNT;
   datalines;
    1      1   1520
    1      2    266
    1      3    124
    1      4     66
    2      1    234
    2      2   1512
    2      3    432
    2      4     78
    3      1    117
    3      2    362
    3      3   1772
    3      4    205
    4      1     36
    4      2     82
    4      3    179
    4      4    492
;
run;
```

CONTROL.WEIGHT

```
data control.weight (label='California Results');
   input ID TREAT $8.  IBW INITWT WT3MOS WT6MOS WT9MOS AGE MMPI1 MMPI2
       MMPI3 MMPI4 MMPI5;
   datalines;
 1  Other   149  166.28  146.98  138.26    .     35   62    68    67   55    67
 2  Other   137  214.42  210.22    .   213.87    54   57    56    59   57    47
 3  Other   138  172.46  159.42  146.01  143.84    33   54    69    63   87    34
 5  Other   122  175.41  160.66  154.30    .     37   56    67    64   71    32
 6  Other   134  173.13  169.40  176.12    .     20   42    51    63   71    45
 7  Other   160  181.25  170.94    .     .     30   72    58    70   71    80
 9  Other   152  212.83  179.93  169.74  164.47  49100   65    87    71   74     4
10  Other   145  239.83  214.48  208.28    .     48   56    51    56   53    53
11  Other   158  175.32  162.66  161.39    .     51   66    60    71   62    84
12  Other   174  227.01  211.06  202.87  205.17    29   77    70    69   65    64
13  Other   137  274.82  251.82  248.18    .     31   66    82    66   69    55
15  Other   152  168.75  156.58  154.61  156.58    42   52    58    65   67    51
```

```
 16 Other   162 187.81 172.07   .     .     40   57   68   67   67   74
 17 Other   123 226.63   .   219.72   .     21   58   49   59   74   70
 18 Other   146 176.03 160.27 160.27   .     41   48   55   49   68   43
 19 Other   166 190.96 159.04   .     .     32   53   52   51   62   71
 21 Other   148 165.54   .   166.22   .     48   57   60   65   74   55
 22 Other   125 193.60 184.00   .     .     28   64   67   70   69   54
 24 Other   152 267.43 230.26 206.09   .     30   48   45   55   50   41
 25 Other   151 193.38 185.43   .     .     33   54   99   67   75   74
 26 Other   134 252.61 227.61 217.72 223.88   31   62   51   70   57   39
 27 Other   140 193.93 191.43 196.43   .     25   54   65   66   55   55
 29 Other   119 182.77   .     .     .     38   66   63   64   60   41
 30 Other   134 189.93 172.39 175.37   .     35   70   92   73   98   39
 31 Other   138 190.22 181.88 178.99 181.16   36   56   69   61   62   34
 32 Other   134 182.09 169.40 163.81 163.81   34   42   65   54   55   41
 34 Other   133 200.00 189.47   .     .     24   52   61   64   62   39
 35 Other   149 221.81 216.11   .     .     46   66   52   69   71   61
 36 Other   133 241.35 247.37   .     .     34   50   65   57   74   47
 37 Other   134 223.13 217.91   .     .     33   64   63   63   60   41
 38 Other   140 235.36 228.57 210.71   .     50   56   61   70   74   39
 39 Other   125 178.60 178.40   .     .     39   50   73   58   58   32
 40 Other   143 243.01 226.57 210.49   .     38   64   66   70   54   46
 41 Other   134 282.65 239.55   .     .     26   66   65   75   74   53
 44 Other   139 282.37 258.99 238.13 241.01   43   66   69   68   65   39
 45 Other   134 216.04 182.09   .     .     39   50   55   59   67   43
 46 Other   115 190.00 171.30   .     .     36   64   59   58   58   44
 47 Other   134 175.19 167.16   .     .     43   57   48   52   71   47
 48 Other   118 179.87   .     .     .     37   52   57   45   50   30
 50 Other   137 173.54 166.97 164.60   .     32   54   76   63   69   25
 51 Other   125 180.60 162.40 152.00 157.60   32   50   56   64   53   53
 52 Other   155 235.32 225.16 210.32 208.39   35   62   64   55   60   51
 53 Other   143 183.39 169.23   .     .     38   64   57   72   81   61
 54 Other   131 212.60 208.40 211.45   .     27   50   67   53   74   47
 63 Surgery 146 219.18 167.12 139.73 119.18   35   58   53   67   48   55
 65 Surgery 123 192.68 155.28 127.64 115.45   31   52   74   54   64   32
 66 Surgery 134 199.25 173.88 161.19 144.03   38   68   64   70   77   30
 71 Surgery 139 209.35 172.66 156.83 138.13   39   66   56   66   71   42
 77 Surgery 137 179.56 150.36 132.12 123.36   29   46   49   42   47   49
 80 Surgery 170 138.24 121.76 100.00   .     31   56   57   64   64   39
 81 Surgery 129 206.98 173.64 132.56 121.71   28   42   78   52   62   41
 84 Surgery 143 220.28 178.32   .     .     29   58   67   54   46   59
 85 Surgery 152 189.47 156.58 140.79 140.79   34   75   73   62   74   51
 86 Surgery 210 157.14 138.57 121.90 109.05   30   88   75   73   69   65
 92 Surgery 139 203.60 169.78 143.88   .     38   62   59   68   60   49
 93 Surgery 151 171.52 150.33 123.18 109.27   42   72   69   59   53   47
 95 Surgery 134 207.46 155.22   .     .     41   51   52   56   63   57
 96 Surgery 138 201.45 172.46 172.46 155.07   55   57   49   57   67   37
 97 Surgery 128 209.38 182.81 162.50 153.91   34   68   88   73   74    .
101 Surgery 166 185.54 146.39 139.76 132.53   42   82   80   89   79   51
102 Surgery 127 218.11 173.23 152.76 137.80   39   76   80   70   74   45
107 Surgery 128 227.34 192.97 184.38 170.31   49   50   51   59   50   55
110 Surgery 143 251.75 207.69 183.22 165.03   42   48   84   52   76   38
111 Surgery 152 197.37 164.47 148.68 132.89   56   90   70   86   76   70
112 Surgery 140 202.14 156.43 137.86   .     31   48   51   50   41   41
116 Surgery 139 273.38 235.25 194.24   .     31   58   55   66   81   45
117 Surgery 125 192.00 156.80 140.00 127.20   42   58   44   63   53   51
120 Surgery 119 277.31 231.93 208.40 192.44   31   60   69   59   95   34
```

```
122 Surgery 138 165.22 130.43 119.57   .    44   66   67   70   62   34
125 Surgery 152 257.89 200.00 182.89 134.21   39   47   84   53   88   74
126 Surgery 138 170.29 142.75   .    .    39   64   64   66   65   46
132 Surgery 149 208.05 173.83   .  126.85   34   82   57   75   65   45
133 Surgery 136 230.15 205.15 190.44 180.88   39   52   71   52   65   30
134 Surgery 168 177.98 140.48 119.64   .    45   70   77   62   67   69
137 Surgery 138 188.41 164.49 152.90 135.51   39   58   61   57   60   46
138 Surgery 141 268.09 220.57 185.82   .    32   52   59   66   74   53
158 Surgery 130 220.00 190.77 173.85   .    27   68   73   66   58   43
223 Surgery 157 220.38 185.35 152.23 138.85   43   78   76   70   57   53
266 Surgery 135 184.44 148.15 123.70   .    41   56   55   59   64   49
269 Surgery 155 201.29 151.61 140.65   .    57   82   84   86   77   73
293 Surgery 160 163.75 130.63   .    .    47   52   67   47   64   76
295 Surgery 128 211.72 172.66 151.56 137.50   45   93   65   80   60   41
298 Surgery 140 243.57 195.00   .  166.43   40   63   62   62   76   52
;
run;
```

CONTROL.WGHT

```
data control.wght (label='California Results');
   input ID TREAT $8.  IBW INITWT WT3MOS WT6MOS WT9MOS AGE MMPI1 MMPI2
        MMPI3 MMPI4 MMPI5;
   datalines;
 1 Other   149 166.28 146.98 138.26   .    35   62   68   67   55   67
 2 Other   137 214.42 210.22   .  213.87   54   57   56   59   57   47
 3 Other   138 172.46 159.42 146.01 143.84   33   54   69   63   87   34
 5 Other   122 175.41 160.66 154.30   .    37   56   67   64   71   32
 6 Other   134 173.13 169.40 176.12   .    20   42   51   63   71   45
 7 Other   160 181.25 170.94   .    .    30   72   58   70   71   80
 9 Other   152 212.83 179.93 169.74 164.47 49100   65   87   71   74    4
10 Other   145 239.83 214.48 208.28   .    48   56   51   56   53   53
11 Other   158 175.32 162.66 161.39   .    51   66   60   71   62   84
12 Other   174 227.01 211.06 202.87 205.17   29   77   70   69   65   64
13 Other   137 274.82 251.82 248.18   .    31   66   82   66   69   55
15 Other   152 168.75 156.58 154.61 156.58   42   52   58   65   67   51
16 Other   162 187.81 172.07   .    .    40   57   68   67   67   74
17 Other   123 226.63   .  219.72   .    21   58   49   59   74   70
18 Other   146 176.03 160.27 160.27   .    41   48   55   49   68   43
19 Other   166 190.96 159.04   .    .    32   53   52   51   62   71
21 Other   148 165.54   .  166.22   .    48   57   60   65   74   55
22 Other   125 193.60 184.00   .    .    28   64   67   70   69   54
24 Other   152 267.43 230.26 206.09   .    30   48   45   55   50   41
25 Other   151 193.38 185.43   .    .    33   54   99   67   75   74
26 Other   134 252.61 227.61 217.72 223.88   31   62   51   70   57   39
27 Other   140 193.93 191.43 196.43   .    25   54   65   66   55   55
29 Other   119 182.77   .    .    .    38   66   63   64   60   41
30 Other   134 189.93 172.39 175.37   .    35   70   92   73   98   39
31 Other   138 190.22 181.88 178.99 181.16   36   56   69   61   62   34
32 Other   134 182.09 169.40 163.81 163.81   34   42   65   54   55   41
34 Other   133 200.00 189.47   .    .    24   52   61   64   62   39
35 Other   149 221.81 216.11   .    .    46   66   52   69   71   61
36 Other   133 241.35 247.37   .    .    34   50   65   57   74   47
```

```
 37 Other   134 223.13 217.91   .     .     33   64   63   63   60   41
 38 Other   140 235.36 228.57 210.71   .     50   56   61   70   74   39
 39 Other   125 178.60 178.40   .     .     39   50   73   58   58   32
 40 Other   143 243.01 226.57 210.49   .     38   64   66   70   54   46
 41 Other   134 282.65 239.55   .     .     26   66   65   75   74   53
 44 Other   139 282.37 258.99 238.13 241.01   43   66   69   68   65   39
 45 Other   134 216.04 182.09   .     .     39   50   55   59   67   43
 46 Other   115 190.00 171.30   .     .     36   64   59   58   58   44
 47 Other   134 175.19 167.16   .     .     43   57   48   52   71   47
 48 Other   118 179.87   .     .     .     37   52   57   45   50   30
 50 Other   137 173.54 166.97 164.60   .     32   54   76   63   69   25
 51 Other   125 180.60 162.40 152.00 157.60   32   50   56   64   53   53
 52 Other   155 235.32 225.16 210.32 208.39   35   62   64   55   60   51
 53 Other   143 183.39 169.23   .     .     38   64   57   72   81   61
 54 Other   131 212.60 208.40 211.45   .     27   50   67   53   74   47
 63 Surgery 146 219.18 167.12 139.73 119.18   35   58   53   67   48   55
 65 Surgery 123 192.68 155.28 127.64 115.45   31   52   74   54   64   32
 66 Surgery 134 199.25 173.88 161.19 144.03   38   68   64   70   77   30
 71 Surgery 139 209.35 172.66 156.83 138.13   39   66   56   66   71   42
 77 Surgery 137 179.56 150.36 132.12 123.36   29   46   49   42   47   49
 80 Surgery 170 138.24 121.76 100.00   .     31   56   57   64   64   39
 81 Surgery 129 206.98 173.64 132.56 121.71   28   42   78   52   62   41
 84 Surgery 143 220.28 178.32   .     .     29   58   67   54   46   59
 85 Surgery 152 189.47 156.58 140.79 140.79   34   75   73   62   74   51
 86 Surgery 210 157.14 138.57 121.90 109.05   30   88   75   73   69   65
 92 Surgery 139 203.60 169.78 143.88   .     38   62   59   68   60   49
 93 Surgery 151 171.52 150.33 123.18 109.27   42   72   69   59   53   47
 95 Surgery 134 207.46 155.22   .     .     41   51   52   56   63   57
 96 Surgery 138 201.45 172.46 172.46 155.07   55   57   49   57   67   37
 97 Surgery 128 209.38 182.81 162.50 153.91   34   68   88   73   74    .
101 Surgery 166 185.54 146.39 139.76 132.53   42   82   80   89   79   51
102 Surgery 127 218.11 173.23 152.76 137.80   39   76   80   70   74   45
107 Surgery 128 227.34 192.97 184.38 170.31   49   50   51   59   50   55
110 Surgery 143 251.75 207.69 183.22 165.03   42   48   84   52   76   38
111 Surgery 152 197.37 164.47 148.68 132.89   56   90   70   86   76   70
112 Surgery 140 202.14 156.43 137.86   .     31   48   51   50   41   41
116 Surgery 139 273.38 235.25 194.24   .     31   58   55   66   81   45
117 Surgery 125 192.00 156.80 140.00 127.20   42   58   44   63   53   51
120 Surgery 119 277.31 231.93 208.40 192.44   31   60   69   59   95   34
122 Surgery 138 165.22 130.43 119.57   .     44   66   67   70   62   34
125 Surgery 152 257.89 200.00 182.89 134.21   39   47   84   53   88   74
126 Surgery 138 170.29 142.75   .     .     39   64   64   66   65   46
132 Surgery 149 208.05 173.83   .   126.85   34   82   57   75   65   45
133 Surgery 136 230.15 205.15 190.44 180.88   39   52   71   52   65   30
134 Surgery 168 177.98 140.48 119.64   .     45   70   77   62   67   69
137 Surgery 138 188.41 164.49 152.90 135.51   39   58   61   57   60   46
138 Surgery 141 268.09 220.57 185.82   .     32   52   59   66   74   53
158 Surgery 130 220.00 190.77 173.85   .     27   68   73   66   58   43
223 Surgery 157 220.38 185.35 152.23 138.85   43   78   76   70   57   53
266 Surgery 135 184.44 148.15 123.70   .     41   56   55   59   64   49
269 Surgery 155 201.29 151.61 140.65   .     57   82   84   86   77   73
293 Surgery 160 163.75 130.63   .     .     47   52   67   47   64   76
295 Surgery 128 211.72 172.66 151.56 137.50   45   93   65   80   60   41
298 Surgery 140 243.57 195.00   .   166.43   40   63   62   62   76   52
;
run;
```

CUSTOMER_RESPONSE

```
data customer_response;
  input Customer Factor1-Factor4 Source1-Source3
      Quality1-Quality3;
  datalines;
1 . . 1 1 1 1 . 1 . .
2 1 1 . 1 1 1 . 1 1 .
3 . . 1 1 1 1 . . . .
4 1 1 . 1 . 1 . . . 1
5 . 1 . 1 1 . . . . 1
6 . 1 . 1 1 . . . . .
7 . 1 . 1 1 . . 1 . .
8 1 . . 1 1 1 . 1 1 .
9 1 1 . 1 1 . . . . 1
10 1 . . 1 1 1 . 1 1 .
11 1 1 1 1 1 . 1 . 1 1 1
12 1 1 . 1 1 1 . . . .
13 1 1 . 1 . 1 . 1 1 .
14 1 1 . 1 1 1 . . . .
15 1 1 . 1 . 1 . 1 1 1
16 1 . . 1 1 . . 1 . .
17 1 1 . 1 1 1 . . 1 .
18 1 1 . 1 1 1 1 . . 1
19 . 1 . 1 1 1 1 . 1 .
20 1 . . 1 1 1 . 1 1 1
21 . . . 1 1 1 . 1 . .
22 . . . 1 1 1 . 1 1 .
23 1 . . 1 . . . . . 1
24 . 1 . 1 1 . . 1 . 1
25 1 1 . 1 1 . . . 1 1
26 1 1 . 1 1 . . 1 . .
27 1 . . 1 1 . . . 1 .
28 1 1 . 1 . . . 1 1 1
29 1 . . 1 1 1 . 1 . 1
30 1 . 1 1 1 . . 1 1 .
31 . . . 1 1 . . 1 1 .
32 1 1 1 1 1 . . 1 1 1
33 1 . . 1 1 . . 1 . 1
34 . . 1 1 . . . 1 1 .
35 1 1 1 1 1 . 1 1 . .
36 1 1 1 1 . 1 . 1 . .
37 1 1 . 1 . . . 1 . .
38 . . . 1 1 1 . 1 . .
39 1 1 . 1 1 . . 1 . 1
40 1 . . 1 . . 1 1 . 1
41 1 . . 1 1 1 1 1 . 1
42 1 1 1 1 . . 1 1 . .
43 1 . . 1 1 1 . 1 . .
44 1 . 1 1 . 1 . 1 . 1
```

45 . . . 1 . . 1 . . 1
46 . . . 1 1 . . . 1 .
47 1 1 . 1 . . 1 1 . .
48 1 . 1 1 1 . 1 1 . .
49 . . 1 1 1 1 . 1 . 1
50 . 1 . 1 1 . . 1 1 .
51 1 . 1 1 1 1 . . . .
52 1 1 1 1 1 1 1 . 1 . .
53 . 1 1 1 . 1 . 1 1 1
54 1 . . 1 1 . . 1 1 .
55 1 1 . 1 1 1 . 1 . .
56 1 . . 1 1 . . 1 1 .
57 1 1 . 1 1 . 1 . . 1
58 . 1 . 1 . 1 . . 1 1
59 1 1 1 1 . . 1 1 1 .
60 . 1 1 1 1 1 . . 1 1
61 1 1 1 1 1 1 1 . 1 . .
62 1 1 . 1 1 . . 1 1 .
63 . . . 1 . . . 1 1 1
64 1 . . 1 1 1 . 1 . .
65 1 . . 1 1 1 . 1 . .
66 1 . . 1 1 1 1 1 1 .
67 1 1 . 1 1 1 . 1 1 .
68 1 1 . 1 1 1 . 1 1 .
69 1 1 . 1 1 . 1 . . .
70 . . . 1 1 1 . 1 . .
71 1 . . 1 1 . 1 . . 1
72 1 . 1 1 1 1 . . 1 .
73 1 1 . 1 . 1 . 1 1 .
74 1 1 1 1 1 1 1 . 1 . .
75 . 1 . 1 1 1 . . 1 .
76 1 1 . 1 1 1 . 1 1 1
77 . . . 1 1 1 . . . .
78 1 1 1 1 1 1 1 . 1 1 .
79 1 . . 1 1 1 . 1 1 .
80 1 1 1 1 1 1 . 1 1 . 1
81 1 1 . 1 1 1 1 1 1 1 .
82 . . . 1 1 1 1 . . .
83 1 1 . 1 1 1 . 1 1 .
84 1 . . 1 1 . . 1 1 .
85 . . . 1 . 1 . 1 . .
86 1 . . 1 1 1 . 1 1 1
87 1 1 . 1 1 1 . 1 . .
88 . . . 1 . 1 . . . .
89 1 . . 1 . 1 . . 1 1
90 1 1 . 1 1 1 . 1 . 1
91 . . . 1 1 . . . 1 .
92 1 . . 1 1 1 . 1 1 .
93 1 . . 1 1 . . 1 1 .
94 1 . . 1 1 1 1 1 . .
95 1 . . 1 . 1 1 1 1 .
96 1 . 1 1 1 1 . . 1 .
97 1 1 . 1 1 . . . 1 .
98 1 . 1 1 1 1 1 1 . .
99 1 1 . 1 1 1 1 1 1 .
100 1 . 1 1 1 . . . 1 1

```
101 1 . 1 1 1 1 . . . .
102 1 . . 1 1 . 1 1 . .
103 1 1 . 1 1 1 . 1 . .
104 . . . 1 1 1 . 1 1 1
105 1 . 1 1 1 . . 1 . 1
106 1 1 1 1 1 1 1 1 1 1
107 1 1 1 1 . . . 1 . 1
108 1 . . 1 . 1 1 1 . .
109 . 1 . 1 1 . . 1 1 .
110 1 . . 1 . . . . . .
111 1 . . 1 1 1 . 1 1 .
112 1 1 . 1 1 1 . . . 1
113 1 1 . 1 1 . 1 1 1 .
114 1 1 . 1 1 . . . . .
115 1 1 . 1 1 . . 1 . .
116 . 1 . 1 1 1 1 1 . .
117 . 1 . 1 1 1 . . . .
118 . 1 1 1 1 . . 1 1 .
119 . . . 1 . . . 1 . .
120 1 1 . 1 . . . . 1 .
;
```

DJIA

```
data djia;
    input Year HighDate date9. High LowDate date9. Low;
    format highdate lowdate date9.;
    datalines;
1968 03DEC1968   985.21 21MAR1968   825.13
1969 14MAY1969   968.85 17DEC1969   769.93
1970 29DEC1970   842.00 06MAY1970   631.16
1971 28APR1971   950.82 23NOV1971   797.97
1972 11DEC1972  1036.27 26JAN1972   889.15
1973 11JAN1973  1051.70 05DEC1973   788.31
1974 13MAR1974   891.66 06DEC1974   577.60
1975 15JUL1975   881.81 02JAN1975   632.04
1976 21SEP1976  1014.79 02JAN1976   858.71
1977 03JAN1977   999.75 02NOV1977   800.85
1978 08SEP1978   907.74 28FEB1978   742.12
1979 05OCT1979   897.61 07NOV1979   796.67
1980 20NOV1980  1000.17 21APR1980   759.13
1981 27APR1981  1024.05 25SEP1981   824.01
1982 27DEC1982  1070.55 12AUG1982   776.92
1983 29NOV1983  1287.20 03JAN1983  1027.04
1984 06JAN1984  1286.64 24JUL1984  1086.57
1985 16DEC1985  1553.10 04JAN1985  1184.96
1986 02DEC1986  1955.57 22JAN1986  1502.29
1987 25AUG1987  2722.42 19OCT1987  1738.74
1988 21OCT1988  2183.50 20JAN1988  1879.14
1989 09OCT1989  2791.41 03JAN1989  2144.64
1990 16JUL1990  2999.75 11OCT1990  2365.10
```

```
1991 31DEC1991  3168.83 09JAN1991  2470.30
1992 01JUN1992  3413.21 09OCT1992  3136.58
1993 29DEC1993  3794.33 20JAN1993  3241.95
1994 31JAN1994  3978.36 04APR1994  3593.35
1995 13DEC1995  5216.47 30JAN1995  3832.08
1996 27DEC1996  6560.90 10JAN1996  5032.94
1997 06AUG1997  8259.30 11APR1997  6391.69
1998 23NOV1998  9374.27 31AUG1998  7539.06
1999 31DEC1999 11497.12 22JAN1999  9120.67
2000 14JAN2000 11722.98 07MAR2000  9796.04
2001 21MAY2001 11337.92 21SEP2001  8235.81
2002 19MAR2002 10635.25 09OCT2002  7286.27
2003 31DEC2003 10453.92 11MAR2003  7524.06
2004 28DEC2004 10854.54 25OCT2004  9749.99
2005 04MAR2005 10940.55 20APR2005 10012.36
2006 27DEC2006 12510.57 20JAN2006 10667.39
2007 09OCT2007 14164.53 05MAR2007 12050.41
2008 02MAY2008 13058.20 10OCT2008  8451.19
2009 24DEC2009 10520.10 09MAR2009  6507.O4
2010 31DEC2010 11577.51 02JUL2010  9686.48
2011 29APR2011 12810.54 23SEP2011 10771.48
2012 05OCT2012 13610.15 01JUN2012 12118.57
2013 31DEC2013 16576.66 04JAN2013 13435.21
2014 05DEC2014 17958.79 11APR2014 16026.75
2015 15MAY2015 18272.56 04SEP2015 16102.38
2016 23DEC2016 19993.81 12FEB2015 15973.84
2017 22DEC2017 24754.06 06JAN2017 19963.80
;
```

EDUCATION

```
data education;
  input State $14. +1 Code $ DropoutRate Expenditures MathScore Region $;
  label dropoutrate='Dropout Percentage - 2017'
        expenditures='Expenditure Per Pupil - 2017'
         mathscore='8th Grade Math Exam - 2018';
  datalines;
Alabama        AL 22.3 3197 252 SE
Alaska         AK 35.8 7716 .   W
Arizona        AZ 31.2 3902 259 W
Arkansas       AR 11.5 3273 256 SE
California     CA 32.7 4121 256 W
Colorado       CO 24.7 4408 267 W
Connecticut    CT 16.8 6857 270 NE
Delaware       DE 28.5 5422 261 NE
Florida        FL 38.5 4563 255 SE
Georgia        GA 27.9 3852 258 SE
Hawaii         HI 18.3 4121 251 W
Idaho          ID 21.8 2838 272 W
Illinois       IL 21.5 4906 260 MW
Indiana        IN 13.8 4284 267 MW
```

```
Iowa          IA 13.6 4285 278 MW
Kansas        KS 17.9 4443 .   MW
Kentucky      KY 32.7 3347 256 SE
Louisiana     LA 43.1 3317 246 SE
Maine         ME 22.5 4744 .   NE
Maryland      MD 26.0 5758 260 NE
Massachusetts MA 28.0 5979 .   NE
Michigan      MI 29.3 5116 264 MW
Minnesota     MN 11.4 4755 276 MW
Mississippi   MS 39.9 2874 .   SE
Missouri      MO 26.5 4263 .   MW
Montana       MT 15.0 4293 280 W
Nebraska      NE 13.9 4360 276 MW
Nevada        NV 28.1 3791 .   W
New Hampshire NH 25.9 4807 273 NE
New Jersey    NE 20.4 7549 269 NE
New Mexico    NM 28.5 3473 256 W
New York      NY 35.0 .    261 NE
North Carolina NC 31.2 3874 250 SE
North Dakota  ND 12.1 3952 281 MW
Ohio          OH 24.4 4649 264 MW
;
```

EMPDATA

```
data empdata;
input IdNumber $ 1-4 LastName $ 8-18 FirstName $ 19-28
    City $ 29-41 State $ 42-43
    Gender $ 45 JobCode $ 49-51 Salary 55-60 @63 Birth date9.
    @73 Hired date9. HomePhone $ 85-98;
format birth hired date9.;
datalines;
1919  Adams     Gerald    Stamford    CT M  TA2  34376  12SEP1990  04JUN2007  203/781-1255
1653  Ahmad     Azeem     Bridgeport  CT F  ME2  35108  15OCT1984  09AUG2010  203/675-7715
1400  Alvarez   Gloria    New York    NY M  ME1  29769  05NOV1987  16OCT2010  212/586-0808
1350  Arthur    Barbara   New York    NY F  FA3  32886  31AUG1985  29JUL2010  718/383-1549
1401  Avery     Jerry     Paterson    NJ M  TA3  38822  13DEC1970  17NOV2005  201/732-8787
1499  Barefoot  Joseph    Princeton   NJ M  ME3  43025  26APR1974  07JUN2010  201/812-5665
1101  Basquez   Richardo  New York    NY M  SCP  18723  06JUN1982  01OCT2010  212/586-8060
1333  Beaulieu  Armando   Stamford    CT M  PT2  88606  30MAR1983  10FEB2011  203/781-1777
1402  Blalock   Ralph     New York    NY M  TA2  32615  17JAN1983  02DEC2010  718/384-2849
1479  Bostic    Marie     New York    NY F  TA3  38785  22DEC1988  05OCT2007  718/384-8816
1403  Bowden    Earl      Bridgeport  CT M  ME1  28072  28JAN1989  21DEC2016  203/675-3434
1739  Boyce     Jonathan  New York    NY M  PT1  66517  25DEC1984  27JAN2011  212/587-1247
1658  Bradley   Jeremy    New York    NY M  SCP  17943  08APR1987  29FEB2012  212/587-3622
1428  Brady     Christine Stamford    CT F  PT1  68767  04APR1980  16NOV2011  203/781-1212
1782  Brown     Jason     Stamford    CT M  ME2  35345  04DEC1980  22FEB2012  203/781-0019
1244  Bryant    Leonard   New York    NY M  ME2  36925  31AUG1983  17JAN2008  718/383-3334
1383  Burnette  Thomas    New York    NY M  BCK  25823  25JAN1985  20OCT2009  718/384-3569
1574  Cahill    Marshall  New York    NY M  FA2  28572  27APR1980  20DEC2012  718/383-2338
1789  Canales   Viviana   New York    NY M  SCP  18326  23JAN1977  11APR2008  212/587-9000
```

```
1404 Carter    Donald    New York     NY M  PT2  91376  24FEB1973  01JAN2010  718/384-2946
1437 Carter    Dorothy   Bridgeport   CT F  A3   33104  20SEP1980  31AUG2004  203/675-4117
1639 Carter    Karen     Stamford     CT F  A3   40260  26JUN1977  28JAN2014  203/781-8839
1269 Caston    Franklin  Stamford     CT M  NA1  41690  03MAY1972  28NOV2012  203/781-3335
1065 Chapman   Neil      New York     NY M  ME2  35090  26JAN1977  07JAN2007  718/384-5618
1876 Chin      Jack      New York     NY M  TA3  39675  20MAY1978  27APR2015  212/588-5634
1037 Chow      Jane      Stamford     CT F  TA1  28558  10APR1984  13SEP2012  203/781-8868
1129 Cook      Brenda    New York     NY F  ME2  34929  08DEC1981  17AUG2011  718/383-2313
1988 Cooper    Anthony   New York     NY M  FA3  32217  30NOV1979  18SEP2004  212/587-1228
1405 Davidson  Jason     Paterson     NJ M  SCP  18056  05MAR1986  26JAN2012  201/732-2323
1430 Dean      Sandra    Bridgeport   CT F  TA2  32925  28FEB1982  27APR2017  203/675-1647
1983 Dean      Sharon    New York     NY F  FA3  33419  28FEB1962  30APR2005  718/384-1647
1134 Delgado   Maria     Stamford     CT F  TA2  33462  05MAR1989  21DEC2016  203/781-1528
1118 Dennis    Roger     New York     NY M  PT3  111379 16JAN1964  18DEC2000  718/383-1122
1438 Desai     Aakash    Stamford     CT F  TA3  39223  15MAR1985  18NOV2007  203/781-2229
1125 Dunlap    Donna     New York     NY F  FA2  28888  08NOV1988  11DEC2007  718/383-2094
1475 Eaton     Alicia    New York     NY F  FA2  27787  15DEC1981  13JUL2010  718/383-2828
1117 Edgerton  Joshua    New York     NY M  TA3  39771  05JUN1983  13AUG2012  212/588-1239
1935 Fernandez Katrina   Bridgeport   CT F  NA2  51081  28MAR1974  16OCT2011  203/675-2962
1124 Fields    Diana     White Plains NY F  FA1  23177  10JUL1978  01OCT2010  914/455-2998
1422 Fletcher  Marie     Princeton    NJ F  FA1  22454  04JUN1984  06APR2011  201/812-0902
1616 Flowers   Annette   New York     NY F  TA2  34137  01MAR1990  04JUN2013  718/384-3329
1406 Foster    Gerald    Bridgeport   CT M  ME2  35185  08MAR1981  17FEB2007  203/675-6363
1120 Garcia    Jack      New York     NY M  ME1  28619  11SEP1992  07OCT2013  718/384-4930
1094 Gomez     Alan      Bridgeport   CT M  FA1  22268  02APR1990  17APR2011  203/675-7181
1389 Gordon    Levi      New York     NY M  BCK  25028  15JUL1979  18AUG2010  718/384-9326
1905 Graham    Alvin     New York     NY M  PT1  65111  16APR1992  29MAY2012  212/586-8815
1407 Grant     Daniel    Mt. Vernon   NY M  PT1  68096  23MAR1989  18MAR2010  914/468-1616
1114 Green     Janice    New York     NY F  TA2  32928  18SEP1989  27JUN2007  212/588-1092
;
```

ENERGY

```
data energy;
  length State $2;
  input Region Division state $ Type Expenditures;
  datalines;
1 1 ME 1 708
1 1 ME 2 379
1 1 NH 1 597
1 1 NH 2 301
1 1 VT 1 353
1 1 VT 2 188
1 1 MA 1 3264
1 1 MA 2 2498
1 1 RI 1 531
1 1 RI 2 358
1 1 CT 1 2024
1 1 CT 2 1405
1 2 NY 1 8786
1 2 NY 2 7825
```

```
1 2 NJ 1 4115
1 2 NJ 2 3558
1 2 PA 1 6478
1 2 PA 2 3695
4 3 MT 1 322
4 3 MT 2 232
4 3 ID 1 392
4 3 ID 2 298
4 3 WY 1 194
4 3 WY 2 184
4 3 CO 1 1215
4 3 CO 2 1173
4 3 NM 1 545
4 3 NM 2 578
4 3 AZ 1 1694
4 3 AZ 2 1448
4 3 UT 1 621
4 3 UT 2 438
4 3 NV 1 493
4 3 NV 2 378
4 4 WA 1 1680
4 4 WA 2 1122
4 4 OR 1 1014
4 4 OR 2 756
4 4 CA 1 10643
4 4 CA 2 10114
4 4 AK 1 349
4 4 AK 2 329
4 4 HI 1 273
4 4 HI 2 298
;
```

EXP ライブラリ

EXP.RESULTS

次のセクションでは、EXP ライブラリの生データと DATA ステップを示します。

```
options ps=40 ls=64 nodate pageno=1;

LIBNAME exp 'library-name';

data exp.results;
  set exp.wght(firstobs=1 obs=11 keep=id treat initwt wt3mos
     age);
  if age>100 then delete;
run;
```

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

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

EXP.SUR

```
data exp.sur;
   input id  treat $  initwt  wt3mos  wt6mos  age;
   datalines;
14   surgery   203.60   169.78   143.88    38
17   surgery   171.52   150.33   123.18    42
18   surgery   207.46   155.22      .      41
;
run;
```

EXPREV

```
data exprev;
input Country $ 1-24 Emp_ID $ 25-32 Order_Date $  Ship_Date $  Sale_Type $ 67-75 Quantity Price Cost;

datalines;
Antarctica            99999999    1/1/18     1/7/18          Internet    2    92.60     20.70
Puerto Rico           99999999    1/1/18     1/5/18          Catalog    14    51.20     12.10
Virgin Islands (U.S.) 99999999    1/1/18     1/4/18          In Store   25    31.10     15.65
Aruba                 99999999    1/1/18     1/4/18          Catalog    30   123.70     59.00
Bahamas               99999999    1/1/18     1/4/18          Catalog     8   113.40     28.45
Bermuda               99999999    1/1/18     1/4/18          Catalog     7    41.00      9.25
Belize                120458      1/2/18     1/2/18          In Store    2   146.40     36.70
British Virgin Islands 99999999   1/2/18     1/5/18          Catalog    11    40.20     20.20
Canada                99999999    1/2/18     1/5/18          Catalog   100   11.80      5.00
```

```
Cayman Islands        120454      1/2/18     1/2/18         In Store   20    71.00     32.30
Costa Rica            99999999    1/2/18     1/6/18         Internet   31    53.00     26.60
Cuba                  121044      1/2/18     1/2/18         Internet   12    42.40     19.35
Dominican Republic    121040      1/2/18     1/2/18         Internet   13    48.00     23.95
El Salvador           99999999    1/2/18     1/6/18         Catalog    21    266.40    66.70
Guatemala             120931      1/2/18     1/2/18         In Store   13    144.40    65.70
Haiti                 121059      1/2/18     1/2/18         Internet   5     47.90     23.45
Honduras              120455      1/2/18     1/2/18         Internet   20    66.40     30.25
Jamaica               99999999    1/2/18     1/4/18         In Store   23    169.80    38.70
Mexico                120127      1/2/18     1/2/18         In Store   30    211.80    33.65
Montserrat            120127      1/2/18     1/2/18         In Store   19    184.20    36.90
Nicaragua             120932      1/2/18     1/2/18         Internet   16    122.00    28.75
Panama                99999999    1/2/18     1/6/18         Internet   20    88.20     38.40
Saint Kitts/Nevis     99999999    1/2/18     1/6/18         Internet   20    41.40     18.00
St. Helena            120360      1/2/18     1/2/18         Internet   19    94.70     47.45
St. Pierre/Miquelon   120842      1/2/18     1/6/18         Internet   16    103.80    47.25
Turks/Caicos Islands  120372      1/2/18     1/2/18         Internet   10    57.70     28.95
United States         120372      1/2/18     1/2/18         Internet   20    88.20     38.40
Anguilla              99999999    1/2/18     1/6/18         In Store   15    233.50    22.25
Antigua/Barbuda       120458      1/2/18     1/2/18         In Store   31    99.60     45.35
Argentina             99999999    1/2/18     1/6/18         In Store   42    408.80    87.15
Barbados              99999999    1/2/18     1/6/18         In Store   26    94.80     42.60
Bolivia               120127      1/2/18     1/2/18         In Store   26    66.00     16.60
Brazil                120127      1/2/18     1/2/18         Catalog    12    73.40     18.45
Chile                 120447      1/2/18     1/2/18         In Store   20    19.10     8.75
Colombia              121059      1/2/18     1/2/18         Internet   28    361.40    90.45
Dominica              121043      1/2/18     1/2/18         Internet   35    121.30    57.80
Ecuador               121042      1/2/18     1/2/18         In Store   11    100.90    50.55
Falkland Islands      120932      1/2/18     1/2/18         In Store   15    61.40     30.80
French Guiana         120935      1/2/18     1/2/18         Catalog    15    96.40     43.85
Grenada               120931      1/2/18     1/2/18         Catalog    19    56.30     25.05
Guadeloupe            120445      1/2/18     1/2/18         Internet   21    231.60    48.70
Guyana                120455      1/2/18     1/2/18         In Store   25    132.80    30.25
Martinique            120841      1/2/18     1/3/18         In Store   16    56.30     31.05
Netherlands Antilles  99999999    1/2/18     1/6/18         In Store   31    41.80     19.45
Paraguay              120603      1/2/18     1/2/18         Catalog    17    117.60    58.90
Peru                  120845      1/2/18     1/2/18         Catalog    12    93.80     41.75
St. Lucia             120845      1/2/18     1/2/18         Internet   19    64.30     28.65
Suriname              120538      1/3/18     1/3/18         Internet   22    110.80    29.35
;
run;
```

GROC

```
data groc;
  input Region $9. Manager $ Department $ Sales;
  datalines;
Southeast   Hayes     Paper      250
Southeast   Hayes     Produce    100
Southeast   Hayes     Canned     120
Southeast   Hayes     Meat       80
```

```
Southeast   Michaels   Paper      40
Southeast   Michaels   Produce    300
Southeast   Michaels   Canned     220
Southeast   Michaels   Meat       70
Northwest   Jeffreys   Paper      60
Northwest   Jeffreys   Produce    600
Northwest   Jeffreys   Canned     420
Northwest   Jeffreys   Meat       30
Northwest   Duncan     Paper      45
Northwest   Duncan     Produce    250
Northwest   Duncan     Canned     230
Northwest   Duncan     Meat       73
Northwest   Aikmann    Paper      45
Northwest   Aikmann    Produce    205
Northwest   Aikmann    Canned     420
Northwest   Aikmann    Meat       76
Southwest   Royster    Paper      53
Southwest   Royster    Produce    130
Southwest   Royster    Canned     120
Southwest   Royster    Meat       50
Southwest   Patel      Paper      40
Southwest   Patel      Produce    350
Southwest   Patel      Canned     225
Southwest   Patel      Meat       80
Northeast   Rice       Paper      90
Northeast   Rice       Produce    90
Northeast   Rice       Canned     420
Northeast   Rice       Meat       86
Northeast   Fuller     Paper      200
Northeast   Fuller     Produce    300
Northeast   Fuller     Canned     420
Northeast   Fuller     Meat       125
;
```

MATCH_11

```
data match_11;
  input Pair Low Age Lwt Race Smoke Ptd Ht UI @@;
  select(race);
    when (1) do;
      race1=0;
      race2=0;
    end;
    when (2) do;
      race1=1;
      race2=0;
    end;
    when (3) do;
      race1=0;
      race2=1;
    end;
```

```
  end;
  datalines;
1  0 14 135 1 0 0 0 0    1  1 14 101 3 1 1 0 0
2  0 15  98 2 0 0 0 0    2  1 15 115 3 0 0 0 1
3  0 16  95 3 0 0 0 0    3  1 16 130 3 0 0 0 0
4  0 17 103 3 0 0 0 0    4  1 17 130 3 1 1 0 1
5  0 17 122 1 1 0 0 0    5  1 17 110 1 1 0 0 0
6  0 17 113 2 0 0 0 0    6  1 17 120 1 1 0 0 0
7  0 17 113 2 0 0 0 0    7  1 17 120 2 0 0 0 0
8  0 17 119 3 0 0 0 0    8  1 17 142 2 0 0 1 0
9  0 18 100 1 1 0 0 0    9  1 18 148 3 0 0 0 0
10 0 18  90 1 1 0 0 1    10 1 18 110 2 1 1 0 0
11 0 19 150 3 0 0 0 0    11 1 19  91 1 1 1 0 1
12 0 19 115 3 0 0 0 0    12 1 19 102 1 0 0 0 0
13 0 19 235 1 1 0 1 0    13 1 19 112 1 1 0 0 1
14 0 20 120 3 0 0 0 1    14 1 20 150 1 1 0 0 0
15 0 20 103 3 0 0 0 0    15 1 20 125 3 0 0 0 1
16 0 20 169 3 0 1 0 1    16 1 20 120 2 1 0 0 0
17 0 20 141 1 0 1 0 1    17 1 20  80 3 1 0 0 1
18 0 20 121 2 1 0 0 0    18 1 20 109 3 0 0 0 0
19 0 20 127 3 0 0 0 0    19 1 20 121 1 1 1 0 1
20 0 20 120 3 0 0 0 0    20 1 20 122 2 1 0 0 0
21 0 20 158 1 0 0 0 0    21 1 20 105 3 0 0 0 0
22 0 21 108 1 1 0 0 1    22 1 21 165 1 1 0 1 0
23 0 21 124 3 0 0 0 0    23 1 21 200 2 0 0 0 0
24 0 21 185 2 1 0 0 0    24 1 21 103 3 0 0 0 0
25 0 21 160 1 0 0 0 0    25 1 21 100 3 0 1 0 0
26 0 21 115 1 0 0 0 0    26 1 21 130 1 1 0 1 0
27 0 22  95 3 0 0 1 0    27 1 22 130 1 1 0 0 0
28 0 22 158 2 0 1 0 0    28 1 22 130 1 1 1 0 1
29 0 23 130 2 0 0 0 0    29 1 23  97 3 0 0 0 1
30 0 23 128 3 0 0 0 0    30 1 23 187 2 1 0 0 0
31 0 23 119 3 0 0 0 0    31 1 23 120 3 0 0 0 0
32 0 23 115 3 1 0 0 0    32 1 23 110 1 1 1 0 0
33 0 23 190 1 0 0 0 0    33 1 23  94 3 1 0 0 0
34 0 24  90 1 1 1 0 0    34 1 24 128 2 0 1 0 0
35 0 24 115 1 0 0 0 0    35 1 24 132 3 0 0 1 0
36 0 24 110 3 0 0 0 0    36 1 24 155 1 1 1 0 0
37 0 24 115 3 0 0 0 0    37 1 24 138 1 0 0 0 0
38 0 24 110 3 0 1 0 0    38 1 24 105 2 1 0 0 0
39 0 25 118 1 1 0 0 0    39 1 25 105 3 0 1 1 0
40 0 25 120 3 0 0 0 1    40 1 25  85 3 0 0 0 1
41 0 25 155 1 0 0 0 0    41 1 25 115 3 0 0 0 0
42 0 25 125 2 0 0 0 0    42 1 25  92 1 1 0 0 0
43 0 25 140 1 0 0 0 0    43 1 25  89 3 0 1 0 0
44 0 25 241 2 0 0 1 0    44 1 25 105 3 0 1 0 0
45 0 26 113 1 1 0 0 0    45 1 26 117 1 1 1 0 0
46 0 26 168 2 1 0 0 0    46 1 26  96 3 0 0 0 0
47 0 26 133 3 1 1 0 0    47 1 26 154 3 0 1 1 0
48 0 26 160 3 0 0 0 0    48 1 26 190 1 1 0 0 0
49 0 27 124 1 1 0 0 0    49 1 27 130 2 0 0 0 1
50 0 28 120 3 0 0 0 0    50 1 28 120 3 1 1 0 1
51 0 28 130 3 0 0 0 0    51 1 28  95 1 1 0 0 0
52 0 29 135 1 0 0 0 0    52 1 29 130 1 0 0 0 1
53 0 30  95 1 1 0 0 0    53 1 30 142 1 1 1 0 0
54 0 31 215 1 1 0 0 0    54 1 31 102 1 1 1 0 0
```

```
55 0 32 121 3 0 0 0 0    55 1 32 105 1 1 0 0 0
56 0 34 170 1 0 1 0 0    56 1 34 187 2 1 0 1 0
;
```

PROCLIB.DELAY

```
data proclib.delay;
  input flight $3. +5 date date7. +2 orig $3. +3 dest $3. +3
      delaycat $15. +2 destype $15. +8 delay;
  informat date date7.;
  format date date7.;
  datalines;
114   01MAR18  LGA   LAX   1-10 Minutes    Domestic            8
202   01MAR18  LGA   ORD   No Delay       Domestic           -5
219   01MAR18  LGA   LON   11+ Minutes    International      18
622   01MAR18  LGA   FRA   No Delay       International      -5
132   01MAR18  LGA   YYZ   11+ Minutes    International      14
271   01MAR18  LGA   PAR   1-10 Minutes    International       5
302   01MAR18  LGA   WAS   No Delay       Domestic           -2
114   02MAR18  LGA   LAX   No Delay       Domestic            0
202   02MAR18  LGA   ORD   1-10 Minutes    Domestic            5
219   02MAR18  LGA   LON   11+ Minutes    International      18
622   02MAR18  LGA   FRA   No Delay       International       0
132   02MAR18  LGA   YYZ   1-10 Minutes    International       5
271   02MAR18  LGA   PAR   1-10 Minutes    International       4
302   02MAR18  LGA   WAS   No Delay       Domestic            0
114   03MAR18  LGA   LAX   No Delay       Domestic           -1
202   03MAR18  LGA   ORD   No Delay       Domestic           -1
219   03MAR18  LGA   LON   1-10 Minutes    International       4
622   03MAR18  LGA   FRA   No Delay       International      -2
132   03MAR18  LGA   YYZ   1-10 Minutes    International       6
271   03MAR18  LGA   PAR   1-10 Minutes    International       2
302   03MAR18  LGA   WAS   1-10 Minutes    Domestic            5
114   04MAR18  LGA   LAX   11+ Minutes    Domestic           15
202   04MAR18  LGA   ORD   No Delay       Domestic           -5
219   04MAR18  LGA   LON   1-10 Minutes    International       3
622   04MAR18  LGA   FRA   11+ Minutes    International      30
132   04MAR18  LGA   YYZ   No Delay       International      -5
271   04MAR18  LGA   PAR   1-10 Minutes    International       5
302   04MAR18  LGA   WAS   1-10 Minutes    Domestic            7
114   05MAR18  LGA   LAX   No Delay       Domestic           -2
202   05MAR18  LGA   ORD   1-10 Minutes    Domestic            2
219   05MAR18  LGA   LON   1-10 Minutes    International       3
622   05MAR18  LGA   FRA   No Delay       International      -6
132   05MAR18  LGA   YYZ   1-10 Minutes    International       3
271   05MAR18  LGA   PAR   1-10 Minutes    International       5
114   06MAR18  LGA   LAX   No Delay       Domestic           -1
202   06MAR18  LGA   ORD   No Delay       Domestic           -3
219   06MAR18  LGA   LON   11+ Minutes    International      27
132   06MAR18  LGA   YYZ   1-10 Minutes    International       7
302   06MAR18  LGA   WAS   1-10 Minutes    Domestic            1
```

```
114    07MAR18  LGA   LAX   No Delay       Domestic            -1
202    07MAR18  LGA   ORD   No Delay       Domestic            -2
219    07MAR18  LGA   LON   11+ Minutes    International       15
622    07MAR18  LGA   FRA   11+ Minutes    International       21
132    07MAR18  LGA   YYZ   No Delay       International       -2
271    07MAR18  LGA   PAR   1-10 Minutes   International       4
302    07MAR18  LGA   WAS   No Delay       Domestic            0
;
```

PROCLIB.EMP95

```
data proclib.emp95;
  input #1 idnum $4. @6 name $15.
       #2 address $42.
       #3 salary 6.;
  datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
2776 Robert Jones
12988 Wellington Farms Ave. Cary NC 27512
29025
8699 Jerry Capalleti
222 West L St. Oxford NC 27587
39985
2100 Lanny Engles
293 Manning Pl. Raleigh NC 27606
30998
9857 Kathy Krupski
1000 Taft Ave. Morrisville NC 27508
38756
0987 Dolly Lunford
2344 Persimmons Branch  Apex NC 27505
44010
3286 Hoa Nguyen
2818 Long St. Cary NC 27513
87734
6579 Bryan Samosky
3887 Charles Ave. Garner NC 27508
50234
3888 Kim Siu
5662 Magnolia Blvd Southeast Cary NC 27513
77558
;
```

PROCLIB.EMP96

```
data proclib.emp96;
  input #1 idnum $4. @6 name $15.
        #2 address $42.
        #3 salary 6.;
  datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
2776 Robert Jones
12988 Wellington Farms Ave. Cary NC 27511
29025
8699 Jerry Capalleti
222 West L St. Oxford NC 27587
39985
3278 Mary Cravens
211 N. Cypress St. Cary NC 27512
35362
2100 Lanny Engles
293 Manning Pl. Raleigh NC 27606
30998
9857 Kathy Krupski
100 Taft Ave. Morrisville NC 27508
40456
0987 Dolly Lunford
2344 Persimmons Branch Trail Apex NC 27505
45110
3286 Hoa Nguyen
2818 Long St. Cary NC 27513
89834
6579 Bryan Samosky
3887 Charles Ave. Garner NC 27508
50234
3888 Kim Siu
5662 Magnolia Blvd Southwest Cary NC 27513
79958
6544 Roger Monday
3004 Crepe Myrtle Court Raleigh NC 27604
47007
;
```

PROCLIB.INTERNAT

```
data proclib.internat;
  input flight $3.  +5 date date7. +2 dest $3. +8 boarded;
  informat date date7.;
  format date date7.;
  datalines;
219    01MAR18  LON      198
622    01MAR18  FRA      207
132    01MAR18  YYZ      115
271    01MAR18  PAR      138
219    02MAR18  LON      147
622    02MAR18  FRA      176
132    02MAR18  YYZ      106
271    02MAR18  PAR      172
219    03MAR18  LON      197
622    03MAR18  FRA      180
132    03MAR18  YYZ      75
271    03MAR18  PAR      147
219    04MAR18  LON      232
622    04MAR18  FRA      137
132    04MAR18  YYZ      117
271    04MAR18  PAR      146
219    05MAR18  LON      160
622    05MAR18  FRA      185
132    05MAR18  YYZ      157
271    05MAR18  PAR      177
219    06MAR18  LON      163
132    06MAR18  YYZ      150
219    07MAR18  LON      241
622    07MAR18  FRA      210
132    07MAR18  YYZ      164
271    07MAR18  PAR      155
;
```

PROCLIB.LAKES

```
data proclib.lakes;
   input region $ 1-2 lake $ 5-13 pol_a1 pol_a2 pol_b1-pol_b4;
   datalines;
NE Carr      0.24   0.99   0.95   0.36   0.44   0.67
NE Duraleigh 0.34   0.01   0.48   0.58   0.12   0.56
NE Charlie   0.40   0.48   0.29   0.56   0.52   0.95
NE Farmer    0.60   0.65   0.25   0.20   0.30   0.64
```

```
NW  Canyon    0.63    0.44    0.20    0.98    0.19    0.01
NW  Morris    0.85    0.95    0.80    0.67    0.32    0.81
NW  Golf      0.69    0.37    0.08    0.72    0.71    0.32
NW  Falls     0.01    0.02    0.59    0.58    0.67    0.02
SE  Pleasant  0.16    0.96    0.71    0.35    0.35    0.48
SE  Juliette  0.82    0.35    0.09    0.03    0.59    0.90
SE  Massey    1.01    0.77    0.45    0.32    0.55    0.66
SE  Delta     0.84    1.05    0.90    0.09    0.64    0.03
SW  Alumni    0.45    0.32    0.45    0.44    0.55    0.12
SW  New Dam   0.80    0.70    0.31    0.98    1.00    0.22
SW  Border    0.51    0.04    0.55    0.35    0.45    0.78
SW  Red       0.22    0.09    0.02    0.10    0.32    0.01
;
```

PROCLIB.MARCH

```
data proclib.march;
  input flight $3. +5 date date7. +3 depart time5. +2 orig $3.
      +3 dest $3.  +7 miles +6 boarded +6 capacity;
  format date date7. depart time5.;
  informat date date7. depart time5.;
  datalines;
114    01MAR18   7:10  LGA   LAX     2475     172     210
202    01MAR18  10:43  LGA   ORD      740     151      210
219    01MAR18   9:31  LGA   LON     3442     198     250
622    01MAR18  12:19  LGA   FRA     3857     207     250
132    01MAR18  15:35  LGA   YYZ      366     115     178
271    01MAR18  13:17  LGA   PAR     3635     138     250
302    01MAR18  20:22  LGA   WAS      229     105     180
114    02MAR18   7:10  LGA   LAX     2475     119     210
202    02MAR18  10:43  LGA   ORD      740     120     210
219    02MAR18   9:31  LGA   LON     3442     147     250
622    02MAR18  12:19  LGA   FRA     3857     176     250
132    02MAR18  15:35  LGA   YYZ      366     106     178
302    02MAR18  20:22  LGA   WAS      229      78     180
271    02MAR18  13:17  LGA   PAR     3635     104     250
114    03MAR18   7:10  LGA   LAX     2475     197     210
202    03MAR18  10:43  LGA   ORD      740     118     210
219    03MAR18   9:31  LGA   LON     3442     197     250
622    03MAR18  12:19  LGA   FRA     3857     180     250
132    03MAR18  15:35  LGA   YYZ      366      75     178
271    03MAR18  13:17  LGA   PAR     3635     147     250
302    03MAR18  20:22  LGA   WAS      229     123     180
114    04MAR18   7:10  LGA   LAX     2475     178     210
202    04MAR18  10:43  LGA   ORD      740     148     210
219    04MAR18   9:31  LGA   LON     3442     232     250
622    04MAR18  12:19  LGA   FRA     3857     137     250
132    04MAR18  15:35  LGA   YYZ      366     117     178
271    04MAR18  13:17  LGA   PAR     3635     146     250
302    04MAR18  20:22  LGA   WAS      229     115     180
114    05MAR18   7:10  LGA   LAX     2475     117     210
```

```
202    05MAR18   10:43  LGA   ORD      740     104     210
219    05MAR18    9:31  LGA   LON     3442     160     250
622    05MAR18   12:19  LGA   FRA     3857     185     250
132    05MAR18   15:35  LGA   YYZ      366     157     178
271    05MAR18   13:17  LGA   PAR     3635     177     250
114    06MAR18    7:10  LGA   LAX     2475     128     210
202    06MAR18   10:43  LGA   ORD      740     115     210
219    06MAR18    9:31  LGA   LON     3442     163     250
132    06MAR18   15:35  LGA   YYZ      366     150     178
302    06MAR18   20:22  LGA   WAS      229      66     180
114    07MAR18    7:10  LGA   LAX     2475     160     210
202    07MAR18   10:43  LGA   ORD      740     175     210
219    07MAR18    9:31  LGA   LON     3442     241     250
622    07MAR18   12:19  LGA   FRA     3857     210     250
132    07MAR18   15:35  LGA   YYZ      366     164     178
271    07MAR18   13:17  LGA   PAR     3635     155     250
302    07MAR18   20:22  LGA   WAS      229     135     180
;
```

PROCLIB.PAYLIST2

```
proc sql;
  create table proclib.paylist2
    (IdNum char(4),
     Gender char(1),
     Jobcode char(3),
     Salary num,
     Birth num informat=date9.
           format=date9.,
     Hired num informat=date9.
           format=date9.);

insert into proclib.paylist2
values('1919','M','TA2',34376,'12SEP1990'd,'04JUN2007'd)
values('1653','F','ME2',31896,'15OCT1984'd,'09AUG2010'd)
values('1350','F','FA3',36886,'31AUG1985'd,'29JUL2010'd)
values('1401','M','TA3',38822,'13DEC1970'd,'17NOV2005'd)
values('1499','M','ME1',23025,'26APR1974'd,'07JUN2010'd);

title 'PROCLIB.PAYLIST2 Table';
select * from proclib.paylist2;
```

PROCLIB.PAYROLL

このデータセット(テーブル)は、"PROC SQL テーブルのデータの更新" (*SAS SQL プロシジャユーザーガイド*)で更新され、その更新データはその後の例で使用されます。

```
data proclib.payroll;
  input IdNumber $4. +3 Gender $1. +4 Jobcode $3. +8 Salary 6.
      +2 Birth date9. +2 Hired date9.;
  informat birth date9. hired date9.;
  format birth date9. hired date9.;
  datalines;
1919   M    TA2       34376  12SEP1990  04JUN2007
1653   F    ME2       35108  15OCT1984  09AUG2010
1400   M    ME1       29769  05NOV1987  16OCT2010
1350   F    FA3       32886  31AUG1985  29JUL2010
1401   M    TA3       38822  13DEC1970  17NOV2005
1499   M    ME3       43025  26APR1974  07JUN2010
1101   M    SCP       18723  06JUN1982  01OCT2010
1333   M    PT2       88606  30MAR1981  10FEB2011
1402   M    TA2       32615  17JAN1983  02DEC2010
1479   F    TA3       38785  22DEC1988  05OCT2007
1403   M    ME1       28072  28JAN1989  21DEC2016
1739   M    PT1       66517  25DEC1984  27JAN2011
1658   M    SCP       17943  08APR1987  29FEB2012
1428   F    PT1       68767  04APR1980  16NOV2011
1782   M    ME2       35345  04DEC1980  22FEB2012
1244   M    ME2       36925  31AUG1983  17JAN2008
1383   M    BCK       25823  25JAN1985  20OCT2009
1574   M    FA2       28572  27APR1980  20DEC2012
1789   M    SCP       18326  23JAN1977  11APR2008
1404   M    PT2       91376  24FEB1973  01JAN2010
1437   F    FA3       33104  20SEP1980  31AUG2004
1639   F    TA3       40260  26JUN1977  28JAN2014
1269   M    NA1       41690  03MAY1972  28NOV2012
1065   M    ME2       35090  26JAN1977  07JAN2007
1876   M    TA3       39675  20MAY1978  27APR2015
1037   F    TA1       28558  10APR1984  13SEP2012
1129   F    ME2       34929  08DEC1981  17AUG2011
1988   M    FA3       32217  30NOV1979  18SEP2004
1405   M    SCP       18056  05MAR1986  26JAN2012
1430   F    TA2       32925  28FEB1982  27APR2017
1983   F    FA3       33419  28FEB1962  30APR2005
1134   F    TA2       33462  05MAR1989  21DEC2016
1118   M    PT3      111379  16JAN1964  18DEC2000
1438   F    TA3       39223  15MAR1985  18NOV2007
1125   F    FA2       28888  08NOV1988  11DEC2007
1475   F    FA2       27787  15DEC1981  13JUL2010
1117   M    TA3       39771  05JUN1983  13AUG2012
1935   F    NA2       51081  28MAR1974  16OCT2011
1124   F    FA1       23177  10JUL1978  01OCT2010
```

```
1422  F   FA1      22454  04JUN1984  06APR2011
1616  F   TA2      34137  01MAR1990  04JUN2013
1406  M   ME2      35185  08MAR1981  17FEB2007
1120  M   ME1      28619  11SEP1992  07OCT2013
1094  M   FA1      22268  02APR1990  17APR2011
1389  M   BCK      25028  15JUL1979  18AUG2010
1905  M   PT1      65111  16APR1992  29MAY2012
1407  M   PT1      68096  23MAR1989  18MAR2010
1114  F   TA2      32928  18SEP1989  27JUN2007
1410  M   PT2      84685  03MAY1987  07NOV2006
1439  F   PT1      70736  06MAR1984  10SEP2010
1409  M   ME3      41551  19APR1970  22OCT2001
1408  M   TA2      34138  29MAR1980  14OCT2007
1121  M   ME1      29112  26SEP1981  07DEC2011
1991  F   TA1      27645  07MAY1982  12DEC2012
1102  M   TA2      34542  01OCT1979  15APR2011
1356  M   ME2      36869  26SEP1977  22FEB2003
1545  M   PT1      66130  12AUG1979  29MAY2010
1292  F   ME2      36691  28OCT1984  02JUL2009
1440  F   ME2      35757  27SEP1982  09APR2011
1368  M   FA2      27808  11JUN1981  03NOV2014
1369  M   TA2      33705  28DEC1981  13MAR2017
1411  M   FA2      27265  27MAY1981  01DEC2017
1113  F   FA1      22367  15JAN1988  17OCT2011
1704  M   BCK      25465  30AUG1986  28JUN2007
1900  M   ME2      35105  25MAY1982  27OCT2007
1126  F   TA3      40899  28MAY1983  21NOV2010
1677  M   BCK      26007  05NOV1983  27MAR2009
1441  F   FA2      27158  19NOV1989  23MAR2011
1421  M   TA2      33155  08JAN1979  28FEB2010
1119  M   TA1      26924  20JUN1982  06SEP2008
1834  M   BCK      26896  08FEB1982  02JUL2012
1777  M   PT3     109630  23SEP1971  21JUN2001
1663  M   BCK      26452  11JAN1987  11AUG2011
1106  M   PT2      89632  06NOV1977  16AUG2004
1103  F   FA1      23738  16FEB1988  23JUL2012
1477  M   FA2      28566  21MAR1984  07MAR2008
1476  F   TA2      34803  30MAY1986  17MAR2017
1379  M   ME3      42264  08AUG1981  10JUN2004
1104  M   SCP      17946  25APR1983  10JUN2011
1009  M   TA1      28880  02MAR1979  26MAR2012
1412  M   ME1      27799  18JUN1976  05DEC2011
1115  F   FA3      32699  22AUG1980  28FEB2010
1128  F   TA2      32777  23MAY1985  20OCT2010
1442  F   PT2      84536  05SEP1986  12APR2008
1417  M   NA2      52270  27JUN1984  07MAR2009
1478  M   PT2      84203  09AUG1979  24OCT2010
1673  M   BCK      25477  27FEB1980  15JUL2011
1839  F   NA1      43433  29NOV1980  03JUL2013
1347  M   TA3      40079  21SEP1987  06SEP2014
1423  F   ME2      35773  14MAY1988  19AUG2010
1200  F   ME1      27816  10JAN1981  14AUG2012
1970  F   FA1      22615  25SEP1984  12MAR2011
1521  M   ME3      41526  12APR1983  13JUL2008
1354  F   SCP      18335  29MAY1981  16JUN2012
1424  F   FA2      28978  04AUG1979  11DEC2009
```

```
1132 F  FA1     22413 30MAY1972 22OCT2003
1845 M  BCK     25996 20NOV1979 22MAR2000
1556 M  PT1     71349 22JUN1984 11DEC2011
1413 M  FA2     27435 16SEP1975 02JAN2010
1123 F  TA1     28407 31OCT1982 05DEC2012
1907 M  TA2     33329 15NOV1980 06JUL2007
1436 F  TA2     34475 11JUN1984 12MAR2007
1385 M  ME3     43900 16JAN1982 01APR2016
1432 F  ME2     35327 03NOV1981 10FEB2015
1111 M  NA1     40586 14JUL1983 31OCT2012
1116 F  FA1     22862 28SEP1979 21MAR2011
1352 M  NA2     53798 02DEC1980 16OCT2006
1555 F  FA2     27499 16MAR1978 04JUL2012
1038 F  TA1     26533 09NOV1989 23NOV2011
1420 M  ME3     43071 19FEB1985 22JUL2017
1561 M  TA2     34514 30NOV1983 07OCT2007
1434 F  FA2     28622 11JUL1982 28OCT2010
1414 M  FA1     23644 24MAR1982 12APR2012
1112 M  TA1     26905 29NOV1984 07DEC2012
1390 M  FA2     27761 19FEB1985 23JUN2011
1332 M  NA1     42178 17SEP1980 04JUN2011
1890 M  PT2     91908 20JUL1981 25NOV2017
1429 F  TA1     27939 28FEB1980 07AUG2012
1107 M  PT2     89977 09JUN1974 10FEB2009
1908 F  TA2     32995 10DEC1979 23APR2010
1830 F  PT2     84471 27MAY1977 29JAN2013
1882 M  ME3     41538 10JUL1977 21NOV2008
1050 M  ME2     35167 14JUL1983 24AUG2016
1425 F  FA1     23979 28DEC1981 28FEB2013
1928 M  PT2     89858 16SEP1974 13JUL2010
1480 F  TA3     39583 03SEP1977 25MAR2001
1100 M  BCK     25004 01DEC1980 07MAY2008
1995 F  ME1     28810 24AUG1983 19SEP2013
1135 F  FA2     27321 20SEP1980 31MAR2010
1415 M  FA2     28278 09MAR1978 12FEB2008
1076 M  PT1     66558 14OCT1975 03OCT2011
1426 F  TA2     32991 05DEC1986 25JUN2010
1564 F  SCP     18833 12APR1982 01JUL2012
1221 F  FA2     27896 22SEP1987 04OCT2011
1133 M  TA1     27701 13JUL1986 12FEB2012
1435 F  TA3     38808 12MAY1979 08FEB2010
1418 M  ME1     28005 29MAR1977 06JAN2012
1017 M  TA3     40858 28DEC1977 16OCT2011
1443 F  NA1     42274 17NOV1988 29AUG2011
1131 F  TA2     32575 26DEC1981 19APR2011
1427 F  TA2     34046 31OCT1990 30JAN2010
1036 F  TA3     39392 19MAY1985 23OCT2004
1130 F  FA1     23916 16MAY1981 05JUN2012
1127 F  TA2     33011 09NOV1984 07DEC2006
1433 F  FA3     32982 08JUL1986 17JAN2007
1431 F  FA3     33230 09JUN1984 05APR2008
1122 F  FA2     27956 01MAY1983 27NOV2008
1105 M  ME2     34805 01MAR1982 13AUG2010
;
```

PROCLIB.PAYROLL2

```
data proclib.payroll2;
  input idnum $4. +3 gender $1. +4 jobcode $3. +9 salary 5.
      +2 birth date9. +2 hired date9.;
  informat birth date9. hired date9.;
  format birth date9. hired date9.;
  datalines;
1639   F   TA3       42260  26JUN1977  28JAN2004
1065   M    ME3       38090  26JAN1964  07JAN2007
1561   M   TA3       36514  30NOV1983  07OCT2007
1221   F   FA3       29896  22SEP1987  04OCT2011
1447   F   FA1       22123  07AUG1982  29OCT2002
1998   M    SCP       23100  10SEP1970  02NOV2012
1036   F   TA3       42465  19MAY1965  23OCT2004
1106   M    PT3       94039  06NOV1957  16AUG2004
1129   F   ME3       36758  08DEC1961  17AUG2011
1350   F   FA3       36098  31AUG1985  29JUL2010
1369   M    TA3       36598  28DEC1961  13MAR2007
1076   M    PT1       69742  14OCT1955  03OCT2011
;
```

PROCLIB.SCHEDULE

```
data proclib.schedule;
  input flight $3. +5 date date7. +2 dest $3. +3 idnum $4.;
  format date date7.;
  informat date date7.;
  datalines;
132    01MAR18  YYZ   1739
132    01MAR18  YYZ   1478
132    01MAR18  YYZ   1130
132    01MAR18  YYZ   1390
132    01MAR18  YYZ   1983
132    01MAR18  YYZ   1111
219    01MAR18  LON   1407
219    01MAR18  LON   1777
219    01MAR18  LON   1103
219    01MAR18  LON   1125
219    01MAR18  LON   1350
219    01MAR18  LON   1332
271    01MAR18  PAR   1439
271    01MAR18  PAR   1442
271    01MAR18  PAR   1132
```

```
271   01MAR18  PAR  1411
271   01MAR18  PAR  1988
271   01MAR18  PAR  1443
622   01MAR18  FRA  1545
622   01MAR18  FRA  1890
622   01MAR18  FRA  1116
622   01MAR18  FRA  1221
622   01MAR18  FRA  1433
622   01MAR18  FRA  1352
132   02MAR18  YYZ  1556
132   02MAR18  YYZ  1478
132   02MAR18  YYZ  1113
132   02MAR18  YYZ  1411
132   02MAR18  YYZ  1574
132   02MAR18  YYZ  1111
219   02MAR18  LON   1407
219   02MAR18  LON   1118
219   02MAR18  LON   1132
219   02MAR18  LON   1135
219   02MAR18  LON   1441
219   02MAR18  LON   1332
271   02MAR18  PAR  1739
271   02MAR18  PAR  1442
271   02MAR18  PAR  1103
271   02MAR18  PAR  1413
271   02MAR18  PAR  1115
271   02MAR18  PAR  1443
622   02MAR18  FRA  1439
622   02MAR18  FRA  1890
622   02MAR18  FRA  1124
622   02MAR18  FRA  1368
622   02MAR18  FRA  1477
622   02MAR18  FRA  1352
132   03MAR18  YYZ  1739
132   03MAR18  YYZ  1928
132   03MAR18  YYZ  1425
132   03MAR18  YYZ  1135
132   03MAR18  YYZ  1437
132   03MAR18  YYZ  1111
219   03MAR18  LON   1428
219   03MAR18  LON   1442
219   03MAR18  LON   1130
219   03MAR18  LON   1411
219   03MAR18  LON   1115
219   03MAR18  LON   1332
271   03MAR18  PAR  1905
271   03MAR18  PAR  1118
271   03MAR18  PAR  1970
271   03MAR18  PAR  1125
271   03MAR18  PAR  1983
271   03MAR18  PAR  1443
622   03MAR18  FRA  1545
622   03MAR18  FRA  1830
622   03MAR18  FRA  1414
622   03MAR18  FRA  1368
622   03MAR18  FRA  1431
```

```
622   03MAR18 FRA  1352
132   04MAR18 YYZ  1428
132   04MAR18 YYZ  1118
132   04MAR18 YYZ  1103
132   04MAR18 YYZ  1390
132   04MAR18 YYZ  1350
132   04MAR18 YYZ  1111
219   04MAR18 LON  1739
219   04MAR18 LON  1478
219   04MAR18 LON  1130
219   04MAR18 LON  1125
219   04MAR18 LON  1983
219   04MAR18 LON  1332
271   04MAR18 PAR  1407
271   04MAR18 PAR  1410
271   04MAR18 PAR  1094
271   04MAR18 PAR  1411
271   04MAR18 PAR  1115
271   04MAR18 PAR  1443
622   04MAR18 FRA  1545
622   04MAR18 FRA  1890
622   04MAR18 FRA  1116
622   04MAR18 FRA  1221
622   04MAR18 FRA  1433
622   04MAR18 FRA  1352
132   05MAR18 YYZ  1556
132   05MAR18 YYZ  1890
132   05MAR18 YYZ  1113
132   05MAR18 YYZ  1475
132   05MAR18 YYZ  1431
132   05MAR18 YYZ  1111
219   05MAR18 LON  1428
219   05MAR18 LON  1442
219   05MAR18 LON  1422
219   05MAR18 LON  1413
219   05MAR18 LON  1574
219   05MAR18 LON  1332
271   05MAR18 PAR  1739
271   05MAR18 PAR  1928
271   05MAR18 PAR  1103
271   05MAR18 PAR  1477
271   05MAR18 PAR  1433
271   05MAR18 PAR  1443
622   05MAR18 FRA  1545
622   05MAR18 FRA  1830
622   05MAR18 FRA  1970
622   05MAR18 FRA  1441
622   05MAR18 FRA  1350
622   05MAR18 FRA  1352
132   06MAR18 YYZ  1333
132   06MAR18 YYZ  1890
132   06MAR18 YYZ  1414
132   06MAR18 YYZ  1475
132   06MAR18 YYZ  1437
132   06MAR18 YYZ  1111
219   06MAR18 LON  1106
```

```
219    06MAR18  LON   1118
219    06MAR18  LON   1425
219    06MAR18  LON   1434
219    06MAR18  LON   1555
219    06MAR18  LON   1332
132    07MAR18  YYZ   1407
132    07MAR18  YYZ   1118
132    07MAR18  YYZ   1094
132    07MAR18  YYZ   1555
132    07MAR18  YYZ   1350
132    07MAR18  YYZ   1111
219    07MAR18  LON   1905
219    07MAR18  LON   1478
219    07MAR18  LON   1124
219    07MAR18  LON   1434
219    07MAR18  LON   1983
219    07MAR18  LON   1332
271    07MAR18  PAR   1410
271    07MAR18  PAR   1777
271    07MAR18  PAR   1103
271    07MAR18  PAR   1574
271    07MAR18  PAR   1115
271    07MAR18  PAR   1443
622    07MAR18  FRA   1107
622    07MAR18  FRA   1890
622    07MAR18  FRA   1425
622    07MAR18  FRA   1475
622    07MAR18  FRA   1433
622    07MAR18  FRA   1352
;
```

PROCLIB.STAFF

```
data proclib.staff;
  input idnum $4. +3 lname $15. +2 fname $15. +2 city $15. +2
      state $2. +5 hphone $12.;
  datalines;
1919   ADAMS          GERALD         STAMFORD       CT     203/781-1255
1653   ALIBRANDI      MARIA          BRIDGEPORT     CT     203/675-7715
1400   ALHERTANI      ABDULLAH       NEW YORK       NY     212/586-0808
1350   ALVAREZ        MERCEDES       NEW YORK       NY     718/383-1549
1401   ALVAREZ        CARLOS         PATERSON       NJ     201/732-8787
1499   BAREFOOT       JOSEPH         PRINCETON      NJ     201/812-5665
1101   BAUCOM         WALTER         NEW YORK       NY     212/586-8060
1333   BANADYGA       JUSTIN         STAMFORD       CT     203/781-1777
1402   BLALOCK        RALPH          NEW YORK       NY     718/384-2849
1479   BALLETTI       MARIE          NEW YORK       NY     718/384-8816
1403   BOWDEN         EARL           BRIDGEPORT     CT     203/675-3434
1739   BRANCACCIO     JOSEPH         NEW YORK       NY     212/587-1247
1658   BREUHAUS       JEREMY         NEW YORK       NY     212/587-3622
1428   BRADY          CHRISTINE      STAMFORD       CT     203/781-1212
```

1782 BREWCZAK JAKOB STAMFORD CT 203/781-0019
1244 BUCCI ANTHONY NEW YORK NY 718/383-3334
1383 BURNETTE THOMAS NEW YORK NY 718/384-3569
1574 CAHILL MARSHALL NEW YORK NY 718/383-2338
1789 CARAWAY DAVIS NEW YORK NY 212/587-9000
1404 COHEN LEE NEW YORK NY 718/384-2946
1437 CARTER DOROTHY BRIDGEPORT CT 203/675-4117
1639 CARTER-COHEN KAREN STAMFORD CT 203/781-8839
1269 CASTON FRANKLIN STAMFORD CT 203/781-3335
1065 COPAS FREDERICO NEW YORK NY 718/384-5618
1876 CHIN JACK NEW YORK NY 212/588-5634
1037 CHOW JANE STAMFORD CT 203/781-8868
1129 COUNIHAN BRENDA NEW YORK NY 718/383-2313
1988 COOPER ANTHONY NEW YORK NY 212/587-1228
1405 DACKO JASON PATERSON NJ 201/732-2323
1430 DABROWSKI SANDRA BRIDGEPORT CT 203/675-1647
1983 DEAN SHARON NEW YORK NY 718/384-1647
1134 DELGADO MARIA STAMFORD CT 203/781-1528
1118 DENNIS ROGER NEW YORK NY 718/383-1122
1438 DABBOUSSI KAMILLA STAMFORD CT 203/781-2229
1125 DUNLAP DONNA NEW YORK NY 718/383-2094
1475 ELGES MARGARETE NEW YORK NY 718/383-2828
1117 EDGERTON JOSHUA NEW YORK NY 212/588-1239
1935 FERNANDEZ KATRINA BRIDGEPORT CT 203/675-2962
1124 FIELDS DIANA WHITE PLAINS NY 914/455-2998
1422 FUJIHARA KYOKO PRINCETON NJ 201/812-0902
1616 FUENTAS CARLA NEW YORK NY 718/384-3329
1406 FOSTER GERALD BRIDGEPORT CT 203/675-6363
1120 GARCIA JACK NEW YORK NY 718/384-4930
1094 GOMEZ ALAN BRIDGEPORT CT 203/675-7181
1389 GOLDSTEIN LEVI NEW YORK NY 718/384-9326
1905 GRAHAM ALVIN NEW YORK NY 212/586-8815
1407 GREGORSKI DANIEL MT. VERNON NY 914/468-1616
1114 GREENWALD JANICE NEW YORK NY 212/588-1092
1410 HARRIS CHARLES STAMFORD CT 203/781-0937
1439 HASENHAUER CHRISTINA BRIDGEPORT CT 203/675-4987
1409 HAVELKA RAYMOND STAMFORD CT 203/781-9697
1408 HENDERSON WILLIAM PRINCETON NJ 201/812-4789
1121 HERNANDEZ ROBERTO NEW YORK NY 718/384-3313
1991 HOWARD GRETCHEN BRIDGEPORT CT 203/675-0007
1102 HERMANN JOACHIM WHITE PLAINS NY 914/455-0976
1356 HOWARD MICHAEL NEW YORK NY 212/586-8411
1545 HERRERO CLYDE STAMFORD CT 203/781-1119
1292 HUNTER HELEN BRIDGEPORT CT 203/675-4830
1440 JACKSON LAURA STAMFORD CT 203/781-0088
1368 JEPSEN RONALD STAMFORD CT 203/781-8413
1369 JONSON ANTHONY NEW YORK NY 212/587-5385
1411 JOHNSEN JACK PATERSON NJ 201/732-3678
1113 JOHNSON LESLIE NEW YORK NY 718/383-3003
1704 JONES NATHAN NEW YORK NY 718/384-0049
1900 KING WILLIAM NEW YORK NY 718/383-3698
1126 KIMANI ANNE NEW YORK NY 212/586-1229
1677 KRAMER JACKSON BRIDGEPORT CT 203/675-7432
1441 LAWRENCE KATHY PRINCETON NJ 201/812-3337
1421 LEE RUSSELL MT. VERNON NY 914/468-9143
1119 LI JEFF NEW YORK NY 212/586-2344

1834 LEBLANC RUSSELL NEW YORK NY 718/384-0040
1777 LUFKIN ROY NEW YORK NY 718/383-4413
1663 MARKS JOHN NEW YORK NY 212/587-7742
1106 MARSHBURN JASPER STAMFORD CT 203/781-1457
1103 MCDANIEL RONDA NEW YORK NY 212/586-0013
1477 MEYERS PRESTON BRIDGEPORT CT 203/675-8125
1476 MONROE JOYCE STAMFORD CT 203/781-2837
1379 MORGAN ALFRED STAMFORD CT 203/781-2216
1104 MORGAN CHRISTOPHER NEW YORK NY 718/383-9740
1009 MORGAN GEORGE NEW YORK NY 212/586-7753
1412 MURPHEY JOHN PRINCETON NJ 201/812-4414
1115 MURPHY ALICE NEW YORK NY 718/384-1982
1128 NELSON FELICIA BRIDGEPORT CT 203/675-1166
1442 NEWKIRK SANDRA PRINCETON NJ 201/812-3331
1417 NEWKIRK WILLIAM PATERSON NJ 201/732-6611
1478 NEWTON JAMES NEW YORK NY 212/587-5549
1673 NICHOLLS HENRY STAMFORD CT 203/781-7770
1839 NORRIS DIANE NEW YORK NY 718/384-1767
1347 O'NEAL BRYAN NEW YORK NY 718/384-0230
1423 OSWALD LESLIE MT. VERNON NY 914/468-9171
1200 OVERMAN MICHELLE STAMFORD CT 203/781-1835
1970 PARKER ANNE NEW YORK NY 718/383-3895
1521 PARKER JAY NEW YORK NY 212/587-7603
1354 PARKER MARY WHITE PLAINS NY 914/455-2337
1424 PATTERSON RENEE NEW YORK NY 212/587-8991
1132 PEARCE CAROL NEW YORK NY 718/384-1986
1845 PEARSON JAMES NEW YORK NY 718/384-2311
1556 PENNINGTON MICHAEL NEW YORK NY 718/383-5681
1413 PETERS RANDALL PRINCETON NJ 201/812-2478
1123 PETERSON SUZANNE NEW YORK NY 718/383-0077
1907 PHELPS WILLIAM STAMFORD CT 203/781-1118
1436 PORTER SUSAN NEW YORK NY 718/383-5777
1385 RAYNOR MILTON BRIDGEPORT CT 203/675-2846
1432 REED MARILYN MT. VERNON NY 914/468-5454
1111 RHODES JEREMY PRINCETON NJ 201/812-1837
1116 RICHARDS CASEY NEW YORK NY 212/587-1224
1352 RIVERS SIMON NEW YORK NY 718/383-3345
1555 RODRIGUEZ JULIA BRIDGEPORT CT 203/675-2401
1038 RODRIGUEZ MARIA BRIDGEPORT CT 203/675-2048
1420 ROUSE JEREMY PATERSON NJ 201/732-9834
1561 SANDERS RAYMOND NEW YORK NY 212/588-6615
1434 SANDERSON EDITH STAMFORD CT 203/781-1333
1414 SANDERSON NATHAN BRIDGEPORT CT 203/675-1715
1112 SANYERS RANDY NEW YORK NY 718/384-4895
1390 SMART JONATHAN NEW YORK NY 718/383-1141
1332 STEPHENSON ADAM BRIDGEPORT CT 203/675-1497
1890 STEPHENSON ROBERT NEW YORK NY 718/384-9874
1429 THOMPSON ALICE STAMFORD CT 203/781-3857
1107 THOMPSON WAYNE NEW YORK NY 718/384-3785
1908 TRENTON MELISSA NEW YORK NY 212/586-6262
1830 TRIPP KATHY BRIDGEPORT CT 203/675-2479
1882 TUCKER ALAN NEW YORK NY 718/384-0216
1050 TUTTLE THOMAS WHITE PLAINS NY 914/455-2119
1425 UNDERWOOD JENNY STAMFORD CT 203/781-0978
1928 UPCHURCH LARRY WHITE PLAINS NY 914/455-5009
1480 UPDIKE THERESA NEW YORK NY 212/587-8729

```
1100  VANDEUSEN      RICHARD        NEW YORK       NY    212/586-2531
1995  VARNER         ELIZABETH      NEW YORK       NY    718/384-7113
1135  VEGA           ANNA           NEW YORK       NY    718/384-5913
1415  VEGA           FRANKLIN       NEW YORK       NY    718/384-2823
1076  VENTER         RANDALL        NEW YORK       NY    718/383-2321
1426  VICK           THERESA        PRINCETON      NJ    201/812-2424
1564  WALTERS        ANNE           NEW YORK       NY    212/587-3257
1221  WALTERS        DIANE          NEW YORK       NY    718/384-1918
1133  WANG           CHIN           NEW YORK       NY    212/587-1956
1435  WARD           ELAINE         NEW YORK       NY    718/383-4987
1418  WATSON         BERNARD        NEW YORK       NY    718/383-1298
1017  WELCH          DARIUS         NEW YORK       NY    212/586-5535
1443  WELLS          AGNES          STAMFORD       CT    203/781-5546
1131  WELLS          NADINE         NEW YORK       NY    718/383-1045
1427  WHALEY         CAROLYN        MT. VERNON     NY    914/468-4528
1036  WONG           LESLIE         NEW YORK       NY    212/587-2570
1130  WOOD           DEBORAH        NEW YORK       NY    212/587-0013
1127  WOOD           SANDRA         NEW YORK       NY    212/587-2881
1433  YANCEY         ROBIN          PRINCETON      NJ    201/812-1874
1431  YOUNG          DEBORAH        STAMFORD       CT    203/781-2987
1122  YOUNG          JOANN          NEW YORK       NY    718/384-2021
1105  YOUNG          LAWRENCE       NEW YORK       NY    718/384-0008
;
```

PROCLIB.STAFF2

```
data proclib.staff;
  input Name & $16. IdNumber $ Salary
      Site $ HireDate date9.;
  format hiredate date9.;
  datalines;
Capalleti, Jimmy  2355 21163 BR1 30JAN2009
Chen, Len         5889 20976 BR1 18JUN2006
Davis, Brad       3878 19571 BR2 20MAR2004
Leung, Brenda     4409 34321 BR2 18SEP2014
Martinez, Maria   3985 49056 US2 10JAN2013
Orfali, Philip    0740 50092 US2 16FEB2003
Patel, Mary       2398 35182 BR3 02FEB2010
Smith, Robert     5162 40100 BR5 15APR2006
Sorrell, Joseph   4421 38760 US1 19JUN2011
Zook, Carla       7385 22988 BR3 18DEC2010
;
```

PROCLIB.SUPERV

```
data proclib.superv;
  input supid $4. +8 state $2. +5  jobcat  $2.;
  label supid='Supervisor Id' jobcat='Job Category';
  datalines;
1677       CT    BC
1834       NY    BC
1431       CT    FA
1433       NJ    FA
1983       NY    FA
1385       CT    ME
1420       NJ    ME
1882       NY    ME
1935       CT    NA
1417       NJ    NA
1352       NY    NA
1106       CT    PT
1442       NJ    PT
1118       NY    PT
1405       NJ    SC
1564       NY    SC
1639       CT    TA
1401       NJ    TA
1126       NY    TA
;
```

RADIO

この DATA ステップは、INFILE ステートメントを使用して、外部ファイルに保存されているデータを読み込みます。

```
data radio;
  infile 'input-file' missover;
  input /(time1-time7) ($1. +1);
  listener=_n_;
run;
```

外部ファイルに保存されているデータを次に示します。

```
967 32 f 5 3 5
7 5 5 5 7 0 0 0 8 7 0 0 8 0
781 30 f 2 3 5
5 0 0 0 5 0 0 0 4 7 5 0 0 0
859 39 f 1 0 5
```

1 0 0 0 1 0 0 0 0 0 0 0 0
859 40 f 6 1 5
7 5 0 5 7 0 0 0 0 0 5 0 0
467 37 m 2 3 1
1 5 5 5 5 4 4 8 8 0 0 0 0
220 35 f 3 1 7
7 0 0 0 7 0 0 0 7 0 0 0 0
833 42 m 2 2 4
7 0 0 0 7 5 4 7 4 0 1 4 4 0
967 39 f .5 1 7
7 0 0 0 7 7 0 0 0 0 0 8 0
677 28 m .5 .5 7
7 0 0 0 0 0 0 0 0 0 0 0 0
833 28 f 3 4 1
1 0 0 0 0 1 1 1 1 0 0 0 1 1
677 24 f 3 1 2
2 0 0 0 0 0 0 2 0 8 8 0 0 0
688 32 m 5 2 4
5 5 0 4 8 0 0 5 0 8 0 0 0 0
542 38 f 6 8 5
5 0 0 5 5 5 0 5 5 5 5 5 5 0
677 27 m 6 1 1
1 1 0 4 4 0 0 1 4 0 0 0 0 0
779 37 f 2.5 4 7
7 0 0 0 7 7 0 7 7 4 4 7 8 0
362 31 f 1 2 2
8 0 0 0 8 0 0 0 0 0 8 8 0 0
859 29 m 10 3 4
4 4 0 2 2 0 0 4 0 0 0 4 4 0
467 24 m 5 8 1
7 1 1 1 7 1 1 0 1 7 1 1 1 1
851 34 m 1 2 8
0 0 0 0 8 0 0 0 4 0 0 0 8 0
859 23 f 1 1 8
8 0 0 0 8 0 0 0 0 0 0 0 0 8
781 34 f 9 3 1
2 1 0 1 4 4 4 0 1 1 1 1 4 4
851 40 f 2 4 5
5 0 0 0 5 0 0 5 0 0 5 5 0 0
783 34 m 3 2 4
7 0 0 0 7 4 4 0 0 4 4 0 0 0
848 29 f 4 1.5 7
7 4 4 1 7 0 0 0 7 0 0 7 0 0
851 28 f 1 2 2
2 0 2 0 2 0 0 0 0 2 2 2 0 0
856 42 f 1.5 1 2
2 0 0 0 0 0 0 2 0 0 0 0 0 0
859 29 m .5 .5 5
5 0 0 0 1 0 0 0 0 0 8 8 5 0
833 29 m 1 3 2
2 0 0 0 2 2 0 0 4 2 0 2 0 0
859 23 f 10 3 1
1 5 0 8 8 1 4 0 1 1 1 1 1 4
781 37 f .5 2 7
7 0 0 0 1 0 0 0 1 7 0 1 0 0
833 31 f 5 4 1

1 0 0 0 1 0 0 0 4 0 4 0 0 0
942 23 f 4 2 1
1 0 0 0 1 0 1 0 1 1 0 0 0 0
848 33 f 5 4 1
1 1 0 1 1 0 0 0 1 1 1 0 0 0
222 33 f 2 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0
851 45 f .5 1 8
8 0 0 0 8 0 0 0 0 0 8 0 0 0
848 27 f 2 4 1
1 0 0 0 1 1 0 0 4 1 1 1 1 1
781 38 m 2 2 1
5 0 0 0 1 0 0 0 0 0 1 1 0 0
222 27 f 3 1 2
2 0 2 0 2 2 0 0 2 0 0 0 0 0
467 34 f 2 2 1
1 0 0 0 0 1 0 1 0 0 0 0 1 0
833 27 f 8 8 1
7 0 1 0 7 4 0 0 1 1 1 4 1 0
677 49 f 1.5 0 8
8 0 8 0 8 0 0 0 0 0 0 0 0 0
849 43 m 1 4 1
1 0 0 0 4 0 0 0 4 0 1 0 0 0
467 28 m 2 1 7
7 0 0 0 7 0 0 7 0 0 1 0 0 0
732 29 f 1 0 2
2 0 0 0 2 0 0 0 0 0 0 0 0 0
851 31 m 2 2 2
2 5 0 6 0 0 8 0 2 2 8 2 0 0
779 42 f 8 2 2
7 2 0 2 7 0 0 0 0 0 0 2 0
493 40 m 1 3 3
3 0 0 0 5 3 0 5 5 0 0 0 1 1
859 30 m 1 0 7
7 0 0 0 7 0 0 0 0 0 0 0 0 0
833 36 m 4 2 5
7 5 0 5 0 5 0 0 7 0 0 0 5 0
467 30 f 1 4 1
0 0 0 0 1 0 6 0 0 1 1 1 0 6
859 32 f 3 5 2
2 2 2 2 2 2 6 6 2 2 2 2 2 6
851 43 f 8 1 5
7 5 5 5 0 0 0 4 0 0 0 0 0 0
848 29 f 3 5 1
7 0 0 0 7 1 0 0 1 1 1 1 1 0
833 25 f 2 4 5
7 0 0 0 5 7 0 0 7 5 0 0 5 0
783 33 f 8 3 8
8 0 8 0 7 0 0 0 8 0 5 4 0 5
222 26 f 10 2 1
1 1 0 1 1 0 0 0 3 1 1 0 0 0
222 23 f 3 2 2
2 2 2 2 7 0 0 2 2 0 0 0 0 0
859 50 f 1 5 4
7 0 0 0 7 0 0 5 4 4 4 7 0 0
833 26 f 3 2 1

1 0 0 1 1 0 0 5 5 0 1 0 0 0
467 29 m 7 2 1
1 1 1 1 1 0 0 1 1 1 0 0 0 0
859 35 m .5 2 2
7 0 0 0 2 0 0 7 5 0 0 4 0 0
833 33 f 3 3 6
7 0 0 0 6 8 0 8 0 0 0 8 6 0
221 36 f .5 1 5
0 7 0 0 0 7 0 0 7 0 0 7 7 0
220 32 f 2 4 5
5 0 5 0 5 5 5 0 5 5 5 5 5 5
684 19 f 2 4 2
0 2 0 2 0 0 0 0 0 2 2 0 0 0
493 55 f 1 0 5
5 0 0 5 0 0 0 0 7 0 0 0 0 0
221 27 m 1 1 7
7 0 0 0 0 0 0 0 5 0 0 0 5 0
684 19 f 0 .5 1
7 0 0 0 0 1 1 0 0 0 0 0 1 1
493 38 f .5 .5 5
0 8 0 0 5 0 0 0 5 0 0 0 0 0
221 26 f .5 2 1
0 1 0 0 0 1 0 0 5 5 5 1 0 0
684 18 m 1 .5 1
0 2 0 0 0 0 1 0 0 0 0 1 1 0
684 19 m 1 1 1
0 0 0 1 1 0 0 0 0 0 1 0 0 0
221 29 m .5 .5 5
0 0 0 0 0 5 5 0 0 0 0 0 5 5
683 18 f 2 4 8
0 0 0 0 8 0 0 0 8 8 8 0 0 0
966 23 f 1 2 1
1 5 5 5 1 0 0 0 0 1 0 0 1 0
493 25 f 3 5 7
7 0 0 0 7 2 0 0 7 0 2 7 7 0
683 18 f .5 .5 2
1 0 0 0 0 0 5 0 0 1 0 0 0 1
382 21 f 3 1 8
0 8 0 0 5 8 8 0 0 8 8 0 0 0
683 18 f 4 6 2
2 0 0 0 2 2 2 0 2 0 2 2 2 0
684 19 m .5 2 1
0 0 0 0 1 1 0 0 0 1 1 1 1 5
684 19 m 1.5 3.5 2
2 0 0 0 2 0 0 0 0 0 2 5 0 0
221 23 f 1 5 1
7 5 1 5 1 3 1 7 5 1 5 1 3 1
684 18 f 2 3 1
2 0 0 1 1 1 1 7 2 0 1 1 1 1
683 19 f 3 5 2
2 0 0 2 0 6 1 0 1 1 2 2 6 1
683 19 f 3 5 1
2 0 0 2 0 6 1 0 1 1 2 0 2 1
221 35 m 3 5 5
7 5 0 1 7 0 0 5 5 5 0 0 0 0
221 43 f 1 4 5

1 0 0 0 5 0 0 5 5 0 0 0 0 0
493 32 f 2 1 6
0 0 0 6 0 0 0 0 0 0 0 4 0
221 24 f 4 5 2
2 0 5 0 0 2 4 4 4 5 0 0 2 2
684 19 f 2 3 2
0 5 5 2 5 0 1 0 5 5 2 2 2 2
221 19 f 3 3 8
0 1 1 8 8 8 4 0 5 4 1 8 8 4
221 29 m 1 1 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5
221 21 m 1 1 1
1 0 0 0 0 0 5 1 0 0 0 0 0 5
683 20 f 1 2 2
0 0 0 0 2 0 0 0 2 0 0 0 0 0
493 54 f 1 1 5
7 0 0 5 0 0 0 0 0 0 5 0 0 0
493 45 m 4 6 5
7 0 0 0 7 5 0 0 5 5 5 5 5 5
850 44 m 2.5 1.5 7
7 0 7 0 4 7 5 0 5 4 3 0 0 4
220 33 m 5 3 5
1 5 0 5 1 0 0 0 0 0 0 5 5
684 20 f 1.5 3 1
1 0 0 0 1 0 1 0 1 0 0 1 1 0
966 63 m 3 5 3
5 4 7 5 4 5 0 5 0 0 5 5 4 0
683 21 f 4 6 1
0 1 0 1 1 1 1 0 1 1 1 1 1 1
493 23 f 5 2 5
7 5 0 4 0 0 0 0 1 1 1 1 1 0
493 32 f 8 8 5
7 5 0 0 7 0 5 5 5 0 0 7 5 5
942 33 f 7 2 5
0 5 5 4 7 0 0 0 0 0 0 7 8 0
493 34 f .5 1 5
5 0 0 0 5 0 0 0 0 0 6 0 0 0
382 40 f 2 2 5
5 0 0 0 5 0 0 5 0 0 5 0 0 0
362 27 f 0 3 8
0 0 0 0 0 0 0 0 0 0 0 0 8 0
542 36 f 3 3 7
7 0 0 0 7 1 0 0 0 7 1 1 0 0
966 39 f 3 6 5
7 0 0 0 7 5 0 0 7 0 5 0 5 0
849 32 m 1 .5 7
7 0 0 0 5 0 0 0 7 4 4 5 7 0
677 52 f 3 2 3
7 0 0 0 0 7 0 0 0 7 0 0 3 0
222 25 m 2 4 1
1 0 0 0 1 0 0 0 1 0 1 0 0 0
732 42 f 3 2 7
7 0 0 0 1 7 5 5 7 0 0 3 4 0
467 26 f 4 4 1
7 0 1 0 7 1 0 0 7 7 4 7 0 0
467 38 m 2.5 0 1

1 0 0 0 1 0 0 0 0 0 0 0 0
382 37 f 1.5 .5 7
7 0 0 0 7 0 0 0 3 0 0 0 3 0
856 45 f 3 3 7
7 0 0 0 7 5 0 0 7 7 4 0 0 0
677 33 m 3 2 7
7 0 0 4 7 0 0 0 7 0 0 0 0 0
490 27 f .5 1 2
2 0 0 0 2 0 0 0 2 0 2 0 0 0
362 27 f 1.5 2 2
2 0 0 0 1 0 4 0 1 0 0 0 4 4
783 25 f 2 1 1
1 0 0 0 1 7 0 0 0 0 1 1 1 0
546 30 f 8 3 1
1 1 1 1 1 0 0 1 0 5 5 0 0 0
677 30 f 2 0 1
1 0 0 0 0 1 0 0 0 0 0 0 0 1
221 35 f 2 2 1
1 0 0 0 1 0 1 0 1 1 1 0 0 0
966 32 f 6 1 7
7 1 1 1 7 4 0 1 7 1 8 8 4 0
222 28 f 1 5 4
7 0 0 0 4 0 0 4 4 4 4 0 0 0
467 29 f 5 3 4
4 5 5 5 1 4 4 5 1 1 1 1 4 4
467 32 m 3 4 1
1 0 1 0 4 0 0 0 4 0 0 0 1 0
966 30 m 1.5 1 7
7 0 0 0 7 5 0 7 0 0 0 0 5 0
967 38 m 14 4 7
7 7 7 7 7 0 4 8 0 0 0 0 4 0
490 28 m 8 1 1
7 1 1 1 1 0 0 7 0 0 8 0 0 0
833 30 f .5 1 6
6 0 0 0 6 0 0 0 0 6 0 0 6 0
851 40 m 1 0 7
7 5 5 5 7 0 0 0 0 0 0 0 0 0
859 27 f 2 5 2
6 0 0 0 2 0 0 0 0 0 0 2 2 2
851 22 f 3 5 2
7 0 2 0 2 2 0 0 2 0 8 0 2 0
967 38 f 1 1.5 7
7 0 0 0 7 5 0 7 4 0 0 7 5 0
856 34 f 1.5 1 1
0 1 0 0 0 1 0 0 4 0 0 0 0 0
222 33 m .1 .1 7
7 0 0 0 7 0 0 0 0 0 7 0 0 0
856 22 m .50 .25 1
0 1 0 0 1 0 0 0 0 0 0 0 0 0
677 30 f 2 2 4
1 0 4 0 4 0 0 0 4 0 0 0 0 0
859 25 m 2 3 7
0 0 0 0 0 7 0 0 7 0 2 0 0 1
833 35 m 2 6 7
7 0 0 0 7 1 1 0 4 7 4 7 1 1
677 35 m 10 4 1

1 1 1 1 1 8 6 8 1 0 0 8 8 8
848 29 f 5 3 8
8 0 0 0 8 8 0 0 0 8 8 8 0 0
688 26 m 3 1 1
1 1 7 1 1 7 0 0 0 8 8 0 0 0
490 41 m 2 2 5
5 0 0 0 0 0 5 5 0 0 0 0 0 5
493 35 m 4 4 7
7 5 0 5 7 0 0 7 7 7 7 0 0 0
677 27 m 15 11 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
848 27 f 3 5 1
1 1 0 0 1 1 0 0 1 1 1 1 0 0
362 30 f 1 0 1
1 0 0 0 7 5 0 0 0 0 0 0 0 0
783 29 f 1 1 4
4 0 0 0 4 0 0 0 4 0 0 0 4 0
467 39 f .5 2 4
7 0 4 0 4 4 0 0 4 4 4 4 4 4
677 27 m 2 2 7
7 0 0 0 7 0 0 7 7 0 0 7 0 0
221 23 f 2.5 1 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0
677 29 f 1 1 7
0 0 0 0 7 0 0 0 7 0 0 0 0 0
783 32 m 1 2 5
4 5 5 5 4 2 0 0 0 0 3 2 2 0
833 25 f 1 0 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0
859 24 f 7 3 7
1 0 0 0 1 0 0 0 0 1 0 0 1 0
677 29 m 2 2 8
0 8 8 0 8 0 0 0 8 8 8 0 0 0
688 31 m 8 2 5
7 5 5 5 5 7 0 0 7 7 0 0 0 0
856 31 m 9 4 1
1 1 1 1 1 0 0 0 0 0 0 0 1 0
856 44 f 1 0 6
6 0 0 0 6 0 0 0 0 0 0 0 0 0
677 37 f 3 3 1
0 0 1 0 0 0 0 0 4 4 0 0 0 0
859 27 m 2 .5 2
2 2 2 2 2 2 2 2 0 0 0 0 0 2
781 30 f 10 4 2
2 0 0 0 2 0 2 0 0 0 0 0 0 2
362 27 m 12 4 3
3 1 1 1 1 3 3 3 0 0 0 0 3 0
362 33 f 2 4 1
1 0 0 0 7 0 0 7 1 1 1 1 1 0
222 26 f 8 1 1
1 1 1 1 0 0 0 1 0 0 0 0 0 0
779 37 f 6 3 1
1 1 1 1 1 0 0 1 1 0 0 0 1 0
467 32 f 1 1 2
2 0 0 0 0 0 0 0 2 0 0 2 0 0
859 23 m 1 1 1

1 0 0 0 1 1 0 1 0 0 0 0 1 1
781 33 f 1 .5 6
6 0 0 0 6 0 0 0 0 0 0 0 0 0
779 28 m 5 2 1
1 1 1 1 1 0 0 0 0 7 7 1 1 0
677 28 m 3 1 5
7 5 5 5 5 6 0 0 6 6 6 6 6 0
677 25 f 9 2 5
1 5 5 5 5 1 1 0 1 1 1 1 1 1
848 30 f 6 2 8
8 0 0 0 2 7 0 0 0 0 2 0 2 0
546 36 f 4 6 4
7 0 0 0 4 4 0 5 5 5 5 2 4 4
222 30 f 2 3 2
2 2 0 0 2 0 0 0 2 0 2 2 0 0
383 32 m 4 1 2
2 0 0 0 2 0 0 2 0 0 0 0 0 0
851 43 f 8 1 6
4 6 0 6 4 0 0 0 0 0 0 0 0 0
222 27 f 1 3 1
1 1 0 1 1 1 0 0 1 0 0 0 4 0
833 22 f 1.5 2 1
1 0 0 0 1 1 0 0 1 1 1 0 0 0
467 29 f 2 1 8
8 0 8 0 8 0 0 0 0 0 8 0 0 0
856 28 f 2 3 1
1 0 0 0 1 0 0 0 1 0 0 1 0 0
580 31 f 2.5 2.5 6
6 6 6 6 6 6 6 6 1 1 1 1 6 6
688 39 f 8 8 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3
677 37 f 1.5 .5 1
6 1 1 1 6 6 0 0 1 1 6 6 6 0
859 38 m 3 6 3
7 0 0 0 7 3 0 0 3 0 3 0 0 0
677 25 f 7 1 1
0 1 1 1 2 0 0 0 1 2 1 1 1 0
848 36 f 7 1 1
0 1 0 1 1 0 0 0 0 0 0 1 1 0
781 31 f 2 4 1
1 0 0 0 1 1 0 1 1 1 1 1 0 0
781 40 f 2 2 8
8 0 0 8 8 0 0 0 0 0 8 8 0 0
677 25 f 3 5 1
1 6 1 6 6 3 0 0 2 2 1 1 1 1
779 33 f 3 2 1
1 0 1 0 0 0 1 0 1 0 0 0 1 0
677 25 m 7 1.5 1
1 1 0 1 1 0 0 0 0 0 1 0 0 0
362 35 f .5 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0
677 41 f 6 2 7
7 7 0 7 7 0 0 0 0 0 8 0 0 0
677 24 m 5 1 5
1 5 0 5 0 0 0 0 1 0 0 0 0 0
833 29 f .5 0 6

6 0 0 0 6 0 0 0 0 0 0 0 0
362 30 f 1 1 1
1 0 0 0 1 0 0 0 1 0 0 0 0 0
850 26 f 6 12 6
6 0 0 0 2 2 2 6 6 6 0 0 6 6
467 25 f 2 3 1
1 0 0 6 1 1 0 0 0 0 1 1 1 1
967 29 f 1 2 7
7 0 0 0 7 0 0 7 7 0 0 0 0 0
833 31 f 1 1 7
7 0 7 0 7 3 0 0 3 3 0 0 0 0
859 40 f 7 1 5
1 5 0 5 5 1 0 0 1 0 0 0 0 0
848 31 m 1 2 1
1 0 0 0 1 1 0 0 4 4 1 4 0 0
222 32 f 2 3 3
3 0 0 0 0 7 0 0 3 0 8 0 0 0
783 33 f 2 0 4
7 0 0 0 7 0 0 0 4 0 4 0 0 0
856 28 f 8 4 2
0 2 0 2 2 0 0 0 2 0 2 0 4 0
781 30 f 3 5 1
1 1 1 1 1 1 0 0 1 1 1 1 1 0
850 25 f 6 3 1
7 5 0 5 7 1 0 0 7 0 1 0 1 0
580 33 f 2.5 4 2
2 0 0 0 2 0 0 0 0 0 8 8 0 0
677 38 f 3 3 1
1 0 0 0 1 0 1 1 1 0 1 0 0 4
677 26 f 2 2 1
1 0 1 0 1 0 0 0 1 1 1 0 0 0
467 52 f 3 2 2
2 6 6 6 6 2 0 0 2 2 2 2 0 0
542 31 f 1 3 1
1 0 1 0 1 0 0 0 1 1 1 1 1 0
859 50 f 9 3 6
6 6 6 6 6 6 6 6 6 3 3 3 6 6
779 26 f 1 2 1
7 0 1 0 1 1 4 1 4 1 1 1 4 4
779 36 m 1.5 2 4
1 4 0 4 4 0 0 4 4 4 4 0 0 0
222 31 f 0 3 7
1 0 0 0 7 0 0 0 0 0 0 0 0 0
362 27 f 1 1 1
1 0 1 0 1 4 0 4 4 1 0 4 4 0
967 32 f 3 2 7
7 0 0 0 7 0 0 0 1 0 0 1 0 0
362 29 f 10 2 2
2 2 2 2 2 2 2 2 2 2 2 7 0 0
677 27 f 3 4 1
0 5 1 1 0 5 0 0 0 1 1 1 0 0
546 32 m 5 .5 8
8 0 0 0 8 0 0 0 8 0 0 0 0 0
688 38 m 2 3 2
2 0 0 0 2 0 0 0 2 0 0 0 1 0
362 28 f 1 1 1

1 0 0 0 1 1 0 4 0 0 0 0 4 0
851 32 f .5 2 4
5 0 0 0 4 0 0 0 0 0 0 0 2 0
967 43 f 2 2 1
1 0 0 0 1 0 0 1 7 0 0 0 1 0
467 44 f 10 4 6
7 6 0 6 6 0 6 0 0 0 0 0 0 6
467 23 f 5 3 1
0 2 1 2 1 0 0 0 1 1 1 1 1 1
783 30 f 1 .5 1
1 0 0 0 1 0 0 0 0 0 0 7 0 0
677 29 f 3 1 2
2 2 2 2 2 0 0 0 0 0 0 0 0 0
859 26 f 9.5 1.5 2
2 2 2 2 2 0 0 2 2 0 0 0 0 0
222 28 f 3 0 2
2 0 0 0 2 0 0 0 0 0 2 0 0 0
966 37 m 2 1 1
7 1 1 1 7 0 0 0 7 0 0 0 0 0
859 31 f 10 10 1
0 1 1 1 1 0 0 0 1 1 0 0 1 0
781 27 f 2 1 2
2 0 0 0 1 0 0 0 4 0 0 0 0 0
677 31 f .5 .5 6
7 0 0 0 0 0 0 0 6 0 0 0 0 0
848 28 f 5 1 2
2 2 0 2 0 0 0 0 2 0 0 0 0 0
781 24 f 3 3 6
1 6 6 6 1 6 0 0 0 0 1 0 1 1
856 27 f 1.5 1 6
2 6 6 6 2 5 0 2 0 0 5 2 0 0
382 30 m 1 2 7
7 0 0 0 7 0 4 7 0 0 0 7 4 4
848 25 f 9 3 1
7 1 1 5 1 0 0 0 1 1 1 1 1 0
382 30 m 1 2 4
7 0 0 0 7 0 4 7 0 0 0 7 4 4
688 40 m 2 3 1
1 0 0 0 1 3 1 0 5 0 4 4 7 1
856 40 f .5 5 5
3 0 0 0 3 0 0 0 0 0 5 5 0 0
966 25 f 2 .5 2
1 0 0 0 2 6 0 0 4 0 0 0 0 0
859 30 f 2 4 2
2 0 0 0 0 2 0 0 0 0 2 0 0 0
849 29 m 10 1 5
7 5 5 5 7 5 5 0 0 0 0 0 7 0
781 28 m 1.5 3 4
1 0 0 0 1 4 4 0 4 4 1 1 4 0
467 35 f 4 2 6
7 6 7 6 6 7 6 7 7 7 7 7 7 6
222 32 f 10 5 1
1 1 0 1 1 0 0 1 1 1 0 0 1 0
677 32 f 1 0 1
1 0 1 0 0 0 0 0 0 0 0 0 0 0
222 54 f 21 4 3

5 0 0 0 7 0 0 7 0 0 0 0 0 0
677 30 m 4 6 1
7 0 0 0 0 1 1 1 7 1 1 0 8 1
683 29 f 1 2 8
8 0 0 0 8 0 0 0 0 8 8 0 0 0
467 38 m 3 5 1
1 0 0 0 1 0 0 1 1 0 0 0 0 0
781 29 f 2 3 8
8 0 0 0 8 8 0 0 8 8 0 8 8 0
781 30 f 1 0 5
5 0 0 0 0 5 0 0 0 0 0 0 0 0
783 40 f 1.5 3 1
1 0 0 0 1 4 0 0 1 1 1 0 0 0
851 30 f 1 1 6
6 0 0 0 6 0 0 0 6 0 0 6 0 0
851 40 f 1 1 5
5 0 0 0 5 0 0 0 0 1 0 0 0 0
779 40 f 1 0 2
2 0 0 0 2 0 0 0 0 0 0 0 0 0
467 37 f 4 8 1
1 0 0 0 1 0 3 0 3 1 1 1 0 0
859 37 f 4 3 3
0 3 7 0 0 7 0 0 0 7 8 3 7 0
781 26 f 4 1 2
2 2 0 2 1 0 0 0 2 0 0 0 0 0
859 23 f 8 3 3
3 2 0 2 3 0 0 0 1 0 0 3 0 0
967 31 f .5 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0
851 38 m 4 2 5
7 5 0 5 4 0 4 7 7 0 4 0 8 0
467 30 m 2 1 2
2 2 0 2 0 0 0 0 2 0 2 0 0 0
848 33 f 2 2 7
7 0 0 0 0 7 0 7 7 0 0 0 7 0
688 35 f 5 8 3
2 2 2 2 2 0 0 3 3 3 3 0 0 0
467 27 f 2 3 1
1 0 1 0 0 1 0 0 1 1 1 0 0 0
783 42 f 3 1 1
1 0 0 0 1 0 0 0 1 0 1 1 0 0
687 40 m 1.5 2 1
7 0 0 0 1 1 0 0 1 0 7 0 1 0
779 30 f 4 8 7
7 0 0 0 7 0 6 7 4 2 2 0 0 6
222 34 f 9 0 8
8 2 0 2 8 0 0 0 0 0 0 0 0 0
467 28 m 3 1 2
2 0 0 0 2 2 0 0 0 2 2 0 0 0
222 28 f 8 4 2
1 2 1 2 2 0 0 1 2 2 0 0 2 0
542 35 m 2 3 2
6 0 7 0 7 0 7 0 0 0 2 2 0 0
677 31 m 12 4 3
7 3 0 3 3 4 0 0 4 4 4 0 0 0
783 45 f 1.5 2 6

6 0 0 0 6 0 0 6 6 0 0 0 0 0
942 34 f 1 .5 4
4 0 0 0 1 0 0 0 0 0 2 0 0 0
222 30 f 8 4 1
1 1 1 1 1 0 0 0 1 1 0 0 0 0
967 38 f 1.5 2 7
7 0 0 0 7 0 0 7 1 1 1 1 0 0
783 37 f 2 1 1
6 6 1 1 6 6 0 0 6 1 1 1 6 0
467 31 f 1.5 2 2
2 0 7 0 7 0 0 7 7 0 0 0 7 0
859 48 f 3 0 7
7 0 0 0 0 0 0 0 0 7 0 0 0 0
490 35 f 1 1 7
7 0 0 0 7 0 0 0 0 0 0 0 8 0
222 27 f 3 2 3
8 0 0 0 3 8 0 3 3 0 0 0 0 0
382 36 m 3 2 4
7 0 5 4 7 4 4 0 7 7 4 7 0 4
859 37 f 1 1 2
7 0 0 0 0 2 0 2 2 0 0 0 0 2
856 29 f 3 1 1
1 0 0 0 1 1 1 1 0 0 1 1 0 1
542 32 m 3 3 7
7 0 0 0 0 7 7 7 0 0 0 0 7 7
783 31 m 1 1 1
1 0 0 0 1 0 0 0 1 1 1 0 0 0
833 35 m 1 1 1
5 4 1 5 1 0 0 1 1 0 0 0 0 0
782 38 m 30 8 5
7 5 5 5 5 0 0 4 4 4 4 4 0 0
222 33 m 3 3 1
1 1 1 1 1 1 1 1 4 1 1 1 1 1
467 24 f 2 4 1
0 0 1 0 1 0 0 0 1 1 1 0 0 0
467 34 f 1 1 1
1 0 0 0 1 0 0 1 1 0 0 0 0 0
781 53 f 2 1 5
5 0 0 0 5 5 0 0 0 0 5 5 5 0
222 30 m 2 5 3
6 3 3 3 6 0 0 0 3 3 3 3 0 0
688 26 f 2 2 1
1 0 0 0 1 0 0 0 1 0 1 1 0 0
222 29 m 8 5 1
1 6 0 6 1 0 0 1 1 1 1 0 0 0
783 33 m 1 2 7
7 0 0 0 7 0 0 0 7 0 0 0 7 0
781 39 m 1.5 2.5 2
2 0 2 0 2 0 0 0 2 2 2 0 0 0
850 22 f 2 1 1
1 0 0 0 1 1 1 0 5 0 0 1 0 0
493 36 f 1 0 5
0 0 0 0 7 0 0 0 0 0 0 0 0 0
967 46 f 2 4 7
7 5 0 5 7 0 0 0 4 7 4 0 0 0
856 41 m 2 2 4

7 4 0 0 7 4 0 4 0 0 0 7 0 0
546 25 m 5 5 8
8 8 0 0 0 0 0 0 0 0 0 0 0 0
222 27 f 4 4 3
2 2 2 3 7 7 0 2 2 2 3 3 3 0
688 23 m 9 3 3
3 3 3 3 3 7 0 0 3 0 0 0 0 0
849 26 m .5 .5 8
8 0 0 0 8 0 0 0 0 8 0 0 0 0
783 29 f 3 3 1
1 0 0 0 4 0 0 4 1 0 1 0 0 0
856 34 f 1.5 2 1
7 0 0 0 7 0 0 7 4 0 0 7 0 0
966 33 m 3 5 4
7 0 0 0 7 4 5 0 7 0 0 7 4 4
493 34 f 2 5 1
1 0 0 0 1 0 0 0 7 0 1 1 8 0
467 29 m 2 4 2
2 0 0 0 2 0 0 2 2 2 2 2 2 2
677 28 f 1 4 1
1 1 1 1 1 0 0 0 1 0 1 0 0 0
781 27 m 2 2 1
1 0 1 0 4 2 4 0 2 2 1 0 1 4
467 24 m 4 4 1
7 1 0 1 1 1 0 7 1 0 0 0 0 0
859 26 m 5 5 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1
848 27 m 7 2 5
7 5 0 5 4 5 0 0 0 7 4 4 0 4
677 25 f 1 2 8
8 0 0 0 0 5 0 0 8 0 0 0 2 0
222 26 f 3.5 0 2
2 0 0 0 2 0 0 0 0 0 0 0 0 0
833 32 m 1 2 1
1 0 0 0 1 0 0 0 5 0 1 0 0 0
781 28 m 2 .5 7
7 0 0 0 7 0 0 0 4 0 0 0 0 0
783 28 f 1 1 1
1 0 0 0 1 0 0 0 0 0 1 1 0 0
222 28 f 5 5 2
2 6 6 2 2 0 0 0 2 2 0 0 2 2
851 33 m 4 5 3
1 0 0 0 7 3 0 3 3 3 3 3 7 5
859 39 m 2 1 1
1 0 0 0 1 0 0 0 0 0 0 1 0 0
848 45 m 2 2 7
7 0 0 0 7 0 0 0 7 0 0 0 0 0
467 37 m 2 2 7
7 0 0 0 0 7 0 0 0 7 0 0 7 0
859 32 m .25 .25 1
1 0 0 0 0 0 0 0 1 0 0 0 0 0

SALES

```
data sales;
  input Region $ CitySize $ Population Product $ SaleType $ Units NetSales;
  cards;
NC S 25000 A100 R 150 3750.00
NC M  125000 A100 R 350 8650.00
NC L 837000 A100 R 800 20000.00
NC S 25000 A100 W 150 3000.00
NC M 125000 A100 W 350 7000.00
NC M 625000 A100 W 750 15000.00
TX M 227000 A100 W 350 7250.00
TX L 5000 A100 W 750 5000.00
;
```

付録 3

UNICODE ライセンス

UNICODE ライセンス V3

Copyright © 1991-2024 Unicode, Inc.

Copyright (c) 1995-2015 International Business Machines Corporation and others

NOTICE TO USER: Carefully read the following legal agreement.BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING DATA FILES, AND/OR SOFTWARE, YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT.IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

Permission is hereby granted, free of charge, to any person obtaining a copy of data files and any associated documentation (the "Data Files") or software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that either (a) this copyright and permission notice appear with all copies of the Data Files or Software, or (b) this copyright and permission notice appear in associated Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.

IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR

CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.
