はじめよう SAS System® 9.4

第 2 版

SAS® ドキュメント

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *Step-by-Step Programming with Base SAS® 9.4, Second Edition*. Cary, NC: SAS Institute Inc.

Step-by-Step Programming with Base SAS® 9.4, Second Edition

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

January 2015

目次

本書について . . . . . *xiii*
はじめよう SAS System 9.4 の新機能 . . . . . *xix*
はじめよう SAS System 9.4 のユーザー補助機能 . . . . . *xxi*

1 部 SAS System について 1

1 章 • SAS System について **3**
SAS System について 3
Base SAS のコンポーネント 4
SAS System によって作成された出力 7
SAS プログラムの実行方法 10
SAS ウィンドウ環境でのプログラムの実行 12
要約 14
詳細情報 15

2 章 • 出力デフォルトの操作 **17**
SAS 9.3 で開始する出力デフォルトの操作 17
詳細情報 24

2 部 データの加工 25

3 章 • DATA ステップ処理について **27**
DATA ステップ処理について 27
SAS データセット: SAS System のキー要素 28
DATA ステップの機能: 基礎知識 34
SAS データセットの作成に必要な情報を提供する 41
要約 49
詳細情報 49

4 章 • 生データから作成する: 基本 **51**
生データについて 52
生データの構造の検証: 注意が必要な要素 52
整列されていないデータの読み込み 53
列状に配置されたデータの読み込み 57
特別な処理が必要なデータの読み込み 60
整列されていないデータを、より柔軟に読み込む 63
複数の入力スタイルを使用する 65
要約 68
詳細情報 69

5 章 • 生データから作成する: 応用 **71**
生データを使用した応用について 71
オブザベーションを作成する前に条件をテストする 72
1 つのレコードから複数のオブザベーションを作成する 74
複数のレコードを読み込み、1 つのオブザベーションを作成する 78

問題の解決: 入力レコードに十分な値が含まれていない場合 . . . 85
要約 . . . 88
詳細情報 . . . 90

6 章 • SAS データセットから作成する . . . **91**
SAS データセットから作成する場合について . . . 91
基礎知識 . . . 92
例で使用される入力 SAS データセット . . . 92
選択したオブザベーションの読み込み . . . 95
選択した変数の読み込み . . . 96
1 つの DATA ステップで複数のデータセットを作成する . . . 100
効率的に処理するため、DROP=データセットオプションと
KEEP=データセットオプションを使用する . . . 102
要約 . . . 103
詳細情報 . . . 105

3 部 基本プログラミング 107

7 章 • DATA ステップ処理の基礎知識 . . . **109**
DATA ステップ処理の概要 . . . 109
例で使用される入力 SAS データセット . . . 110
SAS データセットへの情報の追加 . . . 111
十分な容量の変数の記憶域を定義する . . . 116
オブザベーションの条件付き削除 . . . 117
要約 . . . 117
詳細情報 . . . 118

8 章 • 数値変数の操作 . . . **119**
数値変数の操作について . . . 119
SAS の数値変数について . . . 120
例で使用される入力 SAS データセット . . . 120
数値変数を使用した計算 . . . 121
数値変数の比較 . . . 125
数値変数を効率的に格納する . . . 127
要約 . . . 128
詳細情報 . . . 129

9 章 • 文字変数の操作 . . . **131**
文字変数の操作について . . . 131
例で使用される入力 SAS データセット . . . 132
文字変数の識別および文字値の表示 . . . 133
文字変数の長さの設定 . . . 134
欠損値の処理 . . . 136
文字値の新規作成 . . . 139
数値を文字として処理し、記憶域を節約する . . . 143
要約 . . . 145
詳細情報 . . . 145

10 章 • 選択したオブザベーションの操作 . . . **147**
選択したオブザベーションの操作について . . . 147
例で使用される入力 SAS データセット . . . 148
オブザベーションの選択 . . . 149
条件の作成 . . . 153
文字の比較 . . . 161

要約 . . . 165
詳細情報 . . . 166

11 章 • オブザベーションのサブセットの作成 . . . **167**
オブザベーションのサブセットの作成について . . . 167
例で使用される入力 SAS データセット . . . 168
新しい SAS データセットのオブザベーションを選択する . . . 169
1 つ以上の SAS データセットにオブザベーションを条件付きで書き込む . . . 173
要約 . . . 178
詳細情報 . . . 179

12 章 • グループ化されたオブザベーションや並べ替えられたオブザベーションの操作 . . . **181**
グループ化されたオブザベーションや並べ替えられたオブザベーションの操作について . . . 181
例で使用される入力 SAS データセット . . . 182
グループ化されたデータの操作 . . . 183
並べ替えられたデータの操作 . . . 190
要約 . . . 194
詳細情報 . . . 195

13 章 • 計算に複数のオブザベーションを使用する . . . **197**
計算への複数オブザベーションの使用について . . . 197
例で使用される入力ファイルおよび入力 SAS データセット . . . 198
データセット全体の合計を累計する . . . 199
BY グループの合計の取得 . . . 202
異なるデータセットへの書き込み . . . 204
前のオブザベーションの値の使用 . . . 207
要約 . . . 210
詳細情報 . . . 211

14 章 • プログラミングを簡単にする . . . **213**
ショートカットについて . . . 213
入力ファイルと入力 SAS データセット . . . 214
IF-THEN ステートメントで複数の処理を実行する . . . 215
複数の変数に同じ処理を実行する . . . 217
要約 . . . 220
詳細情報 . . . 221

15 章 • SAS System での日付の操作 . . . **223**
日付の操作について . . . 224
SAS での日付の処理方法について . . . 224
例で使用される入力ファイルおよび入力 SAS データセット . . . 226
日付の入力 . . . 226
日付の表示 . . . 231
計算に日付を使用する . . . 235
SAS 日付関数の使用 . . . 237
期間と SAS 日付値の比較 . . . 239
要約 . . . 240
詳細情報 . . . 241

4 部 SAS データセットの結合 243

16 章 • SAS データセットの結合方法 . . . **245**
SAS データセットの結合について . . . 245

連結の定義 . . . 246
インタリーブの定義 . . . 246
マージの定義 . . . 247
更新の定義 . . . 248
変更の定義 . . . 249
データセットの変更、マージ、更新の比較 . . . 250
詳細情報 . . . 251

17 章 • SAS データセットの連結 . . . **253**
SAS データセットの連結について . . . 253
SET ステートメントを使用したデータセットの連結 . . . 254
APPEND プロシジャを使用したデータセットの連結 . . . 271
SET ステートメントまたは APPEND プロシジャの選択 . . . 275
要約 . . . 276
詳細情報 . . . 276

18 章 • SAS データセットのインタリーブ . . . **279**
SAS データセットのインタリーブについて . . . 279
BY グループ処理の概念について . . . 280
データセットのインタリーブ . . . 280
要約 . . . 284
詳細情報 . . . 285

19 章 • SAS データセットのマージ . . . **287**
SAS データセットのマージについて . . . 288
MERGE ステートメントについて . . . 288
1 対 1 のマージ . . . 288
マッチマージ . . . 294
1 対 1 のマージまたはマッチマージの選択 . . . 309
要約 . . . 314
詳細情報 . . . 314

20 章 • SAS データセットの更新 . . . **315**
SAS データセットの更新について . . . 315
UPDATE ステートメントについて . . . 316
BY 変数の選択方法について . . . 316
データセットの更新 . . . 317
増分値での更新 . . . 322
更新とマージの違いについて . . . 324
欠損値の処理 . . . 327
要約 . . . 331
詳細情報 . . . 331

21 章 • SAS データセットの変更 . . . **333**
SAS データセットの変更について . . . 333
例で使用される入力 SAS データセット . . . 334
SAS データセットの変更: 最も単純な場合 . . . 335
マスタデータセットをトランザクションデータセットのオブザベーションで更新する . . . 336
重複した BY 変数がファイル更新に与える影響について . . . 340
欠損値の処理 . . . 343
要約 . . . 345
詳細情報 . . . 346

22 章 • 複数の SAS データセットのオブザベーションの条件付き処理 . . . **347**
複数の SAS データセットの条件付き処理について . . . 347
例で使用される入力 SAS データセット . . . 348

オブザベーションが読み込まれたデータセットを特定する 351
複数のデータセットから選択したオブザベーションを結合する 356
最後のオブザベーションに基づき計算を実行する 357
要約 359
詳細情報 360

5 部 SAS プログラムのデバッグ 361

23 章 • SAS ログを使用した SAS セッションの分析 **363**
SAS ログを使用した SAS セッションの分析について 364
SAS ログについて 364
SAS ログの場所 366
ログ構造について 366
SAS ログへの書き込み 370
SAS ログで情報を非表示にする 376
ログの表示形式の変更 379
要約 379
詳細情報 381

24 章 • SAS 出力と SAS ログの出力指定 **383**
SAS 出力と SAS ログの出力指定について 383
例で使用される入力ファイルおよび入力 SAS データセット 384
PROC PRINTTO を使用した出力と SAS ログの出力先の指定 385
SAS ウィンドウ環境での出力と SAS ログの格納 387
バッチ(非対話型)環境でのデフォルトの出力先の再定義 388
要約 389
詳細情報 390

25 章 • エラーの診断と回避 **391**
エラーの診断と回避について 391
SAS Supervisor でのジョブのチェック方法について 392
SAS でのエラーの処理方法について 392
エラーの種類について 393
エラーの診断 394
品質管理チェックリストの使用 401
詳細情報 402

26 章 • プログラムの論理エラーの検出 **403**
プログラムの論理エラーの検出 403
DATA ステップデバッガの使用 404
基本的な使用 404
マクロ機能とデバッガの併用 406
例 407

6 部 レポートの作成 419

27 章 • PRINT プロシジャを使用した詳細レポートの作成 **421**
PRINT プロシジャを使用したレポートの作成について 422
例で使用される入力ファイルおよび入力 SAS データセット 422
単純なレポートの作成 424
拡張レポートの作成 434

カスタマイズされたレポートの作成 446
レポートを変更しやすくする 453
要約 457
詳細情報 460

28 章 • TABULATE プロシジャを使用した要約テーブルの作成 **461**
TABULATE プロシジャを使用した要約テーブルの作成について 462
要約テーブルの設計について 462
TABULATE プロシジャの基礎知識 464
例で使用される入力ファイルおよび入力 SAS データセット 467
単純な要約テーブルの作成 468
複雑な要約テーブルの作成 473
要約 484
詳細情報 487

29 章 • REPORT プロシジャを使用した詳細レポートと要約レポートの作成 **489**
REPORT プロシジャを使用した詳細レポートと要約レポートの作成について 489
レポートの作成方法について 490
例で使用される入力ファイルおよび入力 SAS データセット 492
単純なレポートの作成 493
高度なレポートの作成 502
要約 510
詳細情報 514

7 部 プロットとチャートの作成 517

30 章 • 変数間のリレーションシップのプロット **519**
変数間のリレーションシップのプロットについて 519
例で使用される入力ファイルおよび入力 SAS データセット 520
1 つの組み合わせの変数のプロット 524
プロットの拡張 525
複数の組み合わせの変数のプロット 530
要約 536
詳細情報 537

31 章 • 変数を要約するチャートを作成する **539**
変数を要約するチャートの作成について 540
チャート作成ツールについて 540
例で使用される入力ファイルおよび入力 SAS データセット 541
CHART プロシジャを使用した度数グラフの作成 543
度数グラフのカスタマイズ 551
高解像度ヒストグラムの作成 562
要約 574
詳細情報 578

8 部 独自の出力の設計 581

32 章 • SAS ログや出力ファイルに行を書き込む **583**
SAS ログや出力ファイルへの行の書き込みについて 583
PUT ステートメントについて 584
データセットを作成せずに出力を書き込む 584

単純なテキストを書き込む 585
レポートの書き込み 589
要約 596
詳細情報 597

33 章 • SAS 出力とそのカスタマイズについて: 基本 **599**
SAS 出力とそのカスタマイズの基礎知識について 600
出力について 600
例で使用される入力 SAS データセット 602
プロシジャ出力の出力先 603
出力に情報を付加する 604
リスト出力での出力表示の制御 610
ページの表示形式の調整 613
欠損値の表示 621
要約 623
詳細情報 624

34 章 • SAS 出力とそのカスタマイズについて: The Output Delivery System (ODS) **625**
Output Delivery System (ODS)を使用した SAS 出力のカスタマイズについて 626
例で使用される入力データセット 626
ODS 出力形式およびその出力先について 627
出力形式の選択 629
フォーマットされた出力の作成 630
フォーマットする出力の選択 642
ODS 出力のカスタマイズ 647
ODS 出力へのリンクの格納 657
要約 659
詳細情報 662

9 部 SAS ファイルでのデータの格納と管理 665

35 章 • SAS ライブラリの基礎知識 **667**
SAS ライブラリについて 667
SAS ライブラリ概要 668
SAS ライブラリへのアクセス 668
SAS ライブラリにファイルを格納する 670
SAS ライブラリの SAS データセットを参照する 671
要約 673
詳細情報 674

36 章 • SAS ライブラリの管理 **675**
SAS ライブラリの管理について 675
ツールの選択 675
DATASETS プロシジャについて 676
PROC DATASETS セッションについて 677
要約 678
詳細情報 679

37 章 • SAS データセットの情報の表示 **681**
SAS データセットの情報の表示について 681
例で使用される入力データライブラリ 682
SAS ライブラリのディレクトリリストを表示する 682
SAS データセットのコンテンツ情報を表示する 684
異なる形式でコンテンツ情報を表示する 688

要約 . . . 690
詳細情報 . . . 691

38 章 • SAS データセット名と変数属性の変更 . . . **693**
SAS データセット名と変数属性の変更について . . . 693
例で使用される入力データライブラリ . . . 694
SAS データセットの名前の変更 . . . 694
変数属性の変更 . . . 695
要約 . . . 702
詳細情報 . . . 703

39 章 • SAS データセットのコピー、移動、削除 . . . **705**
SAS データセットのコピー、移動、削除について . . . 705
例で使用される入力データライブラリ . . . 706
SAS データセットのコピー . . . 706
特定の SAS データセットのコピー . . . 710
SAS ライブラリと SAS データセットの移動 . . . 711
SAS データセットの削除 . . . 713
SAS ライブラリのすべてのファイルの削除 . . . 714
要約 . . . 715
詳細情報 . . . 716

10 部 SAS 環境の基礎知識 717

40 章 • SAS 環境について . . . **719**
SAS 環境について . . . 719
SAS セッションの開始 . . . 720
SAS 処理モードの選択 . . . 721
要約 . . . 727
詳細情報 . . . 729

41 章 • SAS ウィンドウ環境の使用 . . . **731**
SAS ウィンドウ環境の使用について . . . 732
データの整理 . . . 733
オンラインヘルプの検索 . . . 736
SAS ウィンドウ環境で使用できるコマンドの種類 . . . 736
SAS ウィンドウの操作 . . . 739
テキストの操作 . . . 744
ファイルの操作 . . . 748
SAS プログラムでの作業 . . . 753
出力の操作 . . . 759
要約 . . . 768
詳細情報 . . . 770

42 章 • SAS 環境のカスタマイズ . . . **771**
SAS 環境のカスタマイズについて . . . 771
現在のセッションのカスタマイズ . . . 772
セッション間で引き継がれる設定のカスタマイズ . . . 776
SAS ウィンドウ環境のカスタマイズ . . . 781
要約 . . . 785
詳細情報 . . . 787

11 部 付録 789

付録 1 • 選択例の完全な DATA ステップ **791**
選択例の完全な DATA ステップ 791
CITY データセット 792
UNIVERSITY_TEST_SCORES データセット 793
YEAR_SALES データセット 794
HIGHLOW データセット 795
GRADES データセット 796
USCLIM データセット 797
CLIMATE、PRECIP および STORM データセット 798

付録 2 • DATA ステップデバッガコマンド **801**
ディクショナリ 801

用語集 **817**
キーワード **837**

本書について

SAS 言語の構文規則

SAS 言語の構文規則の概要

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

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

構文の構成要素

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

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

これらの例の SAS 構文では、キーワードには太字が使用されています。

CHAR (*string, position*)

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

ALTER (*alter-password*)

BEST *w.*

REMOVE <*data-set-name*>

この例では、CALL ルーチンの最初の 2 つの単語がキーワードです。

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

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

DO;
... SAS code ...

END;

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

DUPLEX | **NODUPLEX**

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

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

引数

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

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

CHAR (*string, position*)

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

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

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

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

argument(s)

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

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

MISSING *character(s)*;

<LITERAL_ARGUMENT> *argument-1* <<LITERAL_ARGUMENT> *argument-2 ...* >

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

たとえば、BY ステートメントはこの引数を含みます。

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

argument-1 <option(s)> <argument-2 <option(s)> ...>

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

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

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

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

たとえば、LABEL ステートメントは、この形式で複数の引数を含みます。

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

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

次に、この形式で指定された複数の引数の例を示します。

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

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

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

スタイル規則

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

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

ERROR <*message*>;

大文字
リテラルの引数を示します。

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

CMPMODEL=BOTH | CATALOG | XML |

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

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

  LINK *label*;

- 引数に割り当てられる非リテラル値。

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

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

特殊文字

SAS 言語要素の構文には、次の特殊文字も使用されます。

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

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

MAPS = *location-of-maps*

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

この CAT 関数の例では、少なくとも項目が 1 つ必要です。

CAT (*item-1* <, *item-2*, …>)

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

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

CMPMODEL=BOTH | CATALOG | XML

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

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

CAT (*item-1* <, *item-2*, …>)

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

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

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

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

この例では、各ステートメントがセミコロンで終了しています。

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

SAS ライブラリと外部ファイルへの参照

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

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

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

はじめよう SAS System 9.4 の新機能

概要

はじめよう SAS System 9.4 では、SAS プログラムの作成手法を段階的に示します。SAS 概念を示す概念の情報と例が提供されます。このドキュメントのプログラムを実行し、結果を表示することができます。このドキュメントには、SAS コードの記述とデバッグを開始するために必要な基本情報が含まれています。

このドキュメントに対して次の拡張が行われました。

- SAS プログラムのデバッグについての追加情報
- SAS 変数の新しい連結手法
- Output Delivery System (ODS)の更新セクション

SAS 9.4 のメンテナンスリリース 3 では、ドキュメントに対して次の拡張が行われました。

- リスト入力についてのドキュメントに DSD オプションの説明を追加
- ODS スタイルテンプレートの使用方法を更新(“SAS ジョブレベルでの ODS 出力のカスタマイズ” (647 ページ)を参照)
- データセットのマージについてのドキュメントに IN=データセットの説明を追加

SAS プログラムのデバッグ

追加情報と SAS ログ出力例が追加されました。独自の SAS プログラムをより容易にデバッグできるように、SAS ログの項目が説明されています。

DATA ステップデバッガのドキュメントが追加されました。DATA ステップデバッガは、プログラムの論理エラーを検出できるツールです。ツールの説明と例が提供されます。デバッガとあわせて使用するコマンドのリストも提供されます。

SAS 変数の連結

SAS 変数連結の優先手法が導入されました。CAT 関数を使用して、連結された文字列を返します。

Output Delivery System (ODS)

Output Delivery System (ODS)のセクションが更新され、新しい情報が追加されました。ODS では、SAS プロシジャと DATA ステップ出力の生成、格納、再作成の柔軟性が増すとともに、広範囲のフォーマットオプションが提供されます。ODS は、個別プロシジャや DATA ステップを ODS なしで使用する際には使用できないフォーマット機能を備えています。

SAS 9.3 から、SAS ウィンドウ環境のデフォルト出力先は HTML となり、ODS Graphics はデフォルトで有効化されます。これらの新しいデフォルトには複数の利点があります。グラフはテーブルと統合され、すべての出力は新しいスタイルを使用して同じ HTML ファイルに表示されます。この新しいスタイルの HTMLBlue は、オールカラースタイルで、テーブルとモダンな統計グラフを統合するために設計されています。このドキュメントの例は、現在 HTML 出力を示しています。

はじめよう SAS System 9.4 のユーザー補助機能

概要

Base SAS のユーザー補助の詳細については、*SAS 9.4 Companion for Windows* を参照してください。

1 部

SAS System について

1 章
SAS System について 3

2 章
出力デフォルトの操作 17

1 章

SAS System について

SAS System について **3**

Base SAS のコンポーネント **4**
Base SAS の概要 4
データ管理機能 4
プログラミング言語 5
データ分析とレポート作成のユーティリティ 6

SAS System によって作成された出力 **7**
従来の出力 7
Output Delivery System (ODS)の出力 8

SAS プログラムの実行方法 **10**
方法の選択 10
SAS ウィンドウ環境 10
SAS/ASSIST 11
非対話型モード 11
バッチモード 11
対話型ラインモード 12

SAS ウィンドウ環境でのプログラムの実行 **12**

要約 **14**
ステートメント 14
プロシジャ 14

詳細情報 **15**

SAS System について

SAS は、次のタスクの実行を可能にするソフトウェアソリューションの統合システムです。

- データの入力、検索、管理
- レポートやグラフィックの作成
- 統計および数理解析
- ビジネスの予測および意思決定支援
- オペレーションズリサーチおよびプロジェクト管理
- アプリケーション開発

SAS の使用方法は、達成目標によって異なります。多数の SAS System 機能を使用する場合もあれば、少ない機能の使用だけで十分な場合もあります。

SAS System のコア製品は、Base SAS ソフトウェアです。本ドキュメントでは、このソフトウェア製品の使用方法について学習します。このセクションでは、Base SAS の概要を説明します。Base SAS の機能を説明し、SAS 実行方法について扱い、各種出力の概要を示します。

Base SAS のコンポーネント

Base SAS の概要

Base SAS には次が含まれます。

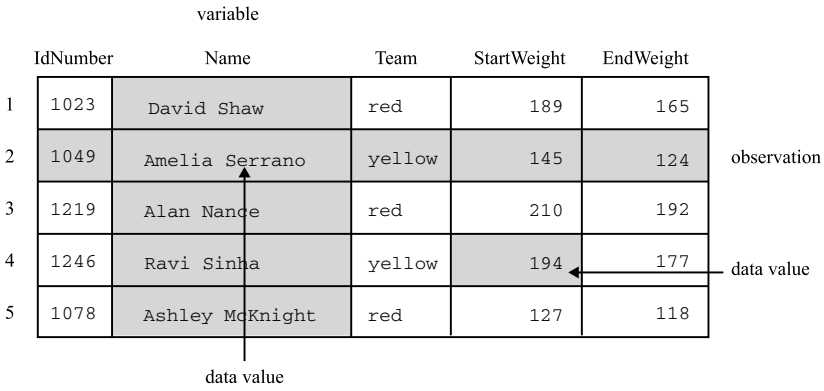
- データ管理機能
- プログラミング言語
- データ分析とレポート作成のユーティリティ

Base SAS の使用法を学習すると、これらの SAS 機能を操作できるようになります。また、SAS 製品はすべて同じ原則に従っているので、これが他の SAS 製品を学習するための下準備にもなります。

データ管理機能

SAS では、SAS データセットと呼ばれる長方形つまり表形式にデータが編成されます。次の図は、SAS データセットを示しています。このデータは、ヘルスクラブでの 16 週間の減量プログラムの参加者について示しています。各参加者のデータには、ID 番号、名前、チーム名、プログラム開始時と終了時の体重(米国ポンド)が含まれます。

図 1.1 *長方形の SAS データセット*



SAS データセットの各行は、個々のエンティティについての情報を表し、オブザベーションと呼ばれます。各列は同じ種類の情報を表し、変数と呼ばれます。個々の情報はデータ値です。SAS データセットでは、エンティティに対するすべてのデータ値がオブザベーションに含まれ、すべてのエンティティに対する同じ種類のデータ値が変数に含まれます。

Base SAS で SAS データセットを作成するには、SAS プログラミング言語のステートメントを使用するプログラムを書きます。DATA ステートメントから始まり、通常は SAS データセットかレポートを作成する SAS プログラムは、DATA ステップと呼ばれます。

次の SAS プログラムでは、ヘルスクラブのデータから WEIGHT_CLUB という SAS データセットが作成されます。

```
data weight_club; 1
   input IdNumber 1-4 Name $ 6-24 Team $ StartWeight EndWeight;2
   Loss=StartWeight-EndWeight; 3
   datalines; 4
1023 David Shaw          red 189 165 5
1049 Amelia Serrano      yellow 145 124 5
1219 Alan Nance          red 210 192 5
1246 Ravi Sinha          yellow 194 177 5
1078 Ashley McKnight     red 127 118 5
;6
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 DATA ステートメントは、WEIGHT_CLUB という SAS データセットの作成を開始するように SAS に指示します。

2 INPUT ステートメントは、入力データから読み取るフィールドを識別し、そこから作成する SAS 変数の名前(IdNumber、Name、Team、StartWeight、EndWeight)を指定します。

3 3 番目のステートメントは割り当てステートメントです。各人の体重の減少を計算し、その結果を新しい変数 Loss に割り当てます。

4 DATALINES ステートメントは、この後にデータ行が続くことを示します。

5 DATALINES ステートメントの後にデータ行が続きます。データが数行しかない場合は、生データを処理するこの方法が役立ちます。(後のセクションで、ファイルに格納された大量のデータへのアクセス方法を説明します。)

6 セミコロンは生データの終わりを表し、ステップ境界となります。これにより、前のステートメントの実行準備が整ったことが示されます。

注: デフォルトでは、WEIGHT_CLUB は一時データセットです。現在のジョブまたはセッションの間のみ存在します。永久 SAS データセットの作成方法については、"DATA ステップ処理について" (27 ページ)を参照してください。

プログラミング言語

SAS 言語の要素

データセット WEIGHT_CLUB を作成したステートメントは、SAS プログラミング言語の一部です。SAS 言語には、ステートメント、式、関数および CALL ルーチン、オプション、出力形式、入力形式が含まれます。これは多くのプログラミング言語が共有する要素です。ただし、SAS 言語要素を使用する方法は、特定のプログラミング規則によって決まります。最も重要な規則を次の 2 つのセクションに記載します。

SAS ステートメントの規則

このドキュメントのプログラムで示される規則(従属ステートメントのインデント、追加スペース、ブランク行など)は、明確さと使いやすさを目的としています。SAS で必須とされているわけではありません。SAS ステートメントの記述規則はほんの少数です。

- SAS ステートメントはセミコロンで終了します。
- SAS ステートメントは、小文字、大文字、またはその 2 つの混合のいずれでも入力できます。

- SAS ステートメントは行の任意のカラム位置から開始でき、同じ行に複数のステートメントを記述できます。
- ステートメントは複数の行にまたがって記述できますが、1 つのワードを 2 行に分けることはできません。
- SAS ステートメントのワードはブランクや特殊文字(WEIGHT_CLUB の例では Loss 変数の計算での等号やマイナス記号など)によって区切られます。

大部分の SAS 名の規則

SAS 名は、SAS データセット名、変数名およびその他の項目に使用されます。次の規則が適用されます。

- SAS 名には、1 文字から 32 文字までを使用できます。
- 最初の文字は、アルファベットまたはアンダースコア(_)でなければなりません。
- 2 文字目以降の文字には、アルファベット、数字、またはアンダースコア(_)を使用する必要があります。
- SAS 名の中にブランクスペースは使用できません。

変数名の特別規則

SAS では、変数名の場合にのみ、変数の作成時に使用した大文字と小文字の組み合わせが記憶されます。内部的には、大文字でも小文字でも処理には影響しません。“CAT”、“cat”、“Cat”はすべて同じ変数を表します。ただし、SAS では、表示のために、各文字の初期の大文字/小文字の区別が記憶され、出力時にはそれを使用して変数名が表示されます。

データ分析とレポート作成のユーティリティ

SAS プログラミング言語は強力かつ柔軟です。これにより分析やレポートをいくつでもプログラミングできます。また、SAS では、SAS プロシジャというビルトインプログラムのライブラリを使用して、プログラミングを単純化できます。SAS プロシジャでは、SAS データセットのデータ値によって、事前にプログラムされたレポートが作成されるので、最小限の労力しか必要としません。

たとえば、次の SAS プログラムでは、SAS データセット WEIGHT_CLUB の変数値を表示するレポートが作成されます。体重の値は米国ポンド単位で表示されます。

```
proc print data=weight_club;
   title 'Health Club Data';
run;
```

PRINT プロシジャというこのプロシジャでは、簡単で系統立てられた形式で変数が表示されます。次の出力は結果を示しています。

図 1.2 SAS データセットの値の表示

Health Club Data

Obs	IdNumber	Name	Team	StartWeight	EndWeight	Loss
1	1023	David Shaw	red	189	165	24
2	1049	Amelia Serrano	yellow	145	124	21
3	1219	Alan Nance	red	210	192	18
4	1246	Ravi Sinha	yellow	194	177	17
5	1078	Ashley McKnight	red	127	118	9

各チームの開始時の体重、終了時の体重、減少した体重の平均を示すテーブルを作成するには、TABULATE プロシジャを使用します。

```
proc tabulate data=weight_club;
   class team;
   var StartWeight EndWeight Loss;
   table team, mean*(StartWeight EndWeight Loss);
   title 'Mean Starting Weight, Ending Weight,';
   title2 'and Weight Loss';
run;
```

次の出力は結果を示しています。

図 1.3 *各チームの平均値のテーブル*

Mean Starting Weight, Ending Weight, and Weight Loss

	Mean		
	StartWeight	EndWeight	Loss
Team			
red	175.33	158.33	17.00
yellow	169.50	150.50	19.00

SAS プログラムで、PROC (プロシジャ)ステートメントから始まり、RUN ステートメントで終了する(または別の PROC または DATA ステートメントによって終了する)部分は、PROC ステップと呼ばれます。前の 2 つの出力を作成する PROC ステップは両方とも次の要素で構成されます。

- PROC ステートメント。PROC というワード、使用するプロシジャの名前、および値を含む SAS データセットの名前から構成されます。(DATA=オプションとデータセット名を省略した場合、プロシジャでは、プログラムで最後に作成された SAS データセットが使用されます。)
- より詳細な指定をするための追加ステートメント(例: CLASS、VAR、TABLE、TITLE ステートメント)。
- RUN ステートメント。これより前のステートメントグループの実行準備が整っていることを示します。

SAS System によって作成された出力

従来の出力

SAS プログラムは、次の種類の出力の一部または全部を生成できます。

SAS データセット
: オブザベーションと変数のテーブルとして格納されたデータ値が含まれます。また、ここには、変数の名前や配置、オブザベーション数、およびデータセットの作成日付など、データセットについての詳細情報も格納されます。SAS データセットは、一時データセットか永久データセットです。このセクションの例では、一時データセット WEIGHT_CLUB を作成します。

SAS ログ
: 入力した SAS ステートメントと、プログラムの実行に関する SAS からのメッセージの記録です。これはディスク上のファイル、モニタでの表示、またはハードコピーリストとして参照できます。SAS ログの正確な表示は、ユーザーの動作環境とサイトによって異なります。ログ 1.1 (8 ページ)にある出力は、このセクションのプログラムに対する典型的な SAS ログを示しています。

レポートまたは単純なリスト
: データ値の単純なリストから大きなデータセットのサブセットのリストまで、または、データのグループ化と要約を行って統計量を表示する複雑な要約レポートなど、多岐にわたります。プロシジャ出力の表示は、サイトと、プログラムで指定するオプションによって異なりますが図 1.2 (6 ページ)および図 1.3 (7 ページ)の出力では、典型的なプロシジャ出力を示しています。また、DATA ステップを使用すると、完全にカスタマイズされたレポートを作成できます。詳細については、“カスタマイズされたレポートの作成” (446 ページ)を参照してください。

カタログなどのその他 SAS ファイル
: データ値のテーブルとして表すことができない情報が含まれます。SAS カタログに格納できる項目の例としては、ファンクションキー設定、SAS/FSP で生成される文字、SAS/GRAPH で生成される表示などがあります。

外部ファイルまたは他のデータベースのエントリ
: SAS プログラムで作成や更新が可能です。SAS/ACCESS では、Oracle などのデータベースで、ファイルを作成し、格納されたファイルを更新できます。

ログ 1.1 *従来の出力: SAS ログ*

```
NOTE: Additional host information: W32_7PRO DNTHOST 6.1.7601 Service Pack 1
Workstation NOTE: SAS initialization used: real time           1.15 seconds cpu
time            0.87 seconds 1    data weight_club; 2       input IdNumber 1-4
Name $ 6-24 Team $ StartWeight EndWeight; 3       Loss=StartWeight-EndWeight;
4       datalines; NOTE: The data set WORK.WEIGHT_CLUB has 5 observations and 6
variables.NOTE: DATA statement used (Total process time): real time
0.01 seconds cpu time            0.01 seconds 10    ; 11   proc tabulate
data=weight_club; NOTE: Writing HTML Body file: sashtml.htm 12      class team;
13      var StartWeight EndWeight Loss; 14      table team, mean*(StartWeight
EndWeight Loss); 15      title 'Mean Starting Weight, Ending Weight,'; 16
title2 'and Weight Loss'; 17   run; NOTE: There were 5 observations read from
the data set WORK.WEIGHT_CLUB.NOTE: PROCEDURE TABULATE used (Total process
time): real time           0.93 seconds cpu time            0.64 seconds
```

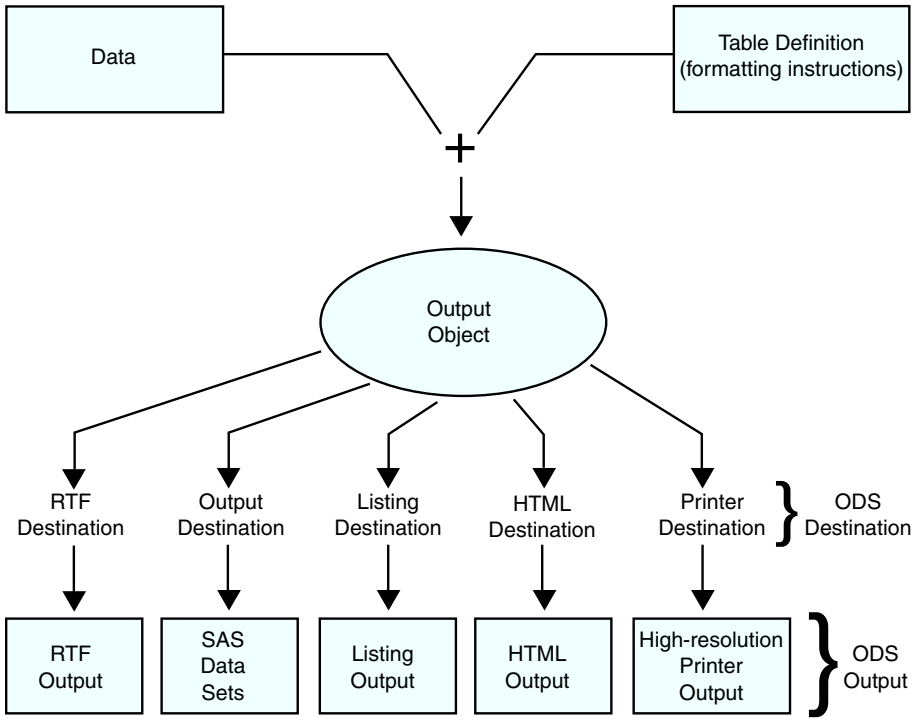
Output Delivery System (ODS)の出力

Output Delivery System (ODS)を使用すると、次のようにさまざまな形式で出力を作成できます。

- HTML ファイル
- 従来の SAS リスト(モノスペース)
- PostScript ファイル
- RTF ファイル(Microsoft Word 用)
- 出力データセット

次の図は、SAS バージョン 8 の出力の概念を示しています。

図 1.4 *ODS 出力の作成モデル*



次の定義は、上図の用語についての説明です。

データ
ODS をサポートする各プロシジャおよび各 DATA ステップでデータが作成されます。データには、SAS データセットと同様の形式のステップ結果(数値と文字)が含まれています。

テーブル定義
テーブル定義は、データのフォーマット方法を記述する命令セットです。この記述には以下が含まれますが、これはほんの一部です。

- 列の順序
- 列ヘッダーのテキストと順序
- データの出力形式
- フォントサイズとフォントの種類

出力オブジェクト
ODS は、フォーマット命令とデータを組み合わせ、出力オブジェクトを作成します。したがって、出力オブジェクトには、プロシジャまたは DATA ステップの結果と、その結果のフォーマット方法についての情報の両方が含まれます。出力オブジェクトには、名前、ラベルおよびパスが含まれます。

注: 多くの出力オブジェクトにはフォーマット命令が含まれていますが、含まれていないものもあります。出力オブジェクトがデータのみで構成されている場合もあります。

ODS 出力先
ODS 出力先には、出力の特定の種類を指定します。ODS では、次のように多数の出力先がサポートされています。

RTF
Microsoft Word 用にフォーマットされた出力を作成します。

出力
SAS データセットを作成します。

リスト
従来の SAS 出力(モノスペース形式)を作成します。

HTML
HTML (Hyper Text Markup Language)でフォーマットされた出力を作成します。ブラウザを使用して Web 上の出力にアクセスできます。

プリンタ
高解像度プリンタ用にフォーマットされた出力を作成します。この種類の出力の例は PostScript ファイルです。

ODS 出力
ODS 出力は、任意の ODS 出力先からのフォーマットされた出力で構成されます。

ODS 出力の詳細については、24 章, “SAS 出力と SAS ログの出力指定” (383 ページ)および 34 章, “SAS 出力とそのカスタマイズについて: The Output Delivery System (ODS)” (625 ページ)を参照してください。

ODS の詳細については、*SAS Output Delivery System: User's Guide* を参照してください。

SAS プログラムの実行方法

方法の選択

SAS プログラムを実行するには、いくつかの方法があります。それらの方法は、実行速度、コンピュータリソースの必要量、プログラムでの対話処理量(すなわち、プログラムの実行時に加えられる変更の種類)に違いがあります。

このドキュメントの例では、プログラムの実行方法にかかわらず、同じ結果が出ます。ただし、少数の例で、プログラムの実行方法によって出力の表示が決定されることもあります。次のセクションでは、SAS プログラムのさまざまな実行方法について簡単に説明します。

SAS ウィンドウ環境

SAS ウィンドウ環境を使用すると、一連のウィンドウによって SAS と直接対話できます。これらのウィンドウを使用すると、ファイルの検索と編成、プログラムの入力と編集、ログ情報の確認、プロシジャ出力の参照、オプションの設定などの共通タスクを実行できます。必要な場合は、この環境からオペレーティングシステムコマンドを発行できます。または、現在の SAS ウィンドウ環境セッションを一時停止し、オペレーティングコマンドを入力して、その後で SAS ウィンドウ環境セッションを再開することもできます。

SAS ウィンドウ環境を使用すると、SAS でのプログラミングが迅速かつ便利になります。SAS の学習や、小さなテストファイルでのプログラム開発では、特にこれが役立ちます。SAS ウィンドウ環境では、他の方法よりも多くのコンピュータリソースを使用しますが、プログラム開発の時間を大幅に節約できます。

SAS ウィンドウ環境の詳細については、41 章, “SAS ウィンドウ環境の使用” (731 ページ)を参照してください。

SAS/ASSIST

SAS の重要な機能の 1 つが、SAS/ASSIST の可用性です。SAS/ASSIST で提供されているポイントアンドクリックのインターフェイスを使用すると、実行するタスクを選択できます。次に、そのタスクを完了するための SAS ステートメントがサブミットされます。SAS/ASSIST を使用するために、SAS 言語でのプログラミング方法を学ぶ必要はありません。

このセクションで前述したとおり、SAS/ASSIST は SAS ステートメントをサブミットすると機能します。その場合、多数の機能が提供されますが、SAS の全機能が備えられているわけではありません。SAS/ASSIST で使用可能なタスク以外のことを実行する場合は、このドキュメントに記載されている SAS でのプログラミングを学習する必要があります。

非対話型モード

非対話型モードでは、SAS ステートメントや、各自の動作環境で必要とされるシステムステートメントを含むファイルを準備し、プログラムをサブミットします。プログラムは即座に実行され、現在のワークステーションセッションが占有されます。プログラムの実行中は、そのセッションでの作業は続行できません。[1] また、通常はプログラムとの対話もできません。[2] ログとプロシジャ出力は、事前に指定された出力先に送られ、通常はプログラムが終了するまで表示されません。プログラムを変更したりエラーを修正したりするには、プログラムを編集して再サブミットする必要があります。

非対話型実行の方がバッチ実行よりも速い場合があります。これは、数ある中から各自のプログラムがスケジュールされるまで待たなくても、コンピュータシステムで即座にプログラムが実行されるためです。

バッチモード

プログラムをバッチモードで実行するには、SAS ステートメントや、各自の動作環境で必要とされるシステムステートメントを含むファイルを準備して、プログラムをサブミットします。

この場合、ワークステーションでは別のタスクの作業ができます。作業中に、その動作環境で(他のユーザーがサブミットしたジョブと一緒に)ジョブの実行がスケジュールされ、実行されます。実行が完了すると、ログとプロシジャ出力を参照できます。

バッチ実行の重要な特徴は、ワークステーション上で他のアクティビティと完全に切り離せることです。実行中のプログラムは表示されず、エラーの発生時にそれを修正することはできません。ログとプロシジャ出力は事前に指定した出力先に送られ、プログラムの実行が終了しなければ参照できません。SAS プログラムを変更するには、動作環境でサポートされているエディタでプログラムを編集して、新しいバッチジョブをサブミットします。

コンピュータリソースに課金される場合、バッチ処理の方が比較的安価にプログラムを実行できます。大きなプログラムの場合や、プログラム実行時に他のタスクのためにワークステーションの使用が必要な場合は特に有用です。ただし、SAS の学習や、新しいプログラムの開発やテストにおいては、バッチモードの使用が効率的ではない場合もあります。

---

1 ワークステーション環境では、別のウィンドウに切り替えて作業を継続できます。

2 使用できる対話処理方法は限られています。たとえば、プログラム内で%INCLUDE ステートメントでアスタリスク(*)オプションを使用できます。

対話型ラインモード

対話型ラインモードセッションでは、SAS プログラムの入力は一度に 1 行ずつで、DATA または PROC の各ステップの終了が SAS に認識されるとすぐにそのステップは自動実行されます。通常は、プロシジャ出力が即座にディスプレイモニタに表示されます。ログやプロシジャ出力の別の箇所を前後にスクロールして表示できるか、または、これらの箇所がトップ画面から消えると失われるかは、サイトのコンピュータシステムやワークステーションによって異なります。プログラム変更とエラー修正には限定的な機能があります。

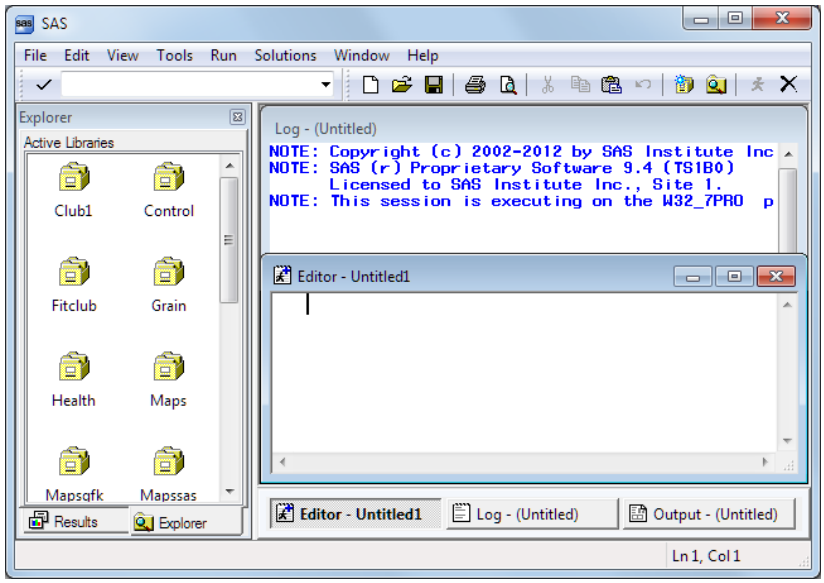
対話型ラインモードセッションでは、ウィンドウ環境よりもコンピュータリソースの使用量が少なくてすみます。ラインモードを使用する場合は、*SAS Statements: Reference* の%INCLUDE、%LIST および RUN ステートメントを把握しておいてください。

SAS ウィンドウ環境でのプログラムの実行

ここまでのセクションで説明した方法のいずれかを使用すれば、このドキュメントの大部分のプログラムを実行できます。このドキュメントでは、SAS セッション内でプログラミングの表示が必要になった場合に、SAS ウィンドウ環境を使用します(Windows および UNIX 動作環境での表示どおり)。SAS ウィンドウ環境の表示は、使用する動作環境によって異なります。SAS ウィンドウ環境の詳細については、41 章, “SAS ウィンドウ環境の使用” (731 ページ)を参照してください。

次の例では、SAS ウィンドウ環境を使用する SAS セッションの概要を説明します。SAS を起動すると、次のウィンドウが表示されます。

図 1.5　*SAS ウィンドウ環境*

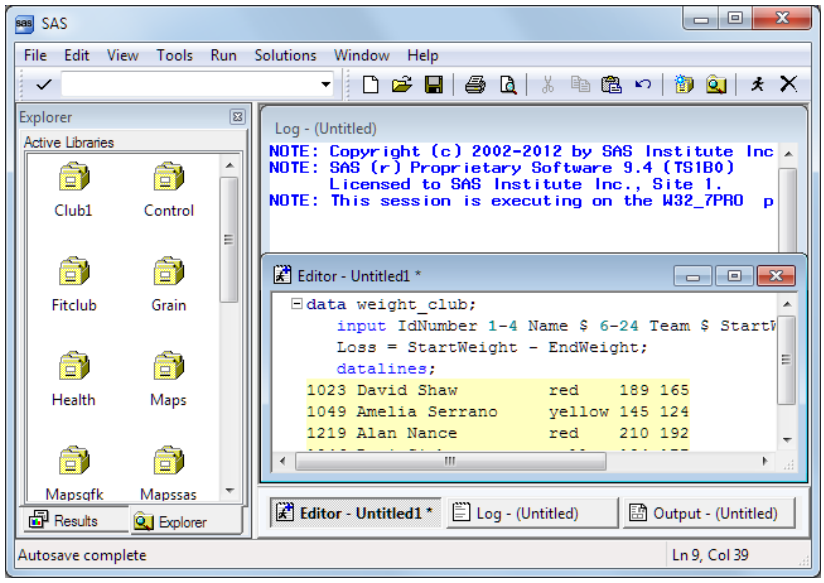


特定のウィンドウ配置、表示色、メッセージ、およびその他の詳細は、サイト、モニタおよび動作環境によって異なります。表示の左側のウィンドウは **SAS エクスプローラ**ウィンドウで、SAS ライブラリ、ファイル、およびその他の項目の割り当てや検索に使用します。右上のウィンドウは**ログ**ウィンドウです。ここには、セッションの SAS ログが表示されます。右下のウィンドウは**プログラムエディタ**ウィンドウです。このウィンドウでは、SAS プログラムの編集に使用するエディタが提供されます。

ヘルスクラブのプログラムを作成するには、**プログラムエディタ**ウィンドウでステートメントを入力します。行番号のオンとオフを切り替えると、プログラムの作成が容易になります。

次の表示画面は、プログラムの先頭を示しています。

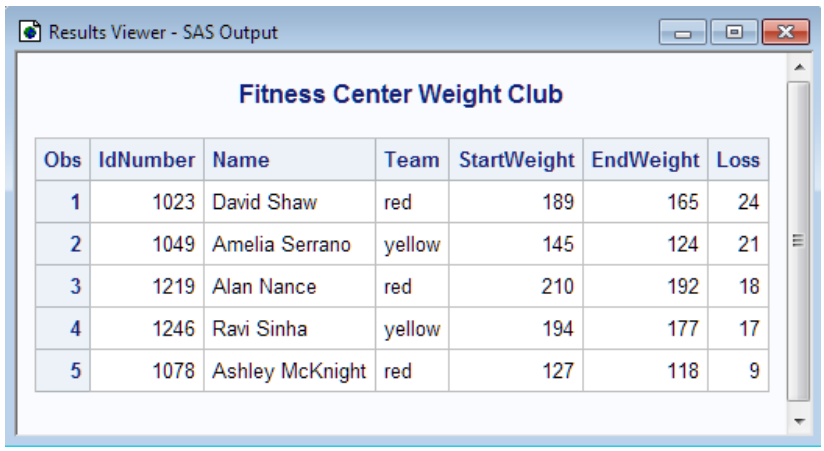
図 1.6 *プログラムエディタウィンドウでのプログラムの編集*



プログラムエディタウィンドウが入力で一杯になった場合は、下にスクロールしてプログラムの入力を続行します。プログラムの編集が終了したら、SAS にサブミットして、出力を参照します。(SAS で出力が作成されない場合は、SAS ログのエラーメッセージを確認します。)

次の表示画面は、**結果ビューア**ウィンドウの 1 ページ目と 2 ページ目を示しています。

図 1.7 *結果ビューアウィンドウの出力の 1 ページ目*



Obs	IdNumber	Name	Team	StartWeight	EndWeight	Loss
1	1023	David Shaw	red	189	165	24
2	1049	Amelia Serrano	yellow	145	124	21
3	1219	Alan Nance	red	210	192	18
4	1246	Ravi Sinha	yellow	194	177	17
5	1078	Ashley McKnight	red	127	118	9

図 1.8 結果ビューアウィンドウの出力の 2 ページ目

出力表示を終了すると、**プログラムエディタ**ウィンドウに戻り、新しいプログラムの作成を開始できます。

デフォルトでは、セッションの終了まで、すべてのサブミットによる出力は**アウトプット**ウィンドウまたは**結果ビューア**ウィンドウに残り、サブミットしたすべてのステートメントがメモリに残っています。出力はいつでも参照できます。前にサブミットしたステートメントを再呼び出しして編集や再サブミットを行うこともできます。また、ウィンドウのコンテンツをクリアすることもできます。

SAS ウィンドウ環境下で使用するコマンドはすべて、ワードまたはファンクションキーとして実行できます。また、どのウィンドウを表示するか決定し、コマンドをファンクションキーに割り当てて、SAS ウィンドウ環境をカスタマイズすることもできます。SAS ウィンドウ環境のカスタマイズの詳細については、42 章, “SAS 環境のカスタマイズ” (771 ページ)を参照してください。

要約

ステートメント

DATA *SAS-data-set*;
DATA ステップが開始され、SAS データセットの作成開始を SAS に指示します。*SAS-data-set* では、作成されるデータセットの名前が指定されます。

%INCLUDE *source(s)* </<SOURCE2> <S2=*length*> <*host-options*>>;
SAS プログラミングステートメントまたはデータ行、あるいはその両方を現在の SAS プログラム内に取り込みます。

RUN;
それより前の SAS ステートメントグループの実行開始を SAS に指示します。

詳細については、*SAS Statements: Reference* を参照してください。

プロシジャ

PROC *procedure* <DATA=*SAS-data-set*>;
PROC ステップを開始し、特定の SAS プロシジャを呼び出して DATA=オプションで指定した SAS データセットを処理するよう SAS に指示します。DATA=オプション

を省略した場合、プロシジャでは、プログラムで最後に作成された SAS データセットが処理されます。

プロシジャ使用の詳細については、*Base SAS Procedures Guide* を参照してください。

詳細情報

基本的な SAS の使用
: 基本的な SAS プログラミング言語の入門レベルの説明については、*The Little SAS® Book: A Primer, Fifth Edition* を参照してください。

DATA ステップ
: SAS データセット作成方法の詳細については、3 章, “DATA ステップ処理について” (27 ページ)を参照してください。

DATA ステップの処理
: DATA ステップの処理の詳細については、7 章, “DATA ステップ処理の基礎知識” (109 ページ)を参照してください。

SAS 環境使用の詳細については、*Getting Started with SAS* を参照してください。

2 章

出力デフォルトの操作

SAS 9.3 で開始する出力デフォルトの操作 **17**
出力デフォルトの操作の概要 17
デフォルト出力先 17
SAS ウィンドウ環境における HTML 出力 18
SAS ウィンドウ環境における LISTING 出力 19
ODS Graphics 20
9.2 の動作を復元する方法 21
詳細情報 **24**

SAS 9.3 で開始する出力デフォルトの操作

出力デフォルトの操作の概要

SAS 9.3 より、SAS ウィンドウ環境の出力はデフォルトで HTML で作成されます。さらに、ODS Graphics がデフォルトで有効化されます。次のセクションでは、これらの新しいデフォルトの利点と、そのデフォルトをこれまでのリリースのデフォルトに一致するように変更する方法について説明します。

- ODS Graphics が SAS 起動時に有効化されます。
- LISTING 出力先が閉じて、HTML 出力先が開きます。
- HTML 出力先のデフォルトスタイルが HTMLBlue です。
- z/OS のデフォルト HTML スタイルが Default から Htmlblue に変更されました。
- PRINTER 出力先のデフォルトスタイルが Pearl です。

LISTING 出力は、これまでのリリースと、SAS 9.3 で SAS をバッチモードで実行するときのデフォルトです。SAS ウィンドウ環境の HTML 出力は、Microsoft Windows 版と UNIX 版の SAS9.3 ではデフォルトですが、他のオペレーティングシステム版ではデフォルトではありません。実際のデフォルトは、レジストリや構成ファイルの設定により異なる場合があります。

デフォルト出力先

9.3 より、デフォルトでは、Windows および UNIX オペレーティングシステムのウィンドウ環境において、LISTING 出力先が閉じて HTML 出力先が開いています。HTML 出力を生成するために ODS HTML ステートメントをサブミットする必要はありません。

また、出力の表示を可能にするために ODS HTML CLOSE ステートメントを使用する必要もありません。ただし、LISTING 出力を作成するには、ODS LISTING ステートメントをサブミットするか、または他の方法で LISTING 出力先を有効化する必要があります。詳細については、“9.2 の動作を復元する方法” (21 ページ)を参照してください。

HTML 出力先では、次の処理が行われます。

- HTML 4.0 埋め込みスタイルシートの生成
- Work ディレクトリに出力ファイルを書き込み
- 出力を表示するための ODS HTML CLOSE ステートメントの指定を不要にする

これらの動作は、ODS HTML CLOSE ステートメントを使用して明示的に ODS HTML 出力先を閉じて、HTML 出力先を再度開くまで持続します。HTML ステートメントを閉じて、新しい ODS HTML ステートメントを発行した後、HTML 出力先では次の処理が行われます。

- 現在のディレクトリに出力ファイルを書き込み
- 出力を表示するための ODS HTML CLOSE ステートメントの指定を必須にする

これらの動作は、SAS セッションを閉じて新しく開くまで持続します。

注意:

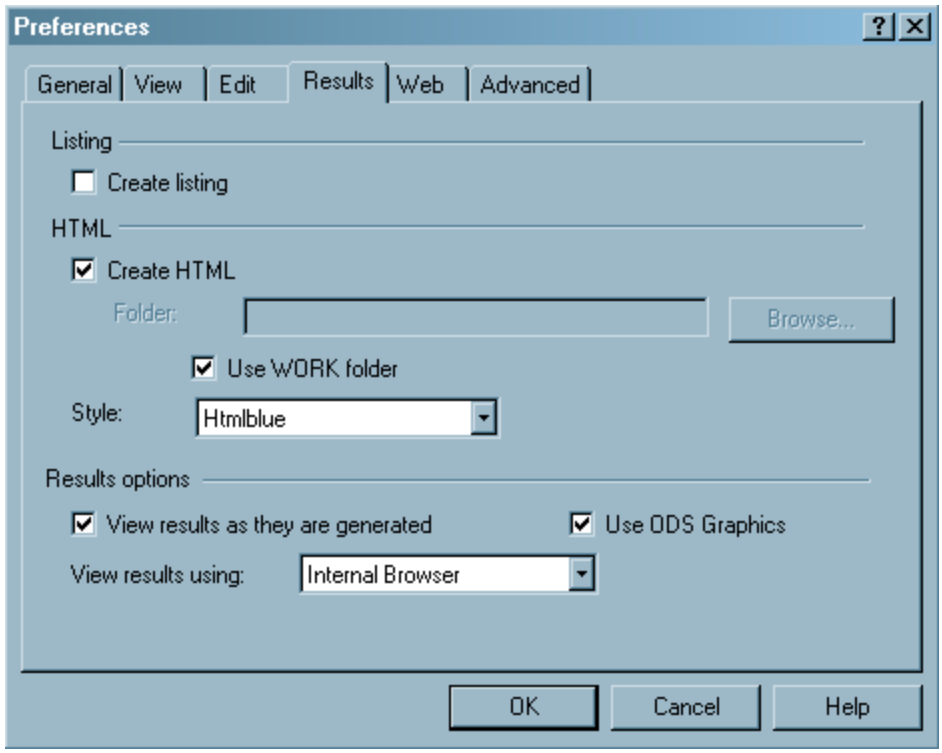
SAS9.3 では、SAS ウィンドウ環境の HTML 出力は、Microsoft Windows 版と UNIX 版のデフォルトですが、他のオペレーティングシステム版およびバッチモードではデフォルトではありません。 SAS をバッチモードまたは他のオペレーティングシステムで実行する場合、デフォルトである LISTING 出力先が開き、ODS Graphics はデフォルトでは有効化されず、HTML 出力のデフォルトスタイルは Styles.Default になります。実際のデフォルトは、レジストリや構成ファイルの設定により異なる場合があります。

SAS ウィンドウ環境における HTML 出力

SAS 9.3 から、SAS ウィンドウ環境のデフォルト出力先は HTML となり、ODS Graphics はデフォルトで有効化されます。これらの新しいデフォルトには複数の利点があります。グラフはテーブルと統合され、すべての出力は新しいスタイルを使用して同じ HTML ファイルに表示されます。この新しいスタイルの HTMLBlue は、オールカラースタイルで、テーブルとモダンな統計グラフを統合するために設計されています。

デフォルト設定の表示と変更を行うには、メイン SAS ウィンドウの上部にあるメニューから**ツール** ⇨ **オプション** ⇨ **プリファレンス**を選択します。次に、**結果**タブを開きます。ニーモニックの TOPR (“topper”と発音)を使用して、この順序を覚えることができます。次の図に、新しいデフォルト設定が指定された SAS **結果**タブを示します。

図 2.1 新しいデフォルト設定が指定されている SAS 結果タブ



結果タブのデフォルト設定を次に示します。

- **リストを作成する**チェックボックスが選択されないので、LISTING 出力は作成されません。
- **HTML を作成する**チェックボックスが選択されるので、HTML 出力が作成されます。
- **WORK フォルダを使用する**チェックボックスが選択されるので、HTML ファイルとグラフイメージファイルの両方が WORK フォルダに保存されます(現在のディレクトリには保存されません)。
- デフォルトスタイルの HTMLBlue が**スタイル**メニューから選択されます。
- **ODS Graphics を使用する**チェックボックスが選択されるので、ODS Graphics が有効化されます。
- **内部ブラウザ**が**ブラウザの選択:**メニューから選択されるので、結果は内部 SAS ブラウザに表示されます。

多くの場合、グラフはデータ分析の重要な要素です。ただし、多数の BY グループを含むプロシジャを使用する場合など、大規模な計算プログラムの実行時は、グラフの作成が望ましくないことがあります。その場合、ODS Graphics を無効化すると、プログラムのパフォーマンスが向上します。SAS プログラムで ODS Graphics の無効化と再有効化を行うには、ODS GRAPHICS OFF ステートメントと ODS GRAPHICS ON ステートメントを使用します。また、**結果**タブの ODS Graphics のデフォルトを変更することもできます。

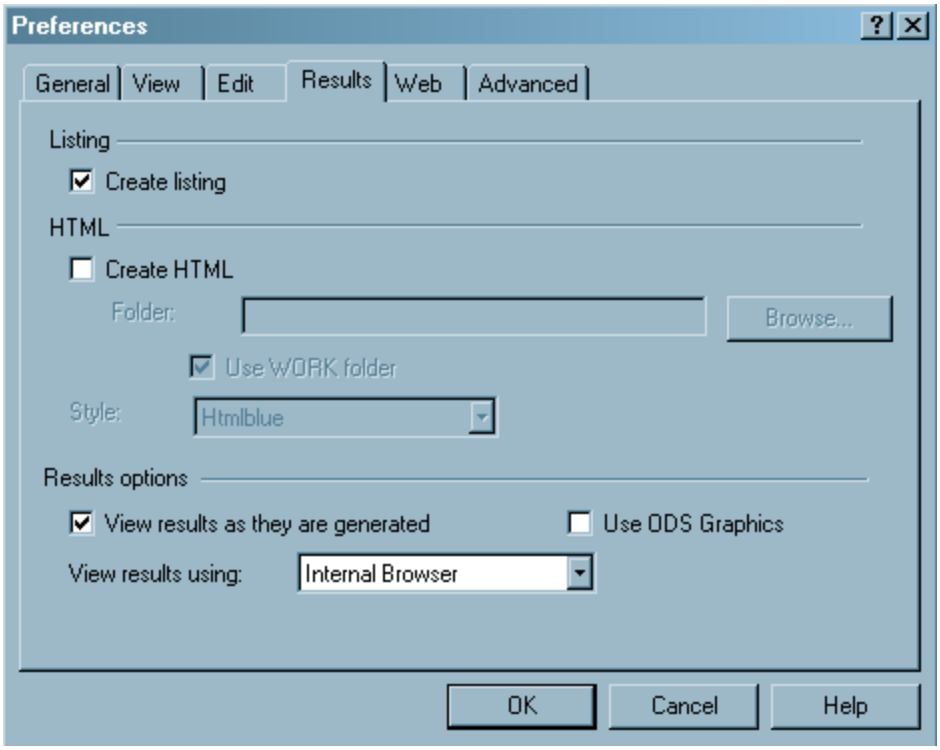
SAS ウィンドウ環境における LISTING 出力

SAS 9.3 より前は、SAS ウィンドウ環境の SAS 出力は、デフォルトで LISTING 出力先に作成されていました。LISTING 出力先では、テーブルがモノスペース形式で表示され、グラフはテーブルと組み合わされていません。

LISTING 出力を作成するには、メイン SAS ウィンドウの上部にあるメニューから**ツール** ⇨ **オプション** ⇨ **プリファレンス**を選択します。次に、**結果**タブを開きます。**リストを作成する**チェックボックスを選択し、**HTML を作成する**チェックボックスは選択しません。

SAS 9.3 より前は、ODS Graphics がデフォルトで無効化されていました。ODS Graphics をデフォルトで有効化または無効化するには、そのチェックボックスを使用します。また、SAS プログラムで ODS Graphics の有効化と無効化を行うには、ODS GRAPHICS ON ステートメントと ODS GRAPHICS OFF ステートメントを使用します。次の図に、旧デフォルト設定が指定された SAS **結果**タブを示します。

図 2.2 *旧デフォルト設定が指定されている SAS 結果タブ*



ODS Graphics

SAS 9.3 より、テンプレートベースのグラフィック(しばしば ODS Graphics と呼ばれる)がデフォルトで作成されます。ODS Graphics には、すべてのグラフィック出力が含まれます。この場合、種類 STATGRAPH のコンパイル済み ODS テンプレートを使用してグラフィック出力が作成されます。用意されているテンプレートは、Sashelp.Tmplmst に格納されています。ODS Graphics の場合、ODS GRAPHICS ステートメントを使用してグラフィック環境を制御する必要があります。Windows および UNIX オペレーティングシステムの SAS ウィンドウ環境では、ODS Graphics を有効化するために ODS GRAPHICS ON ステートメントを指定する必要はありません。

注: SGSCATTER、SGRENDER、SGPLOT、SGPANEL プロシジャでは、ODS Graphics が有効化されなくても、常にグラフが生成されます。

9.2 の動作を復元する方法

概要

次の 3 つの方法のいずれかで、出力デフォルトを変更して 9.2 の動作に戻せます。

- **プリファレンス**ウィンドウの**結果**タブを使用します。これにより、変更を元に戻すまで動作が変更されたままになります。
- ODS ステートメントを使用します。この変更は、現在の SAS セッションの間のみ持続します。
- ODSSTYLE、ODSDEST および ODSGRAPHICS システムオプションを使用します。

プリファレンスウィンドウの使用

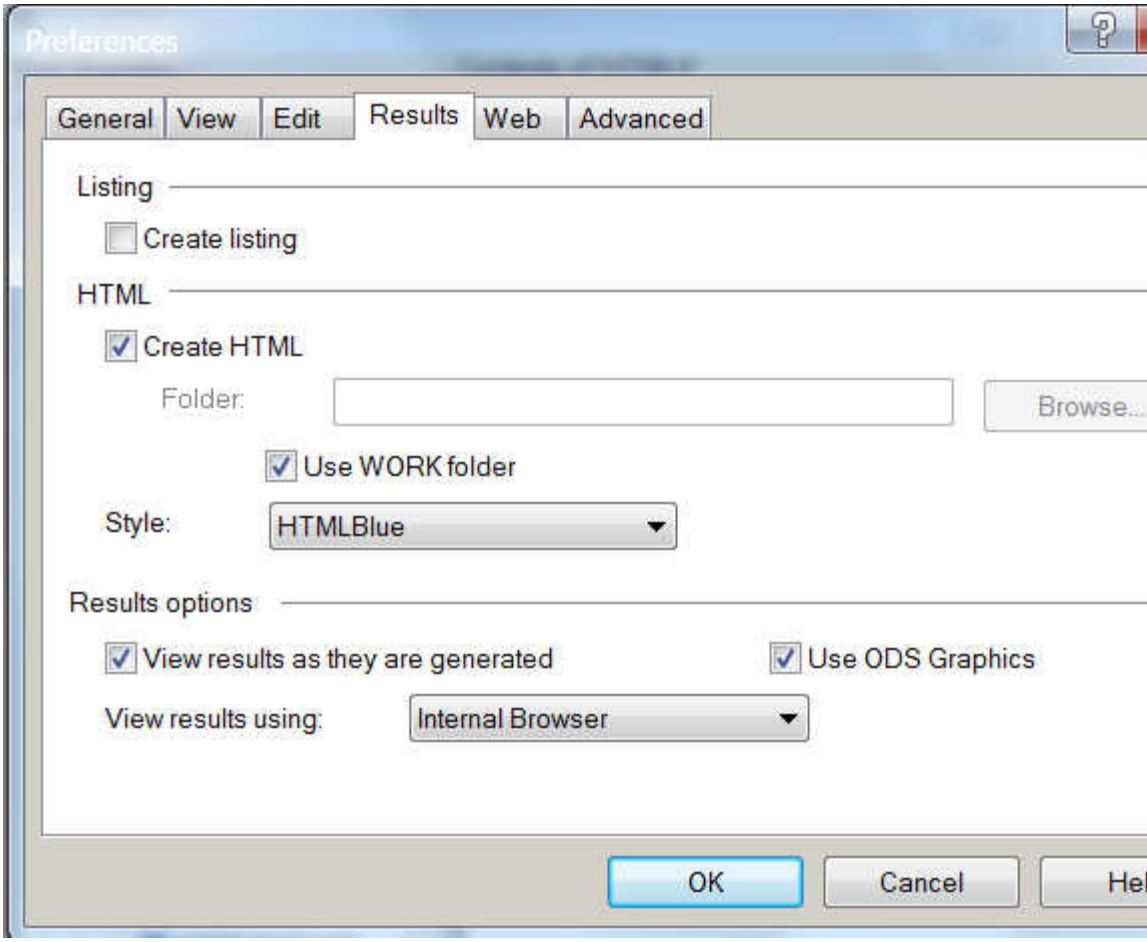
SAS 9.3 から、SAS ウィンドウ環境のデフォルト出力先は HTML となり、ODS Graphics はデフォルトで有効化されます。これらの新しいデフォルトには複数の利点があります。グラフはテーブルと統合され、すべての出力は新しいスタイルを使用して同じ HTML ファイルに表示されます。この新しいスタイルの HTMLBlue は、オールカラースタイルで、テーブルとモダンな統計グラフを統合するために設計されています。

注: デフォルト HTML は HTML4 です。プリファレンスウィンドウで HTML に適用された変更は、デフォルト HTML バージョンにのみ影響します。HTML5 や、その他の HTML バージョンには影響しません。

デフォルト設定の表示と変更を行うには、メイン SAS ウィンドウの上部にあるメニューから**ツール** ⇨ **オプション** ⇨ **プリファレンス**を選択します。次に、**結果**タブを開きます。**プリファレンス**ウィンドウの設定は、明示的に変更するまで持続します。次の図に、新しいデフォルト設定が指定された SAS **結果**タブを示します。

デフォルトで LISTING 出力のみを作成するには、**リストを作成する**を選択し、**HTML を作成する**を選択解除します。ODS Graphics を無効化するには、“ODS Graphics を使用する”を選択解除します。

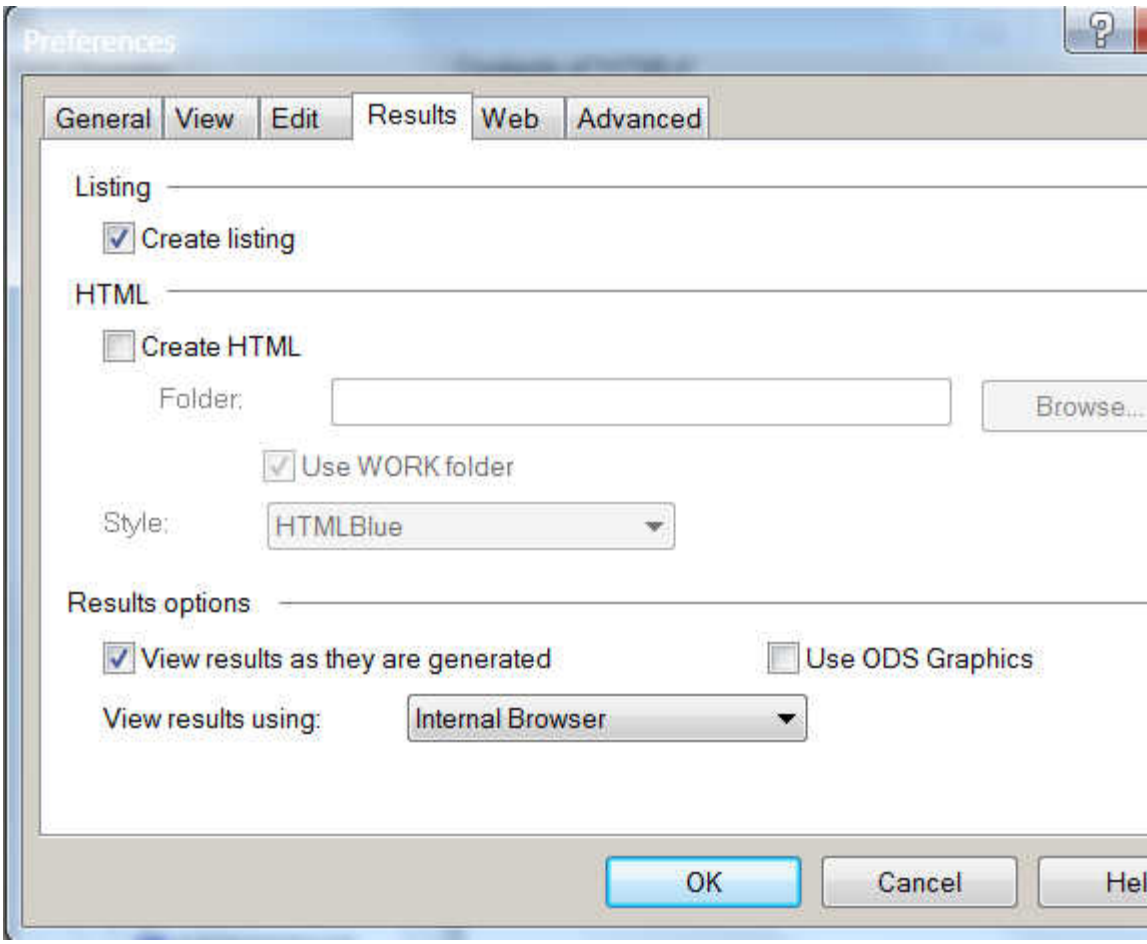
図 2.3 ウィンドウ環境のデフォルトプリファレンスウィンドウ



次の選択を設定する場合、デフォルト出力先は、ODS ステートメント、ODSDEST システムオプションまたは**プリファレンス**ウィンドウを使用して明示的に変更するまでは LISTING です。次の選択設定がデフォルトの場合、ODS GRAPHICS ON ステートメントを指定するか、ODSGRAPHICS システムオプションを使用するか、または**プリファレ**

ンスウィンドウで設定を変更して ODS Graphics を有効化するまで、ODS Graphics は常に無効化されます。

図 2.4 9.3 より前のデフォルトに設定された結果タブ



ODS ステートメントの使用

デフォルト出力先を HTML から LISTING に変更し、ODS Graphics を無効化するには、次の ODS ステートメントを使用します。

```
ods graphics off;
ods html close;
ods listing;
```

これらのステートメントによって現在の SAS セッションの動作が変更されます。新しい SAS セッションを開始すると、デフォルトは SAS 9.3 の動作に戻ります。

システムオプションの使用

SAS9.3 には、デフォルト出力を制御する 3 つの新しいシステムオプションがあります。

ODSSTYLE=システムオプション
: デフォルトスタイルを指定します。デフォルトスタイルを Styles.Default に変更するには、**`ODSSTYLE=styles.default`** を指定します。“ODSSTYLE= System Option” (*SAS Output Delivery System: User's Guide*)。

ODSGRAPHICS=システムオプション
: ODS Graphics がデフォルトで有効化されるかどうかを指定します。ODS Graphics をデフォルトで無効化するには、**`ODSGRAPHICS=OFF`** を指定します。

ODSGRAPHICS=システムオプションの詳細については、“ODSGRAPHICS= System Option” (*SAS Output Delivery System: User's Guide*)を参照してください。

ODSDEST=システムオプション
SAS ウィンドウ環境のデフォルト出力先を指定します。デフォルト出力先を LISTING に変更するには、**`ODSDEST=LISTING`** を指定します。ODSDEST=システムオプションの詳細については、“ODSDEST= System Option” (*SAS Output Delivery System: User's Guide*)を参照してください。

詳細情報

- ODS GRAPHICS ステートメントの詳細については、“ODS GRAPHICS Statement” (*SAS Output Delivery System: User's Guide*)を参照してください。
- ODS Graphics テンプレート作成の詳細については、“TEMPLATE Procedure: Creating ODS Graphics”(SAS Output Delivery System: Procedures Guide)を参照してください。
- *SAS ODS Graphics: Procedures Guide*
- *SAS Graph Template Language: ユーザーガイド*
- *SAS Graph Template Language: リファレンス*
- *SAS ODS Graphics Designer: ユーザーガイド*
- *SAS ODS Graphics Editor: ユーザーガイド*

2 部

データの加工

3 章
DATA ステップ処理について 27

4 章
生データから作成する: 基本 51

5 章
生データから作成する: 応用 71

6 章
SAS データセットから作成する 91

3 章

DATA ステップ処理について

DATA ステップ処理について **27**
目的 27
前提条件 28

SAS データセット: SAS System のキー要素 **28**
SAS データセットの機能について 28
SAS データセットの構造について 29
FILENAME ステートメントと LIBNAME ステートメントの使用 31
一時 SAS データセットと永久 SAS データセット 32

DATA ステップの機能: 基礎知識 **34**
DATA ステップの概要 34
コンパイルフェーズ 35
実行フェーズ 36
DATA ステップの例 37

SAS データセットの作成に必要な情報を提供する **41**
SAS データセット作成の概要 41
データ読み込み方法の指示: 入力スタイル 42
年が 2 桁と 4 桁の日付値の読み込み 43
SAS での変数定義 43
データの場所指定 45
SAS ジョブでの外部ファイルの使用 46
外部ファイルの直接指定 46
ファイル参照名を使用した外部ファイルの参照 47

要約 **49**
ステートメント 49

詳細情報 **49**

DATA ステップ処理について

目的

DATA ステップは、SAS プログラミングの基本要素の 1 つです。これにより、SAS プログラムの分析やレポート作成のプロシジャで使用するデータセットが作成されます。独自の SAS データセットの作成方法を学習するには、DATA ステップの基本構造、機能およびコンポーネントを理解することが重要です。このセクションでは、次を学習します。

- SAS データセットの概要とその必要性
- DATA ステップの機能
- SAS データセットの作成を可能にするために、SAS に提供する必要がある情報

前提条件

先に進む前に、1 章, “SAS System について” (3 ページ)で説明した概念を理解している必要があります。

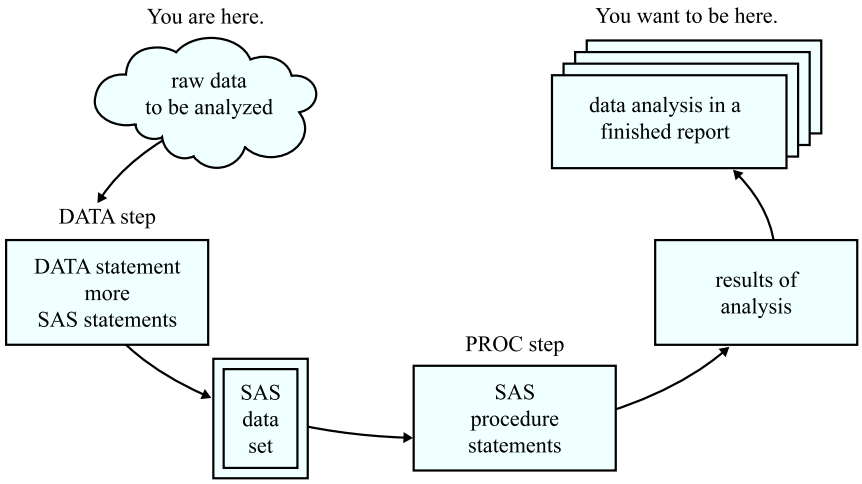
SAS データセット: SAS System のキー要素

SAS データセットの機能について

SAS を使用すると、なんらかの方法でデータの分析や処理を行う方法が提供されることで、問題を解決できます。まず、データを、SAS で認識、処理できる形式に変換する必要があります。データをその形式にすると、分析やレポート生成が可能になります。

次の図は、この処理が最も単純な場合を示しています。

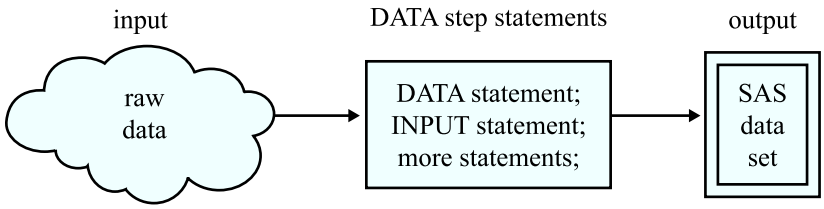
図 3.1　生データから最終分析まで



最初は生データから始まります。これは、SAS で未処理のデータの集合です。DATA ステップと呼ばれる一連のステートメントを使用して、データを SAS データセットに取り込みます。この後で、追加の DATA ステッププログラミングや SAS プロシジャを使用して、データをさらに処理できます。

最も単純な形式の DATA ステップは、次の図に示す 3 つのコンポーネントで表されます。

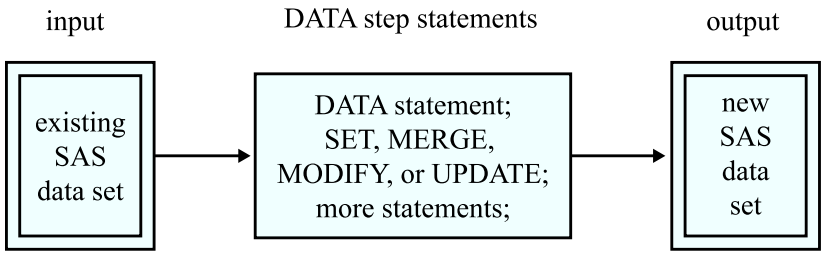
図 3.2 生データから SAS データセットまで



SAS で生データ形式の入力が処理され、SAS データセットが作成されます。SAS データセットがあれば、それを他の DATA ステップへの入力として使用できます。

次の図は、SAS データセットの新規作成に使用できる SAS ステートメントを示しています。

図 3.3 1 つの SAS データセットを使用して別のデータセットを作成



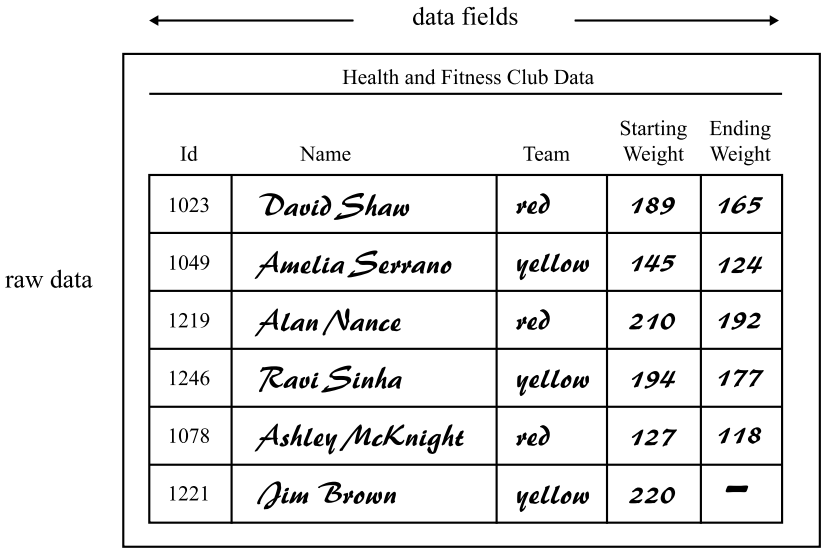
SAS データセットの構造について

SAS データセットを、データを識別、格納する長方形構造と見なします。SAS データセット内のデータについては、追加の DATA ステップを使用してさらに処理したり、SAS プロシジャで多くの種類の分析を実行したりできます。

長方形構造の SAS データセットは、データ値が格納された行と列から成ります。SAS データセットの行はオブザベーション、列は変数と呼ばれます。生データファイルでは、行がレコード、列がフィールドと呼ばれます。変数には、オブザベーションの項目すべてのデータ値が含まれます。

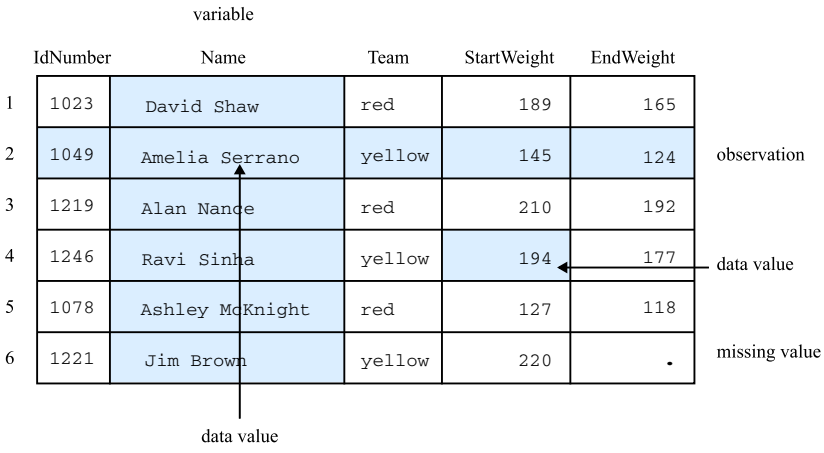
たとえば、次の図は、ヘルスクラブの参加者についての生データの集合を示しています。各レコードに、1 人の参加者についての情報が含まれます。

図 3.4 ヘルスクラブの生データ



次の図は、ヘルスクラブのレコードを簡単に SAS データセットの一部に変換できることを示しています。各レコードがオブザベーションになります。この場合、各オブザベーションが 1 人のプログラム参加者を表します。レコードの各フィールドが変数になります。変数は、各参加者の ID 番号、名前、チーム名、16 週間プログラムの開始時と終了時の体重を表しています。

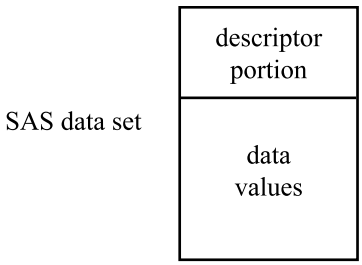
図 3.5 SAS データセットへのデータの組み込み



SAS データセットでは、すべてのオブザベーションそれぞれに対してすべての変数が存在します。各オブザベーションで一部のデータが存在しない場合について考えてみます。あるオブザベーションの数値変数 EndWeigh の値が記録されていないために生データが不完全な場合、この欠損値は、上図のオブザベーション 6 で示されているように、プレースホルダの役割を果たすピリオドで表されます。(文字変数の欠損値はブランクで表されます。文字変数と数値変数については、このセクションで後述します。) 値を欠損としてコーディングすると、データが不完全なデータセットにオブザベーションを追加しても、SAS データセットとして必須の長方形の形状は保たれます。

次の図に示すように、各 SAS データセットには、データ値に加えて、ディスクリプタ部が含まれます。

図 3.6 SAS データセットの各部分



ディスクリプタ部は、すべての変数の名前や属性、データセット内のオブザベーション数、およびデータセットの作成や更新の日時など、SAS がデータセットについて記録する詳細情報から成ります。

動作環境の情報

動作環境や SAS データセットの書き込みに使用するエンジンによっては、SAS データセットについての追加情報をディスクリプタ部に格納できます。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

FILENAME ステートメントと LIBNAME ステートメントの使用

FILENAME ステートメント

FILENAME ステートメントでは、ファイル参照名が外部ファイルに割り当てられます。ファイル参照名と外部ファイル間の関連付けは、SAS セッションの間のみ、または別の FILENAME ステートメントを使用してファイル参照名を変更するまで持続します。

FILENAME ステートメントでは、次の形式が使用されます。

FILENAME *fileref* '*your-input-or-output-file*';

次の FILENAME の例では、myweight.dat という名前のファイルにファイル参照名 MyWeight が割り当てられます。

```
FILENAME MyWeight 'C:\sasuser\MyWeight\myweight.dat';
```

詳細については、“FILENAME Statement” (*SAS Statements: Reference*)を参照してください。

LIBNAME ステートメント

LIBNAME ステートメントでは、SAS データセットか、または SAS データセットと同様にアクセス可能な DBMS ファイルに、ライブラリ参照名が割り当てられます。ライブラリ参照名と SAS ライブラリ間の関連付けは、SAS セッションの間のみ、または別の LIBNAME ステートメントを使用してライブラリ参照名を変更するまで持続します。

LIBNAME ステートメントでは、次の形式が使用されます。

LIBNAME *libref* '*your-SAS-library*';

次の LIBNAME の例では、MyWeight という名前の SAS ライブラリにライブラリ参照名 Weight が割り当てられます。

```
LIBNAME Weight 'C:\sasuser\MyWeight';
```

詳細については、“LIBNAME Statement” (*SAS Statements: Reference*)を参照してください。

一時 SAS データセットと永久 SAS データセット

一時 SAS データセットの作成と使用

DATA ステップを使用して 1 レベル名の SAS データセットを作成する場合は、通常、一時 SAS データセットを作成します。これは現在のセッション中のみ存在するデータセットです。SAS では、このデータセットは WORK と呼ばれる SAS ライブラリに入れられます。ほとんどの動作環境では、SAS で WORK ライブラリに格納されたファイルはすべて、セッションの終了時に削除されます。

一時データセット WEIGHT_CLUB を作成する DATA ステップの例を次に示します。

```
data weight_club;
   input IdNumber Name $ 6-20 Team $ 22-27 StartWeight EndWeight;
   datalines;
1023 David Shaw      red    189 165
1049 Amelia Serrano  yellow 145 124
1219 Alan Nance      red    210 192
1246 Ravi Sinha      yellow 194 177
1078 Ashley McKnight red    127 118
1221 Jim Brown       yellow 220 .
;
```

SASSAS 上のプログラムコードでは、一時データセットが WEIGHT_CLUB.SAS と呼ばれています。ただし、ここですべての一時データセットに第 1 レベル名 WORK が割り当てられ、WEIGHT_CLUB データセットは 2 レベル名の WORK.WEIGHT_CLUB を使用して参照されます。

次の SAS ログ出力では、一時データセット名が表示されています。

ログ 3.1 *SAS ログ: WORK.WEIGHT_CLUB 一時データセット*

```
1    data weight_club; 2       input IdNumber 1-4 Name $ 6-24 Team $ StartWeight
EndWeight; 3       Loss=StartWeight-EndWeight; 4       datalines;
```

SAS では、1 レベルのみの名前を持つすべての SAS データセットに第 1 レベル名 WORK が割り当てられるため、ユーザーは WORK を使用する必要はありません。これらの一時データセットは、WEIGHT_CLUB などの 1 レベル名を使用して参照することができます。

この SAS データセットを後の DATA ステップまたは PROC ステップで参照するには、1 レベル名を使用します。

```
proc print data = weight_club;
run;
```

永久 SAS データセットの作成と使用

永久 SAS データセットを作成するには、WORK 以外の SAS ライブラリを示す必要があります。(WORK は、SAS が一時 SAS ライブラリに自動的に割り当てる予約ライブラリ参照名です。)LIBNAME ステートメントを使用して、使用している動作環境のファイルシステムで SAS ライブラリにライブラリ参照名を割り当てます。ライブラリ参照名は、SAS ライブラリ参照の省略表現方法として機能します。LIBNAME ステートメントの形式は、次のとおりです。

LIBNAME *libref* '*your-data-library*';

libref
SAS ファイルの格納場所に対するショートカット名です。*libref* は有効な SAS 名である必要があります。アルファベットまたはアンダースコアを先頭にする必要があります。また、大文字と小文字、数字、アンダースコアを含められます。ライブラリ参照名の最大長は 8 文字です。

'*your-data-library*'
SAS ライブラリの物理名を指定してください。物理名には動作環境で判別できる名前を指定します。

一部の動作環境では、ライブラリ参照名と物理ファイル名に制限事項を追加適用できます。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

DATA ステップとともに使用している LIBNAME ステートメントの例を次に示します。

```
libname saveit 'your-data-library'; 1
data saveit.weight_club; 2
   ...more SAS statements...
;

proc print data = saveit.weight_club; 3
run;
```

次のリストは、番号付き項目に対応しています。

1 LIBNAME ステートメントで、ライブラリ参照名 SAVEIT と *your-data-library* が関連付けられます。*your-data-library* は SAS ライブラリに対する動作環境での名前です。

2 永久 SAS データセットを新規作成し、この SAS ライブラリに格納するには、DATA ステートメントで 2 レベル名 SAVEIT.WEIGHT_CLUB を使用する必要があります。

3 この SAS データセットを後の DATA ステップまたは PROC ステップで参照するには、PROC ステップで 2 レベル名 SAVEIT.WEIGHT_CLUB を使用する必要があります。

詳細については、35 章, “SAS ライブラリの基礎知識” (667 ページ)を参照してください。

このドキュメントで使用されている規則

例で使用されるデータセットは、通常、1 レベル名で指定された一時データセットとして示されます。

```
data fitness;
```

このドキュメントでも、まれに、データセットが永久 SAS データセットとして作成される場合があります。そのようなデータセットは 2 レベル名で指定され、永久 SAS データセットを作成する各 DATA ステップの前に LIBNAME ステートメントが指定されます。

```
libname saveit 'your-data-library';
data saveit.weight_club;
```

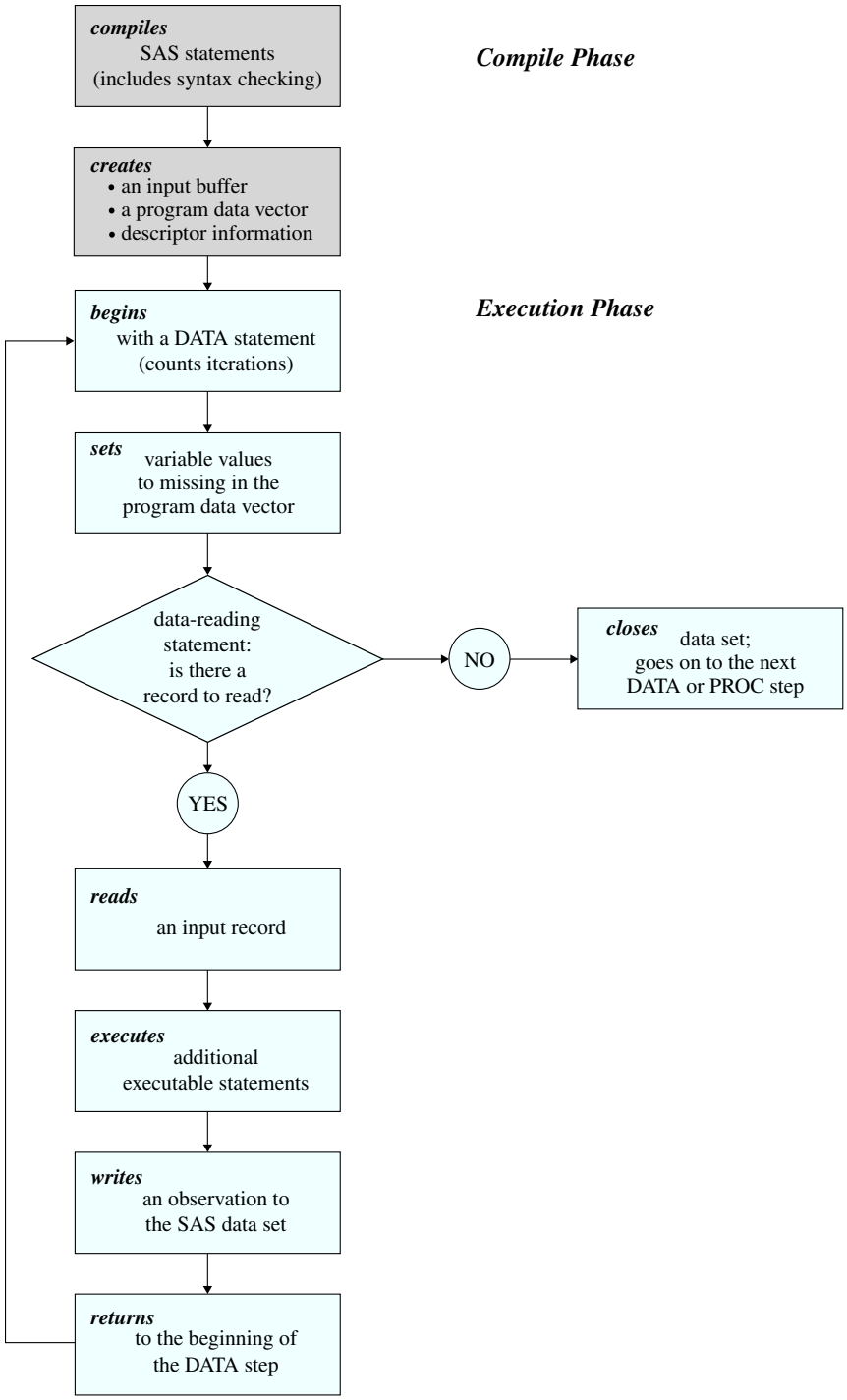
DATA ステップの機能: 基礎知識

DATA ステップの概要

DATA ステップは、DATA ステートメントで始まる SAS ステートメントの集まりから成ります。DATA ステートメントでは、SAS データセット作成プロセスが開始され、データセットの名前が指定されます。DATA ステップを構成するステートメントはコンパイルされ、構文がチェックされます。構文が正しければ、ステートメントが実行されます。最も単純な形式の DATA ステップは、自動的に出力をして前に戻る処理を繰り返すループです。

次の図は、単純な DATA ステップにおける処理フローを示しています。

図 3.7 典型的な DATA ステップにおける処理フロー



コンパイルフェーズ

DATA ステップを実行しようとしてサブミットすると、SAS ステートメントの構文がチェックされ、ステートメントがコンパイルされます。つまり、SAS ステートメントが自動的にマシンコードに変換されます。さらにコードが処理され、次のものが作成されます。

入力バッファ
メモリの論理領域です。プログラムの実行時に、生データファイルから各データレコードが読み込まれます。(ただし、SAS データセットから読み込む場合、データは直接プログラムデータベクトルに書き込まれます。)

プログラムデータベクトル
メモリの論理領域です。ここで SAS System はデータセットを 1 オブザベーションずつ作成します。プログラムを実行すると、入力バッファからデータ値が読み込まれるか、SAS 言語ステートメントが実行されてデータ値が作成されます。読み込まれたり作成された値は、プログラムデータベクトル内の適切な変数に割り当てられます。SAS System は、このベクトルにある値を 1 つのオブザベーションとして SAS データセットに書き出します。

プログラムデータベクトルには、_N_と_ERROR_という 2 つの自動変数も含まれます。自動変数_N_は、DATA ステップが反復された回数をカウントします。自動変数_ERROR_は、DATA ステップの実行中にエラーが発生したかどうかを示す値です。これらの自動変数は、出力データセットには書き込まれません。

ディスクリプタ情報
SAS データセットごとの情報です。データセットの属性と変数の属性が含まれます。SAS System でディスクリプタ情報が作成、保持されます。

実行フェーズ

DATA ステップの実行可能ステートメントはすべて、反復のたびに 1 回ずつ実行されます。入力ファイルに生データが含まれている場合は、レコードが入力バッファに読み込まれます。次に、入力バッファの値が読み込まれ、その値がプログラムデータベクトル内の適切な変数に割り当てられます。プログラムステートメントによって作成された変数の値も計算され、その値がプログラムデータベクトルに書き込まれます。プログラムが DATA ステップの最後に達すると、デフォルトで 3 つの処理が発生します。この処理が、SAS 言語と他の大部分のプログラミング言語とが異なる点です。

1. プログラムデータベクトルの現在のオブザベーションが、データセットに書き込まれます。
2. プログラムがループして、DATA ステップの最初に戻ります。
3. プログラムデータベクトル内の変数が欠損値にリセットされます。

   注: 次の例外が適用されます。

   - RETAIN ステートメントで指定した変数は、欠損値にリセットされません。
   - 自動変数_N_および_ERROR_は、欠損値にリセットされません。

   RETAIN ステートメントの詳細については、“前のオブザベーションの値の使用” (207 ページ)を参照してください。

読み込み対象のレコードが他にあれば、プログラムが再実行されます。2 番目のオブザベーションが作成され、読み込み対象のレコードがなくなるまで処理が続行されます。その後で、データセットが閉じられ、次の DATA または PROC ステップに進みます。

DATA ステップの例

DATA ステップ

次の単純な DATA ステップでは、ヘルスクラブについて収集されたデータから SAS データセットが作成されます。前述のように、入力データには、各参加者の ID 番号、名前、チーム名、16 週間プログラムの開始時と終了時の体重が含まれています。

```
data weight_club; 1
   input IdNumber 1-4 Name $ 6-24 Team $ StartWeight EndWeight; 2
   Loss = StartWeight - EndWeight; 3

   datalines; 4
1023 David Shaw          red    189 165
1049 Amelia Serrano      yellow 145 124
1219 Alan Nance          red    210 192
1246 Ravi Sinha          yellow 194 177
1078 Ashley McKnight     red    127 118
1221 Jim Brown           yellow 220   .
1095 Susan Stewart       blue   135 127
1157 Rosa Gomez          green  155 141
1331 Jason Schock        blue   187 172
1067 Kanoko Nagasaka     green  135 122
1251 Richard Rose        blue   181 166
1333 Li-Hwa Lee          green  141 129
1192 Charlene Armstrong yellow 152 139
1352 Bette Long          green  156 137
1262 Yao Chen            blue   196 180
1087 Kim Sikorski        red    148 135
1124 Adrienne Fink       green  156 142
1197 Lynne Overby        red    138 125
1133 John VanMeter       blue   180 167
1036 Becky Redding       green  135 123
1057 Margie Vanhoy       yellow 146 132
1328 Hisashi Ito         red    155 142
1243 Deanna Hicks        blue   134 122
1177 Holly Choate        red    141 130
1259 Raoul Sanchez       green  189 172
1017 Jennifer Brooks     blue   138 127
1099 Asha Garg           yellow 148 132
1329 Larry Goss          yellow 188 174
; 4
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 DATA ステートメントでは、DATA ステップが開始され、作成するデータセットの名前が指定されます。

2 INPUT ステートメントでは、5 つの変数が作成され、入力バッファからの値の読み込み方法が示され、プログラムデータベクトル内の変数に値が割り当てられます。

3 割り当てステートメントでは、Loss という追加変数が作成され、DATA ステップが反復するたびに Loss の値が計算され、その値がプログラムデータベクトルに書き込まれます。

4 DATALINES ステートメントは、入力データの開始を示します。セミコロンは、入力データと DATA ステップの終了を示します。

注: DATALINES ステートメントを含まない DATA ステップは、RUN ステートメントで終了する必要があります。

プロセス

DATA ステップを実行しようとしてサブミットすると、DATA ステップが自動的にコンパイルされ、その後に実行されます。コンパイル時に、データセット WEIGHT_CLUB に対して入力バッファ、プログラムデータベクトル、およびディスクリプタ情報が作成されます。次の図に示すとおり、プログラムデータベクトルには、INPUT ステートメントで名前を指定した変数に加えて、変数 Loss が含まれています。すべての DATA ステップにおいて、_N_変数と_ERROR_変数の値が自動生成されます。自動変数_N_は、DATA ステップが反復された回数を表しています。自動変数_ERROR_は、バイナリスイッチのように機能する値です。DATA ステップ内にエラーがまったくない場合は 0 になり、エラーが 1 つでもある場合は 1 になります。これらの自動変数は、出力データセットには書き込まれません。

_N_および_ERROR_以外のすべての変数値は、欠損値に初期設定されます。数値欠損値はピリオドで表され、文字欠損値はブランクで表されることに注意してください。

図 3.8 *変数値を欠損値に初期設定*

Input Buffer

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
```

Program Data Vector

IdNumber	Name	Team	StartWeight	EndWeight	Loss
.			.	.	.

構文が正しいので、DATA ステップが実行されます。次の図に示すとおり、INPUT ステートメントを使用すると、生データの最初のレコードが入力バッファに読み込まれます。次に、INPUT ステートメントの命令に従って、入力バッファのデータ値が読み込まれ、その値がプログラムデータベクトル内の変数に割り当てられます。

図 3.9 *INPUT ステートメントによる値の変数への割り当て*

Input Buffer

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
1023 David Shaw          red    189 165
```

Program Data Vector

IdNumber	Name	Team	StartWeight	EndWeight	Loss
1023	David Shaw	red	189	165	.

INPUT ステートメントで列挙したすべての変数に値が割り当てられると、プログラム内の次のステートメントが実行されます。

```
Loss = StartWeight - EndWeight;
```

次の図に示すとおり、この割り当てステートメントでは、変数 Loss の値が計算され、その値がプログラムデータベクトルに書き込まれます。

図 3.10 *値の計算と変数 Loss への割り当て*

Input Buffer

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
1023 David Shaw         red    189 165
```

Program Data Vector

IdNumber	Name	Team	StartWeight	EndWeight	Loss
1023	David Shaw	red	189	165	24

DATA ステップの最後に達すると、プログラムで次が自動実行されます。

- データセットへの最初のオブザベーションの書き込み
- ループして DATA ステップの最初に戻り、次の反復を開始
- _N_自動変数を 1 ずつ増加
- _ERROR_自動変数を 0 にリセット
- 次の図に示すとおり、プログラムデータベクトル内の変数値を、_N_および _ERROR_以外は、欠損値に設定

図 3.11 *欠損値に設定*

Input Buffer

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
1023 David Shaw         red    189 165
```

Program Data Vector

IdNumber	Name	Team	StartWeight	EndWeight	Loss
.			.	.	.

実行が続けられます。INPUT ステートメントで、新たに読み込むレコードが検索されます。他にレコードがなければ、データセットが閉じられ、次の DATA または PROC ステップに進みます。

ただし、次に示すとおり、この例では他にもレコードが存在し、INPUT ステートメントで 2 番目のレコードが入力バッファに読み込まれます。

図 3.12 *2 番目のレコードを入力バッファに読み込み*

Input Buffer

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
1049 Amelia Serrano     yellow 145 124
```

Program Data Vector

IdNumber	Name	Team	StartWeight	EndWeight	Loss
.			.	.	.

次の図に示すとおり、プログラムデータベクトル内の変数に値が割り当てられ、変数 Loss の値が計算されました。その後、最初と同様に 2 番目のオブザベーションが作成されました。

図 3.13 *DATA ステップの 2 回目の反復結果*

Input Buffer

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
1049 Amelia Serrano     yellow 145 124
```

Program Data Vector

IdNumber	Name	Team	StartWeight	EndWeight	Loss
1049	Amelia Serrano	yellow	145	124	21

ファイルの終わりが検出されるまで、このプロセス全体が続けられます。DATA ステップは、読み込むレコードの数だけ反復されます。その後で、データセット WEIGHT_CLUB が閉じられ、次の DATA または PROC ステップの先頭が検索されます。

これで生データからの収集データが SAS データセットに変換されたので、SAS プロシジャで処理できます。

```
proc print data=weight_club;
   title 'Fitness Center Weight Club';
run;
```

PRINT プロシジャで生成された次の出力には、ここで作成したデータセットが表示されています。

図 3.14 WEIGHT_CLUB データセットの PROC PRINT 出力

Fitness Center Weight Club

Obs	IdNumber	Name	Team	StartWeight	EndWeight	Loss
1	1023	David Shaw	red	189	165	24
2	1049	Amelia Serrano	yellow	145	124	21
3	1219	Alan Nance	red	210	192	18
4	1246	Ravi Sinha	yellow	194	177	17
5	1078	Ashley McKnight	red	127	118	9
6	1221	Jim Brown	yellow	220	.	.
7	1095	Susan Stewart	blue	135	127	8
8	1157	Rosa Gomez	green	155	141	14
9	1331	Jason Schock	blue	187	172	15
10	1067	Kanoko Nagasaka	green	135	122	13
11	1251	Richard Rose	blue	181	166	15
12	1333	Li-Hwa Lee	green	141	129	12
13	1192	Charlene Armstrong	yellow	152	139	13
14	1352	Bette Long	green	156	137	19
15	1262	Yao Chen	blue	196	180	16
16	1087	Kim Sikorski	red	148	135	13
17	1124	Adrienne Fink	green	156	142	14
18	1197	Lynne Overby	red	138	125	13
19	1133	John VanMeter	blue	180	167	13
20	1036	Becky Redding	green	135	123	12
21	1057	Margie Vanhoy	yellow	146	132	14
22	1328	Hisashi Ito	red	155	142	13
23	1243	Deanna Hicks	blue	134	122	12
24	1177	Holly Choate	red	141	130	11
25	1259	Raoul Sanchez	green	189	172	17
26	1017	Jennifer Brooks	blue	138	127	11
27	1099	Asha Garg	yellow	148	132	16
28	1329	Larry Goss	yellow	188	174	14

SAS データセットの作成に必要な情報を提供する

SAS データセット作成の概要

生データから SAS データセットを作成できるように、生データを読み込むための特定の情報を SAS に提供します。データセットは、追加処理、データ分析、またはレポート作成に使用できます。DATA ステップで生データを処理するには、次の操作を実行する必要があります。

- INPUT ステートメントを使用して、データの読み込み方法を指示
- 変数を定義して、文字か数値かを示す
- 生データの場所を指定

データ読み込み方法の指示: 入力スタイル

SAS では、生データを SAS データセットに読む込むためのツールが多数用意されています。これらのツールには、3 つの基本的な入力スタイルに加えて、さまざまなフォーマット修飾子やポインタコントロールが含まれます。

リスト入力は、生データの各フィールドが少なくとも 1 つのスペースで区切られ、埋め込みスペースが含まれていない場合に使用されます。INPUT ステートメントでは、変数名のリストを含めるだけです。ただし、リスト入力では、データに多数の制限が加えられます。これらの制限については、4 章, “生データから作成する: 基本” (51 ページ)で詳しく説明します。次の例は、リスト入力を示しています。各データ値の間に少なくとも 1 つブランクがあることに注意してください。

```
data scores;
   input Name $ Test_1 Test_2 Test_3;
   datalines;
Bill 187 97 103
Carlos 156 76 74
Monique 99 102 129
;
```

カラム入力を使用すると、データが固定カラムに配置されていれば、同じデータを読み込むことができます。

```
data scores;
   input Name $ 1-7 Test_1 9-11 Test_2 13-15 Test_3 17-19;
   datalines;
Bill    187  97 103
Carlos  156  76  74
Monique  99 102 129
;
```

フォーマット入力を使用すると、INPUT ステートメントでデータを読み込むための特別な命令を与えることができます。たとえば、特殊記号を含む数値データを読み込むには、データを正しく読み込めるように、特別な命令を SAS に与える必要があります。入力形式と呼ばれるこれらの命令については、4 章, “生データから作成する: 基本” (51 ページ)で詳しく説明します。次の例のように、INPUT ステートメントでは、データ値の読み込みに使用する入力形式を指定できます。

```
data total_sales;
   input Date mmddyy10. +2 Amount comma5.;
   datalines;
09/05/2013  1,382
10/19/2013  1,235
11/30/2013  2,391
;
```

この例では、変数 Date の MMDDYY10.入力形式によって、生データを月、日、年として解釈し、スラッシュは無視することが示されます。また、変数 Amount の COMMA5.入力形式によって、生データを数値として解釈し、カンマは無視することが示されます。+2 は、次項目の検索場所を示すポインタコントロールです。ポインタコントロールの詳細については、4 章, “生データから作成する: 基本” (51 ページ)を参照してください。

また、データレコードでの値の配置によって必要とされるとおりに、複数の入力スタイルを使用することもできます。4 章, “生データから作成する: 基本” (51 ページ)で、入力スタイル(規則と制限事項を含む)および追加のデータ読み込みツールについて詳しく説明します。

年が 2 桁と 4 桁の日付値の読み込み

前述の例の生データでは、日付における年の値は 4 桁でした。

```
09/05/2013
10/19/2013
11/30/2013
```

ただし、SAS では年が 2 桁の値も読み込めます(例: 09/05/99)。この例では、変数 Date に対して MMDDYY8.入力形式を使用します。

ここで、2 桁の年がどの世紀に属すのか判断する方法を説明します。SAS では、YEARCUTOFF= SAS システムオプションの値が使用されます。SAS のバージョン 7 以降では、YEARCUTOFF=オプションのデフォルト値は 1926 です。したがって、00 から 19 までの 2 桁の年は 21 世紀、すなわち 2000 年から 2019 年までだと考えられます。20 から 99 までの 2 桁の年は 20 世紀、すなわち 1920 年から 1999 年までと考えられます。

注: YEARCUTOFF=オプションとデフォルト設定はサイトによって異なる場合があります。

混乱を避けるために、可能であれば生データでは 4 桁の年の値を使用してください。詳細については、*SAS Language Reference: Concepts* の日付、時間および間隔のセクションを参照してください。

SAS での変数定義

ここまでは、INPUT ステートメントによる生データ行の読み込みに関する命令について説明してきました。INPUT ステートメントでは、データ読み込み命令が与えられると同時に、生データから生成するデータセットの変数が定義されます。INPUT ステートメントで、変数属性のデフォルト値を想定すると、非常に作業の役に立ちます。変数定義と変数への属性割り当てを行うための他のステートメントについてもこのドキュメントで後述します。このセクションと4 章, “生データから作成する: 基本” (51 ページ)では、INPUT ステートメントの用法について重点的に説明します。

SAS 変数では次の属性を使用できます。

- 名前
- 種類
- 長さ
- 入力形式
- 出力形式
- ラベル
- オブザベーションの位置
- インデックスの種類

変数属性の詳細については、*SAS Language Reference: Concepts* の SAS 変数に関するセクションを参照してください。

INPUT ステートメントでは、各変数名を指定する必要があります。また、入力形式を指定しなければ、種類は数値、長さは 8 バイトと想定されます。次の INPUT ステートメン

トでは、4 つの数値変数が作成され、それぞれの長さは 8 バイトになります。ユーザーは種類も長さも指定する必要はありません。

```
input IdNumber Test_1 Test_2 Test_3;
```

次の表に、この情報をまとめています。

変数名	種類	長さ
IdNumber	数値	8
Test_1	数値	8
Test_2	数値	8
Test_3	数値	8

数値変数の値には、数字しか含められません。アルファベットや特殊文字を含む値を格納するには、文字変数を作成する必要があります。INPUT ステートメント内の変数名の後にドル記号($)を付けると、文字変数が作成されます。文字変数のデフォルトの長さは 8 バイトです。次のステートメントでは、1 つの文字変数と 4 つの数値変数を含むデータセットが作成されます。変数のデフォルトの長さはすべて 8 バイトです。

```
input IdNumber Name $ Test_1 Test_2 Test_3;
```

次の表に、この情報をまとめています。

変数名	種類	長さ
IdNumber	数値	8
Name	文字	8
Test_1	数値	8
Test_2	数値	8
Test_3	数値	8

INPUT ステートメントでは、変数の種類の指定に加えて、文字変数の長さも指定できます。文字変数の最大長は 32,767 バイトです。INPUT ステートメントで文字変数の長さを指定するには、入力形式を指定するか、カラム番号を使用する必要があります。たとえば、INPUT ステートメントで変数名の後に入力形式$20.かまたは 1-20 のようなカラム指定を付けると、20 バイト長の文字変数が作成されます。

数値変数の長さは、INPUT ステートメントの入力形式やカラム指定の影響を受けないことに注意してください。数値変数と長さの詳細については、*SAS Language Reference: Concepts* を参照してください。

出力形式とラベルという他の 2 つの変数属性は、印刷時や表示時の変数値や変数名の表示方法に影響します。これらの属性は、後述する別のステートメントを使用して割り当てられます。

データの場所指定

データの場所

SAS データセットを作成するには、4 つのうちのいずれかの場所からデータを読み込みます。

- データ(ジョブ)ストリームの生データ。DATALINES ステートメントの後に指定
- INFILE ステートメントで指定するファイル内の生データ
- 既存の SAS データセットからのデータ
- データベース管理システム(DBMS)ファイル内のデータ

ジョブストリームの生データ

DATA ステップを構成するプログラミングステートメントを使用して、ジョブストリームに直接データを配置できます。DATALINES ステートメントで、生データが後に続くことが示されます。データ最終行の後のセミコロンは、データの終わりを示します。DATALINES ステートメントとデータ行は、DATA ステップのステートメントの中で最後に指定する必要があります。

```
data weight_club;
   input IdNumber 1-4 Name $ 6-24 Team $ StartWeight EndWeight;
   Loss = StartWeight - EndWeight;
   datalines;
1023 David Shaw          red    189 165
1049 Amelia Serrano      yellow 145 124
1219 Alan Nance          red    210 192
1246 Ravi Sinha          yellow 194 177
1078 Ashley McKnight     red    127 118
;
```

外部ファイル内のデータ

生データがすでにファイルに格納されている場合は、そのファイルをデータストリームに組み込む必要はありません。INFILE ステートメントを使用して、生データを含むファイルを指定します。INFILE、FILE および FILENAME ステートメントの詳細については、“SAS ジョブでの外部ファイルの使用” (46 ページ)を参照してください。次に示すコード内のステートメントは、同じ例を示していますが、ここでは、生データが外部ファイルに格納されています。

```
data weight_club;
   infile 'your-input-file';
   input IdNumber $ 1-4 Name $ 6-23 StartWeight 24-26
         EndWeight 28-30;
   Loss=StartWeight-EndWeight;
run;
```

SAS データセット内のデータ

すでに SAS データセットに格納されているデータを、新しいデータセットへの入力として使用することもできます。既存の SAS データセットからデータを読み込むには、次のステートメントのいずれかで既存のデータセット名を指定する必要があります。

- SET ステートメント
- MERGE ステートメント

- MODIFY ステートメント
- UPDATE ステートメント

たとえば、次に示すステートメントでは、RED という名前の SAS データセットが新規作成され、変数 LossPercent が追加されます。

```
data red;
   set weight_club;
   LossPercent = Loss / StartWeight * 100;
run;
```

SET ステートメントでは、入力データがすでに SAS データセット構造であることが示され、読み込み対象の SAS データセットの名前が指定されます。この例の SET ステートメントでは、WORK ライブラリの WEIGHT_CLUB データセットを読み込むように指定されています。

DBMS ファイル内のデータ

別のベンダのデータベース管理システム(DBMS)ファイルに格納されたデータがある場合は、SAS/ACCESS を使用して、そのデータを SAS データセットに取り込むことができます。SAS/ACCESS を使用すると、DBMS ファイルを含むライブラリにライブラリ参照名を割り当てられます。この例では、ライブラリ参照名が宣言され、Oracle データを含むライブラリが示されます。SAS で、Oracle ファイルから SAS データセットにデータが読み込まれます。

```
libname dblib oracle user=scott password=tiger path='hrdept_002';
data employees;
   set dblib.employees;
run;
```

SAS/ACCESS を使用した DMBS ファイルへのアクセスの詳細については、*SAS/ACCESS for Relational Databases: Reference* を参照してください。

SAS ジョブでの外部ファイルの使用

SAS プログラムでは、多くの場合、SAS データセットではないファイルから生データを読み込んだり、SAS データセットではないファイルにデータやレポートを書き込む必要があります。SAS データセットではないファイルを SAS プログラムで使用するには、ファイルのある場所を示す必要があります。次の操作を行えます。

- ファイルを使用する、INFILE や FILE などの SAS ステートメントで、ファイルを直接指定します。
- FILENAME ステートメントを使用してファイルのファイル参照名を設定し、そのファイル参照名を INFILE、FILE などの SAS ステートメントで使用します。
- 動作環境コマンドを使用してファイル参照名を設定し、そのファイル参照名を INFILE、FILE などの SAS ステートメントで使用します。

最初の 2 つの方法についてはここで説明します。3 番目の方法は、使用している動作環境によって異なります。

詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

外部ファイルの直接指定

外部ファイルを参照する最も簡単な方法は、ファイルの参照が必要な INFILE、FILE などの SAS ステートメントでファイル名を使用することです。たとえば、使用している動作環境のファイルに格納されている生データを、SAS DATA ステップを使用して読み

込む場合は、INFILE ステートメントにファイル名を加えて、生データのある場所を示します。

```
data temp;
   infile 'your-input-file';
   input IdNumber $ 1-4 Name $ 6-23 StartWeight 24-26
         EndWeight 28-30;
run;
```

さまざまな動作環境におけるこの例の INFILE ステートメントは、次のようになります。

表 3.1 *さまざまな動作環境の INFILE ステートメントの例*

動作環境	INFILE ステートメントの例
z/OS	**infile 'fitness.weight.rawdata(club1)';**
UNIX	**infile '/usr/local/fitness/club1.dat';**
Windows	**infile 'c:\fitness\club1.dat';**

詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

ファイル参照名を使用した外部ファイルの参照

外部ファイル参照の代替方法は、FILENAME ステートメントを使用してファイルのファイル参照名を設定することです。ファイル参照名は、外部ファイル参照の省略表現方法として機能します。ファイル参照名は、FILE ステートメントや INFILE ステートメントなど、ファイルを参照する後の SAS ステートメントで使用します。この方法の利点は、プログラムに同じ外部ファイルへの参照が多数含まれている場合、その外部ファイルの変更時に、ファイル参照箇所をすべて変更しなくても、1 か所変更するだけですむことです。

FILENAME ステートメントの形式は、次のとおりです。

FILENAME *fileref* '*your-input-or-output-file*';

fileref には有効な SAS 名を指定する必要があります。したがって、次の規則を守ってください。

- 文字またはアンダースコアで始める
- 文字、数字またはアンダースコアのみを使用
- 最大長は 8 文字

一部の動作環境では追加制限事項が適用される場合もあります。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

たとえば、使用している動作環境のファイルに格納されている生データを参照するには、まず FILENAME ステートメントでファイルの名前とそのファイル参照名を指定しておいてから、INFILE ステートメントで同じファイル参照名を使用してファイルを参照します。

```
filename fitclub 'your-input-file';

data temp;
   infile fitclub;
```

```
   input IdNumber $ 1-4 Name $ 6-23 StartWeight 24-26 EndWeight 28-30;
run;
```

この例では、INFILE ステートメントはすべての動作環境で同一のままです。ただし、FILENAME ステートメントは、異なる動作環境では記述が異なる場合があります。

次の表に、各動作環境での FILENAME ステートメントの使用例を示します。

表 3.2 *さまざまな動作環境の FILENAME ステートメントの例*

動作環境	FILENAME ステートメントの例
z/OS	**filename fitclub 'fitness.weight.rawdata(club1)';**
UNIX	**filename fitclub '/usr/local/fitness/ club1.dat';**
Windows	**filename fitclub 'c:\fitness\club1.dat';**

同じディレクトリ、区分データセット(PDS)または MACLIB のファイルやメンバを複数使用する必要がある場合は、FILENAME ステートメントを使用して、ディレクトリ、PDS または MACLIB の名前を識別するファイル参照名を作成できます。その後、次の例のように、INFILE ステートメントでそのファイル参照名を使用し、かっこで囲んだファイル、PDS メンバまたは MACLIB メンバの名前をファイル参照名のすぐ後に付け加えます。

```
filename fitclub 'directory-or-PDS-or-MACLIB';

data temp;
   infile fitclub(club1);
   input IdNumber $ 1-4 Name $ 6-23 StartWeight 24-26 EndWeight 28-30;
run;

data temp2;
   infile fitclub(club2);
   input IdNumber $ 1-4 Name $ 6-23 StartWeight 24-26 EndWeight 28-30;
run;
```

この場合、INFILE ステートメントはすべての動作環境で同一のままです。ただし、次の表に示すとおり、FILENAME ステートメントは、異なる動作環境では記述が異なる場合があります。

次の表に、各動作環境での FILENAME ステートメントの使用例を示します。

表 3.3 *さまざまな動作環境でのディレクトリ、PDS および MACLIB の参照*

動作環境	FILENAME ステートメントの例
z/OS	**filename fitclub 'fitness.weight.rawdata';***
UNIX	**filename fitclub '/usr/local/fitness';**
Windows	**filename fitclub 'c:\fitness';**

* z/OS では、外部ファイルは PDS または UFS ファイルにしてください。

要約

ステートメント

DATA *<libref.>SAS-data-set;*
SAS データセットの作成開始が指示されます。ライブラリ参照名を省略すると、一時 SAS データセットが作成されます。(内部処理用にライブラリ参照名 WORK が割り当てられます。)すでに定義されているライブラリ参照名を名前の第 1 レベルとして指定すると、そのライブラリ参照名で参照されるライブラリにデータセットが永久的に格納されます。DATA ステートメントから始まり、RUN ステートメント、別の DATA ステートメントまたは PROC ステートメントで終わる SAS プログラムやプログラムの一部は、DATA ステップと呼ばれます。

FILENAME *fileref 'your-input-or-output-file'*;
ファイル参照名と外部ファイルを関連付けます。外部ファイル名は引用符で囲みます。

INFILE *fileref* | *'your-input-file'*;
INPUT ステートメントで読み込む外部ファイルを指定します。FILENAME ステートメントや適切な動作環境コマンドを使用して割り当てたファイル参照名か、または外部ファイルの実際の名前を指定します。

INPUT *variable <$>*;
リスト入力によって生データを読み込みます。2 つのデータ値の間にはブランクを少なくとも 1 つ入れる必要があります。$は文字変数を示します。

INPUT *variable <$>column-range*;
列状に配置された生データを読み込みます。$は文字変数を示します。

INPUT *variable informat*;
フォーマット入力によって、生データを読み込みます。入力形式によって、データを読み込むための特別な命令が与えられます。

LIBNAME *libref 'your-SAS-data-library'*;
ライブラリ参照名と SAS ライブラリを関連付けます。ライブラリ名は引用符で囲みます。SAS では、2 レベルの SAS データセット名のライブラリ参照名と、LIBNAME ステートメントでそのライブラリ参照名に関連付けられたライブラリとをマッチングして、永久 SAS データセットの場所を突き止めます。SAS ライブラリの作成規則は、使用している動作環境によって異なります。

詳細情報

ATTRIB ステートメント
ATTRIB ステートメントによる属性の変数への割り当ての詳細については、“ATTRIB Statement” (*SAS Statements: Reference*)を参照してください。

DBMS アクセス
このドキュメントでは、SAS を使用した生データファイルおよび SAS データセットの読み込みや、SAS データセットへの書き込みについて説明します。ただし、複数の種類のデータベース管理システム(DBMS)ファイルに格納された情報の SAS での読み書きについては、SAS/ACCESS に関する SAS ドキュメントで詳細に説明します。

入力形式
入力形式での日付の使用法の詳細については、15 章, “SAS System での日付の操作” (223 ページ)を参照してください。

変数の長さ
変数の長さが変数に格納可能な値に与える影響の詳細については、8 章, “数値変数の操作” (119 ページ)および 9 章, “文字変数の操作” (131 ページ)を参照してください。

LINESIZE= *option*
INPUT ステートメントで LINESIZE=オプションを使用して、INPUT ステートメントで読み込まれる各データ行の長さを制限する方法については、“INPUT Statement” (*SAS Statements: Reference*)を参照してください。

MERGE、MODIFY または UPDATE ステートメント
SET ステートメントに加えて、MERGE、MODIFY または UPDATE ステートメントを使用して、SAS データセットを読み込むことができます。詳細については、19 章, “SAS データセットのマージ” (287 ページ)および 20 章, “SAS データセットの更新” (315 ページ)を参照してください。

SET ステートメント
SET ステートメントの詳細については、6 章, “SAS データセットから作成する” (91 ページ)を参照してください。

USER= *SAS system option*
USER= SAS システムオプションを指定すると、1 レベル名を使用して永久 SAS ファイルを示すことができます。(USER=WORK を指定すると、SAS では、1 レベル名で参照されるファイルが、一時作業ファイルを示していると想定されます。)詳細については、*SAS System Options: Reference* を参照してください。

4 章

生データから作成する: 基本

生データについて **52**
目的 52
前提条件 52

生データの構造の検証: 注意が必要な要素 **52**

整列されていないデータの読み込み **53**
リスト入力について 53
プログラム: 基本のリスト入力 53
プログラム: ブランクではなく文字でデータを区切る場合 54
プログラム: 連続した区切り文字が欠損値を示す場合 55
リスト入力: 留意点 56

列状に配置されたデータの読み込み **57**
カラム入力について 57
プログラム: 列状に配置されたデータの読み込み 57
カラム入力が単純なリスト入力に勝る点について 58
埋め込みブランクの読み込みと長い変数の作成 58
プログラム:データレコード読み込み時のフィールドのスキップ 59
カラム入力: 留意点 60

特別な処理が必要なデータの読み込み **60**
フォーマット入力について 60
プログラム: 特別な処理が必要なデータの読み込み 61
ポインタ位置の制御方法について 62
フォーマット入力: 留意点 63

整列されていないデータを、より柔軟に読み込む **63**
より柔軟なリスト入力方法について 63
長い変数の作成と特殊文字を含む数値データの読み込み 64
埋め込みブランクを含む文字データの読み込み 64

複数の入力スタイルを使用する **65**
複数の入力スタイルの使用例 65
入力スタイルのポインタ位置への影響について 66

要約 **68**
ステートメント 68
カラムポインタコントロール 69

詳細情報 **69**

生データについて

目的

生データから SAS データセットを作成するには、先にデータレコードを検証して、読み込むデータ値の配置を決定しておく必要があります。次に、INPUT ステートメントで使用可能な入力読み込みスタイルを検討します。SAS では、基本的な入力スタイルが 3 つ用意されています。

- リスト
- カラム
- フォーマット

これらのスタイルは、個別に使用したり、互いに組み合わせたり、各種のラインホールド指定子、ラインポインタコントロールおよびカラムポインタコントロールと併用したりします。このセクションでは、生データを SAS データセットに変換するためのさまざまな INPUT ステートメント使用方法を説明します。

DATA ステップでデータを直接入力するか、既存の生データファイルを使用します。データをコンピュータで読み込める場合は、SAS によるデータ読み込みツールの使用方法を学習する必要があります。データが未入力の場合は、最も容易にデータを入力できる入力スタイルを選択します。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 1 章, “SAS System について” (3 ページ)
- 3 章, “DATA ステップ処理について” (27 ページ)

生データの構造の検証: 注意が必要な要素

適切な入力スタイルを選択する前に、読み込む生データの構造を検証します。次の要素のいくつかに注意してください。

- 入力レコードのデータ配置。たとえば、データフィールドを列状に配置するか整列させないか。区切り文字をブランクにするか他の文字にするか。
- 文字値に埋め込みブランクを含めるかどうか
- 数値にカンマなどの非数値文字を含めるかどうか
- データに時間や日付の値を含めるかどうか
- 各入力レコードに、2 つ以上のオブザベーションのデータを含めるかどうか
- 1 つのオブザベーションのデータを複数の入力レコードに分散させるかどうか

整列されていないデータの読み込み

リスト入力について

最も単純な形式の INPUT ステートメントでは、リスト入力が使用されます。リスト入力を使用すると、区切り文字(デフォルトではブランク)で区切られたデータ値を読み込むことができます。リスト入力を使用すると、ブランクが検出されるまで、データ値が読み込まれます。値が終了したと見なされると、そのデータはプログラムデータベクトル内の適切な変数に割り当てられます。再びブランク以外の文字に達するまで、レコードのスキャンが続けられます。ブランクかまたは入力レコードの終わりが検出されるまで、データ値が読み込まれます。

プログラム: 基本のリスト入力

このプログラムでは、3 章, “DATA ステップ処理について” (27 ページ) のヘルスクラブデータを使用して、INPUT ステートメントでリスト入力を使用する DATA ステップについて説明します。

```
data club1;
   input IdNumber Name $ Team $ StartWeight EndWeight; 1
   datalines; 2
1023 David red 189 165 3
1049 Amelia yellow 145 124
1219 Alan red 210 192
1246 Ravi yellow 194 177
1078 Ashley red 127 118
1221 Jim yellow 220 . 3
; 2

proc print data=club1;
   title 'Weight of Club Members';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 INPUT ステートメントの変数名は、生データレコード内のフィールドとまったく同じ順序で指定されます。

2 DATALINES ステートメントは、データ行の開始を示します。データ行の後のセミコロンは、データ行の終了と DATA ステップの終了を示します。

3 生データレコードのデータ値同士は、少なくとも 1 つのブランクで区切られます。EndWeight 値の最後のレコードには欠損値が含まれており、ピリオドで表されています。

次の出力は、処理結果のデータセットを示しています。DATA ステップの後の PROC PRINT ステートメントで、リストが作成されます。

図 4.1 リスト入力によって作成されたデータセット

Weight of Club Members

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	124
3	1219	Alan	red	210	192
4	1246	Ravi	yellow	194	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.

プログラム: ブランクではなく文字でデータを区切る場合

このプログラムでもヘルスクラブのデータが使用されますが、ここでは、データが、デフォルト区切り文字のブランクではなくカンマで区切られていることに注意してください。

```
data club1;
   infile datalines dlm=',';1

   input IdNumber Name $ Team $ StartWeight EndWeight;
   datalines;
1023,David,red,189,165 2
1049,Amelia,yellow,145,124
1219,Alan,red,210,192
1246,Ravi,yellow,194,177
1078,Ashley,red,127,118
1221,Jim,yellow,220,.
;
proc print data=club1;
   title 'Weight of Club Members';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 リスト入力では、デフォルトで、入力レコードがスキャンされ、各データ値を区切るブランクが検索されます。DLM=オプションを使用すると、リスト入力で、文字(この場合はカンマ)を区切り文字として認識できます。

2 これらのデータ値は、ブランクのかわりにカンマで区切られています。

注: この例では、DLM=オプションが必須です。このオプションは INFILE ステートメントでのみ使用可能です。通常、このステートメントは、入力データが外部ファイル内に存在する場合のみ使用されます。ただし、ジョブストリームからデータレコードを読む込む場合は、DATALINES を指定すると、INFILE ステートメントオプションを活用できます。

図 4.2 カンマで区切られたデータの読み込み

Weight of Club Members

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	124
3	1219	Alan	red	210	192
4	1246	Ravi	yellow	194	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.

プログラム: 連続した区切り文字が欠損値を示す場合

このプログラムでもヘルスクラブのデータが使用されますここでは、データがコロン(:)で区切られ、Alan に対して、2 つの連続した区切り文字で示された欠損値が存在することに注意してください。

```
data club1;
   infile datalines dsd dlm=':';
   input IdNumber Name $ Team $ StartWeight EndWeight;
   datalines;
1023:David:red:189:165
1049:Amelia:yellow:145:124
1219:Alan::210:192
1246:Ravi:yellow:194:177
1078:Ashley:red:127:118
1221:Jim:yellow:220:.
;
proc print data=club1;
   title 'Weight of Club Members';
run;
```

この例では、DSD オプションが必須です。このオプションは INFILE ステートメントでのみ使用可能です。前の例と同様に、DATALINES の指定によって、INFILE ステートメントを DSD および DLM=オプションと一緒に指定できるようになります。DSD オプションでは、2 個の連続する区切り文字が欠損値として扱われます。(また、DSD オプションでは、デフォルト区切り文字がカンマに設定され、引用文字列内に出現する区切り文字は無視されます。)

この例で DSD オプションを指定しなかった場合、データ値が不一致になり、残りのデータが正しく読み込まれません。次のリストは、DSD オプションが省略された場合、データがどのように読み込まれるかを示しています。Alan の名前の後の値が間違っており、Ravi のオブザベーションが省略されていることに注意してください。Ravi の IdNumber が Alan の EndWeight 値として読み込まれ、そのレコードの残り部分は無視されています。

図 4.3 DSD オプションの間違った省略

INCORRECT: Weight of Club Members

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	124
3	1219	Alan	210	192	1246
4	1078	Ashley	red	127	118
5	1221	Jim	yellow	220	.

INFILE ステートメントに DSD オプションを含めると、連続する区切り文字は欠損値を示し、データが正しく読み込まれます。次のリストは、結果のデータセットを示しています。ここでは、Alan の欠損 Team 値がブランク値として表示され、残りのデータセットが予想どおりに読み込まれていることがわかります。

図 4.4 DSD オプションによって正しく読み込まれた欠損値

CORRECT: Weight of Club Members

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	124
3	1219	Alan		210	192
4	1246	Ravi	yellow	194	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.

リスト入力: 留意点

リスト入力を使用する場合の留意点は次のとおりです。

- 各フィールドが少なくとも 1 つのブランクまたは区切り文字で区切られている場合、リスト入力を使用します。スペース以外の区切り文字を指定するには、INFILE ステートメントで DLM=オプションを使用します。リスト入力の場合、INFILE ステートメントで DATALINES も指定する必要があります。
- 各フィールドは、生データレコードでの出現順序で指定します。
- 欠損値は、ピリオドなどのプレースホルダで表します。

  デフォルトの動作では、ブランクフィールドが存在すると、変数の名前と値が一致しなくなります。ただし、DSD オプションをカンマ区切り文字と一緒に使用した場合、複数のカンマは欠損値を示します。カンマ以外の区切り文字を DSD オプションと一緒に指定するには、DLM=オプションを使用します。

  注: INFILE ステートメントで DSD オプションと DLM=オプションを指定します。リスト入力の場合、INFILE ステートメントで DATALINES も指定する必要があります。

- INFILE ステートメントで DSD オプションと DLM=オプションも指定していなければ、文字値に埋め込みブランクを含めることはできません。
- 文字変数のデフォルト長は 8 バイトです。SAS では、プログラムデータベクトルに値を書き込む際に、これよりも長い値は切り捨てられます。(リスト入力で 9 文字以上の文字変数を読む込むには、LENGTH ステートメントを使用します。“十分な容量の変数の記憶域を定義する” (116 ページ)を参照してください。)
- 標準の文字形式または数値文字形式のデータでなければなりません(すなわち、入力形式なしで読み込み可能)。

注: リスト入力を使用すると、INPUT ステートメントの指定が最も少なくてすみます。ただし、データに制限があるため、他の入力スタイルによるデータ読み込み方法の学習が必要になる場合もあります。たとえば、次のセクションで説明するカラム入力では、制限が弱くなります。このセクションでは、単純なリスト入力のみを説明しました。修飾リスト入力について学習するには、“より柔軟なリスト入力方法について” (63 ページ) を参照してください。

列状に配置されたデータの読み込み

カラム入力について

カラム入力では、データ値が各データレコード内の同じフィールドを占有します。INPUT ステートメントでカラム入力を使用する場合は、変数名を列挙し、対応するデータフィールドの場所を識別するカラム位置を指定します。生データが固定カラムに配置されている場合は、カラム入力を使用できるので、読み込みに入力形式を使用する必要はありません。

プログラム: 列状に配置されたデータの読み込み

次のプログラムでもヘルスクラブのデータが使用されますが、ここでは欠損しているデータ値が 2 つ増えています。データが列状に配置されているので、カラム入力によってデータが読み込まれます。

```
data club1;
   input IdNumber 1-4 Name $ 6-11 Team $ 13-18 StartWeight 20-22
         EndWeight 24-26;
   datalines;
1023 David  red    189 165
1049 Amelia yellow 145
1219 Alan   red    210 192
1246 Ravi   yellow     177
1078 Ashley red    127 118
1221 Jim    yellow 220
;

proc print data=club1;
   title 'Weight Club Members';
run;
```

各変数名の後の指定は、変数値の位置を示す開始カラムと終了カラムを表しています。カラム入力では、欠損値をピリオドなどのプレースホルダで示す必要がないことに注意してください。

次の出力は、処理結果のデータセットを示しています。データセットに 3 回出現する欠損値は、ピリオドで表されています。

図 4.5 カラム入力によって作成されたデータセット

Weight Club Members

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David	red	189	165
2	1049	Amelia	yellow	145	.
3	1219	Alan	red	210	192
4	1246	Ravi	yellow	.	177
5	1078	Ashley	red	127	118
6	1221	Jim	yellow	220	.

カラム入力が単純なリスト入力に勝る点について

カラム入力使用の利点をいくつか次に示します。

- カラム入力では、文字変数に埋め込みブランクを含められます。
- カラム入力では、8 バイトよりも長い変数の作成も可能です。前述の例では、データセット CLUB1 の変数 Name には、会員の名前しか含まれていません。カラム入力を使用すると、姓と名前を 1 つの値として読み込むことができます。入力スタイルにこのような違いが生じる理由は 2 つあります。
  - カラム入力では、ユーザーの指定したカラムを使用して文字変数の長さを決定しているため。
  - ブランクに達するまでデータが読み込まれるリスト入力とは違って、カラム入力では、最後の指定カラムに達するまでデータが読み込まれるため。
- カラム入力を使用すると、生データレコードの読み込み時に一部のデータフィールドをスキップできます。また、データフィールドを任意の順序で読み込んだり、フィールド全体やフィールドの一部を再読み込みしたりできます。

埋め込みブランクの読み込みと長い変数の作成

この DATA ステップでは、カラム入力によって、CLUB2 というデータセットが新規作成されます。このプログラムでも、ヘルスクラブの体重データが使用されます。ただし、データが変更されて、会員の姓と名前が含まれています。ここでは、各レコード(生データ)の 2 番目のデータフィールドに埋め込みブランクが含まれ、そのフィールド長は 18 バイトです。

```
data club2;
   input IdNumber 1-4 Name $ 6-23 Team $ 25-30 StartWeight 32-34
         EndWeight 36-38;
   datalines;
1023 David Shaw         red    189 165
1049 Amelia Serrano     yellow 145 124
1219 Alan Nance         red    210 192
1246 Ravi Sinha         yellow 194 177
```

```
1078 Ashley McKnight     red    127 118
1221 Jim Brown           yellow 220
;

proc print data=club2;
   title 'Weight Club Members';
run;
```

次の出力は、処理結果のデータセットを示しています。

図 4.6 カラム入力によって作成されたデータセット(埋め込みブランク)

Weight Club Members

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David Shaw	red	189	165
2	1049	Amelia Serrano	yellow	145	124
3	1219	Alan Nance	red	210	192
4	1246	Ravi Sinha	yellow	194	177
5	1078	Ashley McKnight	red	127	118
6	1221	Jim Brown	yellow	220	.

プログラム: データレコード読み込み時のフィールドのスキップ

カラム入力を使用すると、フィールドをスキップしたり、フィールドを任意の順序で読み込んだりすることもできます。この例では、カラム入力によって同じヘルスクラブデータが読み込まれますが、変数 Team の値が最初に読み込まれ、変数 IdNumber は完全に省略されます。

カラム入力を使用すると、値の一部の読み込みまたは再読み込みができます。たとえば、チーム名がそれぞれ異なる文字で始まっているので、このプログラムでは、記憶域を節約するために、チーム名を含むフィールドの最初の文字のみが読み込まれます。INPUT ステートメントを次に示します。

```
data club2;
   input Team $ 25 Name $ 6-23 StartWeight 32-34 EndWeight 36-38;
   datalines;
1023 David Shaw          red    189 165
1049 Amelia Serrano      yellow 145 124
1219 Alan Nance          red    210 192
1246 Ravi Sinha          yellow 194 177
1078 Ashley McKnight     red    127 118
1221 Jim Brown           yellow 220
;

proc print data=club2;
   title 'Weight Club Members';
run;
```

次の出力は、処理結果のデータセットを示しています。このデータセットでは ID 番号を含む変数がなくなっています。そのかわり、新しいデータセットでは Team が最初の変数になり、そこに Team 値を表す 1 文字のみが含まれます。

図 4.7 カラム入力によって作成されたデータセット(フィールドのスキップ)

Weight Club Members

Obs	Team	Name	StartWeight	EndWeight
1	r	David Shaw	189	165
2	y	Amelia Serrano	145	124
3	r	Alan Nance	210	192
4	y	Ravi Sinha	194	177
5	r	Ashley McKnight	127	118
6	y	Jim Brown	220	.

カラム入力: 留意点

カラム入力を使用する場合は、次の規則に留意してください。

- 文字変数の最大長は 32,767 バイト(32KB)で、デフォルト長の 8 バイトに制限されません。
- 文字変数に埋め込みブランクを含めることができます。
- フィールドは任意の順序で読み込むことができます。
- プレースホルダで欠損データ値を示す必要はありません。ブランクのフィールドは欠損値として読み込まれるので、他の値が誤って読み込まれることはありません。
- データレコード内のデータの一部をスキップできます。
- フィールドの全部または一部を再読み込みできます。
- 標準の文字および数値のデータのみ読み込むことができます。入力形式は無視されます。

特別な処理が必要なデータの読み込み

フォーマット入力について

INPUT ステートメントでは、データを正しく読み込むために特別な処理が必要になる場合があります。たとえば、SAS では、バイナリ、パック 10 進、日時などの特殊な形式の数値データを読み込めます。また、SAS では、カンマや通貨記号などの特殊文字を含む数値も読み込めます。このような場合には、フォーマット入力を使用します。フォーマット入力は、カラム入力機能と、非標準の数値または文字値の読み込み機能を兼ね備えています。次のデータは、フォーマット入力を示しています。

- 1,262

- $55.64
- 02JAN2013

プログラム: 特別な処理が必要なデータの読み込み

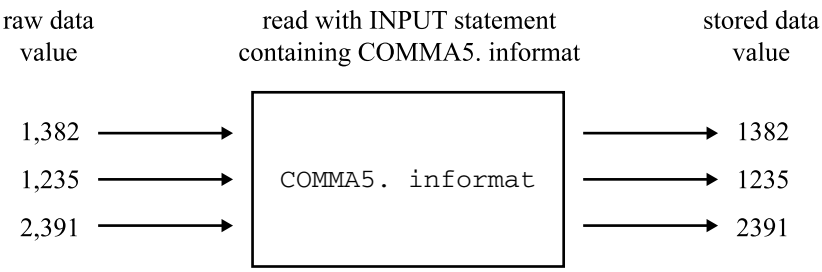
このプログラムのデータには、カンマの入った数値が含まれます。カンマは数値変数では無効な文字です。

```
data january_sales;
   input Item $ 1-16 Amount comma5.;
   datalines;
trucks          1,382
vans            1,235
sedans          2,391
;

proc print data=january_sales;
   title 'January Sales in Thousands';
run;
```

INPUT ステートメントでは、入力形式によって提供される追加処理がなければ、変数 Amount の値を有効な数値として読み込むことができません。入力形式 COMMA5.を使用すると、INPUT ステートメントで、このデータを有効な数値として読み込み、格納できます。

次の図に示すとおり、入力形式 COMMA5.によって、5 文字のデータ(カンマはデータ長の一部としてカウント)を読み込むようプログラムに命令が出され、データからカンマが削除され、処理結果の数値がプログラムデータベクトルに書き込まれます。入力形式名は必ずピリオド(.)で終わることに注意してください。

図 4.8 *入力形式による値の読み込み*



次の図に示すとおり、データ値は生データレコード内とまったく同じ形式で入力バッファに読み込まれます。ただし、特殊文字のない有効な数値としてプログラムデータベクトルに(そしてその後、オブザベーションとしてデータセットに)書き込まれます。

図 4.9 入力値と変数値の比較

Input Buffer

```
----+----1----+----2----+----3
trucks          1,382
```

Program Data Vector

Item	Amount
trucks	1382

次の出力は、処理結果のデータセットを示しています。Amount の値には数字しか含められません。カンマが削除されていることに注意してください。

図 4.10 カラム入力とフォーマット入力によって作成されたデータセット

January Sales in Thousands

Obs	Item	Amount
1	trucks	1382
2	vans	1235
3	sedans	2391

レポートでは、読みやすさを向上させるため数値にカンマを入れる場合があります。入力形式が値の読み込み方法について命令を与え、それと同様に、出力形式は出力の変数値に文字を追加するように命令を与えます。例については、“データセットを作成せずに出力を書き込む” (584 ページ)を参照してください。

ポインタ位置の制御方法について

INPUT ステートメントでは、データ値の読み込み時に、入力ポインタによって入力バッファのデータ位置が追跡されます。カラムポインタコントロールでは、さらにポインタ移動も制御できます。これはフォーマット入力では特に役立ちます。カンマポインタコントロールでは、次の値を読み込む前に、ポインタをどこまで進めるかが示されます。この例では、カラム入力とフォーマット入力を組み合わせて、データ行が読み込まれます。

```
data january_sales;
   input Item $ 1-16 Amount comma5.;
   datalines;
trucks          1,382
vans            1,235
sedans          2,391
;
```

次の例では、カラムポインタコントロールを使用したフォーマット入力によって、データ行が読み込まれます。

```
data january_sales;
```

```
   input Item $10. @17 Amount comma5.;
   datalines;
trucks          1,382
vans            1,235
sedans          2,391
;
```

変数 Item の最初の値の読み込み後、ポインタは隣の位置であるカラム 11 に残されます。次に、絶対指定のカラムポインタコントロール@17 によって、入力バッファのカラム 17 に移動するようポインタに指示が出されます。これにより、正しい位置から変数 Amount の値が読み込まれます。

次のプログラムでは、相対指定のカラムポインタコントロール+6 によって、SAS による次のデータ値の読み込み前に、右に 6 カラム移動するようポインタに命令が出されます。

```
data january_sales;
   input Item $10. +6 Amount comma5.;
   datalines;
trucks          1,382
vans            1,235
sedans          2,391
;
```

これら 2 つのプログラムのデータは、列状に配置されています。カラム入力と同様に、フィールドからフィールドへ移動するようポインタに命令します。カラム入力ではカラム指定を使用します。一方、フォーマット入力では、入力形式で指定した長さとポインタコントロールをあわせて使用します。

フォーマット入力: 留意点

フォーマット入力を使用する場合は、次の規則に留意してください。

- SAS では、入力形式で指定された数のカラムを読み込むまで、フォーマット入力データの読み込みが続けられます。このデータ読み込み方法はリスト入力とは異なります。リスト入力では、ブランク(または定義された他の区切り文字)に達するまで読み込みが続けられます。
- ポインタを配置して次の値を読み込むには、ポインタコントロールを使用します。
- パック 10 進や、カンマを含むデータなど、非標準形式で格納されたデータを読み込めます。
- “カラム入力: 留意点” (60 ページ)で説明したように、カラム入力の全機能を活用すると、入力形式を柔軟に使用できます。

整列されていないデータを、より柔軟に読み込む

より柔軟なリスト入力方法について

リスト入力は最もコーディングが簡単ですが、データに制限が加えられることに注意してください。フォーマット修飾子を使用すると、通常の制限による不都合をなくして、リスト入力の簡易性を生かすことができます。たとえば、修飾リスト入力を使用すると、次の操作を行えます。

- デフォルト長の 8 バイトよりも長い文字変数の作成

- カンマ、ハイフン、通貨記号などの特殊文字を含む数値データの読み込み
- 埋め込みブランクを含む文字データの読み込み
- SAS 日付変数として格納可能なデータ値の読み込み

長い変数の作成と特殊文字を含む数値データの読み込み

コロンフォーマット修飾子(:)を使用してリスト入力を修飾するだけで、次を読み込めるようになります。

- 9 文字以上の文字データ
- 特殊文字を含む数値データ

リスト入力でコロンフォーマット修飾子を使用するには、変数名と入力形式の間にコロンを挿入します。単純なリスト入力の場合と同様に、少なくとも 1 つのブランク(または他の定義済み区切り文字)で各値同士を区切る必要があり、文字値に埋め込みブランク(または他の定義済み区切り文字)を含められません。次の DATA ステップについて考えてみましょう。

```
data january_sales;

   input Item : $12. Amount : comma5.;
   datalines;
Trucks 1,382
Vans 1,235
Sedans 2,391
SportUtility 987
;

proc print data=january_sales;
   title 'January Sales in Thousands';
run;
```

変数 Item は長さが 12 です。また、変数 Amount は、数字を有効な数値として読み込むために、数字からカンマを削除する入力形式(この場合は COMMA5.)を必要としています。この前の例では、フォーマット入力を使用してデータを読み込んでおり、列状に配置する必要がありましたが、ここではデータ値は列状に配置されていません。

次の出力は、処理結果のデータセットを示しています。

図 4.11　修飾リスト入力(: comma5.)によって作成されたデータセット

January Sales in Thousands

Obs	Item	Amount
1	trucks	1382
2	vans	1235
3	sedans	2391

埋め込みブランクを含む文字データの読み込み

リスト入力では、ブランクによって、1 つの値が終了して次の値が開始する位置が決定されるので、通常、値にはブランクを含められません。ただし、アンパサンドフォーマット修飾子(&)を指定すると、リスト入力を使用して、埋め込みブランクを 1 つ含むデータ

を読み込めます。唯一の制限は、レコード内で各値と次のデータ値を 2 つ以上のブランクで区切る必要があることです。

リスト入力でアンパサンドフォーマット修飾子を使用するには、変数名と入力形式の間にアンパサンドを挿入します。次の DATA ステップでは、リスト入力でアンパサンドフォーマット修飾子を使用して、データセット CLUB2 が作成されます。固定カラムのデータではないことに注意してください。そのため、カラム入力には適していません。

```
data club2;
   input IdNumber Name & $18. Team $ StartWeight EndWeight;
   datalines;
1023 David Shaw   red 189 165
1049 Amelia Serrano  yellow 145 124
1219 Alan Nance  red 210 192
1246 Ravi Sinha  yellow 194 177
1078 Ashley McKnight  red 127 118
1221 Jim Brown  yellow 220 .
;

proc print data=club2;
   title 'Weight Club Members';
run;
```

文字変数 Name は長さ 18 で、1 つのブランクで区切った会員の姓と名前を含んでいます。INPUT ステートメントでデータを正しく読み込むには、データ行の変数 Name と変数 Team の値間にブランクを 2 つ入れる必要があります。

次の出力は、処理結果のデータセットを示しています。

図 4.12 *修飾リスト入力(& $18.)によって作成されたデータセット*

Weight Club Members

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David Shaw	red	189	165
2	1049	Amelia Serrano	yellow	145	124
3	1219	Alan Nance	red	210	192
4	1246	Ravi Sinha	yellow	194	177
5	1078	Ashley McKnight	red	127	118
6	1221	Jim Brown	yellow	220	.

複数の入力スタイルを使用する

複数の入力スタイルの使用例

INPUT ステートメントの最初に特定のスタイル(リスト、カラムまたはフォーマット)を使用した場合でも、使用するスタイルがその 1 つに限定されるわけではありません。複数の入力スタイルを使用しても生データレコードの記述が適切であれば、1 つの

INPUT ステートメントで複数の入力スタイルを使用できます。たとえば、次の DATA ステップでは、3 つの入力スタイルがすべて使用されています。

```
data club1;
   input IdNumber 1
         Name $18. 2
         Team $ 25-30 3
         StartWeight EndWeight;

   datalines;1
1023 David Shaw         red    189 165
1049 Amelia Serrano     yellow 145 124
1219 Alan Nance         red    210 192
1246 Ravi Sinha         yellow 194 177
1078 Ashley McKnight    red    127 118
1221 Jim Brown          yellow 220   .
;

proc print data=club1;
   title 'Weight Club Members';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1　変数 IdNumber、StartWeight および EndWeight はリスト入力で読み込まれます。

2　変数 Name はフォーマット入力で読み込まれます。

3　変数 Team はカラム入力で読み込まれます。

次の出力は、データが正しく読み込まれたことを示しています。

図 4.13　複数の入力スタイルによって作成されたデータセット

Weight Club Members

Obs	IdNumber	Name	Team	StartWeight	EndWeight
1	1023	David Shaw	red	189	165
2	1049	Amelia Serrano	yellow	145	124
3	1219	Alan Nance	red	210	192
4	1246	Ravi Sinha	yellow	194	177
5	1078	Ashley McKnight	red	127	118
6	1221	Jim Brown	yellow	220	.

入力スタイルのポインタ位置への影響について

複数の入力スタイルの使用によって問題が発生する可能性がある理由

注意:

1 つの INPUT ステートメントで複数の入力スタイルを使用する際は、入力バッファの値が読み込まれた後で、入力ポインタがどこに位置するのかを理解しておかなければ、予期しない結果が出る可能性があります。INPUT ステートメントでは、入力バッファのレコードからデータ値が読み込まれる際、ポインタによってその位置が追跡されます。次のセ

クションを読み、入力スタイル間のポインタの動きの違いを理解してから、1 つの INPUT ステートメントで複数の入力スタイルを使用します。

カラム入力とフォーマット入力でのポインタ位置

カラム入力とフォーマット入力では、正確なポインタ位置を決定するための命令を与えます。カラム入力では、INPUT ステートメントで指定したカラムが読み込まれます。フォーマット入力では、入力形式によって指定した正確な長さが読み込まれます。どちらの場合も、ポインタは命令された位置まで移動して停止します。ポインタは、最後に読み込まれたカラムの直後のカラムに置かれます。

2 つの入力例を次に示し、その後でポインタ位置について説明します。最初の DATA ステップでは、カラム入力を行います。

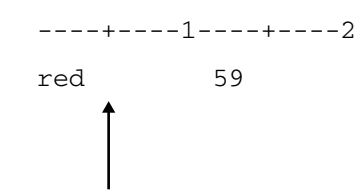
```
data scores;
   input Team $ 1-6 Score 12-13;
   datalines;
red        59
blue       95
yellow     63
green      76
;
```

2 番目の DATA ステップでは、同じデータを使用してフォーマット入力を行います。

```
data scores;
   input Team $6. +5 Score 2.;
   datalines;
red        59
blue       95
yellow     63
green      76
;
```

次の図に示すとおり、前述の 2 つの INPUT ステートメントのどちらでも、最初の値の読み込み後、ポインタはカラム 7 に位置します。

図 4.14 *ポインタ位置: カラム入力とフォーマット入力*



リスト入力とは違って、カラム入力とフォーマット入力では、完全に命令に依存して、ポインタを移動し、2 番目の変数 Score の値を読み込みます。カラム入力では、カラム指定によって、ポインタを各データフィールドに移動します。フォーマット入力では、入力形式とポインタコントロールによって、ポインタ位置を制御します。

次の INPUT ステートメントでは、カラム指定 12-13 のカラム入力によって、ポインタがカラム 12 に移動し、変数 Score の値が読み込まれます。

```
input Team $ 1-6 Score 12-13;
```

次の INPUT ステートメントでは、+5 カラムポインタコントロールのフォーマット入力によって、ポインタがカラム 12 に移動します。その後で、2.数値入力形式を使用して変数 Score の値が読み込まれます。

```
input Team $6. +5 Score 2.;
```

ポインタコントロールを使用して、値の開始カラムにポインタを移動しなければ、この INPUT ステートメントでは、ブランクのカラム 7 と 8 で Score 値の読み込みが試みられます。

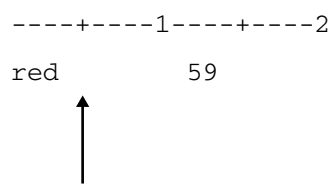
リスト入力でのポインタ位置

これに反して、リスト入力では、レコードのスキャンによってポインタ位置が決定されます。リスト入力では、ポインタはブランクに達するまで読み込みを続け、その次のカラムで停止します。次の変数値を読み込むために、ポインタは、検出された先頭のブランクをすべて切り捨て、自動的にブランク以外の最初のカラムに移動します。リスト入力で読み込まれた同じデータを次に示します。

```
data scores;
   input Team $ Score;
   datalines;
red        59
blue       95
yellow     63
green      76
;
```

次の図に示すとおり、値 **red** の読み込み後、ポインタはカラム 5 に位置しています。次の変数である Score はリスト入力で読み込むため、ポインタはブランク以外の次の値をスキャンしてから、Score 値の読み込みを開始します。カラム入力やフォーマット入力とは違って、リスト入力では、次のフィールドの開始位置に明示的にポインタを移動する必要はありません。

図 4.15 *ポインタ位置: リスト入力*



要約

ステートメント

DATALINES;
DATALINES ステートメントの直後にデータ行が指定されることを示します。最終データ行の直後の行のセミコロンは、データの終了を示します。これにより、DATA ステップのコンパイルと実行が引き起こされます。

INFILE DATALINES DLM='*character*';
ジョブストリームのデータ行として、外部ファイルのかわりに入力レコードソースが示されます。プログラムに入力データを含める場合は、DATALINES ステートメントの直後にデータ行を指定します。INFILE ステートメントで DATALINES を指定できるので、INFILE ステートメントでのみ使用可能な多くのデータ読み込みオプションを利用できます。

DLM=オプションでは、入力レコード内のデータ値を区切るために使用する文字が指定されます。デフォルトでは、ブランクによってデータ値の終了が示されます。リ

スト入力を使用して、ブランク以外の文字でデータ値を区切ったデータレコードを読み込む場合、このオプションが役立ちます。

INPUT *variable* <&><$>;
リスト入力によって、入力データレコードを読み込みます。& (アンパサンドフォーマット修飾子)を使用すると、文字値に埋め込みブランクを含められます。アンパサンドフォーマット修飾子を使用する場合は、データ値の終了を示すためにブランクが 2 つ必要になります。$は文字変数を示します。

INPUT *variable start-column* <– *end-column*>;
カラム入力によって、入力データレコードを読み込みます。データの長さが 1 バイトしかない場合は、end-column を省略できます。この入力スタイルを使用すると、省略するデータのカラムをスキップできます。

INPUT *variable* : *informat*;
INPUT *variable* & *informat*;
修飾リスト入力によって、入力データレコードを読み込みます。: (コロンフォーマット修飾子)を指定すると、その後の入力形式を使用してデータ値を読み込むように命令が出されます。& (アンパサンドフォーマット修飾子)を指定すると、その後の入力形式を使用してデータ値を読み込むように命令が出されます。アンパサンドフォーマット修飾子を使用する場合は、データ値の終了を示すためにブランクが 2 つ必要になります。

INPUT <*pointer-control*> *variable informat*;
フォーマット入力によって、生データを読み込みます。入力形式によって、データを読み込むための特別な命令が与えられます。また、pointer-control を使用すると、特定カラムで読み込みを開始するように指示が出されます。

前述の 3 つの入力スタイルの構文では、変数は 1 つだけです。INPUT ステートメントの後続の変数は、最初の変数と同じ入力スタイルで記述できる場合があります。1 つの INPUT ステートメントで、3 つの入力スタイル(リスト、カラムおよびフォーマット)のいずれでも使用できます。

カラムポインタコントロール

@n
入力バッファで、ポインタが n 番目のカラムに移動します。

+n
入力バッファで、ポインタが前方に n カラム移動します。

/
ポインタが入力バッファの次の行に移動します。

#n
入力バッファで、ポインタが n 番目の行に移動します。

詳細情報

高度な機能
より高度なデータ読み込み機能については、5 章, “生データから作成する: 応用” (71 ページ)を参照してください。

文字で区切られたデータ
ブランク以外の文字で区切られたデータの読み込みの詳細については、“INFILE Statement” (*SAS Statements: Reference*)の DELIMITER=オプションを参照してください。

ポインタコントロール
: カラムポインタコントロール、ラインポインタコントロールおよびラインホールド指定子の詳細な説明とリストについては、*SAS Statements: Reference* を参照してください。

入力の種類
: INPUT ステートメントの詳細については、“INPUT Statement” (*SAS Statements: Reference*)を参照してください。

5 章

生データから作成する: 応用

生データを使用した応用について **71**
目的 71
前提条件 72

オブザベーションを作成する前に条件をテストする **72**

1 つのレコードから複数のオブザベーションを作成する **74**
後置@@ラインホールド指定子の使用 74
後置@@が DATA ステップの実行に及ぼす影響について 74

複数のレコードを読み込み、1 つのオブザベーションを作成する **78**
データレコードの構造 78
方法 1: 複数の INPUT ステートメントを使用 78
方法 2:/ラインポインタコントロールを使用 79
複数のレコードから任意の順序で変数を読み込む 81
#n ラインポインタコントロールが DATA ステップの実行に及ぼす影響について 82

問題の解決: 入力レコードに十分な値が含まれていない場合 **85**
デフォルト動作について 85
制御方法: オプション 86

要約 **88**
カラムポインタコントロール 88
ラインホールド指定子 89
ステートメント 89

詳細情報 **90**

生データを使用した応用について

目的

生データから SAS データセットを作成するには、多くの場合、大部分の基本機能よりも高度な機能が必要になります。このセクションでは、次の高度な生データ読み込み機能を学習します。

- 入力レコードに予想外の欠損値がある場合に発生することを理解し、制御する方法

- レコードを 2 回以上読み込み、現在のレコードを処理する前に条件をテストできるようにする方法
- 1 つの入力レコードから複数のオブザベーションを作成する方法
- 複数のオブザベーションを読み込んで 1 つのレコードを作成する方法

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 1 章, “SAS System について” (3 ページ)
- 3 章, “DATA ステップ処理について” (27 ページ)

オブザベーションを作成する前に条件をテストする

場合によっては、レコードを読み込み、それ以降の処理について判断できるように、指定条件のテストをする間、入力バッファにそのレコードを保持する必要があります。たとえば、必要に応じて、レコードを保持して再読み込みできるようにする機能は、SAS がデータレコードからオブザベーションを作成する前に条件をテストする必要がある場合に役立ちます。これを実行するには、後置アットマーク(@)を使用します。

たとえば、より大きなレコードグループのサブセットとなる SAS データセットを作成するには、オブザベーションの作成に特定のレコードが使用されるかを判断するために、条件をテストする必要がある場合があります。INPUT ステートメントの最後のセミコロンの前に後置アットマークを指定すると、入力バッファに現在のデータ行を保持するよう指示が出されます。これにより、後続の INPUT ステートメントでも、データ行の使用が可能になります。指定しなければ、次の INPUT ステートメントによって、新しいレコードが入力バッファに読み込まれます。

各レコードを 2 回ずつ読み込むプロセスを設定するには、次の操作を実行します。

1. INPUT ステートメントを使用して、レコードの一部を読み込みます。
2. INPUT ステートメントの最後に後置@を使用して、次の INPUT ステートメントの実行用に、入力バッファにレコードを保持します。
3. 条件をテストするために読み込む部分に対して、IF ステートメントを使用します。
4. 条件を満たしている場合は、別の INPUT ステートメントを使用してレコードの残り部分を読み込み、オブザベーションを作成します。
5. 条件を満たしていない場合、レコードは解放され、DATA ステップの先頭に制御が戻されます。

レコードを 2 回読み込むには、次の INPUT ステートメント実行時に、新規レコードを自動的に入力バッファに入れないようにする必要があります。そのためには最初の INPUT ステートメントで後置@を使用します。後置@は 2 つのラインホールド指定子のうちの 1 つで、これを使用すると、以降の処理のために入力バッファにレコードを保持できます。

たとえば、ヘルスクラブのデータには、全会員についての情報が含まれています。次の DATA ステップでは、red チームの会員のみを含む SAS データセットが作成されます。

```
data red_team;
```

```
   input Team $ 13-18 @;  1
   if Team='red';  2
   input IdNumber 1-4 StartWeight 20-22 EndWeight 24-26;3
   datalines;
1023 David  red    189 165
1049 Amelia yellow 145 124
1219 Alan   red    210 192
1246 Ravi   yellow 194 177
1078 Ashley red    127 118
1221 Jim    yellow 220   .
;  4

proc print data=red_team;
   title 'Red Team';
run;
```

この DATA ステップでは、次の処理が発生します。

1 INPUT ステートメントで、レコードが入力バッファに読み込まれ、カラム 13 から 18 までのデータ値が読み取られて、その値がプログラムデータベクトル内の変数 Team に割り当てられます。後置@が指定されているので、入力バッファのレコードは保持されます。

2 IF ステートメントにより、Team の値が **red** の場合のみ、DATA ステップの現在の反復を続行できます。値が red でない場合、SAS は現在の反復を停止して DATA ステップの先頭に戻り、プログラムデータベクトルの値を欠損値にリセットして、保持されていたレコードを入力バッファから解放します。

3 INPUT ステートメントは、Team の値が **red** の場合にのみ実行されます。入力バッファに保持されたレコードから残りのデータ値が読み込まれ、変数 IdNumber、StartWeight および EndWeight にその値が割り当てられます。

4 プログラムが DATA ステップの先頭に戻ると、レコードが入力バッファから解放されます。

次の出力は、処理結果のデータセットを示しています。

図 5.1 *後置@を使用して作成されたサブセットデータセット*

Red Team

Obs	Team	IdNumber	StartWeight	EndWeight
1	red	1023	189	165
2	red	1219	210	192
3	red	1078	127	118

1 つのレコードから複数のオブザベーションを作成する

後置@@ラインホールド指定子の使用

1 つの生データレコードから複数のオブザベーションを作成する必要がある場合があります。そのようなレコードの読み込みを指示する 1 つの方法は、もう 1 つのラインホールド指定子である二重後置アットマーク(@@つまり"後置@@")を使用することです。後置@@が指定されている場合、新しい INPUT ステートメントが検出されたときに、SAS は新しいレコードを入力バッファへ読み込みません。さらに、プログラムが DATA ステップの先頭に戻ったときにレコードの解放も行いません。(後置@を指定しても、DATA ステップを反復すると、入力バッファのレコードは保持されないことに注意してください。)

たとえば、次の DATA ステップでは、INPUT ステートメントで後置@@が使用されます。

```
data body_fat;
   input Gender $ PercentFat @@;
   datalines;
m 13.3 f 22
m 22   f 23.2
m 16   m 12
;

proc print data=body_fat;
    title 'Results of Body Fat Testing';
run;
```

次の出力は、処理結果のデータセットを示しています。

図 5.2 後置@@を使用して作成されたデータセット

Results of Body Fat Testing

Obs	Gender	PercentFat
1	m	13.3
2	f	22.0
3	m	22.0
4	f	23.2
5	m	16.0
6	m	12.0

後置@@が DATA ステップの実行に及ぼす影響について

前述の例におけるデータレコードの読み込み方法を理解するために、前述の DATA ステップで使用されたデータ行について考えてみます。

```
m 13.3 f 22
m 22   f 23.2
```

```
m 16   m 12
```

各レコードに、1 つではなく 2 つのオブザベーションに対する生データが含まれています。3 章, “DATA ステップ処理について” (27 ページ).で説明した、DATA ステップのフローの観点から次の例を考えてみましょう。

SAS は、DATA ステップの最後に達すると、プログラムの先頭に戻り、次の反復を開始しますが、それは読み込むレコードがなくなるまで実行されます。DATA ステップの先頭に戻って INPUT ステートメントを実行するたびに、自動的に新しいレコードが入力バッファに読み込まれます。したがって、各レコードの 2 番目のデータ値セットは読み込まれません。

```
m 13.3 f 22
m 22   f 23.2
m 16   m 12
```

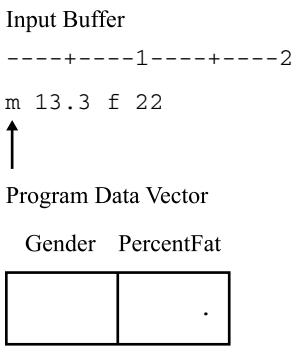
各レコードで 2 番目のデータ値のセットが読み込みできるようにするには、後置@@を使用して、入力バッファにレコードを保持するよう SAS に指示します。各レコードは、レコードの終わりに達するまで、入力バッファに保持されます。このプログラムでは、INPUT ステートメントを実行するたびに次のレコードが自動的に入力バッファに入ることはありません。また、DATA ステップの先頭に戻るときに現在のレコードが自動的に解放されることもありません。その結果、現在のレコードのポインタ位置が維持されるので、プログラムでそのレコード内の各値を読み込めます。DATA ステップの反復が完了するたびに、オブザベーションがデータセットに書き込まれます。

次の 5 つの図は、この例のように、INPUT ステートメントに後置@@を記述した場合、入力バッファで行われる処理内容を示しています。

```
input Gender $ PercentFat @@;
```

最初の図は、プログラムデータベクトル内のすべての値が欠損値に設定されていることを示しています。INPUT ステートメントによって、最初のレコードが入力バッファに読み込まれます。プログラムによって、現在のポインタ位置から値の読み込みが開始されます。現在位置は入力バッファの先頭です。

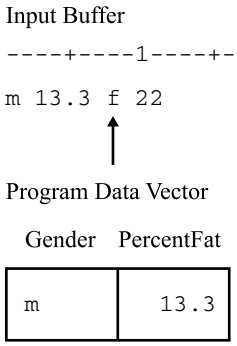
図 5.3 *最初の反復: 最初のレコードの読み込み*



次の図では、値 m がプログラムデータベクトルに書き込まれたことを示しています。ポインタが 13.3 の後のブランクに達すると、変数 PercentFat の値全体が読み込まれま

す。ポインタは次のカラムで停止し、値 13.3 はプログラムデータベクトルに書き込まれます。

図 5.4 *最初のオブザベーションの作成*

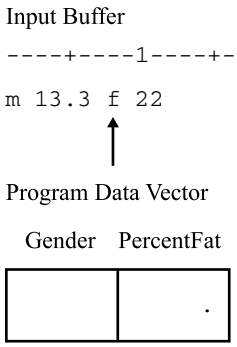


INPUT ステートメントに他の変数がなくて、なおかつ DATA ステップにも他のステートメントがないので、3 つの処理が行われます。

1. 最初のオブザベーションがデータセットに書き込まれます。
2. DATA ステップの次の反復が開始されます。
3. プログラムデータベクトル内の値が欠損値に設定されます。

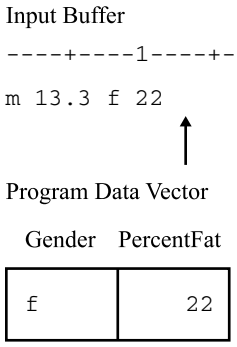
次の図は、現在のポインタ位置を示しています。同じレコード内の次のデータを読み込む準備が整っています。

図 5.5 *2 回目の反復: 最初のレコードが入力バッファに残る*



次の図では、INPUT ステートメントにより、次の 2 つの値が入力バッファから読み込まれ、それらがプログラムデータベクトルに書き込まれたことを示しています。

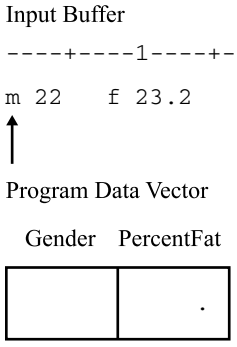
図 5.6 *2 番目のオブザベーションの作成*



DATA ステップが 2 回目の反復を完了すると、プログラムデータベクトル内の値が、2 番目のオブザベーションとしてデータセットに書き込まれます。その後、DATA ステップの 3 回目の反復が開始されます。プログラムデータベクトル内の値が欠損値に設定され、INPUT ステートメントが実行されます。ポインタは、この時点でカラム 13(最後に読み込んだデータ値の 2 カラム右)に位置し、そこから読み込みを続けます。これはリスト入力なので、次の値の読み込みを開始するためにポインタは次の非ブランク文字をスキャンします。ポインタが入力バッファの最後に達するまでに、非ブランク文字が見つからなかった場合は、新しいレコードが入力バッファに読み込まれます。

最後の図は、3 番目のオブザベーションの値が 2 番目のレコードの先頭から読み込まれることを示しています。

図 5.7 *3 回目の反復: 2 番目のレコードを入力バッファに読み込み*



レコードをすべて読み込むまで、このプロセスが続けられます。処理結果の SAS データセットには、オブザベーションが 3 つではなく 6 つ含まれます。

注: このプログラムでは、入力レコード内のデータがすべて正常に読み込まれても、プログラムは次の行に進む必要があることを示すメッセージがログに書き込まれます。

複数のレコードを読み込み、1 つのオブザベーションを作成する

データレコードの構造

前述の例(“埋め込みブランクを含む文字データの読み込み” (64 ページ)を参照)では、1 つの生データレコードに含まれる、複数のオブザベーションのデータを示しています。

```
1023 David Shaw       red 189 165
```

この INPUT ステートメントでは、1 つのレコード内全体に配置されたデータ値がすべて読み込まれます。

```
input IdNumber 1-4 Name $ 6-23 Team $ StartWeight EndWeight;
```

ここで、逆の状況について考えてみましょう。1 つのオブザベーションの情報が、1 つの生データレコードに含まれているのではなく、複数のレコードに散在しているとします。たとえば、ヘルスクラブのデータでは、1 人の会員についての情報を、1 つのレコードに入れるのではなく複数のレコードに分散する形式で作成できます。

```
1023 David Shaw
red
189 165
```

方法 1: 複数の INPUT ステートメントを使用

次の例のように、複数の INPUT ステートメント(レコードごとに 1 つずつ)によって、各レコードを 1 つのオブザベーションに読み込めます。

```
input IdNumber 1-4 Name $ 6-23;
input Team $ 1-6;
input StartWeight 1-3 EndWeight 5-7;
```

複数の INPUT ステートメントの使用法を理解するために、DATA ステップ実行時の処理内容について考えてみましょう。各反復中に各 INPUT ステートメントが検出されると、自動的に 1 レコードが INPUT バッファに読み込まれることに注意してください。SAS で、データ値が入力バッファから読み込まれ、変数値としてプログラムデータベクトルに書き込まれます。DATA ステップの最後に、プログラムデータベクトル内のすべての変数値が、自動的に 1 つのオブザベーションとして書き込まれます。

この例では、DATA ステップで複数の INPUT ステートメントを使用して、選択したデータフィールドのみを読み込み、変数 IdNumber、StartWeight および EndWeight のみを含むデータセットを作成します。

```
data club2;
   input IdNumber 1-4;  1
   input;  2
   input StartWeight 1-3 EndWeight 5-7; 3
   datalines;
1023 David Shaw
red
189 165
1049 Amelia Serrano
yellow
```

```
145 124
1219 Alan Nance
red
210 192
1246 Ravi Sinha
yellow
194 177
1078 Ashley McKnight
red
127 118
1221 Jim Brown
yellow
220  .
;

proc print data=club2;
   title 'Weight Club Members';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 最初の INPUT ステートメントでは、最初のレコードの 1 データフィールドのみが読み込まれ、変数 IdNumber にその値が割り当てられます。

2 2 番目の INPUT ステートメントは、引数のない、ヌルの INPUT ステートメントで、2 番目のレコードを入力バッファに読み込みます。ただし、値は変数に割り当てられません。

3 3 番目の INPUT ステートメントでは、3 番目のレコードが入力バッファに読み込まれ、変数 StartWeight および EndWeight にその値が割り当てられます。

次の出力は、処理結果のデータセットを示しています。

図 5.8 *複数の INPUT ステートメントによって作成されたデータセット*

Weight Club Members

Obs	IdNumber	StartWeight	EndWeight
1	1023	189	165
2	1049	145	124
3	1219	210	192
4	1246	194	177
5	1078	127	118
6	1221	220	.

方法 2:/ラインポインタコントロールを使用

レコードごとに別々の INPUT ステートメントを記述することが、1 つのオブザベーションを作成するための唯一の方法ではありません。INPUT ステートメントを 1 つ記述して、スラッシュ(/)ラインポインタコントロールを使用することもできます。スラッシュライン

ポインタコントロールを使用すると、新しいレコードが強制的に入力バッファに読み込まれ、そのレコードの先頭にポインタが置かれます。

次の例では、複数レコードの読み込みに、INPUT ステートメントを 1 つのみ使用しています。

```
data club2;
   input IdNumber 1-4 / / StartWeight 1-3 EndWeight 5-7;
   datalines;
1023 David Shaw
red
189 165
1049 Amelia Serrano
yellow
145 124
1219 Alan Nance
red
210 192
1246 Ravi Sinha
yellow
194 177
1078 Ashley McKnight
red
127 118
1221 Jim Brown
yellow
220   .
;

proc print data=club2;
   title 'Weight Club Members';
run;
```

前述の例における新しい INPUT ステートメントの先頭と正確に同じ場所に/ラインポインタコントロールを記述します。“方法 1: 複数の INPUT ステートメントを使用” (78 ページ)を参照してください。この DATA ステップ実行時の入力バッファおよびプログラムデータベクトルでの一連のイベントは、前述の方法 1 の例と同じです。/は新しいレコードを入力バッファに読み込むための信号です。DATA ステップで新しい INPUT ステートメントが検出されると、自動的にこの読み込みが行われます。
前述の例に出てくる 2 つのスラッシュ(/ /)は、SAS がレコードをスキップすることを示しています。SAS は、最初のレコードを読み込み、2 番目のレコードはスキップし、3 番目のレコードを読み込みます。

次の出力は、処理結果のデータセットを示しています。

図 5.9 /ラインポインタコントロールを使用して作成されたデータセット

Weight Club Members

Obs	IdNumber	StartWeight	EndWeight
1	1023	189	165
2	1049	145	124
3	1219	210	192
4	1246	194	177
5	1078	127	118
6	1221	220	.

複数のレコードから任意の順序で変数を読み込む

#n ラインポインタコントロールを使用して入力レコードセット内の特定のレコードを指すことにより、複数のレコードを読み込んで 1 つのオブザベーションを作成することもできます。前のセクションで学習したように、/ラインポインタコントロールの使用が複数の INPUT ステートメントに勝る利点は、ステートメント数が少なくてすむことです。一方、*#n* ラインポインタコントロールを使用すると、データ値がどのレコードに含まれていても、任意の順序で変数を読み込めます。これは、データ行をスキップするときにも便利です。

この例では、1 つの INPUT ステートメントを使用して、複数のデータ行を異なる順序で読み込みます。

```
data club2;
   input #2 Team $ 1-6 #1 Name $ 6-23 IdNumber 1-4
         #3 StartWeight 1-3 EndWeight 5-7;
   datalines;
1023 David Shaw
red
189 165
1049 Amelia Serrano
yellow
145 124
1219 Alan Nance
red
210 192
1246 Ravi Sinha
yellow
194 177
1078 Ashley McKnight
red
127 118
1221 Jim Brown
yellow
220   .
;
```

```
proc print data=club2;
   title 'Weight Club Members';
run;
```

次の出力は、処理結果のデータセットを示しています。

図 5.10 #n ラインポインタコントロールを使用して作成されたデータセット

Weight Club Members

Obs	Team	Name	IdNumber	StartWeight	EndWeight
1	red	David Shaw	1023	189	165
2	yellow	Amelia Serrano	1049	145	124
3	red	Alan Nance	1219	210	192
4	yellow	Ravi Sinha	1246	194	177
5	red	Ashley McKnight	1078	127	118
6	yellow	Jim Brown	1221	220	.

オブザベーションの順序は生レコードと同じです(“複数のレコードから任意の順序で変数を読み込む” (81 ページ)のセクションを参照)。ただし、データセットの変数順序は、生の入力データレコードの変数順序とは異なります。これは、INPUT ステートメントの変数順序が、処理結果のデータセットの変数順序に対応するためです。

#n ラインポインタコントロールが DATA ステップの実行に及ぼす影響について

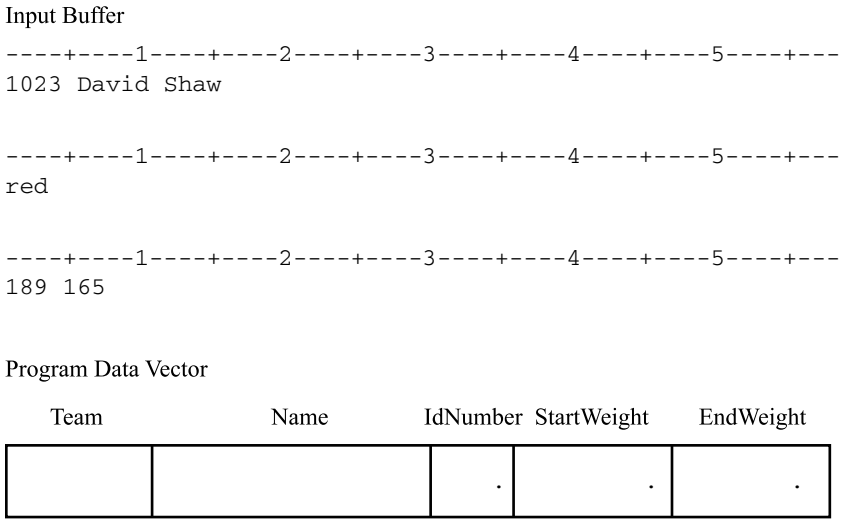
#n ラインポインタコントロールの重要性を理解するため、/ラインポインタコントロールと複数の INPUT ステートメントの処理を示す DATA ステップでの一連のイベントを思い出してください。各レコードは、入力バッファに順次読み込まれます。データが読み込まれると、/または新しい INPUT ステートメントによって、プログラムが次のレコードを入力バッファに読み込みます。2 番目のレコードから値が読み込まれた後は、入力バッファでは最初のレコードはすでに使用できなくなっているので、プログラムが最初のレコードから値を読み込むことはできません。

この問題を解決するには、*#n* ラインポインタコントロールを使用します。*#n* ラインポインタコントロールは、複数行の入力バッファを作成するようプログラムに示すため、プログラムデータベクトルにオブザベーションを作成している間、1 オブザベーションの全データが使用可能になります。*#n* ラインポインタコントロールは、各変数のデータが含まれるレコードも指定します。*#n* ラインポインタコントロールを使用する場合は、生データで各オブザベーションのレコード数が同じである必要があります。たとえば、あるオブザベーションのレコード数が 3、次のオブザベーションのレコード数が 2 という状態は許可されません。

プログラムがコンパイルされて入力バッファが作成される際、INPUT ステートメントが調べられ、1 つのオブザベーションの読み込みに必要なレコード数を含められる行数の入力バッファが 1 つ作成されます。この例では、指定される最大のレコード番号は 3 なので、入力バッファが作成されると、レコードが 3 つずつ含められます。次の図は、この例における DATA ステップのフローを示しています。

次の図は、プログラムデータベクトル内で値が欠損値に設定され、INPUT ステートメントによって最初の 3 レコードが入力バッファに読み込まれたことを示しています。

図 5.11 *3 レコードを 1 オブザベーションとして入力バッファに読み込む*



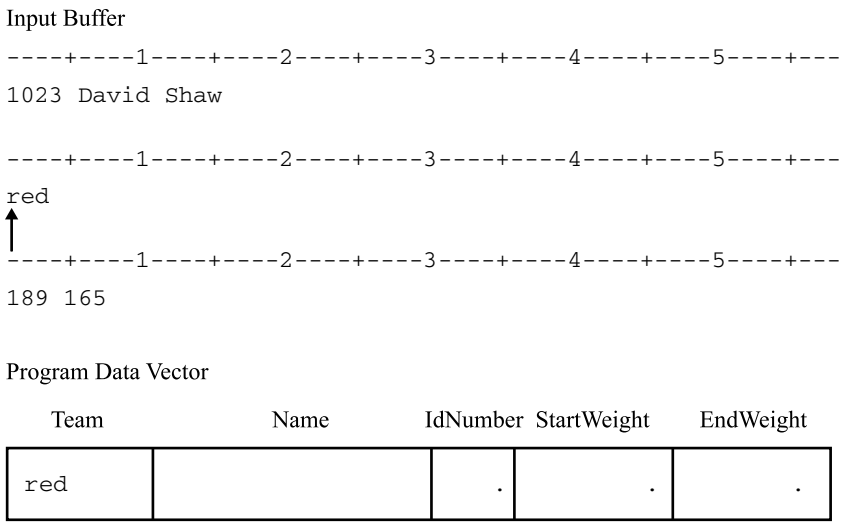
この例の INPUT ステートメントは、次のとおりです。

```
input #2 Team $ 1-6
      #1 Name $ 6-23 IdNumber 1-4
      #3 StartWeight 1-3 EndWeight 5-7;
```

前に#2 が付いている最初の変数は、2 番目のレコードの値を変数 Team に割り当てることを示しています。

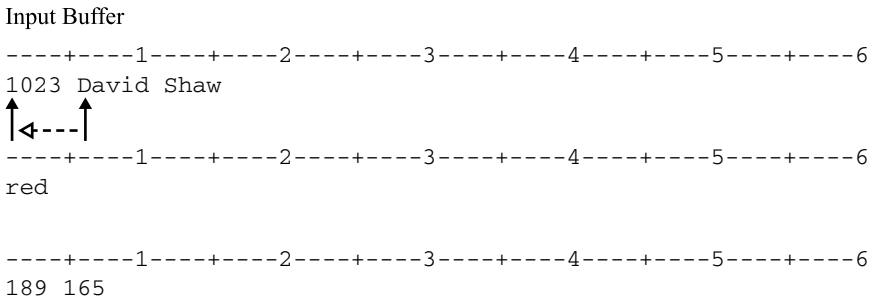
次の図に示すとおり、ポインタが入力バッファの 2 行目に進み、値が読み込まれ、プログラムデータベクトルに書き込まれます。

図 5.12 *最初に 2 番目のレコードから読み込む*



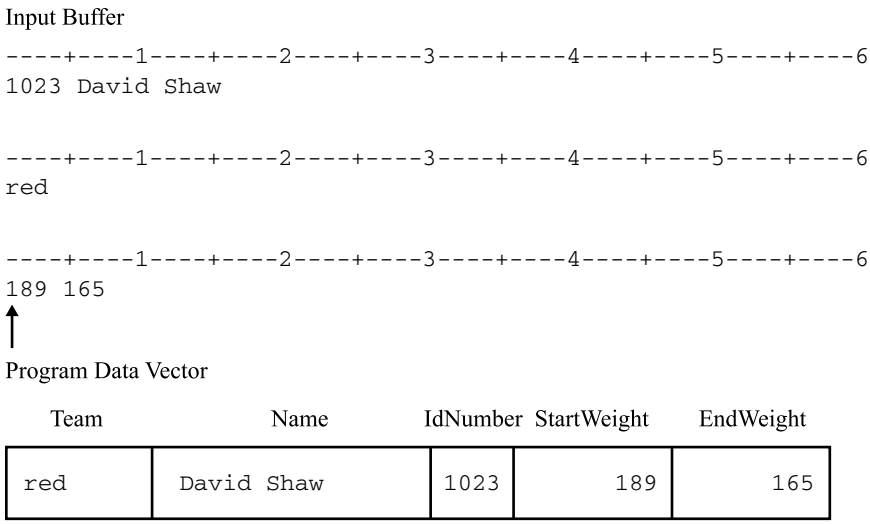
次の図に示すとおり、次にポインタが最初のレコードの 6 カラム目に移動し、値が読み込まれて、プログラムデータベクトル内の変数 Name に割り当てられます。次に最初のカラムに移動して ID 番号が読み込まれ、変数 IdNumber に割り当てられます。

図 5.13 *最初のレコードから読み込む*



次の図では、プロセスが続けられて、最初のオブザベーション内の 3 番目のレコードにポインタが移動したことを示しています。値が読み込まれ、最後に列挙された変数 StartWeight および EndWeight に割り当てられます。

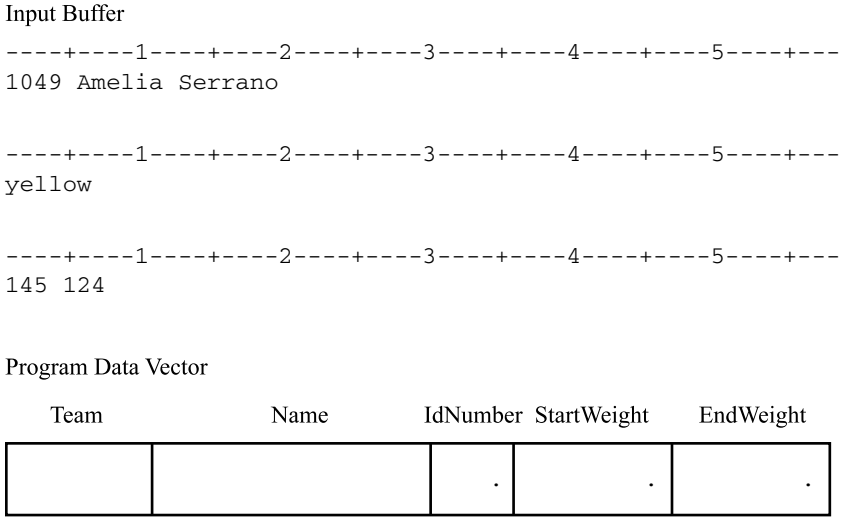
図 5.14 *3 番目のレコードから読み込む*



DATA ステップの最後に達すると、プログラムデータベクトル内の変数値が、1 オブザベーションとしてデータセットに書き込まれます。DATA ステップが先頭に戻り、プログラムデータベクトル内の値が欠損値に設定されます。INPUT ステートメントが再実行されます。

最後の図に示すとおり、次の 3 レコードが入力バッファに読み込まれると、2 番目のオブザベーションの作成準備が整います。

図 5.15 *次の 3 レコードを入力バッファに読み込む*



問題の解決: 入力レコードに十分な値が含まれていない場合

デフォルト動作について

DATA ステップで外部ファイルから生データを読み込む際、INPUT ステートメントに指定されたすべての変数に対するデータを読み込む前に入力行の終わりが検出された場合は、問題が発生する可能性があります。問題が発生する可能性があるのは、可変長レコードまたは欠損値を含むレコード、あるいはその両方を読み込む場合です。

可変長レコードを含む外部ファイルの例を次に示します。

```
----+----1----+----2

22
333
4444
55555
```

この DATA ステップでは、数値入力形式 5.を使用して、生データの各レコード内のフィールドを 1 つ読み込み、変数 TestNumber に値を割り当てます。

```
data numbers;
   infile 'your-external-file';
  input TestNumber 5.;
run;

proc print data=numbers;
   title 'Test DATA Step';
run;
```

DATA ステップで、最初の値(22)が読み込まれます。その値が入力形式で求められる 5 文字よりも短いため、DATA ステップは、次のレコード(333)を使用して値を最後まで入力しようと試みます。この値は PDV に入力され、最初のオブザベーションの TestNumber 変数の値になります。その後、DATA ステップは次のレコードに進みますが、値(4444)が入力形式で求められる値よりも短いため、同じ問題に遭遇します。ここでも DATA ステップは次のレコードに進み、値(55555)を読み込んで、その値を 2 番目のオブザベーションの TestNumber 変数に割り当てます。

次の出力は結果を示しています。プログラムの実行後、SAS ログには、データ値を検索するために次のレコードに移動した箇所を示す NOTE が含められます。

図 5.16 *行末に達した後の生データの読み込み: デフォルト動作*

Test DATA Step

Obs	TestNumber
1	333
2	55555

制御方法: オプション

4 つのオプション: FLOWOVER、STOPOVER、MISSOVER、TRUNCOVER

データ行の終わりを越えて読み込みを試みた後の SAS の動作方法を制御するには、INFILE ステートメントで次のオプションを使用します。

`infile '`*`your-external-file`*`' flowover;`
デフォルト動作です。DATA ステップは、次のレコードをそのまま入力バッファに読み込み、INPUT ステートメントの残りの変数名に割り当てる値を探そうとします。

`infile '`*`your-external-file`*`' stopover;`
INPUT ステートメントで、ステートメント内のすべての変数に対する値が見つからずに現在のレコードの終わりに達した場合、DATA ステップの処理が停止します。外部ファイル内のすべてのデータが指定した基準に合うと想定され、基準に合わないデータレコードが検出されたときは DATA ステップを停止する必要がある場合、このオプションを使用します。

`infile '`*`your-external-file`*`' missover;`
現在のレコードで、INPUT ステートメントのすべての変数に対する値が見つからない場合、DATA ステップが次の行に進まないようにします。そのかわり、DATA ステップによって、値のないすべての変数に対して欠損値が割り当てられます。

`infile '`*`your-external-file`*`' truncover;`
値が INPUT ステートメントで期待される長さに満たなくても、DATA ステップで変数に生データ値が割り当てられます。DATA ステップで入力レコードの終わりが検出された時点で、値のない変数がある場合、そのオブザベーションではその変数に欠損値が割り当てられます。

データ行がプログラム自体に組み込まれている場合(すなわち、データ行が DATALINES ステートメントの後に指定されている場合)でも、これらのオプションを使用できます。外部ファイルを参照するかわりに、DATA ステップ自体にデータレコードを組み込む指定をするには、単に **`datalines`** を使用します。

- **`infile datalines flowover;`**

- **infile datalines stopover;**
- **infile datalines missover;**
- **infile datalines truncover;**

注: このセクションの例では、フォーマット入力での MISSOVER および TRUNCOVER オプションの使用法を示します。これらのオプションは、リスト入力やカラム入力でも使用できます。

MISSOVER オプションについて

MISSOVER オプションを使用して、INPUT ステートメントのすべての変数に対する値が現在のレコードに見つからない場合、DATA ステップが次の行に進まないようにします。そのかわり、DATA ステップでは、任意の指定入力形式に応じた完全な値がないすべての変数に対して欠損値が割り当てられます。入力ファイルには、次の生データが含まれています。

```
----+----1----+----2

22
333
4444
55555
```

次の例では、MISSOVER オプションを使用します。

```
data numbers;
   infile 'your-external-file' missover;
   input TestNumber 5.;
run;

proc print data=numbers;
   title 'Test DATA Step';
run;
```

図 5.17 *MISSOVER オプションによる出力*

Test DATA Step

Obs	TestNumber
1	.
2	.
3	.
4	55555

4 番目のレコードの値のみが入力形式と一致しているので、そのレコードの値のみが TestNumber 変数に割り当てられます。その他のオブザベーションは、欠損値を受け取ります。この例では、これは望ましい結果ではないと考えられますが、MISSOVER オプションが有益な場合もあります。例については、“データセットの更新” (317 ページ)を参照してください。

注: 最後のレコードの終わりにブランク行がある場合、DATA ステップは入力バッファにレコードをもう一つロードしようとします。レコードが残っていないので、MISSOVER オプションで、すべての変数に欠損値を割り当てるよう DATA ステップに指示が出され、余分なオブザベーションがデータセットに加えられます。このような状況が発生しないようにするには、入力データで、最後のレコードの終わりにブランク行が含まれていないことを確認します。

TRUNCOVER オプションについて

TRUNCOVER オプションを使用すると、値が INPUT ステートメントで期待される長さに満たなくても、DATA ステップで変数に生データ値が割り当てられます。DATA ステップで入力レコードの終わりが検出された時点で、値のない変数がある場合、そのオブザベーションではその変数に欠損値が割り当てられます。TRUNCOVER オプションの使用例を次に示します。

```
data numbers;
   infile 'your-external-file' truncover;
   input TestNumber 5.;
run;

proc print data=numbers;
   title 'Test DATA Step';
run;
```

図 5.18 TRUNCOVER オプションによる出力

Test DATA Step

Obs	TestNumber
1	22
2	333
3	4444
4	55555

この結果は、値のうちの 3 つが入力形式と一致しないにもかかわらず、すべての値が TestNumber 変数に割り当てられたことを示しています。TRUNCOVER オプションを使用する別の例については、“例で使用される入力 SAS データセット” (148 ページ)を参照してください。

要約

カラムポインタコントロール

@ *n*
ポインタが入力バッファの *n* 番目のカラムに移動します。

+*n*
ポインタが入力バッファの *n* カラム先に移動します。

/
ポインタが入力バッファの次の行に移動します。

#*n*
ポインタが入力バッファの *n* 番目の行に移動します。

ラインホールド指定子

@
: @(後置@)を使用すると、DATA ステップの同一反復内で新たな INPUT ステートメントが実行されるときに、SAS が新たなデータレコードを入力バッファへ自動的に読み込まないようにします。使用する場合は、後置@を INPUT ステートメント内の最後の項目にする必要があります。

@@
: @@(後置@@)を使用すると、DATA ステップが先頭に戻って次の反復を開始する場合でも、次の INPUT ステートメントが実行されるときに、SAS が新たなデータレコードを入力バッファへ自動的に読み込まないようにします。使用する場合は、後置@@を INPUT ステートメント内の最後の項目にする必要があります。

ステートメント

DATALINES;
: この直後にデータ行が指定されることを示します。最終データ行の直後の行のセミコロンは、データの終了を示します。これにより、DATA ステップのコンパイルと実行が引き起こされます。

INFILE fileref< FLOWOVER | STOPOVER | MISSOVER | TRUNCOVER>;
INFILE '*external-file*' <FLOWOVER | STOPOVER | MISSOVER | TRUNCOVER>;
: INPUT ステートメントで読み込む外部ファイルを指定します。FILENAME ステートメントや適切な動作環境コマンドを使用して割り当てた fileref を指定します。または、外部ファイルの実際の名前を指定します。

  これらのオプションを使用すると、すべての変数に値が割り当てられる前にデータレコードの終わりが検出された場合の SAS の動作方法を制御できます。これらのオプションは、リスト入力、修飾リスト入力およびカラム入力で使用できます。

  FLOWOVER
  : デフォルト動作です。これを指定すると、すべての変数に値が割り当てられる前に現在のレコードの終わりが検出された場合、DATA ステップで次のレコードが調べられます。

  MISSOVER
  : これを指定すると、データレコードの終わりが検出された場合、DATA ステップは値のない変数すべてに欠損値を割り当てます。DATA ステップでは、処理が続行されます。

  STOPOVER
  : これを指定すると、DATA ステップは即座に実行を停止し、SAS ログに NOTE メッセージが書き込まれます。

  TRUNCOVER
  : これを指定すると、DATA ステップでは、INPUT ステートメントで期待された長さに満たない値でも変数に割り当て、データレコードの終わりの検出時に値のない変数すべてに欠損値が割り当てられます。

INPUT *variable* <&> <$>;
: リスト入力によって、入力データレコードを読み込みます。& (アンパサンドフォーマット修飾子)を使用すると、文字値に埋め込みブランクを含められます。アンパサンドフォーマット修飾子を使用する場合は、データ値の終了を示すためにブランクが 2 つ必要になります。$は文字変数を示します。

INPUT variable *start-column <end-column>*;
カラム入力によって、入力データレコードを読み込みます。データの長さが 1 バイトしかない場合は、end-column を省略できます。この入力スタイルを使用すると、省略するデータのカラムをスキップできます。

INPUT *variable* : *informat*;
INPUT *variable* & *informat*;
修飾リスト入力によって、入力データレコードを読み込みます。: (コロンフォーマット修飾子)を指定すると、その後の入力形式を使用してデータ値を読み込むように命令が出されます。& (アンパサンドフォーマット修飾子)を指定すると、その後の入力形式を使用してデータ値を読み込むように命令が出されます。アンパサンドフォーマット修飾子を使用する場合は、データ値の終了を示すためにブランクが 2 つ必要になります。

INPUT *<pointer-control>variable informat*;
フォーマット入力によって、生データを読み込みます。*informat* によって、データを読み込むための特別な命令が与えられます。また、*pointer-control* を使用すると、特定カラムで読み込みを開始するように指示が出されます。

前述の 3 つの入力スタイルの構文では、*variable* は 1 つだけです。INPUT ステートメントの後続の変数は、最初の変数と同じ入力スタイルで記述できる場合があります。1 つの INPUT ステートメントで、3 つの入力スタイル(リスト、カラムおよびフォーマット)のいずれでも使用できます。

詳細情報

欠損データ値の処理
INFILE ステートメントの FLOWOVER、STOPOVER、MISSOVER、TRUNCOVER の各オプションに関する詳細については、“INFILE Statement” (*SAS Statements: Reference*)を参照してください。

条件のテスト
- IF ステートメントを使用した条件付き処理実行の詳細については、10 章, “選択したオブザベーションの操作” (147 ページ) および 11 章, “オブザベーションのサブセットの作成” (167 ページ)を参照してください。
- ラインポインタコントロールおよびラインホールド指定子の詳細な説明とリストについては、“PUT Statement” (*SAS Statements: Reference*)を参照してください。

6 章

SAS データセットから作成する

SAS データセットから作成する場合について **91**
目的 91
前提条件 92
基礎知識 **92**
例で使用される入力 SAS データセット **92**
選択したオブザベーションの読み込み **95**
選択した変数の読み込み **96**
選択した変数の読み込みの概要 96
選択した変数の保持 97
選択した変数の削除 98
データセットオプションまたはステートメントの選択 98
DROP=データセットオプションまたは KEEP=データセットオプションの選択 99
1 つの DATA ステップで複数のデータセットを作成する **100**
効率的に処理するため、DROP=データセットオプションと KEEP=データセットオプションを使用する **102**
要約 **103**
データセットオプション 103
プロシジャ 104
ステートメント 104
詳細情報 **105**

SAS データセットから作成する場合について

目的

このセクションでは、次の操作を行う方法を学習します。

- SAS データセットについての情報を表示
- 生データレコードからではなく既存の SAS データセットから SAS データセットを新規作成

生データを読み込むよりも、DATA ステップで SAS データセットを読み込む方が簡単です。これは SAS に対するデータの記述作業がすでに完了しているためです。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 1 章, “SAS System について” (3 ページ)
- 3 章, “DATA ステップ処理について” (27 ページ)

基礎知識

DATA ステップへの入力として SAS データセットを使用する場合、SAS ではデータセットの記述が可能です。DATA ステップでは、SET、MERGE、MODIFY、UPDATE のいずれかのステートメントを使用して、SAS データセットを読み込みます。データを処理し、出力 SAS データセットを作成するには、SAS プログラミングステートメントを使用します。

DATA ステップでは、元のデータセットのサブセットとなる新たなデータセットを作成できます。たとえば、人事データを含む大きなデータセットがある場合、条件を満たすオブザベーション(特定の日付より後に雇用された従業員のオブザベーションなど)のサブセットの参照が必要なことがあります。あるいは、すべてのオブザベーションが対象であっても、少数の変数のみ(教育年数や勤続年数など)が必要なこともあります。

既存の SAS データセットに加えて、SAS データセットから作成したサブセットを使用すると、生データを使用したり、大きなデータセットを処理したりする場合よりも、コンピュータリソースを効率的に利用できます。読み込まれる変数が少なくなると、作成されるプログラムデータベクトルが小さくなり、読み込まれるオブザベーションが少なくなると、DATA ステップの反復回数が少なくなります。また、生データを読み込むよりも、SAS データセットから直接データを読み込む方が効率的です。これはデータの記述および変換作業がすでに完了しているためです。

SAS データセットを参照する 1 つの方法は、PRINT プロシジャを使用して、SAS データセット内のデータのリストを作成することです。SAS データセットを参照するもう 1 つの方法は、データ値ではなく構造を示す情報を表示することです。データセットの構造についての情報を表示するには、DATASETS プロシジャで CONTENTS ステートメントを指定します。精通していない SAS データセットを処理する必要がある場合は、DATASETS プロシジャで CONTENT ステートメントを指定すると、データセット内のすべての変数の名前、種類、長さなどの情報が表示されます。DATASETS プロシジャでの CONTENTS ステートメントの使用例については、“例で使用される入力 SAS データセット” (92 ページ)を参照してください。

例で使用される入力 SAS データセット

このセクションの例では、CITY という SAS データセットを使用します。これには、ある小都市の支出についての情報が含まれています。ここでは、1980 年から 2000 年までの市の総支出額がレポートされ、その支出が公共サービス費と行政管理費という 2 つの大きなカテゴリに分類されています。

次の例では、DATASETS プロシジャで NOLIST オプションを指定して、CITY データセットを表示します。NOLIST オプションを指定すると、DATASETS プロシジャによって WORK ライブラリ内に他にも存在するデータセットがリスト表示されるのを防止できます。

```
data city;
   input Year 4. @7 ServicesPolice comma6.
         @15 ServicesFire comma6. @22 ServicesWater_Sewer comma6.
         @30 AdminLabor comma6. @39 AdminSupplies comma6.
         @45 AdminUtilities comma6.;
   ServicesTotal=ServicesPolice+ServicesFire+ServicesWater_Sewer;
   AdminTotal=AdminLabor+AdminSupplies+AdminUtilities;
   Total=ServicesTotal+AdminTotal;
   label              Total='Total Outlays'
              ServicesTotal='Services: Total'
             ServicesPolice='Services: Police'
               ServicesFire='Services: Fire'
         ServicesWater_Sewer='Services: Water & Sewer'
                 AdminTotal='Administration: Total'
                 AdminLabor='Administration: Labor'
              AdminSupplies='Administration: Supplies'
             AdminUtilities='Administration: Utilities';
   datalines;
1993  2,819   1,120    422   391      63     98
1994  2,477   1,160    500   172      47     70
1995  2,028   1,061    510   269      29     79
1996  2,754     893    540   227      21     67
1997  2,195     963    541   214      21     59
1998  1,877     926    535   198      16     80
1999  1,727   1,111    535   213      27     70
2000  1,532   1,220    519   195      11     69
2001  1,448   1,156    577   225      12     58
2002  1,500   1,076    606   235      19     62
2003  1,934     969    646   266      11     63
2004  2,195   1,002    643   256      24     55
2005  2,204     964    692   256      28     70
2006  2,175   1,144    735   241      19     83
2007  2,556   1,341    813   238      25     97
2008  2,026   1,380    868   226      24     97
2009  2,526   1,454    946   317      13     89
2010  2,027   1,486  1,043   226       .     82
2011  2,037   1,667  1,152   244      20     88
2012  2,852   1,834  1,318   270      23     74
2013  2,787   1,701  1,317   307      26     66
;

proc datasets library=work nolist;
   contents data=city;
run;
```

次の出力は、CITY データセットに関する情報と、そのコンテンツを表示しています。

図 6.1 パート 1: PROC DATASETS によって表示される CITY の構造

The SAS System

The DATASETS Procedure

Data Set Name	WORK.CITY	Observations	21
Member Type	DATA	Variables	10
Engine	V9	Indexes	0
Created	04/30/2013 15:29:12	Observation Length	80
Last Modified	04/30/2013 15:29:12	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

図 6.2 パート 2: PROC DATASETS によって表示される CITY の構造

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	817
Obs in First Data Page	21
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\userid\AppData\Local\Temp\SAS Temporary Files_TD1234_D56789_\city.sas7bdat
Release Created	9.0401B0
Host Created	W32_7PRO

図 6.3 パート 3: PROC DATASETS によって表示される CITY の構造

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Label
5	AdminLabor	Num	8	Administration: Labor
6	AdminSupplies	Num	8	Administration: Supplies
9	AdminTotal	Num	8	Administration: Total
7	AdminUtilities	Num	8	Administration: Utilities
3	ServicesFire	Num	8	Services: Fire
2	ServicesPolice	Num	8	Services: Police
8	ServicesTotal	Num	8	Services: Total
4	ServicesWater_Sewer	Num	8	Services: Water & Sewer
10	Total	Num	8	Total Outlays
1	Year	Num	8	

次のリストは、上に示した 3 つの SAS 出力の項目に対応しています。

1. 図 6.1 (94 ページ)の Observations と Variables のフィールドは、オブザベーションの数と変数の数を示します。
2. 図 6.2 (94 ページ)の Engine/Host Dependent Information セクションには、データセットについての詳細情報がリスト表示されています。この情報は、ファイルの読み書きのメカニズムであるエンジンによって生成されます。
3. 図 6.3 (95 ページ)の Alphabetic List of Variables and Attributes には、各変数の名前、種類、長さ、位置がリストされています。
4. 図 6.3 (95 ページ)の Label には、各変数の出力形式、入力形式およびラベルが(存在する場合は)リストされています。

動作環境の情報

Engine/Host Dependent Information セクションの出力は、動作環境に依存して異なる場合があります。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

選択したオブザベーションの読み込み

大きなデータセットの一部にのみ興味がある場合は、データセットオプションを使用して、データのサブセットを作成できます。データセットオプションでは、新しいデータセットに含めるオブザベーションが指定されます。11 章, “オブザベーションのサブセットの作成” (167 ページ)では、サブセット化 IF ステートメントを使用して、大きな SAS データセットのサブセットを作成する方法を学習します。このセクションでは、FIRSTOBS=および OBS=データセットオプションを使用して、大きなデータセットのサブセットを作成する方法を学習します。

たとえば、データセットの最初のオブザベーションの読み込みが不要だとします。FIRSTOBS=データセットオプションを使用すると、どのオブザベーションを最初に処理するかを定義できます。この例では、データセット CITY に対して、FIRSTOBS=12 を

指定し、1991 年より前のデータを含むオブザベーションを除外したデータセットを作成します。その結果、SAS では、1991 年より前のデータを含む、最初の 11 オブザベーションは読み込まれません。(CITY データセットの作成プログラムについては、“CITY データセットを作成するための DATA ステップ” (792 ページ)を参照してください。)

次のプログラムで、データセット CITY2 が作成されます。これには、同数の変数が含まれますが、オブザベーション数は CITY よりも少なくなります。

```
data city2;
   set city(firstobs=12);
run;

proc print;
   title 'City Expenditures';
   title2 '1991 - 2000';
run;
```

次の出力は結果を示しています。

図 6.4 *オブザベーションによるデータセットのサブセット化*

City Expenditures
2004 - 2013

Obs	Year	ServicesPolice	ServicesFire	ServicesWater_Sewer	AdminLabor	AdminSupplies	AdminUtilities	ServicesTotal	AdminTotal	Total
1	2004	2195	1002	643	256	24	55	3840	335	4175
2	2005	2204	964	692	256	28	70	3860	354	4214
3	2006	2175	1144	735	241	19	83	4054	343	4397
4	2007	2556	1341	813	238	25	97	4710	360	5070
5	2008	2026	1380	868	226	24	97	4274	347	4621
6	2009	2526	1454	946	317	13	89	4926	419	5345
7	2010	2027	1486	1043	226	.	82	4556	.	.
8	2011	2037	1667	1152	244	20	88	4856	352	5208
9	2012	2852	1834	1318	270	23	74	6004	367	6371
10	2013	2787	1701	1317	307	26	66	5805	399	6204

OBS=データセットオプションでは、新しいデータセットに含める最後のオブザベーションも指定できます。たとえば、次のプログラムでは、1989 年(10 番目のオブザベーション)から 1994 年(15 番目のオブザベーション)までのオブザベーションのみを含む SAS データセットが作成されます。

```
data city3;
   set city (firstobs=10 obs=15);
run;
```

選択した変数の読み込み

選択した変数の読み込みの概要

大きなデータセットのサブセットを作成する際には、オブザベーションを除外するだけでなく、新しいデータセットに含める変数を指定することもできます。新しいデータセットに含める変数を指定して、大きなデータセットからサブセットを作成するには、DATA ステップで、SET ステートメントと KEEP=もしくは DROP=データセットオプション(または DROP および KEEP ステートメント)を使用します。

選択した変数の保持

この例では、SET ステートメントで KEEP=データセットオプションを使用して、データセット CITY の公共サービス関連の支出を表す変数のみを読み込みます。

```
data services;
   set city (keep=Year ServicesTotal ServicesPolice ServicesFire
             ServicesWater_Sewer);
run;

proc print data=services;
   title 'City Services-Related Expenditures';
run;
```

次の出力は、処理結果のデータセットを示しています。データセット SERVICES には、KEEP=オプションで指定する変数のみが含まれることに注意してください。

図 6.5 KEEP=オプションでの変数選択

City Services-Related Expenditures

Obs	Year	ServicesPolice	ServicesFire	ServicesWater_Sewer	ServicesTotal
1	2004	2195	1002	643	3840
2	2005	2204	964	692	3860
3	2006	2175	1144	735	4054
4	2007	2556	1341	813	4710
5	2008	2026	1380	868	4274
6	2009	2526	1454	946	4926
7	2010	2027	1486	1043	4556
8	2011	2037	1667	1152	4856
9	2012	2852	1834	1318	6004
10	2013	2787	1701	1317	5805

次の例では、KEEP=データセットオプションのかわりに KEEP ステートメントを使用して、CITY データセットから変数をすべて読み込みます。KEEP ステートメントでは、KEEP ステートメントに記述した変数のみを含むデータセット(SERVICES)が新規作成されます。次のプログラムでは、前述の例と同じ結果が出ます。

```
data services;
   set city;
   keep Year ServicesTotal ServicesPolice ServicesFire
        ServicesWater_Sewer;
run;
```

次の例では、DATA ステートメントで KEEP=データセットオプションを使用して、同じ結果を出しています。変数はすべてプログラムデータベクトルに読み込まれますが、指定した変数のみが SERVICES データセットに書き込まれます。

```
data services (keep=Year ServicesTotal ServicesPolice ServicesFire
               ServicesWater_Sewer);
   set city;
run;
```

選択した変数の削除

大きなデータセットのサブセットを作成する際に、そこに含める変数ではなく除外する変数を指定する場合は、DROP=オプションを使用します。次の DATA ステップでは、データセット CITY から、DROP=オプションで指定した以外の変数がすべて読み込まれます。そして、SERVICES2 という名前のデータセットが作成されます。

```
data services2;
   set city (drop=Total AdminTotal AdminLabor AdminSupplies
             AdminUtilities);
run;

proc print data=services2;
   title 'City Services-Related Expenditures';
run;
```

次の出力は、処理結果のデータセットを示しています。

図 6.6 DROP=オプションでの変数除外

City Services-Related Expenditures

Obs	Year	ServicesPolice	ServicesFire	ServicesWater_Sewer	ServicesTotal
1	2004	2195	1002	643	3840
2	2005	2204	964	692	3860
3	2006	2175	1144	735	4054
4	2007	2556	1341	813	4710
5	2008	2026	1380	868	4274
6	2009	2526	1454	946	4926
7	2010	2027	1486	1043	4556
8	2011	2037	1667	1152	4856
9	2012	2852	1834	1318	6004
10	2013	2787	1701	1317	5805

次の例では、DROP=データセットオプションのかわりに DROP ステートメントを使用して、CITY データセットから変数をすべて読み込みます。さらに、DROP ステートメントに記述された変数を、新しいデータセットへの書き込み対象から除外します。結果は前述の例と同じになります。

```
data services2;
   set city;
   drop Total AdminTotal AdminLabor AdminSupplies AdminUtilities;
run;
proc print data=services2;
run;
```

データセットオプションまたはステートメントの選択

DATA ステップで作成するデータセットが 1 つのみの場合、変数を削除/保持するデータセットオプションは、変数を削除/保持するステートメントと同じ影響を出力データセットに与えます。プログラムデータベクトルにどの変数を読み込むかを制御する場合、SAS データセットを読み込むステートメント(SET ステートメントなど)でデータセットオプションを使用します。一般的に、オプションを使用する方が、ステートメントを使用する

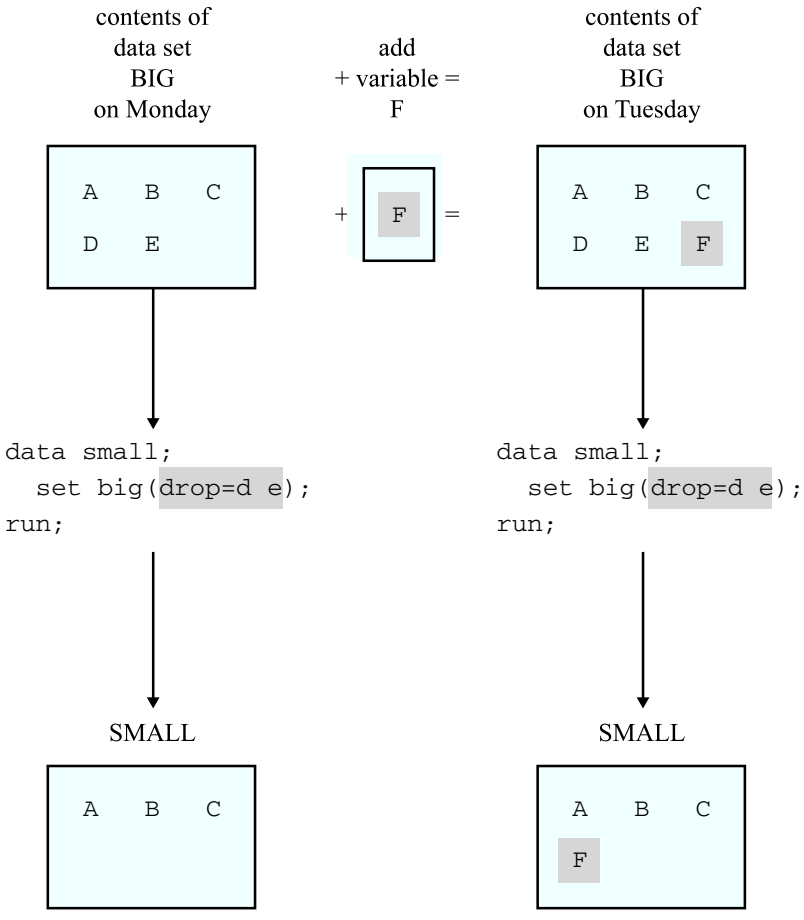
よりも効率的です。場合によってはステートメントが機能しないことがあるので、このセクションに後述するトピックでは、その場合のデータセットオプションの使用方法を示します。

DROP=データセットオプションまたは KEEP=データセットオプションの選択

簡単な例では、指定する変数の数が少なくてすむのはどの方法かによって、DROP=または KEEP=オプションの使用を決定します。データセットを読み込む大きなジョブを処理する際、バッチジョブ実行の合間に変数の追加が予想される場合は、KEEP=オプションを使用して、サブセットデータセットに含める変数を指定します。

次の図は、SMALL という名前の 2 つのデータセットを示しています。この 2 つはコンテンツが異なります。これは火曜日に DATA ステップが実行される前に、新しい変数 F がデータセット BIG に追加されたためです。DATA ステップでは、DROP=オプションが使用され、変数 D および E の出力データセットへの書き込みが防止されます。その結果、データセットには異なるコンテンツが含まれます。2 番目の SMALL データセットには追加の変数 F が存在します。DATA ステップで KEEP=オプションを使用して A、B および C を指定すると、両方の SMALL データセットに同じ変数(A、B および C)が含まれます。元のデータセット BIG に変数 F を追加しても、SMALL データセットの作成に影響は及びません。

図 6.7 DROP=オプションの使用



1 つの DATA ステップで複数のデータセットを作成する

1 つの DATA ステップで、一度に複数のデータセットを作成できます。KEEP=データセットオプションまたは DROP=データセットオプションを使用すると、コンテンツの異なるデータセットが作成される可能性があります。たとえば、次の DATA ステップでは、2 つの SAS データセットが作成されます。SERVICES には公共サービス関連の支出を示す変数、ADMIN には行政管理関連の支出を表す変数が含まれます。DATA ステートメントで各データセット名の後に KEEP=オプションを使用すると、作成される SAS データセットのそれぞれにどの変数を書き込むかが決定されます。

```
data services(keep=ServicesTotal ServicesPolice ServicesFire
              ServicesWater_Sewer)
     admin(keep=AdminTotal AdminLabor AdminSupplies
           AdminUtilities);
   set city;
run;

proc print data=services;
   title 'City Expenditures: Services';
run;

proc print data=admin;
   title 'City Expenditures: Administration';
run;
```

次の 2 つの出力は両方のデータセットを示しています。各データセットには、DATA ステートメントでそれぞれの名前の後に KEEP=オプションによって指定した変数のみが含まれていることに注意してください。

図 6.8 *1 つの DATA ステップで 2 つのデータセットを作成: サービス*

City Expenditures: Services

Obs	ServicesPolice	ServicesFire	ServicesWater_Sewer	ServicesTotal
1	2819	1120	422	4361
2	2477	1160	500	4137
3	2028	1061	510	3599
4	2754	893	540	4187
5	2195	963	541	3699
6	1877	926	535	3338
7	1727	1111	535	3373
8	1532	1220	519	3271
9	1448	1156	577	3181
10	1500	1076	606	3182
11	1934	969	646	3549
12	2195	1002	643	3840
13	2204	964	692	3860
14	2175	1144	735	4054
15	2556	1341	813	4710
16	2026	1380	868	4274
17	2526	1454	946	4926
18	2027	1486	1043	4556
19	2037	1667	1152	4856
20	2852	1834	1318	6004
21	2787	1701	1317	5805

図 6.9 1 つの DATA ステップで 2 つのデータセットを作成: 管理

City Expenditures: Administration

Obs	AdminLabor	AdminSupplies	AdminUtilities	AdminTotal
1	391	63	98	552
2	172	47	70	289
3	269	29	79	377
4	227	21	67	315
5	214	21	59	294
6	198	16	80	294
7	213	27	70	310
8	195	11	69	275
9	225	12	58	295
10	235	19	62	316
11	266	11	63	340
12	256	24	55	335
13	256	28	70	354
14	241	19	83	343
15	238	25	97	360
16	226	24	97	347
17	317	13	89	419
18	226	.	82	.
19	244	20	88	352
20	270	23	74	367
21	307	26	66	399

注: この場合、KEEP=データセットオプションを使用する必要があります。これは、KEEP ステートメントを使用すると、DATA ステップで作成されるすべてのデータセットに同じ変数を含められるためです。

効率的に処理するため、DROP=データセットオプションと KEEP=データセットオプションを使用する

DROP=データセットオプションと KEEP=データセットオプションは、DATA ステートメントと SET ステートメントのどちらにおいても有効です。ただし、SET ステートメントではなく DATA ステートメントにおけるこれらのオプションの使用結果を理解しておけば、より効率的な DATA ステップを作成できます。

DATA ステートメントでは、これらのオプションは、プログラムデータベクトルから処理結果の SAS データセットにどの変数を書き込むかに影響を与えます。SET ステートメントでは、これらのオプションは、入力 SAS データセットからどの変数を読み込むかを

決定します。したがって、プログラムデータベクトルをどのように作成するかが決定されます。

SET ステートメントに DROP=または KEEP=オプションを指定した場合、除外された変数はプログラムデータベクトルに読み込まれません。大きなデータセット(数千や数百万のオブザベーションを含むものなど)を処理する場合は、入力データセットから不必要な変数を読み込まない方が、より効率的な DATA ステップを作成できます。

入力データセットからの変数を使用して計算を実行する場合は、その変数をプログラムデータベクトルに読み込む必要があることにも注意してください。ただし、その変数を新しいデータセットに表示する必要がない場合は、DATA ステートメントで DROP=オプションを使用して除外します。

次の DATA ステップでは、前述の例の DATA ステップと同じ 2 つのデータセットが作成されます。変数 Total はプログラムデータベクトルに読み込まれません。この SET ステートメントと“1 つの DATA ステップで複数のデータセットを作成する” (100 ページ)の SET ステートメントを比較してみます。

```
data services (keep=ServicesTotal ServicesPolice ServicesFire
               ServicesWater_Sewer)
     admin (keep=AdminTotal AdminLabor AdminSupplies
            AdminUtilities);
   set city(drop=Total);
run;

proc print data=services;
   title 'City Expenditures: Services';
run;

proc print data=admin;
   title 'City Expenditures: Administration';
run;
```

前述の例とは対照的に、この例のデータセットオプションは、DATA ステートメントと SET ステートメントの両方に記述されています。SET ステートメントの DROP=オプションでは、プログラムデータベクトルからどの変数を省略するかが決定されます。DATA ステートメントの KEEP=オプションでは、作成される各データセットにプログラムデータベクトルからどの変数を書き込むかが制御されます。

注: DROP ステートメントや KEEP ステートメントの使用法は、DATA ステートメントの DROP=オプションや KEEP=オプションの使用法に似ています。すべての変数は、プログラムデータベクトルに含められますが、オブザベーションがプログラムデータベクトルから新しいデータセットに書き込まれる際に除外されます。1 つの DATA ステップで複数のデータセットを作成する場合は、データセットオプションを使用すると、新しいデータセットのそれぞれで異なる変数を削除または保持できます。一方、DROP ステートメントや KEEP ステートメントでは、作成されるデータセットすべてに影響が及びます。

要約

データセットオプション

DROP=*variable(s)*
除外する変数を指定します。

SET ステートメントで DROP=を使用すると、既存の SAS データセットからプログラムデータベクトルへ読み込まれない変数が指定されます。DATA ステートメントで DROP=を使用すると、作成されるデータセットから除外する変数が指定されます。

FIRSTOBS=*n*
SET ステートメントで指定した SAS データセットから読み込む最初のオブザベーションを指定します。

KEEP=*variable(s)*
含める変数を指定します。

SET ステートメントで KEEP=を使用すると、既存の SAS データセットからプログラムデータベクトルに読み込む変数が指定されます。DATA ステートメントで KEEP=を使用すると、プログラムデータベクトル内のどの変数を、作成されるデータセットに書き込むかが指定されます。

OBS=*n*
SET ステートメントで指定した SAS データセットから読み込む最後のオブザベーションを指定します。

プロシジャ

PROC DATASETS <LIBRARY=*SAS-data-library*>;CONTENTS <DATA=*SAS-data set*>;
データセット内のすべての変数の名前、種類および長さを含む、SAS データセットの構造を記述します。

ステートメント

DATA *SAS-data-set*<(*data-set-options*)>;
DATA ステップを開始し、作成される SAS データセットの名前を指定します。各データセット名の後に、かっこで囲んだ DROP=または KEEP=データセットオプションを指定すると、どの変数をプログラムデータベクトルから出力データセットに書き込むかを制御できます。

DROP *variable(s)*;
作成されるデータセットから除外する変数を指定します。詳細については、“DROP Statement” (*SAS Statements: Reference*)を参照してください。

KEEP *variable(s)*
作成されるデータセットに書き込まれる変数を指定します。詳細については、“KEEP Statement” (*SAS Statements: Reference*)を参照してください。

SET SAS-*data-set*(*data-set-options*);
生データレコードからではなく SAS データセットからオブザベーションを読み込みます。データセット名の後に、かっこで囲んだ DROP=または KEEP=データセットオプションを指定すると、どの変数を入力データセットからプログラムデータベクトルに読み込むかを制御できます。

詳細情報

SAS データセットの作成
: マージ、連結、インタリーブ、更新による他の SAS データセットからの SAS データセット作成の全般的な説明については、16 章, “SAS データセットの結合方法” (245 ページ)を参照してください。

データセットオプション
: *SAS データセットオプション: リファレンス*の“Data Set Options”セクション、および使用する動作環境に関する SAS ドキュメントを参照してください。

DROP ステートメントと KEEP ステートメント
: 詳細については、“DROP Statement” (*SAS Statements: Reference*)および “KEEP Statement” (*SAS Statements: Reference*)を参照してください。

エンジン
: *SAS Language Reference: Concepts*。

サブセット化 IF ステートメント
: 既存の SAS データセットから新しい SAS データセットを作成する場合、サブセット化 IF ステートメントと条件付き(IF-THEN)ロジックを使用できます。詳細については、10 章, “選択したオブザベーションの操作” (147 ページ)および 11 章, “オブザベーションのサブセットの作成” (167 ページ)を参照してください。

3 部

基本プログラミング

7 章
DATA ステップ処理の基礎知識 109

8 章
数値変数の操作 119

9 章
文字変数の操作 131

10 章
選択したオブザベーションの操作 147

11 章
オブザベーションのサブセットの作成 167

12 章
グループ化されたオブザベーションや並べ替えられたオブザベーションの操作 181

13 章
計算に複数のオブザベーションを使用する 197

14 章
プログラミングを簡単にする 213

15 章
SAS System での日付の操作 223

7 章

DATA ステップ処理の基礎知識

DATA ステップ処理の概要 **109**
目的 109
前提条件 109
例で使用される入力 SAS データセット **110**
SAS データセットへの情報の追加 **111**
割り当てステートメントについて 111
変数の作成によるデータの一律変更 111
一部のオブザベーションに情報を追加し他のオブザベーションは対象外とする 112
変数作成なしのデータの一律変更 113
変数の効率的な使用 114
十分な容量の変数の記憶域を定義する **116**
オブザベーションの条件付き削除 **117**
要約 **117**
ステートメント 117
詳細情報 **118**

DATA ステップ処理の概要

目的

SAS データセット内の情報を追加、変更、削除するには、DATA ステップを使用します。このセクションでは、DATA ステップの機能、ステートメントの一般化形式、およびプログラミング技術をいくつか学習します。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 3 章, “DATA ステップ処理について” (27 ページ)
- 4 章, “生データから作成する: 基本” (51 ページ)

例で使用される入力 SAS データセット

Tradewinds Travel 社には、ツアーに関するデータを操作、格納するための外部ファイルがあります。外部ファイルには次の情報が含まれています。

```
1       2   3   4   5
---------------------------
France  8   793 575 Major
Spain  10   805 510 Hispania
India  10     . 489 Royal
Peru    7   722 590 Mundial
```

前述の例の列には、次の情報が含まれています。

1 ツアー先の国名

2 ツアーの宿泊日数

3 米ドルでの航空運賃

4 米ドルでの地上パッケージのコスト

5 ツアー提供会社の名前

India へのツアーの航空運賃のコストが欠損値で、ピリオドで表されていることに注意してください。

次の DATA ステップでは、MYLIB.INTERNATIONALTOURS という永久 SAS データセットが作成されます。

```
libname mylib 'permanent-data-library';

data mylib.internationaltours;
    infile 'input-file';
    input Country $ Nights AirCost LandCost Vendor $;

proc print data = mylib.internationaltours;
   title 'Data Set MYLIB.INTERNATIONALTOURS';
run;
```

DATA ステップの後の PROC PRINT ステートメントによって、MYLIB.INTERNATIONALTOURS データセットが次のように表示されます。

図 7.1 永久 SAS データセットの作成

Data Set MYLIB.INTERNATIONALTOURS

Obs	Country	Nights	AirCost	LandCost	Vendor
1	France	8	793	575	Major
2	Spain	10	805	510	Hispania
3	India	10	.	489	Royal
4	Peru	7	722	590	Mundial

SAS データセットへの情報の追加

割り当てステートメントについて

DATA ステップでプログラムステートメントを使用する最も一般的な理由の 1 つは、元の情報から新しい情報を生成したり、INPUT または SET/MERGE/MODIFY/UPDATE ステートメントで読み込んだ情報を変更したりするためです。DATA ステップでオブザベーションに情報を追加するにはどのようにすればよいでしょうか。

SAS データセットに情報を追加する基本の手法は、DATA ステップで割り当てステートメントを使用して変数を新規作成することです。割り当てステートメントの形式は次のとおりです。

variable=expression;

変数で新しい情報を受け取り、式で新しい情報を作成します。情報の生成に必要な計算を指定し、その計算を式として記述します。式に文字データが含まれている場合、そのデータを引用符で囲む必要があります。式が評価され、指定した変数に新しい情報が格納されます。多くのオブザベーションの内の 1 つか 2 つにのみ情報を追加する必要がある場合、その変数がすべてのオブザベーションに対して作成されることを覚えておくことが重要です。作成される SAS データセットには、すべてのオブザベーションとすべての変数の情報が含まれます。

変数の作成によるデータの一律変更

場合によっては、すべてのオブザベーションにある特定の変更を加えることもあります。たとえば、Tradewinds Travel で、新しい税金のために、すべてのツアーの航空運賃を$10 ずつ値上げする必要があるとします。そのための 1 つの方法は、新しい航空運賃を計算する新しい変数を作成する割り当てステートメントを記述することです。

```
NewAirCost = AirCost+10;
```

このステートメントでは、AirCost の値を読み込み、そこに 10 を加算し、その結果を新しい変数 NewAirCost に割り当てるよう指定されます。

この割り当てステートメントが DATA ステップに含まれている場合、DATA ステップは次のようになります。

```
data newair;
   set mylib.internationaltours;
   NewAirCost = AirCost + 10;

proc print data=newair;
   var Country AirCost NewAirCost;
   title 'Increasing the Air Fare by $10 for All Tours';
run;
```

注: この例では、PROC PRINT ステップの VAR ステートメントによって、出力にどの変数を表示するかが決定されます。

次の出力は、処理結果の SAS データセット NEWAIR を示しています。

図 7.2 新しい変数を使用してすべてのオブザベーションに情報を追加

Increasing the Air Fare by $10 for All Tours

Obs	Country	AirCost	NewAirCost	
1	France	793	803	❶
2	Spain	805	815	
3	India	.	.	❷
4	Peru	722	732	

このデータセットでは次の点に注意してください。

1. DATA ステップの各ステートメントはオブザベーションごとに実行されるため、NewAirCost は DATA ステップが反復するたびに計算されます。
2. India のオブザベーションでは AirCost に欠損値が含まれているので、そのオブザベーションの NewAirCost には欠損値が割り当てられます。

SAS データセットには、すべてのオブザベーションとすべての変数の情報が含まれます。

一部のオブザベーションに情報を追加し他のオブザベーションは対象外とする

一部のオブザベーションにのみ情報追加が必要でその他のオブザベーションには追加不要な場合が多くあります。たとえば、一部の旅行業者が、特定ツアーのスケジューリングに対して旅行代理店にボーナスポイントを与えるとします。今年は Hispania と Mundial の 2 社がボーナスポイントを提供します。

IF-THEN/ELSE ステートメントでは、条件が満たされた場合のみ、割り当てステートメントが実行されます。次の DATA ステップでは、IF ステートメントで変数 Vendor の値がチェックされます。値が Hispania か Mundial のどちらかであれば、ボーナスポイントについての情報がそのオブザベーションに追加されます。

```
data bonus;
   set mylib.internationaltours;
   if Vendor = 'Hispania' then BonusPoints = 'For 10+ people';
   else if Vendor = 'Mundial' then BonusPoints = 'Yes';
run;

proc print data=bonus;
   var Country Vendor BonusPoints;
   title1 'Adding Information to Observations for';
   title2 'Vendors Who Award Bonus Points';
run;
```

次の出力は結果を示しています。

図 7.3 新しい変数を使用して特定のオブザベーションの値を指定

Adding Information to Observations for Vendors Who Award Bonus Points

Obs	Country	Vendor	BonusPoints
1	France	Major	❶
2	Spain	Hispania	For 10+ people ❷
3	India	Royal	❶
4	Peru	Mundial	Yes

新しい変数 BonusPoints には次の情報が含まれます。

1. BonusPoints の値を割り当てられていない 2 つのオブザベーションでは、欠損値が割り当てられます。この場合、欠損値はブランクで表され、文字値が存在しないことが示されます。
2. BonusPoints として検出される最初の値には 14 文字が含まれます。これにより、各オブザベーションで、そのオブザベーションの値の長さには関係なく、BonusPoints 用に 14 バイトの記憶域が確保されます。

変数作成なしのデータの一律変更

場合によっては、新しい変数を追加せずに、既存の変数の値を変更することがあります。たとえば、ある DATA ステップで、新しい変数 NewAirCost が作成され、航空運賃に新しい$10 の税金を加算した値がそこに入れられたとします。

```
NewAirCost = AirCost + 10;
```

新しい変数を作成するのではなく、既存の変数の値を変更するように設定することもできます。例に従って、AirCost は次のように変更されます。

```
AirCost = AirCost + 10;
```

このステートメントは、他の割り当てステートメントと同様に処理されます。等号の右辺の式が評価され、等号の左辺の変数に結果が割り当てられます。等号の右辺と左辺に同じ変数が記述されていても問題はありません。左辺の変数を考慮する前に、等号の右辺の式が評価されます。

次のプログラムには、新しい割り当てステートメントが含まれています。

```
data newair2;
   set mylib.internationaltours;
   AirCost = AirCost + 10;

proc print data=newair2;
   var Country AirCost;
   title 'Adding Tax to the Air Cost Without Adding a New Variable';
run;
```

次の出力は結果を示しています。

図 7.4 変数の情報変更

Adding Tax to the Air Cost Without Adding a New Variable

Obs	Country	AirCost
1	France	803
2	Spain	815
3	India	.
4	Peru	732

変数に含まれる情報の種類を変更すると、変数の意味も変更されます。この場合、AirCost の意味が *airfare without tax*(税抜き航空運賃)から *airfare with tax*(税込み航空運賃)に変更されます。現在の意味を忘れず、なおかつ元の情報が不要だと判明している場合、変数の値変更は有益です。ただし、多くのプログラマにとっては、定義を変更した 1 つの変数を記憶するよりも、変数を別々にする方が容易です。

変数の効率的な使用

1 つか 2 つのオブザベーションにのみ適用される情報を含む変数は、記憶域を必要以上に使用します。可能な場合は、各変数をより多くのオブザベーションに適用してデータセットに作成する変数を減らし、異なるオブザベーションでは異なる値で情報を提供できるようにします。

たとえば、Major 社が、ボーナスポイントではなく、30 人以上のグループに対する割引を提供するとします。非効率的なプログラムでは、次のように、ボーナスポイントと割引に対して別々の変数が作成されます。

```
   /* inefficient use of variables */
options pagesize=60 linesize=80 pageno=1 nodate;
data tourinfo;
   set mylib.internationaltours;
   if Vendor = 'Hispania' then BonusPoints = 'For 10+ people';
   else if Vendor = 'Mundial' then BonusPoints = 'Yes';
        else if Vendor = 'Major' then Discount = 'For 30+ people';
run;

proc print data=tourinfo;
   var Country Vendor BonusPoints Discount;
   title 'Information About Vendors';
run;
```

次の出力は結果を示しています。

図 7.5 非効率的: 複数の変数に情報が散在する変数の使用

Information About Vendors

Obs	Country	Vendor	BonusPoints	Discount
1	France	Major		For 30+ people
2	Spain	Hispania	For 10+ people	
3	India	Royal		
4	Peru	Mundial	Yes	

このように、ここでは記憶域の使用法が非効率的です。BonusPoints と Discount の両方で、非常に多数の欠損値が発生します。

ほんの少し計画性があれば、SAS データセットはずっと効率的になります。次の DATA ステップでは、変数 Remarks に、ボーナスポイント、割引、およびツアーに関するその他の特典が含まれます。

```
   /* efficient use of variables */
data newinfo;
   set mylib.internationaltours;
   if Vendor = 'Hispania' then Remarks = 'Bonus for 10+ people';
   else if Vendor = 'Mundial' then Remarks = 'Bonus points';
        else if Vendor = 'Major' then Remarks = 'Discount: 30+ people';
run;

proc print data=newinfo;
   var Country Vendor Remarks;
   title 'Information About Vendors';
run;
```

次の出力は、より効率的な変数使用法を示しています。

図 7.6 効率的: 変数を使用して最大限の情報を含める

Information About Vendors

Obs	Country	Vendor	Remarks
1	France	Major	Discount: 30+ people
2	Spain	Hispania	Bonus for 10+ people
3	India	Royal	
4	Peru	Mundial	Bonus points

Remarks の欠損値が減少し、非効率的な例において BonusPoints と Discount で使用された情報がすべて含まれます。変数を効率的に使用すると、記憶域を節約し、SAS データセットを最適化することができます。

十分な容量の変数の記憶域を定義する

値が初めて変数に割り当てられる際、変数に割り当てられる最初の値の文字と同じバイト数の変数記憶域を使用できます。時には、変数が必要とする記憶域容量の指定が必要な場合もあります。

たとえば、前述の例に示したように、変数 Remarks にツアーに関するその他の情報を含めるとします。

```
if Vendor = 'Hispania' then Remarks = 'Bonus for 10+ people';
```

この割り当てステートメントでは、Remarks に割り当てられた最初の値が 20 文字なので、Remarks に対して 20 バイトの記憶域を使用できます。最初に割り当てられる値が最長の値ではない場合もあるので、最初の値が割り当てられる前に、変数に対してより適切な長さを指定します。

```
length Remarks $ 30;
```

このステートメントは LENGTH ステートメントと呼ばれ、データセット全体に適用されます。これにより、すべてのオブザベーション内の変数 Remarks に対して使用する記憶域のバイト数が定義されます。LENGTH ステートメントは、個々のオブザベーションに対するステートメント処理時ではなく、コンパイル中に使用されます。次の DATA ステップは、LENGTH ステートメントの使用法を示しています。

```
data newlength;
   set mylib.internationaltours;
   length Remarks $ 30;
   if Vendor = 'Hispania' then Remarks = 'Bonus for 10+ people';
   else if Vendor = 'Mundial' then Remarks = 'Bonus points';
        else if Vendor = 'Major' then Remarks = 'Discount for 30+ people';
run;

proc print data=newlength;
   var Country Vendor Remarks;
   title 'Information About Vendors';
run;
```

次の出力には、NEWLENGTH データセットが表示されています。

図 7.7 LENGTH ステートメントの使用

Information About Vendors

Obs	Country	Vendor	Remarks
1	France	Major	Discount for 30+ people
2	Spain	Hispania	Bonus for 10+ people
3	India	Royal	
4	Peru	Mundial	Bonus points

LENGTH ステートメントは、印刷出力の列のスペース調整ではなく、変数記憶域に影響を与えるので、Remarks 変数は、図 7.7 (116 ページ)でも 図 7.6 (115 ページ)でも同

じように見えます。DATASETS プロシジャを使用して変数記憶域への LENGTH ステートメントの影響を表示するには、37 章, “SAS データセットの情報の表示” (681 ページ)を参照してください。

オブザベーションの条件付き削除

条件に基づいて、プログラムデータベクトルのデータセットへの書き込みを中止する場合は、DATA ステップで DELETE ステートメントを使用します。たとえば、Peru へのツアーが中止になった場合、作成されるデータセットに Peru のオブザベーションを含める必要がなくなります。次の例では、DELETE ステートメントを使用して、そのオブザベーションの出力データセットへの書き込みを防ぎます。

```
data subset;
   set mylib.internationaltours;
   if Country = 'Peru' then delete;
run;

proc print data=subset;
   title 'Omitting a Discontinued Tour';
run;
```

次の出力は結果を示しています。

図 7.8 オブザベーションの削除

Omitting a Discontinued Tour

Obs	Country	Nights	AirCost	LandCost	Vendor
1	France	8	793	575	Major
2	Spain	10	805	510	Hispania
3	India	10	.	489	Royal

Peru のオブザベーションがデータセットから削除されました。

要約

ステートメント

DELETE;
特定オブザベーションの出力データセットへの書き込みが防止されます。通常は IF-THEN/ELSE ステートメントの一部として記述されます。

IF *condition* THEN *action* ELSE *action*;
条件が真かどうかが検証されます。条件が真の場合、THEN ステートメントでアクションの実行が指定されます。条件が偽の場合、ELSE ステートメントで代替アクションが提供されます。アクションには、割り当てステートメントを含む、1 つ以上のステートメントを指定できます。

LENGTH *variable* <$> *length*;
変数に対して記憶域のバイト数(長さ)を割り当てます。変数が文字の場合はドル記号($)が含まれます。LENGTH ステートメントは、変数を最初に使用する前に記述しておく必要があります。

variable = *expression*
割り当てステートメントです。これにより、等号の右辺の *expression* が評価され、左辺の *variable* に結果が割り当てられます。変数の名前を選択し、その値の計算に適した式を作成する必要があります。左辺の変数に結果を割り当てる前に右辺が評価されるので、等号の左辺と右辺に同じ変数名を記述できます。

詳細情報

文字変数
数字に加えてアルファベットや特殊文字を含む式の詳細については、9 章, “文字変数の操作” (131 ページ)を参照してください。

DATA ステップ
全般的な DATA ステップの情報については、3 章, “DATA ステップ処理について” (27 ページ)を参照してください。DATA ステップの詳細については、*SAS Language Reference: Concepts* の“DATA Step Concepts”セクションで確認できます。

IF-THEN/ELSE ステートメント
IF-THEN/ELSE ステートメントについては、10 章, “選択したオブザベーションの操作” (147 ページ)で説明します。

LENGTH ステートメント
LENGTH ステートメントに関する追加情報については、8 章, “数値変数の操作” (119 ページ)および 9 章, “文字変数の操作” (131 ページ)で確認できます。DATASETS プロシジャを使用して変数記憶域への LENGTH ステートメントの影響を表示するには、37 章, “SAS データセットの情報の表示” (681 ページ)を参照してください。

欠損値
欠損値の詳細については、8 章, “数値変数の操作” (119 ページ)および 9 章, “文字変数の操作” (131 ページ)を参照してください。

数値変数
数値変数と式の操作については、8 章, “数値変数の操作” (119 ページ)で確認できます。

SAS ステートメント
IF-THEN/ELSE、LENGTH、DELETE、割り当てステートメント、コメントステートメントの完全な参照情報については、*SAS Statements: Reference* を参照してください。

8 章

数値変数の操作

数値変数の操作について **119**
目的 119
前提条件 119
SAS の数値変数について **120**
例で使用される入力 SAS データセット **120**
数値変数を使用した計算 **121**
割り当てステートメントでの算術演算子の使用 121
数値式と割り当てステートメントについて 123
SAS での欠損値の処理方法について 123
SAS 関数を使用した数の計算 124
数値変数の比較 **125**
数値変数を効率的に格納する **127**
要約 **128**
関数 128
ステートメント 129
詳細情報 **129**

数値変数の操作について

目的

このセクションでは、次の概念を説明しています。

- 算術演算子ならびに SAS 関数 ROUND および SUM を使用して SAS で算術計算を実行する方法
- 論理演算子を使用して数値変数を比較する方法
- ディスク領域が限られている場合に数値変数を効率的に格納する方法

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 第 1 部、“SAS System について”
- 第 2 部、“データの加工”
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)

SAS の数値変数について

数値変数は、値が数字の変数です。

注: 計算には倍精度浮動小数点表記が使用され、デフォルトで、数値変数が SAS データセットに格納されます。

SAS では、指数表示や 16 進法など、多くの形式の数字が受け入れられます。詳細については、*SAS Language Reference: Concepts* で、データ行から読み込み可能な数字の種類についての説明を参照してください。このドキュメントでは、説明をわかりやすくするために、次に示す標準表記の数字に重点を置きます。

```
1254
 336.05
-243
```

SAS では、算術処理を全種類実行できます。DATA ステップで計算を実行するには、算術演算子または SAS 関数、あるいはその 2 つの組み合わせが式に含まれる割り当てステートメントを記述します。数値変数を比較するには、論理演算子を使用して IF-THEN/ELSE ステートメントを記述します。数値関数の詳細については、*SAS Functions and CALL Routines: Reference* の“Functions and CALL Routines”を参照してください。

例で使用される入力 SAS データセット

Tradewinds Travel 社には、最も人気があるツアーについての情報が含まれた外部ファイルがあります。

```
1             2  3   4    5
-----------------------------------
Japan         8  982 1020 Express
Greece       12   .   748 Express
New Zealand  16 1368 1539 Southsea
Ireland       7  787  628 Express
Venezuela     9  426  505 Mundial
Italy         8  852  598 Express
Russia       14 1106 1024 A-B-C
Switzerland   9  816  834 Tour2000
Australia    12 1299 1169 Southsea
Brazil        8  682  610 Almeida
```

番号付きフィールドは次の内容を表しています。

1 ツアー先の国名

2 ツアーの宿泊日数

3 米ドルでの航空運賃

4 米ドルでの地上パッケージのコスト

5 ツアー提供会社の名前

次のプログラムでは、MYLIB.POPULARTOURS という永久 SAS データセットが作成されます。

```
libname mylib 'permanent-data-library';

data mylib.populartours;
   infile 'input-file';
   input Country $ 1-11 Nights AirCost LandCost Vendor $;
run;

proc print data=mylib.populartours;
   title 'Data Set MYLIB.POPULARTOURS';
run;
```

次の出力には、データセットが表示されています。

図 8.1 *データセット MYLIB.POPULARTOURS*

Data Set MYLIB.POPULARTOURS

Obs	Country	Nights	AirCost	LandCost	Vendor
1	Japan	8	982	1020	Express
2	Greece	12	.	748	Express
3	New Zealand	16	1368	1539	Southsea
4	Ireland	7	787	628	Express
5	Venezuela	9	426	505	Mundial
6	Italy	8	852	598	Express
7	Russia	14	1106	1024	A-B-C
8	Switzerland	9	816	834	Tour2000
9	Australia	12	1299	1169	Southsea
10	Brazil	8	682	610	Almeida

MYLIB.POPULARTOURS の変数 Nights、AirCost および LandCost は数字を含み、数値変数として格納されます。それとは対照的に、変数 Country および Vendor には数字に加えてアルファベットや特殊文字も含まれ、それが文字変数として格納されます。

数値変数を使用した計算

割り当てステートメントでの算術演算子の使用

数値変数の計算を実行する 1 つの方法は、算術演算子を使用して割り当てステートメントを記述することです。算術演算子とは、加算、減算、乗算、除算、累乗を示します。

算術式の詳細については、*SAS Language Reference: Concepts* の説明を参照してください。

次の表は、算術式で使用できる演算子を示しています。

表 8.1 *算術式の演算子*

演算	記号	例
加算	+	x = y + z;
減算	–	x = y - z;
乗算	*	x = y * z
除算	/	x = y / z
累乗	**	x = y ** z

次の例は、Tradewinds Travel 社のサンプルデータを使用した典型的な計算をいくつか示しています。

表 8.2 *算術演算子の使用例*

アクション	SAS ステートメント
航空運賃と地上コストを加算して総コストを求めます。	`TotalCost = AirCost + Landcost;`
基本運賃を 10%増額し、$8 の出国税を加算して、ピークの季節の航空運賃を計算します。	`PeakAir = (AirCost * 1.10) + 8;`
各地上パッケージの 1 泊のコストを示します。	`NightCost = LandCost / Nights;`

いずれの場合も、等号の左辺の変数が、等号の右辺の数値式から計算された値を受け取ります。次の DATA ステップにこれらのステートメントを含めると、データセット NEWTOUR が作成されます。

```
data newtour;
   set mylib.populartours;
   TotalCost = AirCost + LandCost;
   PeakAir = (AirCost * 1.10) + 8;
   NightCost = LandCost / Nights;
run;

proc print data=newtour;
   var Country Nights AirCost LandCost TotalCost PeakAir NightCost;
   title 'Costs for Tours';
run;
```

PROC PRINT ステップの VAR ステートメントによって、ステートメントに記述された変数のみが出力に表示されます。

図 8.2 算術式を使用した変数の新規作成

Costs for Tours

Obs	Country	Nights	AirCost	LandCost	TotalCost	PeakAir	NightCost
1	Japan	8	982	1020	2002	1088.2	127.500
2	Greece	12	.	748	.	.	62.333
3	New Zealand	16	1368	1539	2907	1512.8	96.188
4	Ireland	7	787	628	1415	873.7	89.714
5	Venezuela	9	426	505	931	476.6	56.111
6	Italy	8	852	598	1450	945.2	74.750
7	Russia	14	1106	1024	2130	1224.6	73.143
8	Switzerland	9	816	834	1650	905.6	92.667
9	Australia	12	1299	1169	2468	1436.9	97.417
10	Brazil	8	682	610	1292	758.2	76.250

数値式と割り当てステートメントについて

SAS の数値式には、算術式と共通する特徴がいくつかあります。

- 式に演算子が 2 つ以上含まれる場合、演算の優先順位は算術式と同じです。最初が累乗で、次に乗算と除算、最後に加算と減算が処理されます。
- 優先順位の等しい演算子が出現した場合、演算は左から右へと実行されます(累乗は例外で右から左へと実行されます)。
- 式の一部をグループ化するには、算術式と同様にかっこを使用します。かっこ内の演算は最初に実行されます。

注: 割り当てステートメントの等号が実行する機能は、算術式の等号と同じではありません。割り当てステートメントのシーケンス *variable*=では、ステートメントが定義されます。また、変数は等号の左辺に記述する必要があります。算術式のように結果変数と式の位置を切り替えることはできません。

SAS での欠損値の処理方法について

SAS で欠損値を割り当てる理由

オブザベーションで特定の数値変数の値が欠けている場合について考えてみます。たとえば、図 8.2 (123 ページ)で示したように、データセット MYLIB.POPULARTOURS では、Greece のオブザベーションに変数 AirCost の値がありません。SAS データセットの長方形構造を維持するために、そのオブザベーションの変数に欠損値が割り当てられます。欠損値はそのオブザベーションの変数に対する情報が存在しないことを示します。

欠損値のルール

次のルールは、いくつかの状況における欠損値について記述しています。

- データ行では、数値の欠損値はピリオドで表されます。例を次に示します。

```
Greece        8 12    .    748 Express
```

  デフォルトでは、数値フィールドの 1 個のピリオドは欠損値として解釈されます。INPUT ステートメントで、カラム入力と同様に特定のカラムから値を読み取る場合、ブランクのみを含むフィールドでも欠損値が生成されます。

- 式では、数値の欠損値はピリオドで表されます。例を次に示します。

```
if AirCost= . then Status = 'Need air cost';
```

- 比較や並べ替えでは、数値の欠損値は他のどの数値よりも小さい値になります。
- プロシジャ出力では、デフォルトで、数値の欠損値がピリオドで表されます。
- プロシジャには、欠損値が分析から除外されるものもあれば、除外されないものもあります。個別プロシジャのドキュメントで、各プロシジャの欠損値の処理方法が説明されています。

欠損値のプロパゲート

算術式で欠損値を使用すると、その式の結果は自動的に欠損値に設定されます。その計算結果を他の式で使用すると、次の計算結果も欠損値になります。SAS では、この欠損値処理方法を欠損値のプロパゲーションと呼びます。たとえば、図 8.2 (123 ページ)では、データセット NEWTOUR で、Greece のオブザベーションの TOTALCOST と PEAKAIR の値も欠損していることがわかります。

注: SAS では、各種の数値の欠損値を区別できます。*SAS Language Reference: Concepts* の“Missing Values”セクションを参照してください。SAS 言語には、A から Z までの文字とアンダースコア(_)に基づく 27 の特殊欠損値が含まれます。

SAS 関数を使用した数の計算

値の丸め

さまざまなツアーのコストをリストしているデータ例(図 8.1 (121 ページ))では、一部のツアーに端数価格が設定されています($750 ではなく$748、$1300 ではなく$1299 など)。ツアー価格を$10 単位で丸めて求めた、丸められた数を使用すると、処理が容易になります。

算術演算子のみで丸め計算をプログラミングすると、非常に長い処理になります。ただし、SAS には、関数と呼ばれるビルトイン数値式が含まれます。これらの関数は算術演算子と同様に式で使用できます。たとえば、次の割り当てステートメントでは、AirCost の値が$50 単位で丸められます。

```
RoundAir = round(AirCost,50);
```

次のステートメントでは、各ツアーの総コストが計算され、$100 単位で丸められます。

```
TotalCostR = round(AirCost + LandCost,100);
```

欠損値がある場合のコスト計算

他の例のとおり、旅行代理店は、すべての非欠損コストに基づいてツアーの総コストを計算できます。したがって、(Greece のように)航空運賃が欠損している場合、総コストには、欠損値ではなく地上コストが表示されます。(当然ながら、特定の計算において欠損値のスキップが得策かどうかを判断する必要があります。)SUM 関数では、欠損値を無視して、引数の合計が計算されます。SUM 関数の例を次に示します。

```
SumCost = sum(AirCost,LandCost);
```

関数の組み合わせ

関数を組み合わせることが可能です。ROUND 関数では、最初の引数の指定数量が、2 番目の引数の指定単位で丸められます。SUM 関数では、任意の数の引数が、欠損値を無視して合算されます。次の割り当てステートメントでは、非欠損の航空運賃および地上コストの全合計が$100 単位で丸められ、その値が RoundSum に割り当てられます。

```
RoundSum = round(sum(AirCost,LandCost),100);
```

次の DATA ステップでは、ROUND 関数と SUM 関数を使用して、データセット MORETOUR が作成されます。

```
data moretour;
   set mylib.populartours;
   RoundAir = round(AirCost,50);
   TotalCostR = round(AirCost + LandCost,100);
   CostSum = sum(AirCost,LandCost);
   RoundSum = round(sum(AirCost,LandCost),100);
run;

proc print data=moretour;
   var Country AirCost LandCost RoundAir TotalCostR CostSum RoundSum;
   title 'Rounding and Summing Values';
run;
```

次の出力は結果を示しています。

図 8.3　ROUND 関数と SUM 関数による変数の新規作成

Rounding and Summing Values

Obs	Country	AirCost	LandCost	RoundAir	TotalCostR	CostSum	RoundSum
1	Japan	982	1020	1000	2000	2002	2000
2	Greece	.	748	.	.	748	700
3	New Zealand	1368	1539	1350	2900	2907	2900
4	Ireland	787	628	800	1400	1415	1400
5	Venezuela	426	505	450	900	931	900
6	Italy	852	598	850	1500	1450	1500
7	Russia	1106	1024	1100	2100	2130	2100
8	Switzerland	816	834	800	1700	1650	1700
9	Australia	1299	1169	1300	2500	2468	2500
10	Brazil	682	610	700	1300	1292	1300

数値変数の比較

プログラムでは、しばしば、変数が互いに等しいかどうか、または互いよりも大きいか小さいかを確認する必要があります。2 つの数値変数を比較するには、論理演算子を使用して IF-THEN/ELSE ステートメントを記述します。

次の表は、変数比較のために使用できる論理演算子の一部です。

表 8.3 論理演算子

記号	ニーモニック	論理演算
=	eq	等しい
¬=, ^=, ~=	ne	等しくない(キーボードに応じて、¬=、^=、または~=の記号)
>	gt	より大きい
>=	ge	以上
<	lt	より小さい
<=	le	less than or equal to

この例では、greater-than 論理演算子(gt)を使用して、POPULARTOURS データセットの各ツアーの総コストと 2000 が比較されます。ツアーの総コストが 2000 よりも大きい場合、そのツアーはデータセットから除外されます。処理結果のデータセット TOURSUNDER2K には、$2000 以下のツアーが含まれます。

```
data toursunder2K;
   set mylib.popular tours;
   TotalCost = AirCost + LandCost;
 if TotalCost gt 2000 then delete;
run;
proc print data=toursunder2K;
   var Country Nights AirCost Landcost TotalCost Vendor;
   title 'Tours $2000 or Less';
run;
```

次の出力には、総コストが$2000 より低いツアーが表示されています。

図 8.4 数値変数の比較

Tours $2000 or Less

Obs	Country	Nights	AirCost	LandCost	TotalCost	Vendor
1	Greece	12	.	748	.	Express
2	Ireland	7	787	628	1415	Express
3	Venezuela	9	426	505	931	Mundial
4	Italy	8	852	598	1450	Express
5	Switzerland	9	816	834	1650	Tour2000
6	Brazil	8	682	610	1292	Almeida

欠損値を含む計算の結果はすべて欠損値となるため、Greece の TotalCost 値は欠損値になります。比較時には、数値の欠損値は他のどの数値よりも小さいとされます。

変数を 2 つ以上の値と比較する必要がある場合、条件に複数の比較を含められます。欠損値があるツアーを削除するには、2 番目の比較を追加します。

```
data toursunder2K2;
   set mylib.populartours;
   TotalCost = AirCost + LandCost;
   if TotalCost gt 2000 or Totalcost = . then delete;
run;

proc print data=toursunder2K2;
   var Country Nights TotalCost Vendor;
   title 'Tours $2000 or Less';
run;
```

次の出力は結果を示しています。

図 8.5 条件内の複数の比較

Tours $2000 or Less

Obs	Country	Nights	TotalCost	Vendor
1	Ireland	7	1415	Express
2	Venezuela	9	931	Mundial
3	Italy	8	1450	Express
4	Switzerland	9	1650	Tour2000
5	Brazil	8	1292	Almeida

Greece が$2000 以下のツアーに含まれなくなったことに注意してください。

数値変数を効率的に格納する

このセクションで示すデータセットは非常に小さいですが、多くの場合、データセットは非常に大きいです。データセットが大きい場合、データセットが占有する記憶域について考慮が必要な場合もあります。SAS データセットに数値変数を格納する際に領域を節約する方法はいくつもあります。

注: 使用している動作環境に対応する SAS ドキュメントでは、値が、SAS の最小使用バイト数(動作環境に応じて、2 バイトまたは 3 バイト)の 1 または 0 に制限されている数値変数の格納について情報が提供されます。

デフォルトでは、数値変数ごとにデータセットで 8 バイトの記憶域が使用されます。したがって、前述のデータセット MORETOUR 内の各オブザベーションの変数を格納するには 75 バイトが必要です。

```
   56 bytes for numeric variables
      (8 bytes per variable * 7 numeric variables)
   11 bytes for Country
    8 bytes for Vendor

   ____________________________
   75 bytes for all variables
```

数値変数に含まれているのが整数のみであれば、多くの場合、作成されるデータセットの変数を短くできます。たとえば、4 バイトの長さであれば、厳密には、少なくとも 2,000,000 までのすべての整数が格納されます。

注: 一部の動作環境では、最大バイト数が非常に大きくなります。詳細については、使用している動作環境に対応するドキュメントを参照してください。

各変数の使用バイト数を変更するには、LENGTH ステートメントを使用します。

LENGTH ステートメントには、変数名に続けて、格納に使用するバイト数が含められます。数値変数の場合、LENGTH ステートメントは作成されるデータセットにのみ影響を及ぼします。プログラムデータベクトルには影響しません。次のプログラムでは、データセット SHORTER 内にあるすべての数値変数の記憶域が変更されます。

```
data shorter;
   set mylib.populartours;
   length Nights AirCost LandCost RoundAir TotalCostR
          Costsum RoundSum 4;
   RoundAir = round(AirCost,50);
   TotalCostR = round(AirCost + LandCost,100);
   CostSum = sum(AirCost,LandCost);
   RoundSum = round(sum(AirCost,LandCost),100);
run;
```

SHORTER の各オブザベーション内の変数に必要な記憶域を計算すると、LENGTH ステートメントによる記憶域使用量の変更内容がわかります。

```
28 bytes for numeric variables
   (4 bytes per variable in the LENGTH statement X 7 numeric variables)
11 bytes for Country
 8 bytes for Vendor

___________________________
47 bytes for all variables
```

LENGTH ステートメントによって SHORTER 内の 7 変数が短くなるため、各オブザベーション内の変数の記憶域がほぼ半分に削減されます。

注意:

変数値が整数ではない場合、数値変数の長さを短くするには注意が必要です。 整数でない数値を切り捨てると、永久的に精度が失われます。一般的には、ディスク領域が十分でない場合だけ、LENGTH ステートメントを使用して値を切り捨てます。小数を含む変数を格納するには、デフォルトの長さの 8 バイトを使用します。

要約

関数

ROUND (*expression*, *round-off-unit*)
expression の数量を、*round-off-unit* に指定した数字単位で丸めます。**expression** には、数値変数名、数値定数または算術式を指定できます。*round-off-unit* と *expression* はカンマで区切ります。

SUM (*expression-1*<, *expression-2*>, ...)
かっこ内に指定した式の全合計を求めます。SUM 関数では、expressions の合計が計算されるため、欠損値は無視されます。expression のそれぞれに、数値変数、数値定数、別の算術式、別の数値関数のいずれかを指定できます。

ステートメント

LENGTH *variable-listnumber-of-bytes*;
variable-list 内の変数が、指定した *number-of-bytes* に従ってデータセットに格納されることを示します。数値変数は、プログラムデータベクトル内にある間は影響を受けません。数値変数のデフォルトの長さは 8 バイトです。一般に、使用する最小値は、整数を含む変数の場合は 4 バイト、小数を含む変数の場合は 8 バイトです。1 つの LENGTH ステートメントで数値変数と文字変数(次セクションで説明します)の両方に長さを割り当てられます。

variable=expression;
割り当てステートメントです。これにより、等号の右辺の式の値が計算され、左辺の *variable* にその結果が割り当てられます。*variable* が数値の場合、式には算術計算、数値定数または数値関数を指定できます。

詳細情報

変数リストの省略
関数の引数の変数リストを省略する方法は、*SAS Language Reference: Concepts* に記載されています。SUM 関数を含む多くの関数では、省略された変数リストが引数として受け入れられます。

DEFAULT=オプション
LENGTH ステートメントで DEFAULT=オプションを使用して、新規作成されたすべての数値変数にデフォルトの記憶域の長さを割り当てる方法については、*SAS Statements: Reference* で確認できます。

論理式
論理式の使用法の詳細については、*SAS Language Reference: Concepts* で確認できます。

数値精度
数値精度の説明については、*SAS Language Reference: Concepts* を参照してください。コンピュータのハードウェアによって、コンピュータでの数字の格納方法が決定されるため、数字を格納する際の精度は、コンピュータシステムがインストールされているハードウェアによって異なります。ハードウェア固有の制限については、各動作環境の SAS ドキュメントで説明しています。

領域の節約
一部の数値を文字値として処理することで領域を節約する方法については、9 章, “文字変数の操作” (131 ページ)を参照してください。

9 章

文字変数の操作

文字変数の操作について **131**
目的 131
前提条件 132
SAS の文字変数 132
例で使用される入力 SAS データセット **132**
文字変数の識別および文字値の表示 **133**
文字変数の長さの設定 **134**
欠損値の処理 **136**
欠損値の読み取り 136
欠損文字値の確認 137
文字変数値を欠損値に設定 138
文字値の新規作成 **139**
文字値の一部を抽出 139
文字値の結合: 連結の使用 141
数値を文字として処理し、記憶域を節約する **143**
要約 **145**
関数 145
ステートメント 145
詳細情報 **145**

文字変数の操作について

目的

このセクションでは、次の操作を行う方法を学習します。

- 文字変数の識別
- 文字変数の長さの設定
- 文字変数内の文字値の位置調整
- 文字変数の欠損値の処理
- SAS プログラムステートメントにおける文字変数、文字定数および文字式の操作

- 領域を節約するために、数字を含むフィールドを文字変数として読み取るように指示する

前提条件

このセクションを先に進む前に、次のトピックで説明した概念を理解している必要があります。

- 第 1 部、"SAS System について"
- 第 2 部、"データの加工"
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)

SAS の文字変数

文字変数は、値にアルファベット、数字および特殊文字を含む変数で、長さは 1 から 32,767 文字です。文字変数は、宣言ステートメント、比較ステートメントまたは割り当てステートメントで使用できます。これらのステートメントでは、文字変数を操作して新しい文字変数を作成できます。

例で使用される入力 SAS データセット

Tradewinds Travel 社には、ツアーのフライトスケジュールに関するデータを含む外部ファイルがあります。

次の DATA ステップでは、情報を読み取り、AIR.DEPARTURES という名前のデータセットに格納します。

```
libname mylib 'permanent-data-library';

data mylib.departures;
   input Country $ 1-9 CitiesInTour 11-12 USGate $ 14-26
         ArrivalDepartureGates $ 28-48;
   datalines;

1          2 3                     4
-----------------------------------------------
Japan      5 San Francisco          Tokyo, Osaka
Italy      8 New York               Rome, Naples
Australia 12 Honolulu           Sydney, Brisbane
Venezuela  4 Miami            Caracas, Maracaibo
Brazil     4               Rio de Janeiro, Belem
;
proc print data=mylib.departures;
   title 'Data Set AIR.DEPARTURES';
run;
```

番号付きフィールドは次の内容を表しています。

1 ツアー先の国名

2 ツアーの都市数

3 ツアーのアメリカ出国都市(搭乗都市)

4 行き先国の発着都市

DATA ステップの後の PROC PRINT ステートメントによって、AIR.DEPARTURES データセットが次のように表示されます。

図 9.1 データセット AIR.DEPARTURES

Data Set AIR.DEPARTURES

Obs	Country	CitiesInTour	USGate	ArrivalDepartureGates
1	Japan	5	San Francisco	Tokyo, Osaka
2	Italy	8	New York	Rome, Naples
3	Australia	12	Honolulu	Sydney, Brisbane
4	Venezuela	4	Miami	Caracas, Maracaibo
5	Brazil	4		Rio de Janeiro, Belem

AIR.DEPARTURES の変数 Country、USGate および ArrivalDepartureGates には数字以外の情報が含まれているため、文字変数として格納する必要があります。変数 CitiesInTour には数字しか含まれていません。したがって、文字変数と数値変数のどちらとしてでも作成や格納ができます。

文字変数の識別および文字値の表示

SAS データセットに文字値を格納するには、文字値を作成する必要があります。文字変数を作成する 1 つの方法は、入力ステートメントで定義することです。これには、AIR.DEPARTURES を作成する DATA ステップで示したように、単に INPUT ステートメントで変数名の後にドル記号を付けるだけです。

```
input Country $ 1-9 CitiesInTour 11-12 USGate $ 14-26
      ArrivalDepartureGates $ 28-48;
```

また、割り当てステートメントで、文字変数を作成し、そこに値を割り当てることもできます。これには、単に値を引用符で囲むだけです。

```
Schedule = '3-4 tours per season';
```

一重引用符(アポストロフィ)か二重引用符のどちらかが受け入れ可能です。値自体に一重引用符が含まれている場合は、次のように、値を二重引用符で囲みます。

```
Remarks = "See last year's schedule";
```

注: 引用符は、右と左を適切に対応させることが重要です。引用符の数が多いあるいは少ない場合には、誤りのあるステートメントと後続するステートメントは、どちらも誤って解釈されます。

式で文字値を指定する場合も、その値を引用符で囲む必要があります。たとえば、次のステートメントでは、USGate の値と San Francisco が比較され、一致が発生した場合、空港コード SFO が変数 Airport に割り当てられます。

```
if USGate = 'San Francisco' then Airport =
'SFO';
```

文字値では、大文字と小文字が区別されます。たとえば、データセット AIR.DEPARTURES では、Australia のオブザベーションの USGate 値は **Honolulu** です。次の IF 条件は真です。したがって、Airport に値 HNL が割り当てられます。

```
else if USGate = 'Honolulu' then Airport = 'HNL';
```

ただし、次の条件は偽です。

```
if USGate = 'HONOLULU' then Airport = 'HNL';
```

Honolulu と HONOLULU は同じ文字ではないため、そのオブザベーションは選択されません。

次のプログラムでは、DATA ステップにこのような影付きステートメントを入れています。

```
data charvars;
   set mylib.departures;
   Schedule = '3-4 tours per season';
   Remarks = "See last year's schedule";
   if USGate = 'San Francisco' then Airport = 'SFO';
    else if USGate = 'Honolulu' then Airport = 'HNL';
run;

proc print data=charvars noobs;1
   var Country Schedule Remarks USGate Airport;
   title 'Tours By City of Departure';
run;
```

1 PROC PRINT ステートメントの NOOBS オプションでは、出力のオブザベーション番号が非表示にされます。

次の出力には、データセット CHARVARS の文字変数が表示されています。

図 9.2 文字変数の例

Tours By City of Departure

Country	Schedule	Remarks	USGate	Airport
Japan	3-4 tours per season	See last year's schedule	San Francisco	SFO
Italy	3-4 tours per season	See last year's schedule	New York	
Australia	3-4 tours per season	See last year's schedule	Honolulu	HNL
Venezuela	3-4 tours per season	See last year's schedule	Miami	
Brazil	3-4 tours per season	See last year's schedule		

文字変数の長さの設定

この例では、最初の割り当て値によって長さを決定するかわりに、文字変数の長さを指定する必要がある場合の理由を説明します。New York 市には空港が 2 つあるため、John F. Kennedy 国際空港と La Guardia 空港の省略名が両方とも DATA ステップで Airport 変数に割り当てられる可能性があります。

注: 文字変数を作成すると、DATA ステップで最初に出現した文字変数によって変数の長さが決定されます。したがって、変数を記述する最初のステートメントで、可能な限り最も長い値を使用することを考慮に入れる必要があります。最初の変数割り当て時に最も長い値を割り当てなければ、データが切り捨てられる可能性があります。

```
   /* first attempt */
data aircode;
   set mylib.departures;
   if USGate = 'San Francisco' then Airport = 'SFO';
   else if USGate = 'Honolulu' then Airport = 'HNL';
        else if USGate = 'New York' then Airport = 'JFK or LGA';
run;

proc print data=aircode;
   var Country USGate Airport;
   title 'Country by US Point of Departure';
run;
```

次の出力は結果を示しています。

図 9.3 文字値の切り捨て

Country by US Point of Departure

Obs	Country	USGate	Airport
1	Japan	San Francisco	SFO
2	Italy	New York	JFK
3	Australia	Honolulu	HNL
4	Venezuela	Miami	
5	Brazil		

New York のオブザベーションには JFK の文字のみが表示されています。Airport が最初に発生するのは、値 SFO を割り当てるステートメントです。したがって、長さ 3 バイトの Airport が作成され、New York オブザベーションの最初の 3 文字のみが使用されます。

JFK or LGA を書き込む領域を作るには、最初の Airport 参照時に LENGTH ステートメントを使用します。LENGTH ステートメントは宣言ステートメントで、形式は次のとおりです。

LENGTH *variable-list* $ *number-of-bytes*;

variable-list
長さ *number-of-bytes* の割り当て先の 1 つまたは複数の変数を指定します。ドル記号($)は、変数が文字変数であることを示します。LENGTH ステートメントでは、文字変数の長さはプログラムデータベクトルと作成中データセットの両方で決定されます。(それとは対照的に、LENGTH ステートメントでは、数値変数の長さは作成中データセットでのみ決定されます。)SAS 内の文字値の最大長は、32,767 バイトです。

LENGTH ステートメントでは、文字変数 Airport に 10 の長さが割り当てられます。

```
length Airport $ 10;
```

注: LENGTH ステートメントを使用して文字変数に長さを割り当てる場合、それが DATA ステップにおけるその文字変数の最初の参照である必要があります。したがって、DATA ステップでの LENGTH ステートメントの最適な位置は DATA ステートメントの直後です。

次の DATA ステップには、Airport に対する LENGTH ステートメントが含まれています。DATASETS プロシジャを使用すると、SAS データセット内の変数の長さを表示できることに注意してください。

```
   /* correct method */
data aircode2;
   length Airport $ 10;
   set mylib.departures;
   if USGate = 'San Francisco' then Airport = 'SFO';
   else if USGate = 'Honolulu' then Airport = 'HNL';
        else if USGate = 'New York' then Airport = 'JFK or LGA';
             else if USGate = 'Miami' then Airport = 'MIA';
run;

proc print data=aircode2;
   var Country USGate Airport;
   title 'Country by US Point of Departure';
run;
```

次の出力は結果を示しています。

図 9.4 LENGTH ステートメントによる完全な変数情報の取得

Country by US Point of Departure

Obs	Country	USGate	Airport
1	Japan	San Francisco	SFO
2	Italy	New York	JFK or LGA
3	Australia	Honolulu	HNL
4	Venezuela	Miami	MIA
5	Brazil		

欠損値の処理

欠損値の読み取り

SAS では、ブランクを使用して文字変数の欠損値を表します。たとえば、Brazil のデータ行にはアメリカの出発都市が欠けています。

```
Japan      5 San Francisco          Tokyo, Osaka
Italy      8 New York               Rome, Naples
Australia 12 Honolulu           Sydney, Brisbane
Venezuela  4 Miami            Caracas, Maracaibo
Brazil     4               Rio de Janeiro, Belem
```

図 9.1 (133 ページ)に示すように、INPUT ステートメントで Brazil のデータ行が読み取られ、カラム 14 から 26 までの USGate の値が欠損していることが確定すると、そのオブザベーションの USGate に欠損値が割り当てられます。欠損値は印刷時にはブランクで表示されます。

リスト入力で文字データ値を読み取る際に、ある特殊例が発生します。その場合、データ行の欠損値を表示するにはピリオドを使用する必要があります。(リスト入力では

ブランクによって値が区切られます。したがって、ブランクは欠損値ではなく値の検索を続けるためのシグナルと解釈されます。)次の DATA ステップでは、Venezuela の TourGuide 情報が欠損し、ピリオドで表されています。

```
data missingval;
   length Country $ 10 TourGuide $ 10;
   input Country TourGuide;
   datalines;
Japan Yamada
Italy Militello
Australia Edney
Venezuela .
Brazil Cardoso
;

proc print data=missingval;
   title 'Missing Values for Character List Input Data';
run;
```

次の出力は結果を示しています。

図 9.5 *リスト入力でピリオドを欠損文字データとして使用*

Missing Values for Character List Input Data

Obs	Country	TourGuide
1	Japan	Yamada
2	Italy	Militello
3	Australia	Edney
4	Venezuela	
5	Brazil	Cardoso

4 番目のデータ行でピリオドが欠損値として認識されました。これにより、結果データセットに文字変数 TourGuide の欠損値が記録されました。

欠損文字値の確認

欠損文字値の確認が必要な場合は、文字変数を、引用符で囲んだブランクと比較します。

```
if USGate = ' ' then GateInformation = 'Missing';
```

次の DATA ステップには、USGate の欠損情報を確認するために、このステートメントが含まれています。結果は GateInformation に記録されます。

```
data checkgate;
   length GateInformation $ 15;
   set mylib.departures;
   if USGate = ' ' then GateInformation = 'Missing';
   else GateInformation = 'Available';
run;
proc print data=checkgate;
   var Country CitiesIntour USGate ArrivalDepartureGates GateInformation;
   title 'Checking For Missing Gate Information';
run;
```

次の出力は結果を示しています。

図 9.6 欠損文字値の確認

Checking For Missing Gate Information

Obs	Country	CitiesInTour	USGate	ArrivalDepartureGates	GateInformation
1	Japan	5	San Francisco	Tokyo, Osaka	Available
2	Italy	8	New York	Rome, Naples	Available
3	Australia	12	Honolulu	Sydney, Brisbane	Available
4	Venezuela	4	Miami	Caracas, Maracaibo	Available
5	Brazil	4		Rio de Janeiro, Belem	Missing

文字変数値を欠損値に設定

割り当てステートメントで欠損文字値を割り当てるには、文字変数を、引用符で囲んだブランクに設定します。たとえば、次のステートメントでは、ツアーの日数に基づいて出発日が設定されます。ツアーの都市滞在日数が 1 週間以内の場合、出発日は Sunday になります。それ以外の場合は、出発日は不明なので、欠損値に設定されます。

```
if Cities <=7 then DayOfDeparture = 'Sunday';
else DayOfDeparture = ' ';
```

次の DATA ステップには、これらのステートメントが含まれています。

```
data departuredays;
   set mylib.departures;
   length DayOfDeparture $ 8;
   if CitiesInTour <=7 then DayOfDeparture = 'Sunday';
   else DayOfDeparture = ' ';
run;

proc print data=departuredays;
   var Country CitiesInTour DayOfDeparture;
   title 'Departure Day is Sunday or Missing';
run;
```

次の出力は結果を示しています。

図 9.7 欠損文字値の割り当て

Departure Day is Sunday or Missing

Obs	Country	CitiesInTour	DayOfDeparture
1	Japan	5	Sunday
2	Italy	8	
3	Australia	12	
4	Venezuela	4	Sunday
5	Brazil	4	Sunday

文字値の新規作成

文字値の一部を抽出

SCAN 関数について

文字値には、分離して別々の文字変数に割り当てる必要がある複数の情報が含まれる場合があります。たとえば、ArrivalDepartureGates の値には、到着都市と出発都市の 2 つの都市が含まれます。どうすれば個々の値を分離し、2 つの都市に対して別々の変数を作成できるでしょうか。

SCAN 関数は、ソース文字列、必要な文字列の位置および区切り文字を与えられると、文字列を返します。

SCAN (*source*, *n*<,*list-of-delimiters*>)

source は検証が必要な値です。ここには文字変数や文字定数など任意の種類の文字式を指定できます。*n* は、*source* から選択する項目の位置です。*list-of-delimiters* には、区切り文字を 1 つまたは複数指定できます。未指定にしておくこともできます。区切り文字を 2 つ以上指定すると、そのいずれかが使用されます。区切り文字を省略すると、デフォルトの区切り文字リスト(ブランクや特殊文字などを含む)に従って単語が分割されます。

たとえば、ArrivalDepartureGates の値内の最初の項目を選択し、ArrivalGate という新しい変数に割り当てるには、次のように記述します。

```
ArrivalGate = scan(ArrivalDepartureGates,1,',');
```

SCAN 関数では、ArrivalDepartureGates の値が検証され、カンマによって識別された最初の文字列が選択されます。

デフォルト値を区切り文字として使用することもできますが、使用する区切り文字を指定することをお勧めします。Brazil のオブザベーションの処理時に、SCAN 関数でデフォルト区切り文字を使用すると、ブランクが区切り文字として認識され、**`Rio de Janeiro`** ではなく **`Rio`** が最初の項目として選択されます。区切り文字を指定すると、項目の分割が発生する場所を制御できます。

ArrivalDepartureGates から 2 番目の項目を選択し、DEPARTUREGATE という新しい変数項目に割り当てるには、次のように指定します。

```
DepartureGate = scan(ArrivalDepartureGates,2,',');
```

注: DATA ステップ内では、式に SCAN 関数が含まれている場合のターゲット変数のデフォルト長は、最初の SCAN 引数の長さと同じです。SQL プロシジャまたは WHERE 句では、SCAN 関数によって返される文字列の最大長は 200 文字になります。SCAN 関数の詳細については、*SAS Functions and CALL Routines: Reference* を参照してください。

新しい値の位置調整

式で使用された文字値の既存位置調整が保持されることに注意してください。自動位置再調整は実行されません。この例では、新しい変数 DepartureGate の値が ArrivalDepartureGates の値から作成されます。次の出力に示すように、

ArrivalDepartureGates の値では、2 つの都市名の間にカンマとブランクが含まれます。

図 9.8 SCAN 関数を使用して値を別々の語に分割

Data Set Air.Departure

Obs	Country	CitiesInTour	USGate	ArrivalDepartureGates
1	Japan	5	San Francisco	Tokyo, Osaka
2	Italy	8	New York	Rome, Naples
3	Australia	12	Honolulu	Sydney, Brisbane
4	Venezuela	4	Miami	Caracas, Maracaibo
5	Brazil	4		Rio de Janeiro, Belem

SCAN 関数によってカンマで名前を分割すると、2 番目の項目はブランクで始まります。したがって、DepartureGate に割り当てられる値はすべてブランクで始まります。

値を左揃えにするには、LEFT 関数を使用します。

LEFT (*source*)

LEFT 関数では、*source* の先頭のブランクをすべて値の右側に移動した値が生成されます。したがって、結果的に左揃えになります。*source* には、文字変数、引用符で囲んだ文字定数、別の文字関数など、任意の種類の文字式を指定できます。

この例では、2 番目の割り当てステートメントで LEFT 関数を使用します。

```
DepartureGate = scan(ArrivalDepartureGates,2,',');
DepartureGate = left(DepartureGate);
```

2 つの関数をネストすることもできます。

```
DepartureGate = left(scan(ArrivalDepartureGates,2,','));
```

関数をネストすると、最も内側の関数の処理が最初に実行されます。その関数の結果が次の関数の引数として使用され、それ以降も同様に処理されます。

次の DATA ステップでは、出発地と到着地に対して別々の変数が作成されます。

```
data gates;
   set mylib.departures;
   ArrivalGate = scan(ArrivalDepartureGates,1,',');
   DepartureGate = left(scan(ArrivalDepartureGates,2,','));
run;

proc print data=gates;
   var Country ArrivalDepartureGates ArrivalGate DepartureGate;
   title 'Arrival and Departure Gates';
run;
```

次の出力は結果を示しています。

図 9.9 SCAN 関数を使用して値を別々の語に分割

Arrival and Departure Gates

Obs	Country	ArrivalDepartureGates	ArrivalGate	DepartureGate
1	Japan	Tokyo, Osaka	Tokyo	Osaka
2	Italy	Rome, Naples	Rome	Naples
3	Australia	Sydney, Brisbane	Sydney	Brisbane
4	Venezuela	Caracas, Maracaibo	Caracas	Maracaibo
5	Brazil	Rio de Janeiro, Belem	Rio de Janeiro	Belem

SCAN 関数使用時の記憶域の節約

DATA ステップでは、ターゲット変数の長さが見割り当ての場合、SCAN 関数で、割り当てステートメントのターゲット変数に最初の引数の長さが割り当てられます。SQL プロシジャまたはプロシジャの WHERE 句では、SCAN 関数によって返される単語の最大長は 200 文字になります。演算子または他の関数を含む式で SCAN 関数を使用する場合、SCAN 関数によって返される単語の最大長は 32,767 文字になります。

ArrivalGate と DepartureGate の長さを、デフォルト長ではなく必要な値に設定すると、記憶域を大幅に節約できます。文字変数は最初の発生時に長さが設定されるため、LENGTH ステートメントは変数の値を作成する割り当てステートメントの前に記述する必要があります。

```
data gatelength;
   length ArrivalGate $ 14 DepartureGate $ 9;
   set mylib.departures;
   ArrivalGate = scan(ArrivalDepartureGate,1,',');
   DepartureGate = left(scan(ArrivalDepartureGate,2,','));
run;
```

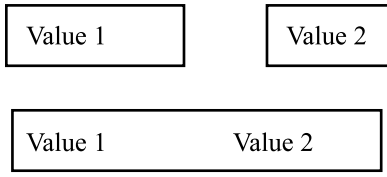
文字値の結合: 連結の使用

変数値の連結について

SAS では、連結と呼ばれる操作により文字値を結合して長くすることができます。連結によって文字値を結合するには、文字値を順々に並べて変数に割り当てます。SAS プログラミングでは、連結演算子は 1 組の縦棒(||)です。キーボードに実線の縦棒がない場合は、破線の縦棒 2 つ(¦¦)かまたは感嘆符 2 つ(!!)を使用します。新しい変数の長さは、各変数の長さの合計か、または LENGTH ステートメントで新しい変数に対して指定された文字数です。

次の図は、連結を示しています。

図 9.10 2 つの値の連結



簡単な連結の実行

次のステートメントでは、CAT 関数を使用して、搭乗地として指定されたすべての都市を、AllGates という 1 つの変数に結合します。

```
AllGates = cat(USGate,ArrivalDepartureGates);
```

注: CAT 関数では、“変数値の連結について” (141 ページ)で説明した||連結演算子は使用されません。

ArrivalDepartureGates の各値の先頭を USGate の各値の末尾に付けて、その結果を AllGates に割り当てます。次の DATA ステップには、このステートメントが含まれています。

```
data all;
   set mylib.departures;
   AllGates = cat(USGate,ArrivalDepartureGates);
run;

proc print data=all;
   var Country USGate ArrivalDepartureGates AllGates;
   title 'All Tour Gates';
run;
```

次の出力は結果を示しています。

図 9.11 *簡単な連結*

All Tour Gates

Obs	Country	USGate	ArrivalDepartureGates	AllGates
1	Japan	San Francisco	Tokyo, Osaka	San FranciscoTokyo, Osaka
2	Italy	New York	Rome, Naples	New York Rome, Naples
3	Australia	Honolulu	Sydney, Brisbane	Honolulu Sydney, Brisbane
4	Venezuela	Miami	Caracas, Maracaibo	Miami Caracas, Maracaibo
5	Brazil		Rio de Janeiro, Belem	Rio de Janeiro, Belem

前述の出力では USGate の長さは 13 バイトですが、そのすべてを使用しているのは San Francisco のみです。したがって、その他の値は最後にブランクが含まれ、Brazil の値は完全にブランクです。

連結される文字値が、属している変数の長さより短い場合、連結時に、その値に後置ブランクが埋め込まれます。ただし、HTML が SAS のデフォルト出力スタイルであり、そこでは連結出力を表示する際、変数値の後置ブランクは無視されます。そのかわりに、HTML では、連結値の間にブランクが 1 つ入れられます。したがって、前述の出力の AllGates の値では、USGate と ArrivalDepartureGates の連結値間にブランクが 1 つ含まれます。たとえば、4 番目のオブザベーションの AllGates の値では、Miami と Caracas の間にブランクが 1 つ含まれます。HTML 出力では、連結変数を表示する際、USGATE 変数の Miami 値の後に続く 8 つの後置ブランクは無視されます。

注: NOBREAKSPACE スタイル属性を使用すると、連結変数間の後置ブランクを保持できます。NOBREAKSPACE スタイル属性については、“Style Attributes” (*SAS Output Delivery System: User's Guide*)を参照してください。

追加文字の追加

TRIM 関数を使用すると、連結変数間のブランクを削除できます。また、連結変数間にカンマ、コロン、ブランクなどの区切り文字を追加することもできます。

注: TRIM 関数では、“変数値の連結について” (141 ページ)で説明した||連結演算子が使用されます。

TRIM 関数を使用して連結変数間のブランクを削除すると、結果が見にくくなる場合があります。結果を見やすくするには、USGate のトリム値と ArrivalDepartureGates の値をカンマとブランクで連結します。また、Brazil の AllGate2 値とその他すべての AllGate2 値の位置を揃えるには、IF-THEN ステートメントを使用して、そのオブザベーションの AllGate2 値と ArrivalDepartureGates 値を等しくします。

```
AllGates2 = trim(USGate)||', '||ArrivalDepartureGates;
if Country = 'Brazil' then AllGates2 = ArrivalDepartureGates;
```

次の DATA ステップには、これらのステートメントが含まれています。

```
data all2;
   set mylib.departures;
   AllGates2 = trim(USGate)||', '||ArrivalDepartureGates;
   if Country = 'Brazil' then AllGates2 = ArrivalDepartureGates;
run;

proc print data=all2;
   var Country USGate ArrivalDepartureGates AllGates2;
   title 'All Tour Gates';
run;
```

注: ALLGATES2 ステートメントでは、', 'はリテラル値です。SAS では、2 つの変数のサイズとリテラルの長さを足して、連結変数のサイズを計算します。HTML では、追加後置ブランクはいずれも無視され、指定されたカンマと 1 つのブランクのみが挿入されます。

次の出力は結果を示しています。

図 9.12 *見やすくするために追加文字を連結*

All Tour Gates

Obs	Country	USGate	ArrivalDepartureGates	AllGates2
1	Japan	San Francisco	Tokyo, Osaka	San Francisco, Tokyo, Osaka
2	Italy	New York	Rome, Naples	New York, Rome, Naples
3	Australia	Honolulu	Sydney, Brisbane	Honolulu, Sydney, Brisbane
4	Venezuela	Miami	Caracas, Maracaibo	Miami, Caracas, Maracaibo
5	Brazil		Rio de Janeiro, Belem	Rio de Janeiro, Belem

後置ブランクの削除または連結変数間の区切り文字の追加の詳細については、“TRIM Function” (*SAS Functions and CALL Routines: Reference*)を参照してください。

数値を文字として処理し、記憶域を節約する

SAS では DATA ステップの数値ごとに 8 バイトの記憶域が使用されることに注意してください。デフォルトでは、SAS で、出力データセットでも数値ごとに 8 バイトの記憶域が使用されます。ただし、文字値には最小 1 文字を含められます。その場合、プログラムデータベクトルと出力データセットの両方で、文字変数に 1 バイトが使用されます。さらに、文字値の 0 から 9 までの数字は、その他すべての文字と同様に処理されます。変数に対して計算を実行しない場合は、数字を含む値を文字値として処理すると、記憶域を節約できます。

たとえば、ホテルの部屋の質に応じて、さまざまな価格を提供するツアーがあるとします。パンフレットでは、客室が 2 つ星、3 つ星などとランク付けされています。この場合、値 2、3 および 4 は実際はカテゴリ名で、その値に対して算術演算が実行されることはないと考えられます。したがって、値は文字変数に読み込むことができます。次の DATA ステップでは、HotelRank が文字変数として読み取られ、1 バイトの長さを割り当てられます。

```
data hotels;
   input Country $ 1-9 HotelRank $ 11 LandCost;
   datalines;
Italy     2  498
Italy     4  698
Australia 2  915
Australia 3 1169
Australia 4 1399
;

proc print data=hotels;
   title 'Hotel Rankings';
run;
```

前述の例では、INPUT ステートメントで HotelRank に 1 バイトの長さが割り当てられます。これは INPUT ステートメントで 1 カラムを読み取ると値が見つかるためです(カラム入力を使用して表示)。リスト入力を使用する場合は、INPUT ステートメントの前に LENGTH ステートメントを配置し、長さを 1 バイトに設定します。

数字を文字値として読み取った後で、数値式に使用する必要があると判明した場合、プログラムに変更を加えずに処理できます。SAS では、式で使用するための数値が文字値から自動生成されます。変換が発生したという NOTE メッセージもログに出力されます。(当然、変換によって DATA ステップでのコンピュータリソース使用量が少し増加します。)元の変数は変更されません。

次の出力は結果を示しています。

図 9.13 *文字変数の作成による記憶域の節約*

Hotel Rankings

Obs	Country	HotelRank	LandCost
1	Italy	2	498
2	Italy	4	698
3	Australia	2	915
4	Australia	3	1169
5	Australia	4	1399

注: 列幅がデフォルト幅の 8 ではないことに注意してください。

要約

関数

LEFT (*source*)
前置ブランクをすべて値の最後に移動することで *source* を左揃えにします。*source* には、文字変数、引用符で囲んだ文字定数、別の文字関数など、任意の種類の文字式を指定できます。左から削除されたブランクがすべて右に追加されるため、結果の長さと *source* の長さは一致します。詳細については、“LEFT Function” (*SAS Functions and CALL Routines: Reference*)を参照してください。

SCAN (*source*, **n** <,*list-of-delimiters*>)
n 番目の項目を *source* から選択します。*source* には、文字変数、引用符で囲んだ文字定数、別の文字関数など、任意の種類の文字式を指定できます。項目を分割する文字を選択するには、区切り文字を使用します。区切り文字を省略すると、SAS でデフォルトの区切り文字リスト(ブランクや特殊文字)を使用して項目が分割されます。詳細については、“SCAN Function” (*SAS Functions and CALL Routines: Reference*)を参照してください。

TRIM (*source*)
source から後置ブランクがトリムされます。*source* には、文字変数、引用符で囲んだ文字定数、別の文字関数など、任意の種類の文字式を指定できます。TRIM 関数は、変数の格納方法には影響を及ぼしません。TRIM 関数を使用して後置ブランクを削除し、トリム値をその値よりも長い変数に割り当てると、その値と新しい変数の長さを一致させるために、値に新しい後置ブランクが埋め込まれます。詳細については、“TRIM Function” (*SAS Functions and CALL Routines: Reference*)を参照してください。

ステートメント

LENGTH *variable-list$number-of-bytes*;
number-of-bytes に指定した長さを、*variable-list* 内の 1 つまたは複数の文字変数に割り当てます。1 つの LENGTH ステートメントで任意の数の長さを割り当てられます。また、同じステートメントで文字変数と数値変数の両方に長さを割り当てられます。すべての文字変数の長さの前にドル記号($)を配置します。詳細については、“LENGTH Statement” (*SAS Statements: Reference*)を参照してください。

詳細情報

文字値
このセクションでは、文字値の操作のために SAS が提供する柔軟性について説明します。このセクションで説明する関数に加えて、次の文字関数も頻繁に使用されます。

COMPBL
文字列から複数のブランクを削除します。

COMPRESS
ソースから指定文字を削除します。

INDEX
ソースデータの中の文字パターンを検索します。

LOWCASE
引数の文字をすべて小文字に変換します。

RIGHT
ソースを右揃えにします。

%SUBSTR
文字のグループを抽出します。

TRANSLATE
文字式内の特定の文字を置換します。

UPCASE
ソースデータを大文字で返します。

INDEX 関数と UPCASE 関数については 10 章, “選択したオブザベーションの操作” (147 ページ)で説明します。すべての文字関数の詳細な説明については、*SAS Functions and CALL Routines: Reference* を参照してください。

文字変数
文字変数の詳細情報については *SAS Language Reference: Concepts* を参照してください。

文字変数の位置調整の追加情報については、*SAS Output Delivery System: User's Guide* の TEMPLATE プロシジャ、および *Base SAS Procedures Guide* の REPORT プロシジャの説明をを参照してください。

大文字と小文字の比較
大文字と小文字の比較方法については、10 章, “選択したオブザベーションの操作” (147 ページ)で説明します。

連結演算子
連結演算子については、*SAS Language Reference: Concepts* で確認できます。

DATASETS プロシジャ
DATASETS プロシジャを使用して SAS データセット内の変数の長さを表示する方法については、37 章, “SAS データセットの情報の表示” (681 ページ)で説明されています。

IF-THEN ステートメント
IF-THEN ステートメントの詳細な説明については、10 章, “選択したオブザベーションの操作” (147 ページ)で確認できます。

入力形式と出力形式
文字変数の読み取りと書き込み用の多数の SAS 入力形式と出力形式の完全な情報については *SAS Formats and Informats: Reference* を参照してください。

欠損値
欠損値の詳細情報については *SAS Language Reference: Concepts* を参照してください。

VLENGTH 関数
VLENGTH 関数については、*SAS Functions and CALL Routines: Reference* で詳細に説明されています。

10 章

選択したオブザベーションの操作

選択したオブザベーションの操作について **147**
目的 147
前提条件 148
例で使用される入力 SAS データセット **148**
オブザベーションの選択 **149**
選択処理について 149
単純条件に基づいたオブザベーションの選択 150
代替処理の指定 151
一連の相互排他的条件の作成 152
条件の作成 **153**
条件の作成について 153
単純条件に基づいたオブザベーションの選択 154
条件に複数の比較を使用 155
文字の比較 **161**
文字の比較のタイプ 161
大文字と小文字の比較 161
同じ文字グループで始まる値をすべて選択 162
文字値範囲の選択 163
別の文字値内全体から値を検索 164
要約 **165**
ステートメント 165
関数 166
詳細情報 **166**

選択したオブザベーションの操作について

目的

最も便利な SAS 機能の 1 つは、選択したオブザベーションに対してのみ処理を実行する機能です。このセクションでは、次の概念を説明しています。

- 選択処理の方法
- 条件に基づいてオブザベーションを選択するステートメントの記述方法
- 数値変数と文字変数の選択に関する特別な点

前提条件

このセクションを先に進む前に、これまでのすべてのセクションで説明した概念を理解している必要があります。

例で使用される入力 SAS データセット

Tradewinds Travel 社は、さまざまな都市の美術館やギャラリーのツアーを提供しています。この会社は、処理を効率化するために、追加情報が必要だと判断しました。たとえば、ツアーで一定期間内にまわる美術館やギャラリーが多すぎる場合は、訪問する美術館の数を減らすか、ツアー日数を変更する必要があります。ツアーに割り当てられているガイドに都合がつかなければ、別のガイドを割り当てる必要があります。処理の大部分は、さまざまな基準を満たすオブザベーション、または満たさないオブザベーションを選択し、必要な処理を実行することです。

Tradewinds Travel 社のツアーデータは、次の情報を含む外部ファイルに格納されます。

```
1         2 3    4   5               6        7
-----------------------------------------------------
Rome      3 1550 7 4 M, 3 G          D'Amico  Torres
Brasilia  8 3360 6 5 M, 1 other      Lucas    Lucas
London    6 2460 5 3 M, 2 G          Wilson   Lucas
Warsaw    6    . 8 5 M, 1 G, 2 other Lucas    D'Amico
Madrid    3  740 5 3 M, 2 other      Torres   D'Amico
Amsterdam 4 1160 6 3 M, 3 G                   Vandever
```

次のリストは、前述のファイルの番号付き項目に対応しています。

1 行く先の市の名前を特定します

2 その市での宿泊数を示します

3 米ドルでの地上パッケージのコストを示します

4 ツアーで提供するイベント(美術館やギャラリーへの訪問など)の数を示します

5 イベントの簡単な説明を提供します。**M** は美術館、**G** はギャラリー、**other** はその他の種類のイベントを示します

6 ツアーガイドの名前を提示します

7 予備ツアーガイドの名前を提示します

次の DATA ステップでは、MYLIB.ARTTOURS が作成されます。

```
libname mylib 'SAS-library';

data mylib.arttours;
   infile 'input-file' truncover;
   input City $ 1-9 Nights 11 LandCost 13-16 NumberOfEvents 18
         EventDescription $ 20-36 TourGuide $ 38-45
         BackUpGuide $ 47-54;
run;

proc print data=mylib.arttours;
```

```
   title 'Data Set MYLIB.ARTTOURS';
run;
```

注: INFILE ステートメントで TRUNCOVER オプションを指定した場合、レコードが INPUT ステートメントで求められる長さよりも短ければ、SAS は変数の長さのレコードを読み込みます。

DATA ステップの後の PROC PRINT ステートメントによって、MYLIB.ARTTOURS データセットが次のように表示されます。

図 10.1 *データセット MYLIB.ARTTOURS*

Data Set MYLIB.ARTTOURS

Obs	City	Nights	LandCost	NumberOfEvents	EventDescription	TourGuide	BackUpGuide
1	Rome	3	1550	7	4 M, 3 G	D'Amico	Torres
2	Brasilia	8	3360	6	5 M, 1 other	Lucas	Lucas
3	London	6	2460	5	3 M, 2 G	Wilson	Lucas
4	Warsaw	6	.	8	5 M, 1 G, 2 other	Lucas	D'Amico
5	Madrid	3	740	5	3 M, 2 other	Torres	D'Amico
6	Amsterdam	4	1160	6	3 M, 3 G		Vandever

次に、出力のフィールドをいくつか説明します。

- NumberOfEvents には、ツアーの間に訪問したアトラクションの数が含まれます。
- EventDescription には、訪問した美術館(`M`)、アートギャラリー(`G`)、およびその他アトラクション(`other`)の数がリストされます。
- TourGuide には、ツアーに割り当てられたツアーガイドの名前がリストされます。
- BackUpGuide には、最初のツアーガイドの都合がつかない場合の代替ツアーガイドがリストされます。

オブザベーションの選択

選択処理について

DATA ステップの処理対象オブザベーションを選択する最も一般的な方法は、IF-THEN ステートメントを使用することです。

IF *condition* **THEN** *action*;

condition は 1 つ以上の比較です。例を次に示します。

- `City = 'Rome'`
- `NumberOfEvents > Nights`
- `TourGuide = 'Lucas' and Nights > 7`

記号>は、より大きいを意味します。比較演算子として記号を使用する方法は、“条件の作成について” (153 ページ)に説明されています。

指定オブザベーションの比較は、真か偽のどちらかになります。最初の例では、City の値は **Rome** かそれ以外かのどちらかです。2 番目の例では、現在のオブザベーションの NumberOfEvents の値は、同じオブザベーションの Nights の値よりも大きいかそうでないかのどちらかです。3 番目の例のように、条件に比較が 2 つ以上含まれている場合、すべての条件がルールに従って評価され(後述)、条件全体が真か偽かが宣言されます。

条件が真の場合、THEN 句の処理が行われます。処理は DATA ステップの個々の反復で実行可能な SAS ステートメントして表す必要があります。そのようなステートメントは実行可能ステートメントと呼ばれます。次の例において示されるように、最も一般的な実行可能ステートメントは割り当てステートメントです。

- `LandCost=LandCost + 30;`
- `Calendar='Check schedule';`
- `TourGuide='Torres';`

このセクションでは、THEN 句の割り当てステートメントのみを扱いますが、他のセクションの例では、THEN 句とあわせて使用する他の種類のステートメントについて説明します。

データセットについて情報を提供するステートメントは実行可能ではありません。そのようなステートメントは宣言ステートメントと呼ばれます。たとえば、LENGTH ステートメントは、変数全体に影響を及ぼしますが、特定のオブザベーション内での変数の処理方法ではありません。したがって、LENGTH ステートメントは THEN 句では使用できません。

条件が偽の場合、SAS は THEN 句を無視して、DATA ステップ内の次のステートメントに処理を進めます。

単純条件に基づいたオブザベーションの選択

次の DATA ステップでは、IF-THEN ステートメントで前述の例の条件と処理が使用されています。

```
data revise;
   set mylib.arttours;
   if City='Rome' then LandCost=LandCost + 30;
   if NumberOfEvents > Nights then Calendar='Check schedule';
   if TourGuide='Lucas' and Nights > 7 then TourGuide='Torres';
run;

proc print data=revise;
   var City Nights LandCost NumberOfEvents TourGuide Calendar;
   title 'Tour Information';
run;
```

次の出力は結果を示しています。

図 10.2 IF-THEN ステートメントによるオブザベーションの選択

Tour Information

Obs	City	Nights	LandCost	NumberOfEvents	TourGuide	Calendar
1	Rome	3	1580	7	D'Amico	Check schedule
2	Brasilia	8	3360	6	Torres	
3	London	6	2460	5	Wilson	
4	Warsaw	6	.	8	Lucas	Check schedule
5	Madrid	3	740	5	Torres	Check schedule
6	Amsterdam	4	1160	6		Check schedule

この出力は次のことを示しています。

- Rome のオブザベーションの地上コストが$30 増加した。
- 4 つのオブザベーションでイベント数がツアー日数よりも大きい。
- Brasilia では、最初のツアーガイドは Lucas で、かつツアーの宿泊日数が 7 より大きいため、ツアーガイドが Torres に置き換えられている。

代替処理の指定

SAS では、すべてのオブザベーションの変数に値を割り当てなくても、すべてのオブザベーションで変数が作成されることに注意してください。前述の出力では、2 つのオブザベーションの Calendar 値がブランクです。次の例のように、2 番目の IF-THEN ステートメントで異なる値を割り当てられます。

```
if NumberOfEvents > Nights then Calendar='Check schedule';
if NumberOfEvents <= Nights then Calendar='No problems';
```

記号<=は以下という意味です。この場合、Events と Nights の値は、IF 条件ごとに 1 回ずつ、あわせて 2 回比較されます。より効率的な代替処理の指定方法は、ELSE ステートメントを使用することです。

ELSE *action*;

ELSE ステートメントでは、IF 条件が偽の場合に行われる代替処理が指定されます。次に示すように、対応する IF-THEN ステートメントの直後に記述する必要があります。

```
if NumberOfEvents > Nights then Calendar='Check schedule';
else Calendar='No problems';
```

REVISE2 DATA ステップでは、この ELSE ステートメントが前述の DATA ステップに追加されています。

```
data revise2;
   set mylib.arttours;
   if City='Rome' then LandCost=LandCost + 30;
   if NumberOfEvents > Nights then Calendar='Check schedule';
   else Calendar='No problems';
```

```
   if TourGuide='Lucas' and Nights > 7 then TourGuide='Torres';
run;

proc print data=revise2;
   var City Nights LandCost NumberOfEvents TourGuide Calendar;
   title 'Tour Information';
run;
```

次の出力は結果を示しています。

図 10.3 *ELSE ステートメントによる代替処理の指定*

Tour Information

Obs	City	Nights	LandCost	NumberOfEvents	TourGuide	Calendar
1	Rome	3	1580	7	D'Amico	Check schedule
2	Brasilia	8	3360	6	Torres	No problems
3	London	6	2460	5	Wilson	No problems
4	Warsaw	6	.	8	Lucas	Check schedule
5	Madrid	3	740	5	Torres	Check schedule
6	Amsterdam	4	1160	6		Check schedule

一連の相互排他的条件の作成

IF-THEN ステートメントの後に ELSE ステートメントを使用すると、IF 条件が偽の場合の代替処理が 1 つ指定されます。ただし、多くの場合、一連の相互排他的条件が伴い、それぞれ別々の処理が必要になります。この例では、ツアー価格を高、中、低に分類できます。一連の IF-THEN ステートメントと ELSE ステートメントによって、ツアー価格が適切に分類されます。

```
if LandCost >= 2500 then Price='High';
else if LandCost >= 1500 then Price='Medium';
     else Price='Low';
```

記号>=は、以上という意味です。一連のステートメントの実行方法を示すために、Amsterdam と Brasilia という 2 つのオブザベーションについて考えてみます。Amsterdam の LandCost 値は 1160、Brasilia の値は 3360 です。

LandCost の値が 1160 である場合、SAS はプログラムを次の方法で処理します。

1. 1160 が 2500 以上かどうかを検証し、比較が偽であると特定すると、THEN 句を無視して、ELSE ステートメントに進みます。
2. ELSE ステートメントの処理は、もう一つの条件を評価することです。1160 が 1500 以上かどうかを検証し、比較が偽であると特定すると、THEN 句を無視して、付随する ELSE ステートメントに進みます。
3. ELSE ステートメントの処理を実行し、Price に値 Low を割り当てます。

LandCost の値が 3360 である場合、SAS はプログラムを次の方法で処理します。

1. 3360 が 2500 以上かどうかを検証し、比較が真であると特定すると、THEN 句の処理を実行します。Price の値が High になります。

2. ELSE ステートメントは無視します。残りの一連のステートメント全体が、最初の ELSE ステートメントに含まれているため、一連のステートメントの残りの処理はすべてスキップされます。

これらの処理の簡単な考え方として、一連の相互排他的な IF-THEN/ELSE ステートメントでオブザベーションが 1 つの条件を満たすと、その THEN 処理が行われ、残りのステートメントはスキップされることに留意してください。したがって、プログラムの効率性を高めるには、IF-THEN/ELSE ステートメントを、最も一般的な条件が先にくるように配置します。

次の DATA ステップには、前述の一連のステートメントが含まれています。

```
data prices;
   set mylib.arttours;
   if LandCost >= 2500 then Price='High  ';
   else if LandCost >= 1500 then Price='Medium';
          else Price='Low';
run;

proc print data=prices;
   var City LandCost Price;
   title 'Tour Prices';
run;
```

次の出力は結果を示しています。

図 10.4 *IF-THEN/ELSE ステートメントによる相互排他的な値の割り当て*

Tour Prices

Obs	City	LandCost	Price
1	Rome	1550	Medium
2	Brasilia	3360	High
3	London	2460	Medium
4	Warsaw	.	Low
5	Madrid	740	Low
6	Amsterdam	1160	Low

4 番目のオブザベーションの Price 値に注目してください。Warsaw ツアーの LandCost 値は欠損値なので、Price 値は Low です。欠損値が可能な限り最小の数値であることに留意してください。

条件の作成

条件の作成について

IF-THEN ステートメントを使用すると、比較が行われます。ある値が別の値と等しいかどうか、別の値よりも大きいかどうかなどを SAS は特定する必要があります。

SAS での主な比較演算子は次の 6 つになります。

表 10.1 比較演算子

記号	ニーモニック演算子	意味
=	EQ	等しい
¬=、^=、~=	NE	等しくない(キーボードに応じて、¬、^または~の記号)
>	GT	より大きい
<	LT	未満
>=	GE	以上
<=	LE	以下

表内の記号は算術記号に基づいています。ニーモニック演算子と呼ばれる文字の省略形には、該当する記号と同じ効果があります。好きな形式を使用してください。ただし、ニーモニック演算子は比較にしか使用できないことに注意してください。たとえば、割り当てステートメントの等号はニーモニック演算子ではなく記号=で表す必要があります。次のステートメントは両方とも、ツアーの宿泊日数と 6 を比較しています。

- `if Nights >= 6 then Stay='Week+';`
- `if Nights ge 6 then Stay='Week+';`

比較演算子の両側の項には、変数、式または定数を指定できます。正しい演算子を使用している限り、特定の項をどの側に記述するかは問題ではありません。次の比較はすべて、SAS ステートメントで使用できるように正しく作成されています。

- `Guide=' '`
- `LandCost ne .`
- `LandCost lt 1200`
- `600 ge LandCost`
- `NumberOfEvents / Nights > 2`
- `2 <= NumberOfEvents / Nights`

単純条件に基づいたオブザベーションの選択

次の DATA ステップは、これらの条件のいくつかを示しています。

```
data changes;
   set mylib.arttours;
   if Nights >= 6 then Stay='Week+';
   else Stay = 'Days';
   if LandCost ne . then Remarks='OK  ';
   else Remarks = 'Redo';
   if LandCost lt 1200 then Budget='Low   ';
   else Budget = 'Medium';
   if NumberOfEvents / Nights > 2 then Pace='Too fast';
```

```
   else Pace='OK';
run;

proc print data=changes;
   var City Nights LandCost NumberOfEvents Stay Remarks Budget Pace;
   title 'Tour Information';
run;
```

次の出力は結果を示しています。

図 10.5 特定の条件に従って値を変数に割り当て

Tour Information

Obs	City	Nights	LandCost	NumberOfEvents	Stay	Remarks	Budget	Pace
1	Rome	3	1550	7	Days	OK	Medium	Too fast
2	Brasilia	8	3360	6	Week+	OK	Medium	OK
3	London	6	2460	5	Week+	OK	Medium	OK
4	Warsaw	6	.	8	Week+	Redo	Low	OK
5	Madrid	3	740	5	Days	OK	Low	OK
6	Amsterdam	4	1160	6	Days	OK	Low	OK

条件に複数の比較を使用

複数の比較を指定

これらの演算子を使用して条件に複数の比較を指定できます。

- **`& or AND`**
- **`| or OR`**

条件には任意の数の AND 演算子または OR 演算子、あるいはその両方を含められます。

条件がすべて真になる必要がある場合の比較

比較が AND で接続されている場合、条件が真になるにはすべての比較が真である必要があります。次の例を考えてみます。

```
if City='Brasilia' and TourGuide='Lucas' then Remarks='Bilingual';
```

オブザベーションの City の値が **`Brasilia`** で TourGuide の値が **`Lucas`** の場合、比較は真になります。

一般的な比較は、ある値が 2 つの数量の間かどうか、すなわち、ある数量より大きくてもう 1 つの数量より小さいかどうかを特定するものです。たとえば、LandCost の値が 2000 以上かつ 2500 以下のオブザベーションを選択するには、AND を使用して比較を記述します。

```
if LandCost >= 2000 and LandCost <= 2500 then Price = '2000-2500';
```

この比較のより簡単な記述方法は次のとおりです。

```
if 2000 <= LandCost <= 2500 then Price = '2000-2500';
```

この比較はその前の比較と同じ意味です。演算子<、<=、>、>=、相当するニーモニックのいずれもがこのようにして使用できます。

次の DATA ステップには、これらの複数の比較ステートメントが含まれています。

```
data showand;
   set mylib.arttours;
   if City='Brasilia' and TourGuide='Lucas' then Remarks='Bilingual';
   if 2000 <= LandCost <= 2500 then Price='2000-2500';
run;

proc print data=showand;
   var City LandCost TourGuide Remarks Price;
   title 'Tour Information';
run;
```

次の出力は結果を示しています。

図 10.6 *複数の比較の実行時に AND を使用*

Tour Information

Obs	City	LandCost	TourGuide	Remarks	Price
1	Rome	1550	D'Amico		
2	Brasilia	3360	Lucas	Bilingual	
3	London	2460	Wilson		2000-2500
4	Warsaw	.	Lucas		
5	Madrid	740	Torres		
6	Amsterdam	1160			

条件が 1 つだけ真になる必要がある場合

比較が OR で接続されている場合、条件が真になるために必要なのは、比較のうちのいずれか 1 つが真であるのみです。次の例について考えます。

```
if LandCost gt 1500 or LandCost / Nights gt 400 then Level='Deluxe';
```

地上コストが$1500 を超えるかまたは 1 泊のコストが$200 を超えるオブザベーション、あるいはその両方に該当するオブザベーションはすべて条件を満たします。次の DATA ステップはこの条件を示しています。

```
data showor;
   set mylib.arttours;
   if LandCost gt 1500 or LandCost / Nights gt 400 then Level='Deluxe';
run;
proc print data=showor;
   var City LandCost Nights Level;
   title 'Tour Information';
run;
```

次の出力は結果を示しています。

図 10.7 複数の比較の実行時に OR を使用

Tour Information

Obs	City	LandCost	Nights	Level
1	Rome	1550	3	Deluxe
2	Brasilia	3360	8	Deluxe
3	London	2460	6	Deluxe
4	Warsaw	.	6	
5	Madrid	740	3	
6	Amsterdam	1160	4	

否定演算子を AND または OR とともに使用

否定演算子と OR を結合する場合には注意が必要です。多くの場合、実際に必要な演算子は AND です。たとえば、変数 TourGuide のデータにいくつか問題があるとします。Brasilia のオブザベーションではツアーガイドと予備ツアーガイドが両方とも **Lucas** です。Amsterdam のオブザベーションではツアーガイド名が欠損しています。TourGuide に問題がないオブザベーションに OK のラベルを付ける必要があります。IF 条件を記述する際は OR と AND のどちらを使用すればよいでしょうか。

次の DATA ステップは両方の条件を示しています。

```
data showorand;
   set mylib.arttours;
   if TourGuide ne BackUpGuide or TourGuide ne ' ' then GuideCheckUsingOR='OK';
   else GuideCheckUsingOR='No';
   if TourGuide ne BackUpGuide and TourGuide ne ' ' then GuideCheckUsingAND='OK';
   else GuideCheckUsingAND='No';
run;

proc print data = showorand;
   var City TourGuide BackUpGuide GuideCheckUsingOR GuideCheckUsingAND;
   title 'Negative Operators with OR and AND';
run;
```

次の出力は結果を示しています。

図 10.8 比較の実行時に否定演算子を使用

Negative Operators with OR and AND

Obs	City	TourGuide	BackUpGuide	GuideCheckUsingOR	GuideCheckUsingAND
1	Rome	D'Amico	Torres	OK	OK
2	Brasilia	Lucas	Lucas	OK	No
3	London	Wilson	Lucas	OK	OK
4	Warsaw	Lucas	D'Amico	OK	OK
5	Madrid	Torres	D'Amico	OK	OK
6	Amsterdam		Vandever	OK	No

GuideCheckUsingOR を作成する IF-THEN/ELSE ステートメントでは、条件を真にするために必要な真である比較は、単に 1 つのみです。データセット MYLIB.ARTTOURS の Brasilia オブザベーションと Amsterdam オブザベーションについて、次の条件が真になることを確認します。

- Brasilia のオブザベーションでは、TourGuide には欠損値がないので、比較 **`TourGuide NE ' '`** は真です。
- Amsterdam の場合、比較 **`TourGuide NE BackUpGuide`** が真です。

それぞれのオブザベーションで OR 比較が 1 つは真になるため、GuideCheckUsingOR ではすべてのオブザベーションに OK のラベルが付きます。GuideCheckUsingAND を作成する IF-THEN/ELSE ステートメントの方がよい結果が出ています。すなわち、AND 演算子では、TourGuide と BackUpGuide の値が同一ではなく、なおかつ欠損もしていないオブザベーションが選択されます。

AND と OR が必要な複雑な比較を使用

条件には AND 演算子と OR 演算子の両方を含められます。その場合、AND 演算子が OR 演算子の前に評価されます。次の例では、都市のリストとガイドのリストが指定されます。

```
   /* first attempt */
if City='Brasilia' or City='Rome' and TourGuide='Lucas' or
   TourGuide="D'Amico" then Topic= 'Art history';
```

AND で接続された項目が最初に結合されます。

```
City='Rome' and TourGuide='Lucas'
```

その後で次の OR 比較が行われます。

```
City='Brasilia'
    or
City='Rome' and TourGuide='Lucas'
    or
TourGuide="D'Amico"
```

City の比較と TourGuide の比較をグループ化するには、かっこを使用します。

```
   /* correct method */
if (City='Brasilia' or City='Rome') and
```

```
(TourGuide='Lucas' or TourGuide="D'Amico") then
Topic='Art history';
```

かっこ内の比較が最初に評価され、その結果がより大きな比較の項として使用されます。どの条件でもかっこを使用して、比較のグループ化を制御したり、条件を見やすくしたりできます。

次の DATA ステップは、これらの条件を示しています。

```
data combine;
   set mylib.arttours;
   if (City='Brasilia' or City='Rome') and
      (TourGuide='Lucas' or TourGuide="D'Amico") then
      Topic='Art history';
run;

proc print data=combine;
   var City TourGuide Topic;
   title 'Tour Information';
run;
```

次の出力は結果を示しています。

図 10.9 *かっこを使用して AND と OR で比較を結合*

Tour Information

Obs	City	TourGuide	Topic
1	Rome	D'Amico	Art history
2	Brasilia	Lucas	Art history
3	London	Wilson	
4	Warsaw	Lucas	
5	Madrid	Torres	
6	Amsterdam		

数値の比較の省略

数値の比較については 2 つの考慮点を知っておくと特に役立ちます。

- 省略形式の比較を使用できます。
- OR を使用した省略比較には注意が必要です。

計算においては、TRUE の値が 1 で FALSE の値が 0 です。SAS では、次が当てはまります。

- 0 以外の数値が真になります。
- 0 または欠損値が偽になります。

したがって、数値変数や数式を単独で条件式として使用できます。その値が 0 以外の数の場合、条件は真になります。その値が 0 または欠損値の場合、条件は偽になります。

次の例では、指定オブザベーションに LandCost の値が存在する場合のみ、変数 Remarks に値が割り当てられます。

```
if LandCost then Remarks='Ready to budget';
```

このステートメントは次に相当します。

```
if LandCost ne . and LandCost ne 0 then Remarks='Ready to budget';
```

OR を使用した比較を省略する場合は注意が必要です。予想外の結果を招きやすくなります。たとえば、この IF-THEN ステートメントで 6 泊または 8 泊のツアーを選択するとします。

```
   /* first try */
if Nights=6 or 8 then Stay='Medium';
```

この条件は次の比較のように処理されます。

```
Nights=6
   or
   8
```

2 番目の比較では Nights の値は使用されません。比較には、数字の 8 が単独で使用されます。数字の 8 は 0 でも欠損値でもないため、常に値が TRUE になります。一連の OR 比較では 1 つの比較が真であるだけで条件も真になるので、すべてのオブザベーションでこの条件は真になります。

次の比較では、6 泊または 8 泊のオブザベーションが正しく選択されます。

```
   /* correct way */
if Nights=6 or Nights=8 then Stay='Medium';
```

次の DATA ステップには、これらの IF-THEN ステートメントが含まれています。

```
   data morecomp;
      set mylib.arttours;
      if LandCost then Remarks='Ready to budget';
      else Remarks='Need land cost';
      if Nights=6 or Nights=8 then Stay='Medium';
      else Stay='Short';
   run;

   proc print data=morecomp;
      var City Nights LandCost Remarks Stay;
      title 'Tour Information';
   run;
```

次の出力は結果を示しています。

図 10.10 数値の比較の省略

Tour Information

Obs	City	Nights	LandCost	Remarks	Stay
1	Rome	3	1550	Ready to budget	Short
2	Brasilia	8	3360	Ready to budget	Medium
3	London	6	2460	Ready to budget	Medium
4	Warsaw	6	.	Need land cost	Medium
5	Madrid	3	740	Ready to budget	Short
6	Amsterdam	4	1160	Ready to budget	Short

文字の比較

文字の比較のタイプ

文字の比較を行うとき特殊な状況が発生し、次のタスクの実行が必要になる場合があります。

- 大文字と小文字を比較します。
- 特定の文字グループで始まる値をすべて選択します。
- 特定の文字範囲で始まる値をすべて選択します。
- 別の文字値内全体から特定の値を検索します。

大文字と小文字の比較

SAS は、比較において大文字と小文字を区別します。たとえば、値 **Madrid** と **MADRID** は等しくありません。大文字でも小文字でも発生する可能性がある値を比較するには、UPCASE 関数を使用して大文字の値を生成します。その後、次に示すように、2 つの大文字の値を比較します。

```
data newguide;
   set mylib.arttours;
   if upcase(City)='MADRID' then TourGuide='Balarezo';
run;

proc print data=newguide;
   var City TourGuide;
   title 'Tour Guides';
run;
```

比較では、City 値の大文字バージョンが生成され、大文字定数 MADRID と比較されます。オブザベーション内の City 値は元のままです。

次の出力は結果を示しています。

図 10.11 大文字比較によって生成されたデータセット

Tour Guides

Obs	City	TourGuide
1	Rome	D'Amico
2	Brasilia	Lucas
3	London	Wilson
4	Warsaw	Lucas
5	Madrid	Balarezo
6	Amsterdam	

ここでは、UPCASE 関数で **Madrid** の大文字値と **MADRID** が比較されたので、**Balarezo** が **Madrid** のツアーガイドとして割り当てられています。UPCASE 関数では、この 2 つの値を等しい値として読み取ることができます。

同じ文字グループで始まる値をすべて選択

名前の頭文字が D のツアーガイド全員などの、文字値グループの選択が必要になる場合があります。

デフォルトでは、長さの異なる値を比較するには、短い方の値の末尾にブランクを追加し、その結果を長い方の値と比較して検証します。次に例を示します。

```
   /* first attempt */
if Tourguide='D' then Chosen='Yes';
else Chosen='No';
```

SAS は比較を次のように解釈します。

```
TourGuide='D       '
```

D の後に 7 つのブランクが続きます(TourGuide がカラム入力によって作成された文字変数なので、長さ 8 バイトであるため)。TourGuide の値が D の 1 文字から成ることは決してないので、比較は決して真になりません。

長い値を短い方の基準に合わせて比較するには、次の例のように、演算子の後にコロン(:)を入れます。

```
   /* correct method */
if TourGuide=:'D' then Chosen='Yes';
else Chosen='No';
```

コロンを使用すると、短い値と長い値で同じ文字数が比較されます。ここでは、短い文字列には 1 文字が含まれています。したがって、長い値からは最初の文字のみが検証されます。D で始まる名前はすべて、比較が真になります。(TourGuide の値がすべて大文字で始まるかどうか不明な場合は、UPCASE 関数を使用します。)次の DATA ステップでは、D で始まる名前が選択されます。

```
   data dguide;
      set mylib.arttours;
      if TourGuide=:'D' then Chosen='Yes';
```

```
   else Chosen='No';
run;

proc print data=dguide;
   var City TourGuide Chosen;
   title 'Guides Whose Names Begin with D';
run;
```

次の出力は結果を示しています。

図 10.12 特定の文字列で始まる値をすべて選択

Guides Whose Names Begin with D

Obs	City	TourGuide	Chosen
1	Rome	D'Amico	Yes
2	Brasilia	Lucas	No
3	London	Wilson	No
4	Warsaw	Lucas	No
5	Madrid	Torres	No
6	Amsterdam		No

文字値範囲の選択

A から L または M から Z で始まるすべての名前など、特定の文字範囲から始まる値の選択が必要な場合があります。文字値の範囲を選択するには、次の点を理解する必要があります。

- コンピュータ処理では、文字に大きさがあります。A がアルファベットにおける最小の文字で、Z が最大です。したがって、比較 A<B は真で、比較 D>C も同様です。[1]
- ブランクはどの文字よりも小さくなります。

次のステートメントでは、比較演算子とコロンを結合することによって、ガイドの名前が A-L で始まるグループと M-Z で始まるグループの 2 つに分割されます。

```
if TourGuide <=: 'L' then TourGuideGroup='A-L';
   else TourGuideGroup='M-Z';
```

次の DATA ステップでは、グループが作成されます。

```
data guidegrp;
   set mylib.arttours;
   if TourGuide <=: 'L' then TourGuideGroup'A-L';
   else TourGuideGroup='M-Z';
run;

proc print data=guidegrp;
```

1 アルファベットにおける文字の大きさは、SAS を実行するすべての動作環境に当てはまります。大文字と小文字のどちらが大きいか、また、文字値内の数字をどのように処理するかなど、その他の点は動作環境によって決まります。さまざまな動作環境での文字値の並べ替え方法の詳細については、12 章, “グループ化されたオブザベーションや並べ替えられたオブザベーションの操作” (181 ページ)を参照してください。

```
    var City TourGuide TourGuideGroup;
    title 'Tour Guide Groups';
run;
```

次の出力は結果を示しています。

図 10.13 特定の文字範囲で始まる値をすべて選択

Tour Guide Groups

Obs	City	TourGuide	TourGuideGroup
1	Rome	D'Amico	A-L
2	Brasilia	Lucas	A-L
3	London	Wilson	M-Z
4	Warsaw	Lucas	A-L
5	Madrid	Torres	M-Z
6	Amsterdam		A-L

A から L で始まるすべての名前、および欠損値がグループ A-L に入ります。ブランクはどの文字よりも小さいため、欠損値はそのグループに入ります。

別の文字値内全体から値を検索

美術館やギャラリーに加えて他のアトラクションも訪問するツアーをリストするデータセットが必要です。データセット MYLIB.ARTTOURS の変数 EventDescription では、そのようなイベントが **other** と呼ばれます。ただし、**other** の語の位置はオブザベーションによって異なります。**other** が指定オブザベーションの EventDescription 値のどこに存在するかの判断方法を考えます。

INDEX 関数では、指定した文字列(excerpt)が特定の文字値(source)内に存在するかどうかが判断されます。

INDEX (*source*, *excerpt*)

source と *excerpt* の両方に、引用符で囲んだ文字列、文字変数、その他の文字関数など、任意の種類の文字式を指定できます。*excerpt* が *source* 内に発生していると、関数によって *excerpt* の最初の文字の位置が正数で返されます。発生していなければ、関数で 0 が返されます。0 より大きい値を検証すると、特定の文字列が別の文字値内に存在するかどうかを判断できます。

次のステートメントでは、文字列 **other** を含むオブザベーションが選択されます。

```
if index(EventDescription,'other') > 0 then OtherEvents='Yes';
else OtherEvents='No';
```

また、次の方法で条件を作成することもできます。

```
if index(EventDescription,'other') then OtherEvents='Yes';
else OtherEvents='No';
```

2 番目の例は、0 と欠損値以外のすべての値では条件が真になるという事実を利用しています。このステートメントが次の DATA ステップに含まれています。

```
data otherevent;
   set mylib.arttours;
```

```
   if index(EventDescription,'other') then OtherEvents='Yes';
   else OtherEvents='No';
run;

proc print data=otherevent;
   var City EventDescription OtherEvents;
   title 'Tour Events';
run;
```

次の出力は結果を示しています。

図 10.14　別の値内で文字列を検索

Tour Events

Obs	City	EventDescription	OtherEvents
1	Rome	4 M, 3 G	No
2	Brasilia	5 M, 1 other	Yes
3	London	3 M, 2 G	No
4	Warsaw	5 M, 1 G, 2 other	Yes
5	Madrid	3 M, 2 other	Yes
6	Amsterdam	3 M, 3 G	No

Brasilia および Madrid のオブザベーションでは、文字列 **other** が変数(Brasilia の場合は **5 M, 1 other** で Madrid の場合は **3 M, 2 other**)の 8 番目に見つかるので、INDEX 関数は値 8 を返します。New York については、文字列 **other** が変数(**5 M, 1 G, 2 other**)の 13 番目に見つかるので、値 13 が返されます。残りのオブザベーションでは、関数で文字列 **other** が見つからないので 0 が返されます。

要約

ステートメント

IF *condition* THEN *action*;
<ELSE *action*>;

condition が真かどうかが検証されます。条件が真の場合、THEN 句の処理が実行されます。*condition* が偽で ELSE ステートメントが存在する場合は、ELSE が処理されます。*condition* が偽で ELSE ステートメントが存在しない場合は、DATA ステップ内の次のステートメントが処理されます。*Condition* には 1 つ以上の数値または文字の比較を指定します。*action* には実行可能なステートメントを指定する必要があります。すなわち、DATA ステップの個々の反復で処理可能なステートメントです。(LENGTH など、DATA ステップ全体に影響するステートメントは、実行可能ではありません。)

SAS の処理では、0 でもなく、欠損値でもない数値が真になります。値 0 と欠損値は偽です。したがって、数値は単独で比較に使用できます。その値が 0 または欠損値の場合、比較は偽になります。それ以外の場合は、比較が真になります。

関数

INDEX(*source*, *excerpt*)
source から *excerpt* に指定された文字列を.検索します。*source* と *excerpt* の両方に、文字変数、引用符で囲んだ文字列、その他の文字関数など、任意の種類の文字式を指定できます。*excerpt* が *source* 内に存在する場合、関数によって *excerpt* の最初の文字の位置(正数)が返されます。*excerpt* が存在しない場合、関数によって 0 が返されます。

UPCASE(*argument*)
argument の大文字値が生成されます。文字変数、引用符で囲んだ文字列、その他の文字関数など、任意の種類の文字式を指定できます。

詳細情報

Base SAS 関数
Base SAS 関数は、*SAS Functions and CALL Routines: Reference* に記載されています。

比較演算子と論理演算子
詳細については、"SAS Operators in Expressions" (*SAS Language Reference: Concepts*)を参照してください。

実行可能ステートメント
IF-THEN/ELSE ステートメントでは実行可能ステートメントのみ発行できます。実行可能ステートメントと非実行可能ステートメントの詳細なリストについては、"DATA Step Statements" (*SAS Statements: Reference*)を参照してください。

IF-THEN/ELSE ステートメントと句
詳細については、"IF-THEN/ELSE Statement" (*SAS Statements: Reference*)を参照してください。

IN 演算子
IN 演算子を使用すると、値を(変数や式ではなく)一連の数値定数や文字定数と比較する際に、比較を短くすることができます。詳細については、"The IN Operator in Character Comparisons" (*SAS Language Reference: Concepts*)および"The IN Operator in Numeric Comparisons" (*SAS Language Reference: Concepts*)を参照してください。

SELECT ステートメント
条件に基づいてオブザベーションを選択する SELECT ステートメントは、一連の IF-THEN/ELSE ステートメントに相当します。条件や処理が多数に上る場合は、SELECT グループに記述すると、DATA ステップを見やすくすることができます。詳細については、"SELECT Statement" (*SAS Statements: Reference*)を参照してください。

TRUNCOVER オプション
INFILE ステートメントの TRUNCOVER オプションについては、"例で使用される入力 SAS データセット" (148 ページ)の注に記載されています。詳細については、"INFILE Statement" (*SAS Statements: Reference*)を参照してください。

11 章

オブザベーションのサブセットの作成

オブザベーションのサブセットの作成について **167**
目的 167
前提条件 168

例で使用される入力 SAS データセット **168**

新しい SAS データセットのオブザベーションを選択する **169**
条件に基づいたオブザベーションの削除 169
条件に基づいたオブザベーションの受け入れ 171
DELETE ステートメントとサブセット化 IF ステートメントの比較 171

1 つ以上の SAS データセットにオブザベーションを条件付きで書き込む **173**
OUTPUT ステートメントについて 173
複数のデータセットにオブザベーションを条件付きで書き込む例 173
複数のデータセットの書き込み時によくある間違いの回避 174
OUTPUT ステートメントの配置が重要な理由について 175
1 つ以上のデータセットに対してオブザベーションを複数回書き込む 177

要約 **178**
ステートメント 178

詳細情報 **179**

オブザベーションのサブセットの作成について

目的

このセクションでは、新しいデータセットを作成するために既存の SAS データセットから特定のオブザベーションを選択する方法を学習します。特に、このセクションでは、次について学習します。

- 入力データソースのオブザベーションの一部のみを含む新しい SAS データセットを作成する方法
- 1 つの DATA ステップを使用して、入力データソースのオブザベーションを書き込んで複数の新しい SAS データセットを作成する方法

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 3 章, “DATA ステップ処理について” (27 ページ)
- 4 章, “生データから作成する: 基本” (51 ページ)
- 5 章, “生データから作成する: 応用” (71 ページ)
- 6 章, “SAS データセットから作成する” (91 ページ)
- 10 章, “選択したオブザベーションの操作” (147 ページ)
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)

例で使用される入力 SAS データセット

Tradewinds Travel 社は、さまざまな美術館やギャラリーへのツアースケジュールを有しています。ツアーについての異なる情報は異なる SAS データセットに格納して保持しておくと便利です。ツアーデータは、次の情報を含む外部ファイルに格納されます。

```
1         2 3    4      5
------------------------------
Rome      3 1550 Medium D'Amico
Brasilia  8 3360 High   Lucas
London    6 2460 Medium Wilson
Warsaw    6    .        Lucas
Madrid    3  740 Low    Torres
Amsterdam 4 1160 Low
```

次のリストでは、入力ファイルのフィールドについて説明しています。

1 行く先の市の名前を提供します

2 ツアーの宿泊日数を示します

3 米ドルでの地上パッケージのコストを示します

4 予算の等級を示します

5 ツアーガイドの名前を提示します

次のプログラムでは、MYLIB.ARTS という永久 SAS データセットが作成されます。

```
libname mylib 'SAS-library';

data mylib.arts;
   infile 'input-file' truncover;
   input City $ 1-9 Nights 11 LandCost 13-16 Budget $ 18-23
         TourGuide $ 25-32;
run;

proc print data=mylib.arts;
   title 'Data Set MYLIB.ARTS';
run;
```

次の出力は結果を示しています。

図 11.1 データセット MYLIB.ARTS

Data Set MYLIB.ARTS

Obs	City	Nights	LandCost	Budget	TourGuide
1	Rome	3	1550	Medium	D'Amico
2	Brasilia	8	3360	High	Lucas
3	London	6	2460	Medium	Wilson
4	Warsaw	6	.		Lucas
5	Madrid	3	740	Low	Torres
6	Amsterdam	4	1160	Low	

新しい SAS データセットのオブザベーションを選択する

条件に基づいたオブザベーションの削除

SAS データセットの新規作成時に SAS データセット内の特定のオブザベーションを選択するには 2 つの方法があります。

1. 条件を満たしていないオブザベーションを削除し、必要なオブザベーションのみ保持します。
2. 条件を満たすオブザベーションのみ受け入れます。

オブザベーションを削除するには、まず IF 条件で識別して、THEN 句で DELETE ステートメントを使用します。

IF *condition* **THEN DELETE**;

オブザベーションに対する DELETE ステートメントの処理では、SAS は、現在のオブザベーションを出力データセットに書き込まず、新しいオブザベーションのために DATA ステップの先頭に速やかに戻ります。DELETE ステートメントでは、出力データセットにオブザベーションは含められませんが、入力データセットからオブザベーションが削除されることはありません。たとえば、次のステートメントでは、LandCost の欠損値を含むオブザベーションが削除されます。

```
if LandCost=. then delete;
```

次の DATA ステップには、このステートメントが含まれています。

```
data remove;
   set mylib.arts;
   if LandCost=. then delete;
run;

proc print data=remove;
   title 'Tours With Complete Land Costs';
```

```
run;
```

次の出力は結果を示しています。

図 11.2 特定の値のオブザベーションを削除

Tours With Complete Land Costs

Obs	City	Nights	LandCost	Budget	TourGuide
1	Rome	3	1550	Medium	D'Amico
2	Brasilia	8	3360	High	Lucas
3	London	6	2460	Medium	Wilson
4	Madrid	3	740	Low	Torres
5	Amsterdam	4	1160	Low	

Warsaw は LandCost の値が欠損しているオブザベーションなので、結果データセット REMOVE には含まれません。

また、外部ファイルからデータを入力する際にオブザベーションを削除することもできます。次の DATA ステップでは、REMOVE データセットと同じ SAS データセットが作成されます。

```
data remove2;
   infile 'input-file' truncover;
   input City $ 1-9 Nights 11 LandCost 13-16 Budget $ 18-23
         TourGuide $ 25-32;
   if LandCost=. then delete;
run;

proc print data=remove2;
   title 'Tours With Complete Land Costs';
run;
```

次の出力は結果を示しています。

図 11.3 外部ファイルからの読み取り時にオブザベーションを削除

Tours With Complete Land Costs

Obs	City	Nights	LandCost	Budget	TourGuide
1	Rome	3	1550	Medium	D'Amico
2	Brasilia	8	3360	High	Lucas
3	London	6	2460	Medium	Wilson
4	Madrid	3	740	Low	Torres
5	Amsterdam	4	1160	Low	

条件に基づいたオブザベーションの受け入れ

旅行代理店で、6 泊ツアーのオブザベーションのみを含むデータセットが必要とされています。選択を行う 1 つの方法は、Nights の値が 6 と等しくないオブザベーションを削除することです。

```
if Nights ne 6 then delete;
```

より直接的な方法は、条件を満たすオブザベーションのみを選択することです。サブセット化 IF ステートメントでは、指定したオブザベーションが選択されます。そこには条件のみが含まれます。

IF *condition*;

サブセット化 IF ステートメントの暗黙的な処理は常に同じです。条件が真の場合は、オブザベーションの処理が続行されます。条件が偽の場合は、オブザベーションの処理が停止され、新しいオブザベーションのために DATA ステップの先頭に戻ります。結果が元のオブザベーションのサブセットなので、このステートメントはサブセット化と呼ばれます。たとえば、Nights の値が 6 と等しいオブザベーションのみ選択する場合は、次のステートメントを指定します。

```
if Nights = 6;
```

次の DATA ステップにはサブセット化 IF ステートメントが含まれています。

```
data subset6;
   set mylib.arts;
   if nights=6;
run;

proc print data=subset6;
   title 'Six-Night Tours';
run;
```

次の出力は結果を示しています。

図 11.4 *サブセット化 IF ステートメントによるオブザベーションの選択*

Six-Night Tours

Obs	City	Nights	LandCost	Budget	TourGuide
1	London	6	2460	Medium	Wilson
2	Warsaw	6	.		Lucas

2 つのオブザベーションが 6 泊ツアーの条件を満たしました。

DELETE ステートメントとサブセット化 IF ステートメントの比較

DELETE ステートメントとサブセット化 IF ステートメントのどちらかを選択する場合に考慮する主な理由は次のとおりです。

- 通常、条件を識別するには比較が最も少なくてすむステートメントを選択する方が容易。
- 通常、否定語よりも肯定語で考える方が容易(この場合はサブセット化 IF が優先されます)。

サブセット化 IF が優先される状況がもう 1 つあります。データに欠損値やスペルミスの値がある場合は、この方法を使用するのが安全です。次の状況について考えてみます。

Tradewinds Travel 社では、低中価格ツアーの SAS データセットが必要です。Budget の値が **Low**、**Medium**、および **High** だと判明している場合、最初に考えるのは **High** の値のオブザベーションを削除することでしょう。次のプログラムでは、Budget 値が **HIGH** のオブザベーションを削除して SAS データセットを作成します。

```
   /* first attempt */
data lowmed;
   set mylib.arts;
   if upcase(Budget)='HIGH' then delete;
run;

proc print data=lowmed;
   title 'Medium and Low Priced Tours';
run;
```

次の出力は結果を示しています。

図 11.5 *削除によるサブセットの作成*

Medium and Low Priced Tours

Obs	City	Nights	LandCost	Budget	TourGuide
1	Rome	3	1550	Medium	D'Amico
2	London	6	2460	Medium	Wilson
3	Warsaw	6	.		Lucas
4	Madrid	3	740	Low	Torres
5	Amsterdam	4	1160	Low	

データセット LOWMED には、必要なツアーと Warsaw へのツアーの両方が含まれています。Warsaw オブザベーションの Budget 値は欠損しているので、Warsaw へのツアーが含まれているのは誤りです。サブセット化 IF ステートメントを使用すると、確実に、データセットに必要なオブザベーションが正確に含まれます。この DATA ステップでは、サブセット化 IF ステートメントでサブセットが作成されます。

```
   /* a safer method */
data lowmed2;
   set mylib.arts;
   if upcase(Budget)='MEDIUM' or upcase(Budget)='LOW';
run;

proc print data=lowmed2;
   title 'Medium and Low Priced Tours';
run;
```

次の出力は結果を示しています。

図 11.6 サブセット化 IF ステートメントでの正確なサブセットの作成

Medium and Low Priced Tours

Obs	City	Nights	LandCost	Budget	TourGuide
1	Rome	3	1550	Medium	D'Amico
2	London	6	2460	Medium	Wilson
3	Madrid	3	740	Low	Torres
4	Amsterdam	4	1160	Low	

結果は Budget に欠損値がない SAS データセットです。

1 つ以上の SAS データセットにオブザベーションを条件付きで書き込む

OUTPUT ステートメントについて

SAS では、1 つの DATA ステップで OUTPUT ステートメントを使用して複数の SAS データセットを作成できます。

OUTPUT <*SAS-data-set(s)*>;

データセット名を指定せずに OUTPUT ステートメントを使用すると、DATA ステートメントで名前を指定したすべてのデータセットに現在のオブザベーションが書き込まれます。選択したデータセットにオブザベーションを書き込む必要がある場合は、OUTPUT ステートメントにデータセット名を直接指定します。OUTPUT ステートメントに記述するデータセット名はすべて、DATA ステートメントにも記述する必要があります。

複数のデータセットにオブザベーションを条件付きで書き込む例

SAS データセットの一方にはツアーガイドの Lucas がガイドするツアーが含まれ、他方にはその他のガイドが案内するツアーが含まれます。次のタスクを実行することで、複数データセットへの書き込みが行われます。

- DATA ステートメントで両方のデータセット名を指定
- IF 条件を使用してオブザベーションを選択
- THEN 句と ELSE 句で OUTPUT ステートメントを使用して、適切な複数データセットにオブザベーションを出力

次の DATA ステップは、複数データセットへの書き込み方法を示します。

```
data lucastour othertours;
   set mylib.arts;
   if TourGuide='Lucas' then output lucastour;
   else output othertours;
run;
```

```
proc print data=lucastour;
   title "Data Set with TourGuide='Lucas'";
run;

proc print data=othertours;
   title "Data Set with Other Guides";
run;
```

次の出力は結果を示しています。

図 11.7 *1 つの DATA ステップで 2 つのデータセットを作成*

Data Set with TourGuide = 'Lucas'

Obs	City	Nights	LandCost	Budget	TourGuide
1	Brasilia	8	3360	High	Lucas
2	Warsaw	6	.		Lucas

Data Set with Other Guides

Obs	City	Nights	LandCost	Budget	TourGuide
1	Rome	3	1550	Medium	D'Amico
2	London	6	2460	Medium	Wilson
3	Madrid	3	740	Low	Torres
4	Amsterdam	4	1160	Low	

複数のデータセットの書き込み時によくある間違いの回避

OUTPUT ステートメントを使用すると、DATA ステップの終わりにオブザベーションは自動出力されません。したがって、DATA ステップで OUTPUT ステートメントを使用する場合は、OUTPUT ステートメントでそのステップのすべての出力をプログラムする必要があります。たとえば、前述の DATA ステップでは、LUCASTOUR と OTHERTOURS の両方に出力が送られました。比較のために、次のプログラムでは、DATA ステップで ELSE ステートメントを省略するとどうなるかを示します。

```
data lucastour2 othertour2;
   set mylib.arts;
   if TourGuide='Lucas' then output lucastour2;
run;

proc print data=lucastour2;
   title "Data Set with Guide='Lucas'";
run;

proc print data=othertour2;
   title "Data Set with Other Guides";
run;
```

次の出力は結果を示しています。

図 11.8 2 番目のデータセットへの直接出力に失敗

Data Set with Guide = 'Lucas'

Obs	City	Nights	LandCost	Budget	TourGuide
1	Brasilia	8	3360	High	Lucas
2	Warsaw	6	.		Lucas

OTHERTOUR2 には、出力が送られなかったため、オブザベーションが書き込まれていません。

OUTPUT ステートメントの配置が重要な理由について

デフォルトでは、各反復の最後に、SAS はオブザベーションを出力データセットに書き込みます。OUTPUT ステートメントを使用すると、自動出力機能は無視されます。したがって、OUTPUT ステートメントの配置場所は非常に重要です。たとえば、OUTPUT ステートメントの実行後に変数値が計算される場合は、オブザベーションを出力データセットに書き込む際にその値を使用できません。

たとえば、次の DATA ステップでは、割り当てステートメントが IF-THEN/ELSE グループの後に位置しています。

```
   /* first attempt to combine assignment and OUTPUT statements */
data lucasdays otherdays;
   set mylib.arts;
   if TourGuide='Lucas' then output lucasdays;
   else output otherdays;
   Days=Nights+1;
run;

proc print data=lucasdays;
   title "Number of Days in Lucas's Tours";
run;

proc print data=otherdays;
   title "Number of Days in Other Guides' Tours";
run;
```

次の出力は結果を示しています。

図 11.9 意図しない結果: 値の割り当て前にオブザベーションの書き込み

Number of Days in Lucas's Tours

Obs	City	Nights	LandCost	Budget	TourGuide	Days
1	Brasilia	8	3360	High	Lucas	.
2	Warsaw	6	.		Lucas	.

Number of Days in Other Guides' Tours

Obs	City	Nights	LandCost	Budget	TourGuide	Days
1	Rome	3	1550	Medium	D'Amico	.
2	London	6	2460	Medium	Wilson	.
3	Madrid	3	740	Low	Torres	.
4	Amsterdam	4	1160	Low		.

割り当てステートメントの処理前に OUTPUT ステートメントでオブザベーションが SAS データセットに書き込まれたので、すべてのオブザベーションで Days 値が欠損しています。データセットに Days 値を表示する必要がある場合は、OUTPUT ステートメントを使用する前に割り当てステートメントを使用します。次のプログラムは正しい位置を示しています。

```
   /* correct position of assignment statement */
data lucasdays2 otherdays2;
   set mylib.arts;
   Days=Nights + 1;
   if TourGuide='Lucas' then output lucasdays2;
   else output otherdays2;
run;

proc print data=lucasdays2;
   title "Number of Days in Lucas's Tours";
run;
proc print data=otherdays2;
   title "Number of Days in Other Guides' Tours";
run;
```

次の出力は結果を示しています。

図 11.10 意図した結果: 値の割り当て後にオブザベーションの書き込み

Number of Days in Lucas's Tours

Obs	City	Nights	LandCost	Budget	TourGuide	Days
1	Brasilia	8	3360	High	Lucas	9
2	Warsaw	6	.		Lucas	7

Number of Days in Other Guides' Tours

Obs	City	Nights	LandCost	Budget	TourGuide	Days
1	Rome	3	1550	Medium	D'Amico	4
2	London	6	2460	Medium	Wilson	7
3	Madrid	3	740	Low	Torres	4
4	Amsterdam	4	1160	Low		5

1 つ以上のデータセットに対してオブザベーションを複数回書き込む

OUTPUT ステートメントの処理後も、オブザベーションはプログラムデータベクトルに残っているので、引き続きプログラミングに使用できます。再度同じ SAS データセットに書き込むことも、異なるデータセットに書き込むこともできます。次の例では、2 組のデータセットが作成されます。1 組はツアーガイド名、もう 1 組は宿泊日数に基づいています。

```
data lucastour othertour weektour daytour;
   set mylib.arts;
   if TourGuide='Lucas' then output lucastour;
   else output othertour;
   if nights >= 6 then output weektour;
   else output daytour;
run;

proc print data=lucastour;
   title "Lucas's Tours";
run;

proc print data=othertour;
   title "Other Guides' Tours";
run;
proc print data=weektour;
   title 'Tours Lasting a Week or More';
run;

proc print data=daytour;
   title 'Tours Lasting Less Than a Week';
run;
```

次の出力は結果を示しています。

図 11.11 *オブザベーションを 2 つ以上のデータセットに割り当て*

Lucas's Tours

Obs	City	Nights	LandCost	Budget	TourGuide
1	Brasilia	8	3360	High	Lucas
2	Warsaw	6	.		Lucas

Other Guides' Tours

Obs	City	Nights	LandCost	Budget	TourGuide
1	Rome	3	1550	Medium	D'Amico
2	London	6	2460	Medium	Wilson
3	Madrid	3	740	Low	Torres
4	Amsterdam	4	1160	Low	

Tours Lasting a Week or More

Obs	City	Nights	LandCost	Budget	TourGuide
1	Brasilia	8	3360	High	Lucas
2	London	6	2460	Medium	Wilson
3	Warsaw	6	.		Lucas

Tours Lasting Less Than a Week

Obs	City	Nights	LandCost	Budget	TourGuide
1	Rome	3	1550	Medium	D'Amico
2	Madrid	3	740	Low	Torres
3	Amsterdam	4	1160	Low	

最初の IF-THEN/ELSE グループでは、すべてのオブザベーションがデータセット LUCASTOUR か OTHERTOUR のいずれかに書き込まれます。2 番目の IF-THEN/ELSE グループでは、同じオブザベーションが、WEEKTOUR と DAYTOUR という別のデータセットのペアに書き込まれます。このような繰り返しが可能なのは、最初の OUTPUT ステートメントの処理後も各オブザベーションがプログラムデータベクトルに残っていて、再度書き込みに使用できるためです。

要約

ステートメント

DATA *<libref-1> SAS-data-set-1< ...<libref-n> SAS-data-set-n>*;
DATA ステップで作成する SAS データセットを指定します。

DELETE;
現在のオブザベーションを削除します。DELETE ステートメントは通常 IF-THEN/ELSE グループの一部として使用されます。

IF *condition*;
condition が真かどうかが検証されます。それが真の場合、SAS は現在のオブザベーションの処理を続行します。それが真でない場合、SAS はそのオブザベーションの処理を停止し、SAS データセットへの追加は行わずに、DATA ステップの先頭に戻ります。使用される condition は IF-THEN/ELSE ステートメントの場合と同じです。このタイプの IF ステートメントは、元のオブザベーションのサブセットを生成するため、サブセット化 IF ステートメントと呼ばれます。

OUTPUT <*SAS-data-set*>;
現在のオブザベーションをすぐに *SAS-data-set* に書き込みます。オブザベーションはプログラムデータベクトルに残っています。SAS データセットへの再書き込みも含め、そのオブザベーションを引き続きプログラミングに使用できます。DATA ステップに OUTPUT ステートメントを記述した場合、オブザベーションの SAS データセットへの自動出力は行われません。DATA ステップで OUTPUT ステートメントを使用してすべての出力に対して出力先を指定する必要があります。OUTPUT ステートメントに指定する SAS データセットはすべて、DATA ステートメントにも記述する必要があります。

詳細情報

比較演算子と論理演算子
詳細については、“SAS Operators in Expressions” (*SAS Language Reference: Concepts*)および 10 章, “選択したオブザベーションの操作” (147 ページ)を参照してください。

DROP=および KEEP=データセットオプション
DROP=および KEEP=データセットオプションを使用して変数のサブセットを SAS データセットに出力する方法については、6 章, “SAS データセットから作成する” (91 ページ)を参照してください。詳細については、“DROP= Data Set Option” (*SAS Data Set Options: Reference*)および“KEEP= Data Set Option” (*SAS Data Set Options: Reference*)を参照してください。

FIRSTOBS=および OBS=データセットオプション
これらのデータセットオプションを使用して、SAS データセットの最初、中間または最後のオブザベーションを選択する方法については、6 章, “SAS データセットから作成する” (91 ページ)を参照してください。詳細については、“FIRSTOBS= Data Set Option” (*SAS Data Set Options: Reference*)および“OBS= Data Set Option” (*SAS Data Set Options: Reference*)を参照してください。

IF-THEN/ELSE ステートメント
詳細については、“IF-THEN/ELSE Statement” (*SAS Statements: Reference*)を参照してください。

DELETE ステートメントおよび OUTPUT ステートメント
詳細については、“DELETE Statement” (*SAS Statements: Reference*)および“OUTPUT Statement” (*SAS Statements: Reference*)を参照してください。

WHERE ステートメント
WHERE ステートメントでは、条件に基づいてオブザベーションが選択されます。この処理はサブセット化 IF ステートメントの処理に類似しています。WHERE ステートメントは PROC ステップで非常に役立ちます。また、一部の DATA ステップでも役立ちます。WHERE ステートメントでは、オブザベーションをプログラムデータベクトルに入れる前に選択します。これとは対照的に、サブセット化 IF ステートメントでは、すでにプログラムデータベクトル内にあるオブザベーションが選択されます。

詳細については、“オブザベーションの選択” (432 ページ)を参照してください。
“WHERE Statement” (*SAS Statements: Reference*)も参照してください。

注: 場合によっては、DATA ステップの WHERE ステートメントとサブセット化 IF ステートメントで同じ条件を指定しても、異なるサブセットが生成されることがあります。この違いについては、“WHERE Statement” (*SAS Statements: Reference*) の WHERE ステートメントの説明に記載されています。DATA ステップで WHERE ステートメントを使用する前に、必ずこの違いを理解しておいてください。この注意点を心に留めておけば、WHERE ステートメントによって DATA ステップの効率性を大幅に上げられます。

12 章

グループ化されたオブザベーションや並べ替えられたオブザベーションの操作

グループ化されたオブザベーションや並べ替えられたオブザベーションの操作について **181**
目的 181
前提条件 182
例で使用される入力 SAS データセット **182**
グループ化されたデータの操作 **183**
データのグループ化の基礎知識 183
SORT プロシジャによるオブザベーションのグループ化 184
2 つ以上の変数によるグループ化 185
グループの降順での並べ替え 186
グループ内の最初のオブザベーションまたは最後のオブザベーションの検索 . 187
並べ替えられたデータの操作 **190**
並べ替えられたデータについて 190
データの並べ替え 190
重複オブザベーションの削除 191
照合順序について 193
ASCII 照合順序 193
EBCDIC 照合順序 194
要約 **194**
プロシジャ 194
ステートメント 194
詳細情報 **195**

グループ化されたオブザベーションや並べ替えられたオブザベーションの操作について

目的

オブザベーションを特定の変数値別にグループ化したレポートや、オブザベーションをアルファベット順に並べ替えたレポートの作成が必要になることがあります。このセクションでは、次の概念を説明しています。

- オブザベーションを変数別にグループ化する方法、およびグループ化したオブザベーションの操作方法

- オブザベーションの並べ替え方法、および並べ替えたオブザベーションの操作方法

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 1 章, “SAS System について” (3 ページ)
- “DATA ステップ処理の概要” (109 ページ)
- 4 章, “生データから作成する: 基本” (51 ページ)
- 5 章, “生データから作成する: 応用” (71 ページ)
- 6 章, “SAS データセットから作成する” (91 ページ)
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)

例で使用される入力 SAS データセット

Tradewinds Travel 社には、建築物か景色のどちらかを重視するツアーについてのデータを含む外部ファイルがあります。SAS データセットにデータを作成し、それらのツアーのオブザベーションをグループ化すると、各グループのレポートを別々に作成できます。さらに、オブザベーションを国のアルファベット順にする必要がある場合は、並べ替えることができます。外部ファイルは次のようになります。

```
1           2             3  4    5
----------------------------------------
Spain       architecture  10 1020 World
Japan       architecture   8 1440 Express
Switzerland scenery        9 1468 World
Brazil      architecture   8 1150 World
Ireland     scenery        7 1116 Express
New Zealand scenery       16 2978 Southsea
Italy       architecture   8  936 Express
Greece      scenery       12 1396 Express
```

次のリストでは、入力ファイルのフィールドについて説明しています。

1 行き先国名を示します

2 ツアーで重視する領域を示します

3 ツアーの宿泊日数を示します

4 米ドルでの地上パッケージのコストを示します

5 ツアー業者名のリスト

次の DATA ステップでは、永久 SAS データセット MYLIB.ARCH_OR_SCEN が作成されます。

```
libname mylib 'SAS-library';

data mylib.arch_or_scen;
   infile 'input-file' truncover;
```

```
   input Country $ 1-11 TourType $ 13-24 Nights LandCost Vendor $;
run;

proc print data=mylib.arch_or_scen;
   title 'Data Set MYLIB.ARCH_OR_SCEN';
run;
```

DATA ステップの後の PROC PRINT ステートメントによって、MYLIB.ARCH_OR_SCEN データセットが次のように表示されます。

図 12.1 *データセット MYLIB.ARCH_OR_SCEN*

Data Set MYLIB.ARCH_OR_SCEN

Obs	Country	TourType	Nights	LandCost	Vendor
1	Spain	architecture	10	1020	World
2	Japan	architecture	8	1440	Express
3	Switzerland	scenery	9	1468	World
4	Brazil	architecture	8	1150	World
5	Ireland	scenery	7	1116	Express
6	New Zealand	scenery	16	2978	Southsea
7	Italy	architecture	8	936	Express
8	Greece	scenery	12	1396	Express

グループ化されたデータの操作

データのグループ化の基礎知識

データをグループ化する基本的方法は、BY ステートメントを使用することです。

BY *list-of-variables*;

BY ステートメントは、DATA ステップで SET、MERGE、MODIFY、UPDATE ステートメントのいずれかと一緒に使用したり、SAS プロシジャで使用したりできます。

SET、MERGE、MODIFY、UPDATE ステートメントのいずれかを使用してグループ化したデータを操作するには、データが次の条件を満たしている必要があります。

- オブザベーションが外部ファイルではなく SAS データセット内に存在する必要があります。
- グループを定義する変数を BY ステートメントに記述する必要があります。
- 入力データセット内のオブザベーションはすべて、BY ステートメントに指定する変数に従って、数値または文字の昇順または降順にするか、カレンダ月やフォーマットされた値などのなんらかの形でグループ化する必要があります。

注: MODIFY ステートメントを使用する場合、入力データを順序どおりに並べる必要はありません。ただし、データを並べ替えるとパフォーマンスを向上させられます。

3 番目の条件が満たされていない場合、データは SAS データセットに格納されていますが、必要なグループ別に並べ替えられていません。SORT プロシジャを使用してデータを並べ替えられます(次セクションで説明します)。

SAS データセットをなんらかの順序で並べ替えると、BY ステートメントを使用して、1 つ以上の共通変数の値をグループ化できるようになります。

SORT プロシジャによるオブザベーションのグループ化

入力データセット内のすべてのオブザベーションは特定の順序で並べる必要があります。この条件を満たすには、MYLIB.ARCH_OR_SCEN 内のオブザベーションを TourType の値である **architecture** または **scenery** で並べ替えます。オブザベーションを TourType で並べ替えるには、SORT プロシジャを使用します。

```
proc sort data=mylib.arch_or_scen out=tourorder;
   by TourType;
run;
```

SORT プロシジャでは、データセット MYLIB.ARCH_OR_SCEN が、TourType 値に従ってアルファベット順に並べ替えられます。並べ替えられたオブザベーションは、OUT= オプションで指定した新しいデータセットに入れられます。この例では、TOURORDER が並べ替えられたデータセットです。OUT=オプションを省略すると、並べ替え済みバージョンのデータセットがデータセット MYLIB.ARCH_OR_SCEN と置換されます。

SORT プロシジャでは、並べ替えられたデータセット以外の出力は生成されません。SAS ログのメッセージによって、SORT プロシジャが実行されたことが示されます。

ログ 12.1 *SORT プロシジャの正常実行を示すメッセージ*

```
880  proc sort data=mylib.arch_or_scen out=tourorder; 881     by TourType; 882  run; NOTE: There were
8 observations read from the data set MYLIB.ARCH_OR_SCEN.NOTE: The data set WORK.TOURORDER has 8
observations and 5 variables.NOTE: PROCEDURE SORT used (Total process time): real time           0.20
seconds cpu time            0.04 seconds
```

並べ替えられたデータセットを参照するには、プログラムに PROC PRINT ステップを追加します。

```
proc sort data=mylib.arch_or_scen out=tourorder;
   by TourType;
run;

proc print data=tourorder;
   var TourType Country Nights LandCost Vendor;
   title 'Tours Sorted by Architecture or Scenery';
run;
```

次の出力は結果を示しています。

図 12.2 並べ替えられた出力の表示

Tours Sorted by Architecture or Scenery

Obs	TourType	Country	Nights	LandCost	Vendor
1	architecture	Spain	10	1020	World
2	architecture	Japan	8	1440	Express
3	architecture	Brazil	8	1150	World
4	architecture	Italy	8	936	Express
5	scenery	Switzerland	9	1468	World
6	scenery	Ireland	7	1116	Express
7	scenery	New Zealand	16	2978	Southsea
8	scenery	Greece	12	1396	Express

デフォルトでは、グループが BY 値の昇順(最小値から最大値へ)で並べ替えられます。データセットを並べ替えても、内部の変数の順序は変更されません。ただし、このセクションの大部分の例では、PRINT プロシジャで VAR ステートメントが使用され、BY 変数が最初の列に表示されています。(このドキュメントで使用される PRINT プロシジャおよびその他のプロシジャでは、BY グループごとに別々のレポートを作成することもできます。)

2 つ以上の変数によるグループ化

オブザベーションは必要な数の変数でグループ化できます。この例では、TourType、Vendor および LandCost でオブザベーションをグループ化します。

```
proc sort data=mylib.arch_or_scen out=tourorder2;
   by TourType Vendor LandCost;
run;

proc print data=tourorder2;
   var TourType Vendor LandCost Country Nights;
   title 'Tours Grouped by Type of Tour, Vendor, and Price';
run;
```

次の出力は結果を示しています。

図 12.3 複数の変数によるグループ化

Tours Grouped by Type of Tour, Vendor, and Price

Obs	TourType	Vendor	LandCost	Country	Nights
1	architecture	Express	936	Italy	8
2	architecture	Express	1440	Japan	8
3	architecture	World	1020	Spain	10
4	architecture	World	1150	Brazil	8
5	scenery	Express	1116	Ireland	7
6	scenery	Express	1396	Greece	12
7	scenery	Southsea	2978	New Zealand	16
8	scenery	World	1468	Switzerland	9

この例に示すように、オブザベーションは、グループ内で指定された最初の変数、2 番目に指定された変数(以下同様に続く)の順にグループ化されます。すべての変数によって定義されたグループのそれぞれにオブザベーションが 1 つだけ含まれます。この例では、2 つの変数ですべてのオブザベーションの値が同じになることはありません。言い換えれば、この例では重複するエントリがありません。

グループの降順での並べ替え

TourType 別にグループ化されたデータセットでは、**architecture** のグループが **scenery** のグループより前に位置します。これは **architecture** の最初の文字“a”が、コンピュータ処理においては“s”よりも小さいためです。(照合順序と呼ばれる文字の順序については、このセクションで後述します。)特定の変数を降順にするには、SORT プロシジャの BY ステートメントで変数名の前に DESCENDING オプションを配置します。次の例のオブザベーションのグループ化では、TourType は降順ですが Vendor と LandCost は昇順です。

```
proc sort data=mylib.arch_or_scen out=tourorder3;
   by descending TourType Vendor LandCost;
run;

proc print data=tourorder3;
   var TourType Vendor LandCost Country Nights;
   title 'Descending Order of TourType';
run;
```

次の出力は結果を示しています。

図 12.4 降順で並べ替えられたオブザベーションと昇順で並べ替えられたオブザベーションの結合

Descending Order of TourType

Obs	TourType	Vendor	LandCost	Country	Nights
1	scenery	Express	1116	Ireland	7
2	scenery	Express	1396	Greece	12
3	scenery	Southsea	2978	New Zealand	16
4	scenery	World	1468	Switzerland	9
5	architecture	Express	936	Italy	8
6	architecture	Express	1440	Japan	8
7	architecture	World	1020	Spain	10
8	architecture	World	1150	Brazil	8

グループ内の最初のオブザベーションまたは最後のオブザベーションの検索

データセット全体を表示する必要がない場合、建築物を呼びものとする最も安いツアーと景色を呼びものとする最も安いツアーのみを含むデータセットを作成できます。

まず、データセットを TourType と LandCost 順に並べ替えます。

```
proc sort data=mylib.arch_or_scen out=tourorder4;
   by TourType LandCost;
run;

proc print data=tourorder4;
   var TourType LandCost Country Nights Vendor;
   title 'Tours Arranged by TourType and LandCost';
run;
```

次の出力は結果を示しています。

図 12.5 *並べ替えて最も安いツアーを探す*

Tours Arranged by TourType and LandCost

Obs	TourType	LandCost	Country	Nights	Vendor
1	architecture	936	Italy	8	Express
2	architecture	1020	Spain	10	World
3	architecture	1150	Brazil	8	World
4	architecture	1440	Japan	8	Express
5	scenery	1116	Ireland	7	Express
6	scenery	1396	Greece	12	Express
7	scenery	1468	Switzerland	9	World
8	scenery	2978	New Zealand	16	Southsea

LandCost を昇順で並べ替えたので、各 TourType 値の最初のオブザベーションで LandCost 値が最も小さくなります。DATA ステップの各 BY グループの最初のオブザベーションを検索できる場合、サブセット化 IF ステートメントを使用してそのオブザベーションを選択できます。SAS では、TourType の各値の最初のオブザベーションを検索する方法が提供されます。

DATA ステップで BY ステートメントを使用すると、BY ステートメントの変数ごとに追加変数が 2 つずつ自動作成されます。一方の変数は FIRST.*variable* (このとき *variable* は BY 変数名)と命名されます。もう一方の変数は LAST.*variable* と命名されます。これらの値は 1 か 0 のどちらかです。プログラムデータベクトルに存在し、DATA ステップのプログラミングに使用できますが、作成される SAS データセットには追加されません。たとえば、DATA ステップがこれらのステートメントで始まるとします。

```
data lowcost;
   set tourorder4;
   by TourType;
   ...more SAS statements...
run;
```

BY ステートメントによって、FIRST.TOURTYPE と呼ばれる変数が 1 つ、LAST.TOURTYPE と呼ばれる変数がもう 1 つ作成されます。値 **architecture** を含む最初のオブザベーションの処理時に、FIRST.TOURTYPE の値が 1 になります。値 **architecture** を含む他のオブザベーションでは、0 になります。同様に、値 **architecture** を含む最後のオブザベーションの処理時に、LAST.TOURTYPE の値が 1 になります。値 **architecture** を含む他のオブザベーションでは、0 になります。scenery グループのオブザベーションについても同じ結果が発生します。

FIRST.変数と LAST.変数は出力データセットに書き込まれないため、PRINT プロシジャでその値を表示することはできません。したがって、FIRST.変数と LAST.変数の値を表示する最も簡単な方法は、その値を他の変数に割り当てることです。この例では、FIRST.TOURTYPE の値が FirstTour という変数に、LAST.TOURTYPE の値が LastTour という変数に割り当てられます。

```
data temp;
   set tourorder4;
```

```
   by TourType;
   FirstTour=first.TourType;
   LastTour=last.TourType;
run;
proc print data=temp;
   var Country Tourtype FirstTour LastTour;
   title 'Specifying FIRST.TOURTYPE and LAST.TOURTYPE';
run;
```

次の出力は結果を示しています。

図 12.6 FIRST.値とLAST.値の表示

Specifying FIRST.TOURTYPE and LAST.TOURTYPE

Obs	Country	TourType	FirstTour	LastTour
1	Italy	architecture	1	0
2	Spain	architecture	0	0
3	Brazil	architecture	0	0
4	Japan	architecture	0	1
5	Ireland	scenery	1	0
6	Greece	scenery	0	0
7	Switzerland	scenery	0	0
8	New Zealand	scenery	0	1

このデータセットでは、Italy が値 **architecture** を含む最初のオブザベーションです。そのオブザベーションの FIRST.TOURTYPE の値は 1 です。Italy は値 **architecture** 含む最後のオブザベーションではないので、LAST.TOURTYPE の値は 0 です。Spain と Brazil は値 **architecture** を含む最初のオブザベーションでも最後のオブザベーションでもありません。その FIRST.TOURTYPE と LAST.TOURTYPE はどちらも 0 です。Japan は値 **architecture** を含む最後のオブザベーションです。LAST.TOURTYPE の値は 1 です。同じルールが **scenery** グループのオブザベーションに適用されます。

これでサブセット化 IF ステートメントで FIRST.TOURTYPE を使用する準備ができました。データを TourType と LandCost 順に並べ替えている場合、ツアーの各種類の最初のオブザベーションを選択すると、そのカテゴリにおける任意のツアーの最低価格が得られます。

```
proc sort data=mylib.arch_or_scen out=tourorder4;
   by TourType LandCost;
run;

data lowcost;
   set tourorder4;
   by TourType;
   if first.TourType;
run;

proc print data=lowcost;
   title 'Least Expensive Tour for Each Type of Tour';
```

```
run;
```

次の出力は結果を示しています。

図 12.7 各 BY グループから 1 つずつオブザベーションを選択

Least Expensive Tour for Each Type of Tour

Obs	Country	TourType	Nights	LandCost	Vendor
1	Italy	architecture	8	936	Express
2	Ireland	scenery	7	1116	Express

並べ替えられたデータの操作

並べ替えられたデータについて

デフォルトでは、グループは BY 値の昇順で表示されます。場合によっては、オブザベーションのグループ化が可能だということではなく、オブザベーションの並べ替え順序を強調する必要があります。たとえば、ツアーを国のアルファベット順に並べる必要があるとします。

データを特定の順序で並べ替えるには、グループ化されたデータの場合と同様に SORT プロシジャを使用します。グループ化よりも並べ替え順序の方が重要な場合、通常、結果データセットでは指定 BY 値を含むオブザベーションは 1 つしか必要ありません。したがって、重複オブザベーションの削除が必要になる場合があります。

動作環境の情報

SORT プロシジャは、SAS の一部として提供される並べ替えユーティリティか、ホスト動作環境で提供される並べ替えユーティリティのどちらかにアクセスします。このドキュメントの例ではすべて SAS 並べ替えユーティリティが使用されます。動作環境ユーティリティでは、このセクションで後述する NODUPRECS オプションなどの特定オプションが受け入れられない場合があります。デフォルトの並べ替えユーティリティは各サイトで設定されます。使用可能なユーティリティの詳細については、使用している動作環境に対応するドキュメントを参照してください。

データの並べ替え

次の例では、データセット MYLIB.ARCH_OR_SCEN が Country 順に並べ替えられます。

```
proc sort data=mylib.arch_or_scen out=bycountry;
   by Country;
run;

proc print data=bycountry;
   title 'Tours in Alphabetical Order by Country';
run;
```

次の出力は結果を示しています。

図 12.8 データの並べ替え

Tours in Alphabetical Order by Country

Obs	Country	TourType	Nights	LandCost	Vendor
1	Brazil	architecture	8	1150	World
2	Greece	scenery	12	1396	Express
3	Ireland	scenery	7	1116	Express
4	Italy	architecture	8	936	Express
5	Japan	architecture	8	1440	Express
6	New Zealand	scenery	16	2978	Southsea
7	Spain	architecture	10	1020	World
8	Switzerland	scenery	9	1468	World

重複オブザベーションの削除

SORT プロシジャで NODUPRECS オプションを使用すると、SAS データセット内の重複オブザベーションを削除できます。次のプログラムでは、SAS データセットを作成して重複オブザベーションを削除する方法を示します。

次に示す外部ファイルには、Switzerland の重複オブザベーションが含まれています。

```
Spain       architecture  10 1020 World
Japan       architecture   8 1440 Express
Switzerland scenery        9 1468 World
Brazil      architecture   8 1150 World
Switzerland scenery        9 1468 World
Ireland     scenery        7 1116 Express
New Zealand scenery       16 2978 Southsea
Italy       architecture   8  936 Express
Greece      scenery       12 1396 Express
```

次の DATA ステップでは、MYLIB.ARCH_OR_SCEN2 という永久 SAS データセットが作成されます。

```
libname mylib 'SAS-library';

data mylib.arch_or_scen2;
   infile 'input-file';
   input Country $ 1-11 TourType $ 13-24 Nights LandCost Vendor $;
run;

proc print data=mylib.arch_or_scen2;
   title 'Data Set MYLIB.ARCH_OR_SCEN2';
run;
```

次の出力では、このデータセットに Switzerland の重複オブザベーションが含まれていることが示されます。

図 12.9 *データセット MYLIB.ARCH_OR_SCEN2*

Data Set MYLIB.ARCH_OR_SCEN2

Obs	Country	TourType	Nights	LandCost	Vendor
1	Spain	architecture	10	1020	World
2	Japan	architecture	8	1440	Express
3	Switzerland	scenery	9	1468	World
4	Brazil	architecture	8	1150	World
5	Switzerland	scenery	9	1468	World
6	Ireland	scenery	7	1116	Express
7	New Zealand	scenery	16	2978	Southsea
8	Italy	architecture	8	936	Express
9	Greece	scenery	12	1396	Express

次のプログラムでは、SORT プロシジャで NODUPRECS オプションを使用して重複オブザベーションを削除します。プログラムで FIXED というデータセットが新規作成されます。

```
proc sort data=mylib.arch_or_scen2 out=fixed noduprecs;
    by Country;
run;

proc print data=fixed;
   title 'Data Set FIXED: MYLIB.ARCH_OR_SCEN2 With Duplicates Removed';
run;
```

次の出力には、SAS ログに発生したメッセージが表示されています。

ログ 12.2 *削除された重複オブザベーションを示す SAS ログ*

```
697  proc sort data=mylib.arch_or_scen2 out=fixed noduprecs; 698      by
Country; 699  run; NOTE: There were 9 observations read from the data set
MYLIB.ARCH_OR_SCEN2.NOTE: 1 duplicate observations were deleted.NOTE: The data
set WORK.FIXED has 8 observations and 5 variables.NOTE: PROCEDURE SORT used
(Total process time): real time           0.01 seconds cpu time            0.01
seconds 700 701  proc print data=fixed; 702     title 'Data Set FIXED:
MYLIB.ARCH_OR_SCEN2 With Duplicates Removed'; 703  run; NOTE: There were 8
observations read from the data set WORK.FIXED.NOTE: PROCEDURE PRINT used (Total
process time): real time           0.03 seconds cpu time            0.03 seconds
```

次の出力は NODUPRECS オプションの結果を示しています。

図 12.10 重複オブザベーションのないデータセット FIXED

Data Set FIXED: MYLIB.ARCH_OR_SCEN2 With Duplicates Removed

Obs	Country	TourType	Nights	LandCost	Vendor
1	Brazil	architecture	8	1150	World
2	Greece	scenery	12	1396	Express
3	Ireland	scenery	7	1116	Express
4	Italy	architecture	8	936	Express
5	Japan	architecture	8	1440	Express
6	New Zealand	scenery	16	2978	Southsea
7	Spain	architecture	10	1020	World
8	Switzerland	scenery	9	1468	World

照合順序について

数値変数と文字変数は両方とも並べ替えて昇順や降順にできます。数値変数については、昇順や降順は容易に理解できます。文字変数については、昇順や降順はより複雑です。文字値には、大文字と小文字、特殊文字、および 0 から 9 までの数字(数値ではなく文字として処理される場合)が含まれます。

文字値を並べ替える順序は照合順序と呼ばれます。デフォルトでは、SAS を実行する動作環境に応じて、EBCDIC または ASCII という 2 つの順序のどちらかで文字が並べ替えられます。参考のため、ここでは両方の順序を示します。

1 つの動作環境で処理を行っている限り、照合順序の詳細について考慮が必要になることはめったにありません。ただし、EBCDIC を使用する動作環境から ASCII を使用する動作環境にファイルを転送する場合やその逆方向の転送をする場合、一方の動作環境で並べ替えられた文字値が、他方の動作環境にとって必ずしも正しい順序であるとは限りません。この問題の最も簡単な解決策は、転送先の動作環境で(数値データではなく)文字データを再度並べ替えることです。照合順序の詳細については、使用している動作環境に対応するドキュメントを参照してください。

ASCII 照合順序

次のオペレーティングシステムでは ASCII 照合順序が使用されます。

- UNIX とその派生
- Windows

英語の ASCII 順序で小さい値から大きい値へと順に並べた表示可能文字を次に示します。

- blank!"#$%&'()*+,− ./0123456789:;<=>?@
- ABCDEFGHIJKLMNOPQRSTUVWXYZ [\ ] ◦_
- abcdefghijklmnopqrstuvwxyz{}~

ASCII 順序の主な特徴は、数字が大文字よりも小さくて、大文字が小文字よりも小さいことです。ブランクが最小の表示可能文字で、その後に他種類の文字が続きます。

ブランク < 数字 < 大文字 < 小文字

EBCDIC 照合順序

z/OS オペレーティングシステムでは EBCDIC 照合順序が使用されます。

英語の EBCDIC 順序で小さい値から大きい値へと順に並べた表示可能文字を次に示します。

- blank.<(+|&!$*);¬ − /,%_>?:#@'="
- abcdefghijklmnopqr~stuvwxyz
- {ABCDEFGHI}JKLMNOPQR\ STUVWXYZ
- 0123456789

EBCDIC 順序の主な特徴は、小文字が大文字よりも小さくて、大文字が数字よりも小さいことです。ブランクが最小の表示可能文字で、その後に他種類の文字が続きます。

ブランク < 小文字 < 大文字 < 数字

要約

プロシジャ

PROC SORT <DATA=*SAS-data-set*> <OUT=*SAS-data-set*> <NODUPRECS>;
SAS データセットを BY ステートメントに記述した変数の値で並べ替えます。OUT= オプションを指定すると、並べ替えられたデータが、入力データとは異なる SAS データセットに格納されます。NODUPRECS オプションでは、PROC SORT での同一オブザベーションの削除が指示されます。

ステートメント

BY <DESCENDING> *variable-1* < ...<DESCENDING> *variable-n*>;
DATA ステップで、BY ステートメントで指定された変数ごとに FIRST.変数と LAST.変数が作成されます。FIRST.*variable-1* の値は、指定 BY 値を含む最初のオブザベーションでは 1 で、他のオブザベーションでは 0 です。同様に、LAST.*variable-1* の値は、指定 BY 値の最後のオブザベーションでは 1 で、他のオブザベーションでは 0 です。BY ステートメントは、DATA ステップの SET、MERGE、MODIFY、UPDATE のいずれかのステートメントの後に指定できます。INPUT ステートメントと一緒には使用できません。デフォルトでは、BY ステートメントで読み込まれるデータは BY 値の昇順だと仮定されます。DESCENDING オプションは、後に続く変数の値を逆順(最大値から最小値へ)にするよう指示します。

詳細情報

オブザベーションの並べ替えの代替処理
: オブザベーションの並べ替えの代替処理方法を使用するには、オブザベーションを変数の特定値で識別するインデックスを作成します。詳細については、“SAS Data Files” (*SAS Language Reference: Concepts*)を参照してください。

BY ステートメントと BY グループ処理
: 詳細については、“BY Statement” (*SAS Statements: Reference*)および “BY-Group Processing in the DATA Step” (*SAS Language Reference: Concepts*)を参照してください。

SAS データセットのインタリーブ、マージ、更新
: 詳細については、18 章, “SAS データセットのインタリーブ” (279 ページ)、19 章, “SAS データセットのマージ” (287 ページ)および 20 章, “SAS データセットの更新” (315 ページ)を参照してください。

  これらの操作は DATA ステップの BY ステートメントによって決まります。インタリーブでは、データセットが並べ替え順序で結合されます。マッチマージでは、BY 変数の値によって識別されたオブザベーションが結合されます。更新では、トランザクションを含むデータセットによってマスタファイルの値が変更されます。

  SAS ステートメントの詳細については、“MERGE Statement” (*SAS Statements: Reference*)および“UPDATE Statement” (*SAS Statements: Reference*)を参照してください。

NOTSORTED オプション
: NOTSORTED オプションは、SORT プロシジャを除けば、DATA ステップと PROC ステップの両方で使用できます。NOTSORTED オプションは、データが変数値に従ってグループ化されるが、そのグループが昇順でも降順でもない場合に役立ちます。BY ステートメントで NOTSORTED オプションを使用すると、処理が可能になります。詳細については、32 章, “SAS ログや出力ファイルに行を書き込む” (583 ページ)を参照してください。

SORT プロシジャ
: SORT プロシジャ、および BY ステートメントの役割の詳細については、“SORT” (*Base SAS Procedures Guide*)を参照してください。BY ステートメントの NOTSORTED オプションの詳細については、“BY Statement” (*SAS Statements: Reference*)を参照してください。

  次の項目では、SAS データセットの並べ替えについて言及しています。

  - 大容量のデータセットを処理する場合は、計画的に作業を行い、データセットの並べ替え回数を可能な限り少なくしてください。たとえば、データセットを、プログラムの最初に State 順で並べ替えて、その後 State 内で City 順に並べ替える必要がある場合は、プログラムの最初にデータセットを State と City で並べ替えておきます。
  - BY 値が他のオブザベーションの BY 値と重複する(ただし他の変数値は必ずしも重複するとは限らない)オブザベーションを削除するには、SORT プロシジャで NODUPKEY オプションを使用します。
  - SAS では、英語の EBCDIC や ASCII 以外の順序でデータを並べ替えられます。例としては、デンマーク語-ノルウェー語およびフィンランド語やスウェーデン語の順序などがあります。

  使用している動作環境に対応する SAS ドキュメントには、SORT プロシジャについて動作環境固有の情報が記載されています。一般に、データの並べ替えに関する

多くの考慮点は、サイトの動作環境および他のローカル条件(各種動作環境ユーティリティが使用可能かどうかなど)によって決まります。

13 章

計算に複数のオブザベーションを使用する

計算への複数オブザベーションの使用について **197**
目的 197
前提条件 197
例で使用される入力ファイルおよび入力 SAS データセット **198**
データセット全体の合計を累計する **199**
累計の作成 199
合計のみの書き込み 201
BY グループの合計の取得 **202**
異なるデータセットへの書き込み **204**
オブザベーションを異なるデータセットに書き込む 204
合計を異なるデータセットに書き込む 205
プログラム 205
前のオブザベーションの値の使用 **207**
要約 **210**
ステートメント 210
詳細情報 **211**

計算への複数オブザベーションの使用について

目的

このセクションでは、複数のオブザベーションを必要とする計算について学習します。このセクションには次の種類の例が含まれています。

- データセットまたは BY グループ全体の合計を累計する
- オブザベーションの値を後のオブザベーションの値と比較するために保存する

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)

- 12 章, “グループ化されたオブザベーションや並べ替えられたオブザベーションの操作” (181 ページ)

例で使用される入力ファイルおよび入力 SAS データセット

Tradewinds Travel 社は、ピークの季節中にさまざまなツアー業者とどのくらい取引をしているのかを知る必要があります。参照の必要なデータは、さまざまな業者のツアーにスケジュールされている総人数です。また、スケジュールされているツアーの総額も参照する必要があります。

次の外部ファイルには、Tradewinds Travel のツアーについてのデータが含まれています。

```
1           2    3        4
----------------------------
Germany     1150 Express  10
Spain       1020 World    12
Brazil      1080 World     6
India        978 Express   .
Japan       1440 Express  10
Greece      1396 Express  20
New Zealand 2978 Southsea  6
Venezuela    850 World     8
Italy        936 Express   9
Russia      1848 World     6
Switzerland 1468 World    20
Australia   2158 Southsea 10
Ireland     1116 Express   9
```

次のリストでは、入力ファイルのフィールドについて説明しています。

1 ツアーの行き先国の名前を提供する

2 米ドルでの地上パッケージのコストを示す

3 旅行業者の名前を示す

4 そのツアーを予約した人数を示す

最初のステップは、永久 SAS データセットを作成することです。次のプログラムは、データセット MYLIB.TOURREVENUE を作成します。

```
libname mylib 'SAS-library';

data mylib.tourrevenue;
   infile 'input-file' truncover;
   input Country $ 1-11 LandCost Vendor $ NumberOfBookings;
run;

proc print data=mylib.tourrevenue;
   title 'SAS Data Set MYLIB.TOURREVENUE';
run;
```

DATA ステップの後の PROC PRINT ステートメントによって、
MYLIB.TOURREVENUE データセットが次のように表示されます。

図 13.1 データセット MYLIB.TOURREVENUE

SAS Data Set MYLIB.TOURREVENUE

Obs	Country	LandCost	Vendor	NumberOfBookings
1	Germany	1150	Express	10
2	Spain	1020	World	12
3	Brazil	1080	World	6
4	India	978	Express	.
5	Japan	1440	Express	10
6	Greece	1396	Express	20
7	New Zealand	2978	Southsea	6
8	Venezuela	850	World	8
9	Italy	936	Express	9
10	Russia	1848	World	6
11	Switzerland	1468	World	20
12	Australia	2158	Southsea	10
13	Ireland	1116	Express	9

データセット MYLIB.TOURREVENUE の各オブザベーションには、ツアーのコストと、そのツアーを予約した人数が含まれます。Tradewinds Travel 社のタスクを次に示します。

- 各業者の使用金額とすべての業者の使用金額合計を特定する
- 個別の業者のレコードとは別の SAS データセットに合計を格納する
- 地上コストにツアーの予約人数を掛けて収益を決定し、最も多く収益をもたらすツアーを見つけ出す

データセット全体の合計を累計する

累計の作成

データセット MYLIB.TOURREVENUE に対する計算実行における最初のタスクは、Tradewinds Travel 社のツアー予約者の総人数を求めることです。したがって、各オブザベーションに、値が 0 で始まり予約の数だけ増加する変数が必要です。合計ステートメントでその機能が得られます。

variable + expression

合計ステートメントでは、プラス記号の左側の *variable* の値は、ステートメントを最初に処理する前は 0 です。ステートメントの処理によって、プラス記号右側の *expression* の値が初期値に加算されます。ステートメントの次の処理までは合計変数に新しい値が保持されます。合計ステートメントでは式の欠損値は無視されます。前の合計は変更されません。

次のステートメントでは、予約の総数が算出されます。

```
TotalBookings + NumberOfBookings;
```

次の DATA ステップには、前述の合計ステートメントが含まれています。

```
data total;
   set mylib.tourrevenue;
   TotalBookings + NumberOfBookings;
run;

proc print data=total;
   var Country NumberOfBookings TotalBookings;
   title 'Total Tours Booked';
run;
```

次の出力は結果を示しています。

図 13.2 *データセットの合計を累計する*

Total Tours Booked

Obs	Country	NumberOfBookings	TotalBookings
1	Germany	10	10
2	Spain	12	22
3	Brazil	6	28
4	India	.	28
5	Japan	10	38
6	Greece	20	58
7	New Zealand	6	64
8	Venezuela	8	72
9	Italy	9	81
10	Russia	6	87
11	Switzerland	20	107
12	Australia	10	117
13	Ireland	9	126

TOTAL データセットの最後のオブザベーションの TotalBookings 変数には、その年の予約総数が含まれます。

合計のみの書き込み

データセットから必要な情報が合計のみの場合、1 オブザベーションと 1 変数 (TotalBookings 変数)のみを含むデータセットを作成するには、次の機能をすべて実行する DATA ステップを記述します。

- SET ステートメントに END=オプションを指定し、現在のオブザベーションが最後のオブザベーションかどうかを決定します。
- サブセット化 IF ステートメントを使用して、最後のオブザベーションのみを SAS データセットに書き込みます。
- DATA ステップに KEEP=オプションを指定して、予約を合計する変数のみを保持します。

SET ステートメントで END=オプションを指定すると、DATA ステップでの最後のオブザベーションの処理時に END=オプションで指定した変数が 1 に設定されます。他のオブザベーションでは END=オプションで指定した変数は 0 に設定されます。

SET *SAS-data-set* <END=*variable*>;

END=変数は作成されるデータセットには追加されません。END=変数の値を検証することによって、どのオブザベーションが最後のオブザベーションか判断できます。

次のプログラムでは、サブセット化 IF ステートメントで最後のオブザベーションが選択され、KEEP=データセットオプションによってデータセットに変数 TotalBookings のみが保持されます。

```
data total2(keep=TotalBookings);
   set mylib.tourrevenue end=Lastobs;
   TotalBookings + NumberOfBookings;
   if Lastobs;
run;

proc print data=total2;
   title 'Total Number of Tours Booked';
run;
```

次の出力は結果を示しています。

図 13.3 *データセットの最後のオブザベーションを選択*

Total Number of Tours Booked

Obs	TotalBookings
1	126

Lastobs の値が 1 の場合、サブセット化 IF ステートメントの条件が真になります。MYLIB.TOURREVENUE の最後のオブザベーションを処理する際、Lastobs に値 1 が割り当てられます。したがって、サブセット化 IF ステートメントでは MYLIB.TOURREVENUE の最後のオブザベーションのみが受け入れられ、データセット TOTAL2 に最後のオブザベーションが書き込まれます。

BY グループの合計の取得

Tradewinds Travel 社の追加要件は、各業者で予約されたツアー数を特定することです。このタスクを完了するためには、プログラムでデータを変数別にグループ化する必要があります。すなわち、プログラムでデータセットを整理し、業者ごとに 1 グループずつのオブザベーショングループにまとめる必要があります。この場合、プログラムでは Vendor 変数別にデータをグループ化する必要があります。一般的に各グループは BY グループとして知られています。グループ化の決定に使用する変数は BY 変数と呼ばれます。

Vendor 変数別にデータをグループ化するためには、次の操作を行います。

- PROC SORT ステップを含めて、オブザベーションを Vendor 変数別にグループ化します。
- DATA ステップで BY ステートメントを使用します。
- 合計ステートメントを使用して予約を合計します。
- 各オブザベーショングループの最初に合計変数を 0 にリセットします。

次のプログラムでは、データセットが Vendor 順に並べ替えられ、各業者の予約総数が合計されます。

```
proc sort data=mylib.tourrevenue out=mylib.sorttour;
    by Vendor;
run;

data totalby;
   set mylib.sorttour;
   by Vendor;
   if First.Vendor then VendorBookings=0;
   VendorBookings + NumberOfBookings;
run;

proc print data=totalby;
   title 'Summary of Bookings by Vendor';
run;
```

前述のプログラムでは、IF-THEN ステートメントで FIRST.Vendor 変数を使用して、各 BY グループの最初のオブザベーションで合計変数(VendorBookings)を 0 にリセットしています。(FIRST.*variable* 一時変数と LAST.*variable* 一時変数の詳細については、“グループ内の最初のオブザベーションまたは最後のオブザベーションの検索” (187 ページ)を参照してください。)

次の出力は結果を示しています。

図 13.4 BY グループ合計の算出

Summary of Bookings by Vendor

Obs	Country	LandCost	Vendor	NumberOfBookings	VendorBookings
1	Germany	1150	Express	10	10
2	India	978	Express	.	10
3	Japan	1440	Express	10	20
4	Greece	1396	Express	20	40
5	Italy	936	Express	9	49
6	Ireland	1116	Express	9	58
7	New Zealand	2978	Southsea	6	6
8	Australia	2158	Southsea	10	16
9	Spain	1020	World	12	12
10	Brazil	1080	World	6	18
11	Venezuela	850	World	8	26
12	Russia	1848	World	6	32
13	Switzerland	1468	World	20	52

この出力には事実上は各業者の予約総数が含まれていますが、無関係の情報も大量に含まれています。各業者の予約総数のレポート作成に必要なのは、各業者の最後のオブザベーションの変数 Vendor および VendorBookings のみです。したがって、プログラムでは次の要素を使用できます。

- DROP=データセットオプションまたは KEEP=データセットオプションを使用して、出力データセットから変数 Country、LandCost および NumberOfBookings を削除します。
- サブセット化 IF ステートメントで LAST.Vendor 変数を使用して、各グループの最後のオブザベーションのみをデータセット TOTALBY に書き込みます。

次のプログラムは、データセット TOTALBY を作成します。

```
proc sort data=mylib.tourrevenue out=mylib.sorttour;
   by Vendor;
run;

data totalby(drop=country landcost NumberOfBookings);
   set mylib.sorttour;
   by Vendor;
   if First.Vendor then VendorBookings=0;
   VendorBookings + NumberOfBookings;
   if Last.Vendor;
run;

proc print data=totalby;
```

```
   title 'Total Bookings by Vendor';
run;
```

次の出力は結果を示しています。

図 13.5 各 BY グループの合計を新しいデータセットに挿入

Total Bookings by Vendor

Obs	Vendor	VendorBookings
1	Express	58
2	Southsea	16
3	World	52

異なるデータセットへの書き込み

オブザベーションを異なるデータセットに書き込む

Tradewinds Travel 社は今年行われたツアーに関する全情報を必要としています。1 つの SAS データセットには、ツアーに費やされた総額など、各ツアーの詳細情報を含める必要があります。もう 1 つの SAS データセットには、各業者の予約総数とその業者の費やした総額を含める必要があります。これらのデータセットは両方とも、これまでに学習したテクニックを使用して作成できます。

プログラムではまず、次の DATA ステートメントと SET ステートメントを使用して、SAS データセット MYLIB.SORTTOUR から 2 つの SAS データセットを作成します。

```
data tourdetails vendordetails;
   set mylib.sorttour;
```

データセット TOURDETAILS には個別レコードが含まれ、VENDORDETAILS には業者についての情報が含まれます。TOURDETAILS のオブザベーションはグループ化する必要はありませんが、VENDORDETAILS のオブザベーションは Vendor 別にグループ化する必要があります。

データがまだ Vendor 別にグループ化されていない場合は、まず SORT プロシジャを使用します。VENDORDETAILS 用に BY ステートメントを DATA ステップに追加します。

```
   proc sort data=mylib.tourrevenue out=mylib.sorttour;
      by Vendor;
   run;

   data tourdetails vendordetails;
      set mylib.sorttour;
      by Vendor;
   run;
```

個別ツアーに必要な計算は、各ツアーの使用金額のみです。したがって、割り当てステートメントで金額が計算され、TOURDETAILS にオブザベーションが書き込まれます。

```
Money=LandCost * NumberOfBookings;
output tourdetails;
```

DATA ステップの TOURDETAILS 作成部分はこれで完了です。

合計を異なるデータセットに書き込む

オブザベーションは OUTPUT ステートメントの実行後もプログラムデータベクトルに残っているので、引き続きプログラミングステートメントでそのオブザベーションを使用できます。DATA ステップの残り部分では VENDORDETAILS データセットの情報が作成されます。

FIRST.Vendor 変数によって、各グループの最初のオブザベーションをいつ処理しているのかが特定されます。

次にそのオブザベーションの合計変数 VendorBookings および VendorMoney が 0 に設定されます。VendorBookings では各業者の予約総数が集計され、VendorMoney では総コストが集計されます。次のステートメントを DATA ステップに追加します。

```
if First.Vendor then
   do;
      VendorBookings=0;
      VendorMoney=0;
   end;
 VendorBookings + NumberOfBookings;
 VendorMoney + Money;
```

注: プログラムでは DO グループが使用されます。DO グループを使用すると、プログラムで条件を 1 度評価すれば、結果として複数の処理を行えます。DO グループの詳細については、“IF-THEN ステートメントで複数の処理を実行する” (215 ページ)を参照してください。

各 BY グループの最後のオブザベーションにその業者の合計が含まれます。したがって、次のステートメントを使用してデータセット VENDORDETAILS に最後のオブザベーションを出力します。

```
if Last.Vendor then output vendordetails;
```

最後のステップとして、KEEP=データセットオプションおよび DROP=データセットオプションを使用して、2 つのデータセットから無関係の変数を削除し、各データセットに必要な変数のみが含まれるようにします。

```
data tourdetails(drop=VendorBookings VendorMoney)
     vendordetails(keep=Vendor VendorBookings VendorMoney);
```

プログラム

VENDORDETAILS データセットと TOURDETAILS データセットを作成する完全なプログラムは次のとおりです。

```
   proc sort data=mylib.tourrevenue out=mylib.sorttour;
      by Vendor;
   run;

   data tourdetails(drop=VendorBookings VendorMoney)
        vendordetails(keep=Vendor VendorBookings VendorMoney);
      set mylib.sorttour;
      by Vendor;
      Money=LandCost * NumberOfBookings;
```

```
   output tourdetails;
   if First.Vendor then
      do;
         VendorBookings=0;
         VendorMoney=0;
      end;
   VendorBookings + NumberOfBookings;
   VendorMoney + Money;
   if Last.Vendor then output vendordetails;
run;

proc print data=tourdetails;
   title 'Detail Records: Dollars Spent on Individual Tours';
run;

proc print data=vendordetails;
   title 'Vendor Totals: Dollars Spent and Bookings by Vendor';
run;
```

次の出力では、一方の SAS データセットに詳細ツアーレコード、他方に業者合計が表示されています。

図 13.6 *TOURDETAILS データセットの詳細ツアーレコード*

Detail Records: Dollars Spent on Individual Tours

Obs	Country	LandCost	Vendor	NumberOfBookings	Money
1	Germany	1150	Express	10	11500
2	India	978	Express	.	.
3	Japan	1440	Express	10	14400
4	Greece	1396	Express	20	27920
5	Italy	936	Express	9	8424
6	Ireland	1116	Express	9	10044
7	New Zealand	2978	Southsea	6	17868
8	Australia	2158	Southsea	10	21580
9	Spain	1020	World	12	12240
10	Brazil	1080	World	6	6480
11	Venezuela	850	World	8	6800
12	Russia	1848	World	6	11088
13	Switzerland	1468	World	20	29360

図 13.7 VENDORDETAILS データセットの業者合計

Vendor Totals: Dollars Spent and Bookings by Vendor

Obs	Vendor	VendorBookings	VendorMoney
1	Express	58	72288
2	Southsea	16	39448
3	World	52	65968

前のオブザベーションの値の使用

Tradewinds Travel 社のさらなる要件は、最も収益を生んだツアーを含む異なる SAS データセットです。(総収益はツアー価格に予約数を掛けた値に等しいです。)データセットを新規作成する 1 つの方法は、次の 3 つの操作を実行することです。

1. DATA ステップで収益を計算します。
2. データセットを収益の降順で並べ替えます。
3. 別の DATA ステップで OBS=データセットオプションを使用して、そのオブザベーションを書き込みます。

より効率的な方法では、1 つの DATA ステップですべてのオブザベーションの収益を比較します。現在のオブザベーションの値を保持して今後のオブザベーションで使用できます。DATA ステップの処理が次のオブザベーションに達すると、保持された値によって前のオブザベーションの情報が示されます。

RETAIN ステートメントを使用すると、DATA ステップで作成される変数で、現在のオブザベーションの値が次のオブザベーションに入れられて保持されます。この変数は DATA ステップの各反復の最初に欠損値に設定されることはありません。RETAIN は宣言ステートメントで、実行可能ステートメントではありません。このステートメントの形式は次のとおりです。

RETAIN *variable-1 < ...variable-n>*;

あるオブザベーションの Revenue 値を次のオブザベーションの Revenue 値と比較するには、HoldRevenue という保持変数を作成し、そこに現在の Revenue 変数値を割り当てます。次のオブザベーションでは、HoldRevenue 変数に前のオブザベーションの Revenue 値が含まれるので、その値を現在のオブザベーションの Revenue 値と比較できます。

RETAIN ステートメントの機能を確認するには、次の例を参照してください。次の DATA ステップでは、現在の収益が HoldRevenue に割り当てられる前に、オブザベーションがデータセット TEMP に書き込まれます。

```
data temp;
   set mylib.tourrevenue;
   retain HoldRevenue;
   Revenue=LandCost * NumberOfBookings;
   output;
   HoldRevenue=Revenue;
run;
```

```
proc print data=temp;
   var Country LandCost NumberOfBookings Revenue HoldRevenue;
   title 'Tour Revenue';
run;
```

次の出力は結果を示しています。

図 13.8 RETAIN ステートメントによる値の保持

Tour Revenue

Obs	Country	LandCost	NumberOfBookings	Revenue	HoldRevenue
1	Germany	1150	10	11500	.
2	Spain	1020	12	12240	11500
3	Brazil	1080	6	6480	12240
4	India	978	.	.	6480
5	Japan	1440	10	14400	.
6	Greece	1396	20	27920	14400
7	New Zealand	2978	6	17868	27920
8	Venezuela	850	8	6800	17868
9	Italy	936	9	8424	6800
10	Russia	1848	6	11088	8424
11	Switzerland	1468	20	29360	11088
12	Australia	2158	10	21580	29360
13	Ireland	1116	9	10044	21580

HoldRevenue の値は最初のオブザベーションの開始時には欠損しています。OUTPUT ステートメントで TEMP に最初のオブザベーションを書き込む際もまだ欠損しています。OUTPUT ステートメントの後で、割り当てステートメントによって Revenue の値が HoldRevenue に割り当てられます。HoldRevenue が保持されるため、DATA ステップの次の反復の開始時には値が存在します。OUTPUT ステートメントが再度実行されるときには、なおもその値が HoldRevenue の値として含まれています。

Revenue の最大値を見つけるには、次のプログラムに示すように、Revenue が HoldRevenue より大きい場合のみ Revenue の値を HoldRevenue に割り当てます。

```
data mostrevenue;
   set mylib.tourrevenue;
   retain HoldRevenue;
   Revenue=LandCost * NumberOfBookings;
   if Revenue > HoldRevenue then HoldRevenue=Revenue;
run;

proc print data=mostrevenue;
   var Country LandCost NumberOfBookings Revenue HoldRevenue;
   title 'Tour Revenue';
```

```
run;
```

次の出力は結果を示しています。

図 13.9 保持変数に最大値を保持

Tour Revenue

Obs	Country	LandCost	NumberOfBookings	Revenue	HoldRevenue
1	Germany	1150	10	11500	11500
2	Spain	1020	12	12240	12240
3	Brazil	1080	6	6480	12240
4	India	978	.	.	12240
5	Japan	1440	10	14400	14400
6	Greece	1396	20	27920	27920
7	New Zealand	2978	6	17868	27920
8	Venezuela	850	8	6800	27920
9	Italy	936	9	8424	27920
10	Russia	1848	6	11088	27920
11	Switzerland	1468	20	29360	29360
12	Australia	2158	10	21580	29360
13	Ireland	1116	9	10044	29360

最後のオブザベーションの HoldRevenue の値は、ツアーで生み出された最大の収益を表します。どのオブザベーションの値か特定するには、HoldCountry という変数を作成し、収益が最大のオブザベーションの国名を保持します。RETAIN ステートメントに HoldCountry を含めると、その値は明示的に変更されるまで保持されます。次に END=データセットオプションを使用して最後のオブザベーションを選択し、KEEP=データセットオプションを使用して MOSTREVENUE の HoldRevenue と HoldCountry のみ保持します。

```
data mostrevenue (keep=HoldCountry HoldRevenue);
   set mylib.tourrevenue  end=LastOne;
   retain HoldRevenue HoldCountry;
   Revenue=LandCost * NumberOfBookings;
   if Revenue > HoldRevenue then
      do;
         HoldRevenue=Revenue;
         HoldCountry=Country;
      end;
   if LastOne;
run;
proc print data=mostrevenue;
   title 'Country with the Largest Value of Revenue';
run;
```

注: プログラムでは DO グループが使用されます。DO グループを使用すると、プログラムで条件を 1 度評価すれば、結果として複数の処理を行えます。DO グループの詳細については、“IF-THEN ステートメントで複数の処理を実行する” (215 ページ)を参照してください。

次の出力は結果を示しています。

図 13.10 *RETAIN ステートメントとサブセット化 IF ステートメントを使用して最大収益を割り出す*

Country with the Largest Value of Revenue

Obs	HoldRevenue	HoldCountry
1	29360	Switzerland

要約

ステートメント

RETAIN *variable-1* < *...variable-n*>;
後続オブザベーションで使用するために *variable* の値を保持します。RETAIN ステートメントは、DATA ステップの先頭に制御が戻ったときに、変数値が再初期化されて欠損値になるのを防ぎます。

RETAIN ステートメントは、現在の DATA ステップで作成される変数(INPUT ステートメントや割り当てステートメントで作成される変数など)に影響を及ぼします。SET、MERGE または UPDATE ステートメントで読み取られた変数は自動的に保持されます。RETAIN ステートメントで名前を指定しても影響はありません。

RETAIN ステートメントでは変数に初期値を割り当てられます。DATA ステップのすべてのオブザベーションで変数の値を同じにする必要がある場合は、割り当てステートメントではなく RETAIN ステートメントで値を書き込むほうが効率的です。DATA ステップのコンパイル時に RETAIN ステートメントで値が割り当てられますが、割り当てステートメントは DATA ステップを実行するたびに実行されます。

SET *SAS-data-set* <END=*variable*>;
指定した *SAS-data-set* から読み取ります。END=オプションで指定された *variable* の値は、データセットの最後のオブザベーションが処理されるまでは 0 です。処理後は変数の値が 1 になります。END=変数は作成されるデータセットには含まれません。

variable + *expression*;
合計ステートメントと呼ばれます。プラス記号右側の *expression* の結果をプラス記号左側の *variable* に加算して、*variable* の新しい値を後続オブザベーションで使用するために保持します。expression は数値変数でも式でもかまいません。*variable* の値は保持されます。expression が欠損値の場合、*variable* ではその前の値が保持されます。合計ステートメントの初回実行前は、*variable* のデフォルト値は 0 です。

合計ステートメントではプラス記号が必要です。開始値から連続した値を引いていくには、合計変数に負の値を足します。

詳細情報

自動変数_N_
DATA ステップの実行回数をカウントする方法が提供されます。詳細については、32 章, “SAS ログや出力ファイルに行を書き込む” (583 ページ)を参照してください。

SAS では各 DATA ステップで_N_が作成されます。初めて DATA ステップの実行を開始するときは_N_の値は 1 です。2 回目の実行を開始するときは値が 2 になります(以下同様に続く)。_N_は出力データセットには追加されません。_N_を使用する方が合計ステートメントを使用するよりも効率的です。

DO グループ
1 単位として実行されるステートメントのグループを示します。DO グループ処理の詳細については、14 章, “プログラミングを簡単にする” (213 ページ)を参照してください。DO ステートメントの詳細については、“DO Statement” (*SAS Statements: Reference*)を参照してください。

END=オプション
SET ステートメントで使用できます。例については、22 章, “複数の SAS データセットのオブザベーションの条件付き処理” (347 ページ)を参照してください。

KEEP=および DROP=データセットオプション
データセットのどのオブザベーションを保持または削除する必要があるかを示します。これらのオプションは入力または出力データセットに対して使用できます。詳細については、6 章, “SAS データセットから作成する” (91 ページ)を参照してください。“DROP= Data Set Option” (*SAS Data Set Options: Reference*)および“KEEP= Data Set Option” (*SAS Data Set Options: Reference*)も参照してください。

LAG 関数ファミリ
オブザベーションの値を保持して後続オブザベーションで使用するための別の方法が提供されます。LAG 関数では、100 オブザベーションまで値を保持できます。詳細については、“LAG Function” (*SAS Functions and CALL Routines: Reference*)を参照してください。

RETAIN ステートメント、合計ステートメント、SET ステートメント
詳細については、“RETAIN Statement” (*SAS Statements: Reference*)、“Sum Statement” (*SAS Statements: Reference*)および“SET Statement” (*SAS Statements: Reference*)を参照してください。

合計ステートメントと SUMBY ステートメント
レポート表示だけのために合計を出す場合に、PRINT プロシジャで使用できます。PRINT プロシジャの合計ステートメントと SUMBY ステートメントについては、27 章, “PRINT プロシジャを使用した詳細レポートの作成” (421 ページ)で説明しています。

SUMMARY プロシジャと MEANS プロシジャ
合計の計算にも使用できます。詳細については、“SUMMARY” (*Base SAS Procedures Guide*)および“MEANS” (*Base SAS Procedures Guide*)を参照してください。

14 章

プログラミングを簡単にする

ショートカットについて **213**
目的 213
前提条件 213
入力ファイルと入力 SAS データセット **214**
IF-THEN ステートメントで複数の処理を実行する **215**
複数の変数に同じ処理を実行する **217**
IF-THEN ステートメントの連続使用 217
変数を配列にグループ化 217
処理の繰り返し 218
現在の変数の選択 218
要約 **220**
ステートメント 220
詳細情報 **221**

ショートカットについて

目的

このセクションでは、コードの読み書きを容易にする DATA ステップのプログラミングテクニックを 2 つ学習します。

- DO グループを使用して、IF 条件の評価後に複数の処理を実行
- 配列を使用して、1 つのステートメントグループで複数の変数に対して同じ処理を実行

前提条件

このセクションを先に進む前に、次のセクションで説明したトピックを理解している必要があります。

- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)
- 10 章, “選択したオブザベーションの操作” (147 ページ)

入力ファイルと入力 SAS データセット

次の例では、Tradewinds Travel 社は、美術館やギャラリーへのツアーについてのデータを調整します。ツアーのデータを次に示します。

```
1         2 3 4 5        6
--------------------------
Rome      4 3 . D'Amico  2
Paris     5 . 1 Lucas    5
London    3 2 . Wilson   3
New York  5 1 2 Lucas    5
Madrid    . . 5 Torres   4
Amsterdam 3 3 .          .
```

次のリストは、前述のファイルの番号付き項目に対応しています。

1 この列は、行き先の国名を示します。

2 この列は、訪れる美術館の数を示します。

3 この列は、ツアーでの美術ギャラリーの数を示します。

4 この列は、観光するその他のアトラクションの数を示します。

5 この列には、ツアーガイドの姓を載せます。

6 この列には、ツアーガイドの経験年数を載せます。

次のプログラムでは、永久 SAS データセット MYLIB.ATTRACTIONS が作成されます。

```
libname mylib 'permanent-data-library';

data mylib.attractions;
   infile 'input-file';
   input City $ 1-9 Museums 11 Galleries 13
         Other 15 TourGuide $ 17-24 YearsExperience 26;
run;

proc print data=mylib.attractions;
   title 'Data Set MYLIB.ATTRACTIONS';
run;
```

次の出力は結果を示しています。

図 14.1 データセット MYLIB.ATTRACTIONS

Data Set MYLIB.ATTRACTIONS

Obs	City	Museums	Galleries	Other	TourGuide	YearsExperience
1	Rome	4	3	.	D'Amico	2
2	Paris	5	.	1	Lucas	5
3	London	3	2	.	Wilson	3
4	New York	5	1	2	Lucas	5
5	Madrid	.	.	5	Torres	4
6	Amsterdam	3	3	.		.

IF-THEN ステートメントで複数の処理を実行する

Madrid と Amsterdam のオブザベーションでは複数の変更が必要です。そのオブザベーションを選択する 1 つの方法は、次のように、連続する IF-THEN ステートメントで IF 条件を評価することです。

```
   /* multiple actions based on the same condition */
data updatedattractions;
   set mylib.attractions;
   if City = 'Madrid' then Museums = 3;
   if City = 'Madrid' then Other = 2;
   if City = 'Amsterdam' then TourGuide = 'Vandever';
   if City = 'Amsterdam' then YearsExperience = 4;
run;
```

IF 条件を都市ごとに 2 回ずつ記述することを避けるため、次の例のように THEN 句で DO グループを使用します。

IF *condition* THEN

DO;

...more SAS statements...

END;

DO ステートメントを使用すると、後に続くステートメントはすべて、対応する END ステートメントが出現するまで 1 単位として処理されます。DO で始まって END で終わる SAS ステートメントのグループは DO グループと呼ばれます。

次の DATA ステップでは、複数の IF-THEN ステートメントが DO グループに置き換えられています。

```
      /* a more efficient method */
   data updatedattractions2;
      set mylib.attractions;
      if City = 'Madrid' then
```

```
        do;
           Museums = 3;
           Other = 2;
        end;
     else if City = 'Amsterdam' then
        do;
           TourGuide = 'Vandever';
           YearsExperience = 4;
        end;
run;

proc print data=updatedattractions2;
   title 'Data Set MYLIB.UPDATEDATTRACTIONS';
run;
```

次の出力は結果を示しています。

図 14.2 *DO グループを使用したデータセットの作成*

Data Set MYLIB.UPDATEDATTRACTIONS

Obs	City	Museums	Galleries	Other	TourGuide	YearsExperience
1	Rome	4	3	.	D'Amico	2
2	Paris	5	.	1	Lucas	5
3	London	3	2	.	Wilson	3
4	New York	5	1	2	Lucas	5
5	Madrid	3	.	2	Torres	4
6	Amsterdam	3	3	.	Vandever	4

DO グループを使用すると、プログラムの書き込みが迅速になり、読み取りが容易になります。また、プログラムにおいて次の 2 点が SAS にとって効率化されます。

1. IF 条件の評価回数が少なくなります。(この DATA ステップには前の DATA ステップよりも多くのステートメントがありますが、DO ステートメントと END ステートメントで必要なコンピュータリソースはほとんどありません。)
2. 複数の IF-THEN ステートメントを 2 つのステートメントに要約すると明らかになるように、条件 **`City = 'Madrid'`** と **`City = 'Amsterdam'`** は相互に排他的です。2 番目の IF-THEN ステートメントを ELSE ステートメントに組み込むことができます。その結果、最初の IF 条件が真の場合は 2 番目の IF 条件は評価されません。

複数の変数に同じ処理を実行する

IF-THEN ステートメントの連続使用

データセット MYLIB.ATTRACTIONS で変数 Museums、Galleries および Other は、そのツアーのアトラクションが対象にならない種類の場合、欠損値になります。欠損値を 0 に変更するには、次のプログラムに示すように、IF-THEN ステートメントを割り当てステートメントともに連続で記述します。

```
   /* same action for different variables */
data changes;
   set mylib.attractions;
   if Museums = . then Museums = 0;
   if Galleries = . then Galleries = 0;
   if Other = . then Other = 0;
run;
```

3 つの IF-THEN ステートメントの処理パターンは同じです。変数名のみ異なります。プログラムを読みやすくするには、同じ処理を複数回実行する SAS ステートメントを記述し、影響のある変数のみを変更します。このテクニックは配列処理と呼ばれ、次の 3 ステップから成ります。

1. 変数を配列にグループ化
2. 処理の繰り返し
3. 処理対象の現在の変数を選択

変数を配列にグループ化

DATA ステップのプログラミングでは、配列と呼ばれる一時グループに変数を入れられます。配列を定義するには、ARRAY ステートメントを使用します。単純な ARRAY ステートメントの形式は次のとおりです。

ARRAY *array-name*{*number-of-variables*} *variable-1* < *...variable-n*>;

Array-name は変数グループを識別するために選択する SAS 名です。*Number-of-variables* は中かっこで囲んで、グループ化する変数の数を指定し、*variable-1*< *...variable-n*>でその名前をリストします。

注: 他のプログラミング言語で配列を処理したことがある場合は、SAS における配列が他の多くの言語における配列とは異なることに注意してください。SAS では、配列とは、エイリアスの割り当てによって変数のグループを一時的に識別する便利な方法にすぎません。永続的なデータ構造ではありません。DATA ステップが処理されている間のみ存在します。配列は *array-name* によって識別され、同一 DATA ステップ内にある他の配列と区別されます。これは変数ではありません。

次の ARRAY ステートメントでは、3 つの変数 Museums、Galleries および Other がリストされます。

```
array changelist{3} Museums Galleries Other;
```

このステートメントでは、次の操作が SAS に指示されます。

- この DATA ステップの処理中に CHANGLIST というグループを作成
- CHANGELIST に Museums、Galleries、Other という 3 つの変数名を入れる

さらに、ARRAY ステートメントに変数をリストすることにより、*array-name {position}*の形式で変数に追加名が割り当てられ、このときの *position* はリスト内の変数の位置(この場合は 1、2 または 3)になります。位置には、数字か、または値が数字である変数の名前を指定できます。この追加名は配列参照と呼ばれ、その位置は添字と呼ばれます。前述の ARRAY ステートメントでは、Museums に CHANGELIST{1}、Galleries に CHANGELIST{2}、Other に CHANGELIST{3}が割り当てられます。DATA ステップのその時点から、元の名前と配列参照のどちらでも変数の呼び名として使用できます。たとえば、Museums と CHANGELIST{1}という名前は同一の値を指します。

処理の繰り返し

同じ処理を繰り返して何回か実行するよう指定するには、次の形式の反復 DO ループを使用します。

DO *index-variable*=1 TO *number-of-variables-in-array*;

...SAS statements...

END;

反復 DO ループは反復 DO ステートメントで始まり、他の SAS ステートメントをはさんで、END ステートメントで終わります。反復 DO ステートメントに記述された指示に従って、ループは繰り返し処理(反復)されます。反復 DO ステートメントに含まれる *index-variable* は、ユーザー選択の名前を使用し、その値はループの反復ごとに変化します。配列処理では、通常、配列に含まれる変数の数だけループを実行する必要があります。したがって、*index-variable* の値が 1 から *number-of-variables-in-array* までであることを指定します。デフォルトでは、*index-variable* の値は、ループの反復が新しく始まる前に毎回 1 ずつ増分します。値が *number-of-variables-in-array* より大きくなると、ループの処理が停止します。デフォルトでは、インデックス変数は作成されるデータセットに追加されます。

処理が 3 回行われ、Count というインデックス変数が含まれる反復 DO ループは次のようになります。

```
do Count = 1 to 3;
   ...SAS statements...
end;
```

ループ初回処理での Count の値は 1、2 回目の値は 2、3 回目の値は 3 です。4 回目の実行開始時には、Count の値は 4 で、1 から 3 までの指定範囲を超えるので、ループ処理が停止されます。

現在の変数の選択

ここまでで、変数はグループ化され、ループの処理回数がわかったので、次は、配列内にある変数のうち、どの変数を反復 DO ループに使用するのかを指定する必要があります。配列に含まれる変数を識別するには、配列参照を使用します。配列参照の添字には、変数名および番号を使用できます。したがって、プログラミングステートメントでは、DO ループのインデックス変数を配列参照の添字にできます。

array-name {*index-variable*}

インデックス変数の値が変化すると、配列参照の添字も変化します。同様に、参照されている変数も変化します。

次のステートメントでは、配列参照の添字としてインデックス変数 Count が使用されます。

```
if changelist{Count} = . then changelist{Count} = 0;
```

このステートメントを反復 DO ループ内に配置します。Count の値が 1 の場合、配列参照は CHANGELIST{1}として読み取られ、CHANGELIST{1} (Museums)に対して IF-THEN ステートメントの処理が行われます。Count の値が 2 または 3 の場合は、CHANGELIST{2} (Galleries)または CHANGELIST{3} (Other)に対してステートメントの処理が行われます。配列参照を使用した完全な反復 DO ループは次のようになります。

```
do Count = 1 to 3;
   if changelist{Count} = . then changelist{Count} = 0;
end;
```

このステートメントでは、次の操作が SAS に指示されます。

- DO ループに含まれる操作を 3 回実行します。
- IF-THEN ステートメントの反復ごとに、配列の添字 Count を Count の現在の値で置き換えます。
- 配列参照を使用して変数を特定し、その変数に対して IF-THEN ステートメントを実行します。

次の DATA ステップでは、ARRAY ステートメントと反復 DO ループが使用されています。

```
libname mylib 'permanent-data-library';

data changes;
   set mylib.attractions;
   array changelist{3} Museums Galleries Other;
   do Count = 1 to 3;
      if changelist{Count} = . then changelist{Count} = 0;
   end;
run;

proc print data=changes;
   title 'Tour Attractions';
run;
```

次の出力は結果を示しています。

図 14.3 *ARRAY と反復 DO ループを使用したデータセットの作成*

Tour Attractions

Obs	City	Museums	Galleries	Other	TourGuide	YearsExperience	Count
1	Rome	4	3	0	D'Amico	2	4
2	Paris	5	0	1	Lucas	5	4
3	London	3	2	0	Wilson	3	4
4	New York	5	1	2	Lucas	5	4
5	Madrid	0	0	5	Torres	4	4
6	Amsterdam	3	3	0		.	4

データセット CHANGES では、変数 Museums、Galleries および Other の欠損値がゼロになったことが示されています。さらに、データセットには値 4 (各オブザベーションのループ処理を中止させる値)の変数 Count が含まれています。

Count をデータセットから除外するには、DROP=データセットオプションを使用します。

```
data changes2 (drop=Count);
   set mylib.attractions;
   array changelist{3} Museums Galleries Other;
   do Count = 1 to 3;
      if changelist{Count} = . then changelist{count} = 0;
   end;
run;

proc print data=changes2;
   title 'Tour Attractions';
run;
```

次の出力は結果を示しています。

図 14.4 *データセットのインデックス変数の削除*

Tour Attractions

Obs	City	Museums	Galleries	Other	TourGuide	YearsExperience
1	Rome	4	3	0	D'Amico	2
2	Paris	5	0	1	Lucas	5
3	London	3	2	0	Wilson	3
4	New York	5	1	2	Lucas	5
5	Madrid	0	0	5	Torres	4
6	Amsterdam	3	3	0		.

要約

ステートメント

ARRAY *array-name*{*number-of-variables*} *variable-1* < *...variable-n*>;
現在の DATA ステップを処理するために存在する、名前と順序を付けた変数のリストを作成します。*Array-name* は有効な SAS 名である必要があります。各 variable は配列に含める変数の名前です。*Number-of-variables* はリストされる変数の数です。

配列内に変数を配置すると、*array-name* {*position*}により変数がアクセス可能になり、このとき *position* はリスト内の変数の位置(1 から *number-of-variables* まで)です。この変数アクセス方法は配列参照、位置は配列参照の添字と呼ばれます。ARRAY ステートメントで変数をリストすると、同じ DATA ステップ内のプログラミングステートメントで元の変数名と配列参照のどちらでも使用できるようになります。

このドキュメントでは、添字は中かっこで囲まれます。丸かっこ( )も受け入れ可能で、大かっこ[ ]はその文字がサポートされている動作環境では受け入れ可能です。サポートされている文字を確認するには、使用している動作環境のドキュメントを参照してください。

DO;
... SAS statements ...
END;
囲まれた *SAS statements* を 1 単位として処理します。DO で始まって END で終わるステートメントのグループは DO グループと呼ばれます。DO グループは通常 THEN 句または ELSE ステートメント内に出現します。

DO *index-variable*=1 TO *number-of-variables-in-array*;
... SAS statements ...
END;
反復 DO ループと呼ばれます。DATA ステップの実行ごとに、*index-variable* の値に基づいて反復 DO ループが繰り返し処理(反復)されます。インデックス変数を作成するには、反復 DO ステートメントで SAS 変数名を使用します。

配列処理に反復 DO ループを使用する場合、通常、*index-variable* の値は 1 で始まり、ループが反復する前に毎回 1 つ増加します。値が *number-of-variables-in-array* (通常は処理される配列内の変数の数)よりも大きくなると、ループの処理が停止され、DATA ステップ内の次のステートメントに進みます。

配列処理では、通常、反復 DO ループ内の SAS ステートメントに、添字がインデックス変数名の配列参照(*array-name* {*index-variable*}など)が含まれます。ループの反復ごとに、参照内の添字がインデックス変数の現在値に置き換えられます。したがって、ループの反復が連続すると、配列内の最初の変数、2 番目の変数(以下同様に続く)の順にステートメントの処理が行われます。

詳細情報

配列
配列は 1 次元にも多次元にもできます。詳細については、“Array Processing” (*SAS Language Reference: Concepts*)、“ARRAY Statement” (*SAS Statements: Reference*)、および“Array Reference Statement” (*SAS Statements: Reference*)を参照してください。

DO グループ
反復 DO ステートメントは柔軟かつ強力です。配列処理以外の多くの状況でも役に立ちます。インデックス変数の範囲は、任意の数字で始めたり終えたりできます。また、増分には任意の正数または負数を指定できます。インデックス変数の範囲は、開始値と終了値として、DIM、LBOUND および HBOUND 関数の値、カンマで区切られた値のリスト、またはこれらを組み合わせて指定できます。範囲には WHILE 句や UNTIL 句も含められます。インデックス変数は文字変数にすることもできます(その場合、範囲は文字値のリストとして指定する必要があります)。詳細については、次の文書を参照してください。

- “DO Statement” (*SAS Statements: Reference*)
- “DO Statement, Iterative” (*SAS Statements: Reference*)
- “DIM Function” (*SAS Functions and CALL Routines: Reference*)
- “HBOUND Function” (*SAS Functions and CALL Routines: Reference*)
- “LBOUND Function” (*SAS Functions and CALL Routines: Reference*)

DO WHILE ステートメントと DO UNTIL ステートメント

DO WHILE ステートメントでは、条件が真の間ループが処理されます。DO UNTIL ステートメントでは、条件が真になるまでループが処理されます。(DO UNTIL ループでは常に、少なくとも 1 回は処理が行われます。DO WHILE ループでは、最初に条件が偽の場合、処理はまったく行われません。)詳細については、“DO UNTIL Statement” (*SAS Statements: Reference*)および“DO WHILE Statement” (*SAS Statements: Reference*)を参照してください。

15 章

SAS System での日付の操作

日付の操作について **224**
目的 224
前提条件 224

SAS での日付の処理方法について **224**
SAS での日付値の格納法 224
年が 2 桁の日付の世紀を決定 225

例で使用される入力ファイルおよび入力 SAS データセット **226**

日付の入力 **226**
日付値の入力形式について 226
日付値の読み取り 227
優れたプログラミング実践による日付の読み取り 228
日付を定数として使用 230

日付の表示 **231**
SAS での値の表示方法について 231
日付値のフォーマット 232
永久日付出力形式を変数に割り当て 233
出力形式の一時変更 235

計算に日付を使用する **235**
日付の並べ替え 235
日付変数の新規作成 236

SAS 日付関数の使用 **237**
曜日の検索 237

期間と SAS 日付値の比較 **239**

要約 **240**
ステートメント 240
日付の出力形式と入力形式 240
関数 241
システムオプション 241

詳細情報 **241**

日付の操作について

目的

SAS では日付を一意の数字として格納し、他の数値変数と同様にプログラムで使用できるようにします。このセクションでは、次の操作を行う方法を学習します。

- 生データファイルの日付を読み取り、SAS 日付値として格納します。
- SAS 日付値の表示に使用するカレンダ形式を指定します。
- 日付を使用して計算します。すなわち、日付間の日数を決定したり、日付が何曜日に当たるかを割り出したり、今日の日付を計算に使用したりします。

前提条件

このセクションを先に進む前に、次のトピックを理解している必要があります。

- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)
- 11 章, “オブザベーションのサブセットの作成” (167 ページ)
- 12 章, “グループ化されたオブザベーションや並べ替えられたオブザベーションの操作” (181 ページ)

SAS での日付の処理方法について

SAS での日付値の格納法

日付はさまざまな方法で書き込まれます。日付には数字しか含まないものもあります。また、数字、英文字、文字の各種組み合わせを含む日付もあります。たとえば、次の形式はすべて 2013 年 7 月 26 日を表します。

表 15.1 *日付のフォーマット方法*

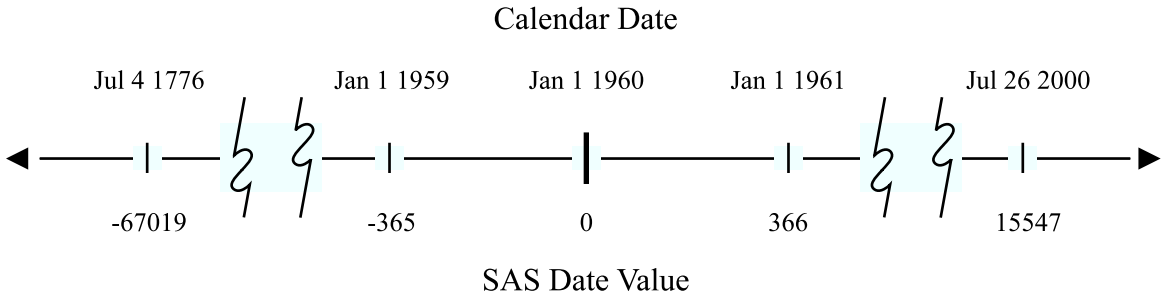
072613	26JUL13	132607
July 26, 2013	26JUL2013	July 26, 2013

さまざまな日付形式があるので、日付の入力方法や表示方法にかかわらず日付を格納して計算に使用する方法が必要になります。

SAS で日付を表すために使用する共通基盤は SAS データ値と呼ばれます。日付の書き込みにどの形式を使用しても、その日付は変換され、1960 年 1 月 1 日と入力日付との間の日数として格納されます。

次の図は、カレンダ形式の日付と SAS 日付値の対応を表したものです。

図 15.1 カレンダ日付と SAS 日付値の比較



SAS では、すべての日付が数直線上の一意の数字になります。1960 年 1 月 1 日より前の日付は負の数、それより後の日付は正の数で表します。SAS 日付値は数値変数なので、簡単に並べ替えて、時間間隔を決定し、日付を定数や SAS 関数の引数として使用したり、計算に使用したりできます。

注: SAS 日付値は、グレゴリオ暦に基づく西暦 1582 年から西暦 19,900 年までの日付に対して有効です。歴史上の日付を処理する場合は注意してください。グレゴリオ暦は 1582 年からヨーロッパのほぼ全土で使用されていますが、イギリスおよびアメリカの植民地では 1752 年までグレゴリオ暦は採用されていませんでした。

年が 2 桁の日付の世紀を決定

外部データソースや SAS プログラムステートメントの日付に 2 桁の年が含まれている場合、YEARCUTOFF=システムオプションを使用して割り当てる西暦の上 2 桁を決定します。YEARCUTOFF=システムオプションでは、2 桁の年の世紀を決定するために使用する 100 年間の最初の年が指定されます。たとえば、YEARCUTOFF=1950 は 2 桁の年 50 - 99 が 1950 - 1999 年に対応することを示します。2 桁の年 00 - 49 は 2000 - 2049 年に対応します。YEARCUTOFF=システムオプションのデフォルト値は 1926 ですが、処理する日付値の範囲に合わせて OPTIONS ステートメントの YEARCUTOFF=値を変更できます。

YEARCUTOFF=システムオプションを使用する前に、データ内の日付を検証します。

- データ内の日付が 100 年間の範囲に収まる場合は、YEARCUTOFF=システムオプションを使用します。
- データ内の日付が 100 年間の範囲内に収まっていない場合は、2 桁の年を 4 桁の年に変換するか、DATA ステップで条件付きロジックを使用して適切な西暦の上 2 桁を割り当てるようにする必要があります。

YEARCUTOFF=システムオプションがデータ範囲にとって適切だと判断したら、使用する設定を決定します。YEARCUTOFF=に最適な設定は、データ内で最も前の年の前年です。たとえば、データの範囲が 2013 から 2112 までの場合は、YEARCUTOFF=を 2013 に設定します。YEARCUTOFF=を 2013 に設定した結果は次のとおりです。

- 13 から 99 までの範囲内にある 2 桁の日付はすべて、2013 から 2099 までとして解釈されます。
- 00 から 12 までの範囲内にある 2 桁の日付はすべて、2100 から 2112 までとして解釈されます。

YEARCUTOFF=を 2013 に設定すると、2 桁の年の 13 は 2013 と解釈され、2 桁の年の 05 は 2105 と解釈されます。

例で使用される入力ファイルおよび入力 SAS データセット

旅行業では、ツアーに関する最重要データに次のような日付が含まれます。

- ツアーの出発日と帰宅日
- 支払い期限
- 払い戻し期限など

Tradewinds Travel には、過去および今後の人気ツアーの日付と、そのツアーの宿泊日数を含むデータがあります。生データは、次のような外部ファイルに格納されます。

```
Japan       13may2000  8
Greece      17oct99   12
New Zealand 03feb2001 16
Brazil      28feb2001  8
Venezuela   10nov00    9
Italy       25apr2001  8
Russia      03jun1997 14
Switzerland 14jan2001  9
Australia   24oct98   12
Ireland     27aug2000  7
```

最初の列には、ツアー先の国名がリスト表示されます。2 番目の列には、出発日がリスト表示されます。3 番目の列には、ツアーの宿泊日数がリスト表示されます。

日付の入力

日付値の入力形式について

値を SAS 日付値として読み取らせるには、入力形式と呼ばれる指示セットを指定する必要があります。デフォルトでは、数値変数は、英文字や特殊文字を含まない標準数値入力形式で読み取られます。データを含むフィールドが標準パターンと一致しない場合は、INPUT ステートメントで適切な入力形式を指定します。

SAS では多くの入力形式が適用されています。日付値の読み取りによく使用される 4 つの入力形式は次のとおりです。

MMDDYY8.
　書き込まれた日付を mm/dd/yy として読み取ります。

MMDDYY10.
　書き込まれた日付を mm/dd/yyyy として読み取ります。

DATE7.
　ddMMMyy 形式で日付を読み取ります。

DATE9.
　ddMMMyyyy 形式で日付を読み取ります。

各入力形式名は、末尾がピリオドで、読み取るカラム数を示す幅指定が含まれていることに注意してください。

日付値の読み取り

Tradewinds Travel データの SAS データセットを作成するには、INPUT ステートメントで DATE9.入力形式を使用して、変数 DepartureDate を読み取ります。

```
input Country $ 1-11 @13 DepartureDate date9. Nights;
```

INPUT ステートメントで入力形式を使用することをフォーマット入力と呼びます。この例のフォーマット入力には次の項目が含まれます。

- 値の開始カラムを示すポインタ(@13)
- 読み取る変数の名前(DepartureDate)
- 使用する入力形式の名前(DATE9.)

次の DATA ステップでは、SAS 日付値を作成するために DATE9.入力形式を使用して MYLIB.TOURDATES が作成されます。

```
options yearcutoff=1920;
libname mylib 'permanent-data-library';

data mylib.tourdates;
   infile 'input-file';
   input Country $ 1-11 @13 DepartureDate date9. Nights;
run;

proc print data=mylib.tourdates;
   title 'Tour Departure Dates as SAS Date Values';
run;
```

次の出力は結果を示しています。

図 15.2　カレンダ日付から SAS 日付値を作成

Tour Departure Dates as SAS Date Values

Obs	Country	DepartureDate	Nights
1	Japan	14743	8
2	Greece	14534	12
3	New Zealand	15009	16
4	Brazil	15034	8
5	Venezuela	14924	9
6	Italy	15090	8
7	Russia	13668	14
8	Switzerland	14989	9
9	Australia	14176	12
10	Ireland	14849	7

変数 DepartureDate の SAS 値と前のセクションで示した生データの値を比較してください。データセット MYLIB.TOURDATES は、出発日が読み取られ、SAS 日付値が作成されたことを示しています。ここで認識可能な形式で日付を表示する方法が必要になります。

優れたプログラミング実践による日付の読み取り

日付の読み取り時には、DATE9.入力形式か MMDDYY10.入力形式を常に使用して、データを正しく読み取るようにすることが優れたプログラミング実践法です。DATE7.入力形式や MMDDYY8.入力形式を使用した場合は、年の最初の 2 桁のみが読み取られます。データに 4 桁の年が含まれている場合、年ではなく世紀が読み取られます。

例では、PRINT プロシジャで FORMAT オプションを使用して日付をフォーマットします。

2 桁の年と 4 桁の年の両方を含む Tradewinds Travel 外部ファイルがあるとします。

```
Japan       13may2000  8
Greece      17oct99   12
New Zealand 03feb2001 16
Brazil      28feb2001  8
Venezuela   10nov00    9
Italy       25apr2001  8
France      03jun1997 14
Switzerland 14jan2001  9
Australia   24oct98   12
Ireland     27aug2000  7
```

次の DATA ステップでは、DATE7.入力形式を使用して SAS データセット MYLIB.TOURDATES7 が作成されます。

```
options yearcutoff=1920;

data mylib.tourdates7;
   infile 'input-file';
   input Country $ 1-11 @13 DepartureDate date7. Nights;
run;

proc print data=mylib.tourdates7;
   title 'Tour Departure Dates Using the DATE7. Informat';
   title2 'Displayed as Two-Digit Calendar Dates';
   format DepartureDate date7.;
run;

proc print data=mylib.tourdates7;
   title 'Tour Departure Dates Using the DATE7. Informat';
   title2 'Displayed as Four-Digit Calendar Dates';
   format DepartureDate date9.;
run;
```

PRINT プロシジャでは、2 桁の年(DATE7.)と 4 桁の年(DATE9.)のカレンダ日付を使用して DepartureDate がフォーマットされます。次の出力は、間違った日付入力形式を

使用すると無効 SAS データセットが作成される可能性があることを示しています。プログラムによる出力を次に示します。

図 15.3 間違った日付入力形式の使用

Tour Departure Dates Using the DATE7. Informat
Displayed as Two-Digit Calendar Dates

Obs	Country	DepartureDate	Nights
1	Japan	13MAY20	0
2	Greece	17OCT99	12
3	New Zealand	03FEB20	1
4	Brazil	28FEB20	1
5	Venezuela	10NOV00	9
6	Italy	25APR20	1
7	Russia	03JUN19	97
8	Switzerland	14JAN20	1
9	Australia	24OCT98	12
10	Ireland	27AUG20	0

図 15.4 間違った入力形式による無効 SAS データセット作成の可能性

Tour Departure Dates Using the DATE7. Informat
Displayed as Four-Digit Calendar Dates

Obs	Country	DepartureDate	Nights
1	Japan	13MAY1920	0
2	Greece	17OCT1999	12
3	New Zealand	03FEB1920	1
4	Brazil	28FEB1920	1
5	Venezuela	10NOV2000	9
6	Italy	25APR1920	1
7	Russia	03JUN2019	97
8	Switzerland	14JAN1920	1
9	Australia	24OCT1998	12
10	Ireland	27AUG1920	0

オブザベーション 1、3、4、6、7、8、10 について入力ファイルの 4 桁の年が MYLIB.TOURDATES7 の年と一致しないことに注意してください。

- 7 文字読み取った後で日付の読み取りが停止されました。完全な 4 桁の年ではなく最初の 2 桁(世紀)が読み取られました。
- 次の変数のデータを読み取るために、ポインタが 1 カラム移動され、次の 2 つの数字(年 00、01 および 97)が変数 Nights の値として読み取られました。入力ファイルの Nights のデータは無視されました。
- 4 桁のカレンダ日付の日付フォーマットを行う際に、YEARCUTOFF= 1920 システムオプションを使用して、2 桁の年の世紀を決定しました。オブザベーション 7 でもともと 1997 だった値が 2019 になり、オブザベーション 1、3、4、6、8、10 でもともと 2000 および 2001 だった値が 1920 になりました。

日付を定数として使用

Switzerland ツアーの出発日を 1 月 14 日のかわりに 2001 年 1 月 21 日にする場合は、次の割り当てステートメントを使用して更新を行います。

```
if Country = 'Switzerland' then DepartureDate = '21jan2001'd;
```

値'21jan2001'D は SAS 日付定数です。SAS 日付定数を記述するには、標準 SAS 形式 ddMMMyyyy の日付を引用符で囲み、最後の引用符のすぐ後に英文字 D を付けます。D 接尾辞によって、カレンダ日付の SAS 日付値への変換が指定されます。次の DATA ステップでは、SAS 日付定数が使用されています。PRINT プロシジャの FORMAT オプションによって、DATE9.出力形式で日付が書き込まれます。

```
data correctdates;
   set mylib.tourdates;
   if Country = 'Switzerland' then DepartureDate = '21jan2001'd;
run;

proc print data=correctdates;
   title 'Corrected Departure Date for Switzerland';
   format DepartureDate date9.;
run;
```

次の出力は結果を示しています。

図 15.5 SAS 日付定数を使用した日付変更

Corrected Departure Date for Switzerland

Obs	Country	DepartureDate	Nights
1	Japan	13MAY2000	8
2	Greece	17OCT1999	12
3	New Zealand	03FEB2001	16
4	Brazil	28FEB2001	8
5	Venezuela	10NOV2000	9
6	Italy	25APR2001	8
7	Russia	03JUN1997	14
8	Switzerland	21JAN2001	9
9	Australia	24OCT1998	12
10	Ireland	27AUG2000	7

日付の表示

SAS での値の表示方法について

出発日の表示方法を理解するには、SAS での通常の値表示方法を理解する必要があります。SAS ではすべてのデータ値が出力形式と呼ばれる指示セットを使用して表示されます。デフォルトでは、数値変数の値を表示するために、カンマ、英文字、またはその他特殊表記を含まない標準数値出力形式が使用されます。図 15.2 (227 ページ)では、SAS 日付値を標準数値出力形式で書き込むと認識の困難な数字が生成されることが示されています。これらの数字をカレンダ日付として表示するには、変数に対して SAS 日付出力形式を指定する必要があります。

SAS 日付出力形式は、最もよく見られるカレンダ日付書き込み方法で使用可能です。DATE9.出力形式では、日付が ddMMMyyyy 形式で表示されます。月、日、年を略さずに表示する場合は、WORDDATE18.出力形式を使用します。WEEKDATE29.出力形式には曜日が含まれます。mm/dd/yy 形式でカレンダ日付を表示する出力形式 MMDDYY8.や、mm/dd/yyyy 形式でカレンダ日付を表示する出力形式 MMDDYY10.など、数字表示に使用可能な出力形式もあります。入力形式名と同様に、各出力形式名は、末尾がピリオドで、日付値の表示時に使用するカラム数を示す幅指定が含まれています。

日付値のフォーマット

使用する出力形式を指定するには、FORMAT ステートメントで変数と出力形式名を指定します。次の FORMAT ステートメントでは、MMDDYY10.出力形式が変数 DepartureDate に割り当てられます。

```
format DepartureDate mmddyy10.;
```

この例では、FORMAT ステートメントに次の項目が含まれています。

- 変数の名前(DepartureDate)
- 使用する出力形式の名前(MMDDYY10.)

次の PRINT プロシジャでは、2 桁の年のカレンダ出力形式と 4 桁の年のカレンダ出力形式の両方で変数 DepartureDate がフォーマットされます。

```
proc print data=mylib.tourdates;
   title 'Departure Dates in Two-Digit Calendar Format';
   format DepartureDate mmddyy8.;
run;

proc print data=mylib.tourdates;
   title 'Departure Dates in Four-Digit Calendar Format';
   format DepartureDate mmddyy10.;
run;
```

次の出力は結果を示しています。

図 15.6 フォーマット日付値の表示: 2 桁のカレンダ出力形式

Departure Dates in Two-Digit Calendar Format

Obs	Country	DepartureDate	Nights
1	Japan	05/13/00	8
2	Greece	10/17/99	12
3	New Zealand	02/03/01	16
4	Brazil	02/28/01	8
5	Venezuela	11/10/00	9
6	Italy	04/25/01	8
7	Russia	06/03/97	14
8	Switzerland	01/14/01	9
9	Australia	10/24/98	12
10	Ireland	08/27/00	7

図 15.7 フォーマット日付値の表示: 4 桁のカレンダ出力形式

Departure Dates in Four-Digit Calendar Format

Obs	Country	DepartureDate	Nights
1	Japan	05/13/2000	8
2	Greece	10/17/1999	12
3	New Zealand	02/03/2001	16
4	Brazil	02/28/2001	8
5	Venezuela	11/10/2000	9
6	Italy	04/25/2001	8
7	Russia	06/03/1997	14
8	Switzerland	01/14/2001	9
9	Australia	10/24/1998	12
10	Ireland	08/27/2000	7

PROC ステップに FORMAT ステートメントを配置すると、そのステップでのみ出力形式と変数が関連付けられます。出力形式と変数を永久的に関連付けるには、DATA ステップで FORMAT ステートメントを使用します。

永久日付出力形式を変数に割り当て

次の例では、永久 SAS データセットが新規作成され、DATA ステップで DATE9.出力形式が割り当てられます。これにより変数 DepartureDate を使用する後続のプロシジャおよび DATA ステップすべてにおいて、デフォルトで DATE9.出力形式が使用されるようになります。PROC CONTENTS ステップでは、データセット MYLIB.TOURDATE の特性が表示されます。

```
options yearcutoff=1920;

data mylib.fmttourdate;
   set mylib.tourdates;
   format DepartureDate date9.;
run;

proc contents data=mylib.fmttourdate nodetails;
run;
```

次の出力は、DATE9.出力形式が DepartureDate と永久的に関連付けられたことを示しています。

図 15.8 DATA ステップでの出力形式の割り当て

The CONTENTS Procedure

Data Set Name	MYLIB.FMTTOURDATE	Observations	10
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	04/30/2013 10:14:06	Observation Length	32
Last Modified	04/30/2013 10:14:06	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	2039
Obs in First Data Page	10
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\Users\lirezn\mylib\fmttourdate.sas7bdat
Release Created	9.0401B0
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Format
1	Country	Char	11	
2	DepartureDate	Num	8	DATE9.
3	Nights	Num	8	

出力形式の一時変更

異なる出力形式の日付を必要とするレポートを準備している場合、PROC ステップで FORMAT ステートメントを使用して永久出力形式を上書きできます。たとえば、データセット MYLIB.TOURDATES の DepartureDate 値を month-name dd, yyyy 形式で表示するには、PROC PRINT ステップで FORMAT ステートメントを発行します。次のプログラムでは、変数 DepartureDate に対して WORDDATE18.出力形式が指定されます。

```
proc print data=mylib.tourdates;
   title 'Tour Departure Dates';
   format DepartureDate worddate18.;
run;
```

次の出力は結果を示しています。

図 15.9　事前指定した出力形式の上書き

Tour Departure Dates

Obs	Country	DepartureDate	Nights
1	Japan	May 13, 2000	8
2	Greece	October 17, 1999	12
3	New Zealand	February 3, 2001	16
4	Brazil	February 28, 2001	8
5	Venezuela	November 10, 2000	9
6	Italy	April 25, 2001	8
7	Russia	June 3, 1997	14
8	Switzerland	January 14, 2001	9
9	Australia	October 24, 1998	12
10	Ireland	August 27, 2000	7

出力形式 DATE9.はまだ DepartureDate に永久的に割り当てられています。残りの例のカレンダ日付は、PROC PRINT ステップに FORMAT ステートメントが含まれない限り、ddMMMyyyy 形式です。

計算に日付を使用する

日付の並べ替え

SAS 日付値は数値変数なので、並べ替えて計算に使用できます。次の例では、データセット MYLIB.TOURDATES を使用して、Tradewinds Travel データについてその他の情報を抽出します。

ツアーのスケジュール頻度の特定に役立てるために、ツアーを時間順にリストしたレポートを印刷できます。最初のステップは、PROC SORT ステップで次の BY ステートメントを指定して、オブザベーションを日付変数 DepartureDate の昇順に並べ替えるよう指定することです。

```
by DepartureDate;
```

次の PROC PRINT ステップで VAR ステートメントを使用することにより、出発日をレポートの最初の列としてリストできます。

```
proc sort data=mylib.fmttourdate out=sortdate;
   by DepartureDate;
run;

proc print data=sortdate;
   var DepartureDate Country Nights;
   title 'Departure Dates Listed in Chronological Order';
run;
```

次の出力は結果を示しています。

図 15.10 *SAS 日付値順の並べ替え*

Departure Dates Listed in Chronological Order

Obs	DepartureDate	Country	Nights
1	03JUN1997	Russia	14
2	24OCT1998	Australia	12
3	17OCT1999	Greece	12
4	13MAY2000	Japan	8
5	27AUG2000	Ireland	7
6	10NOV2000	Venezuela	9
7	14JAN2001	Switzerland	9
8	03FEB2001	New Zealand	16
9	28FEB2001	Brazil	8
10	25APR2001	Italy	8

データセット SORTDATE のオブザベーションは今は時間順に並んでいます。この例では FORMAT ステートメントがないので、データセット MYLIB.FMTTOURDATE の作成時に DepartureDate に割り当てた DATE9.出力形式で日付が表示されることに注意してください。

日付変数の新規作成

各ツアーの出発日と宿泊日数がわかっているので、各ツアーの帰宅日を計算できます。まず初めに、次のように宿泊日数を出発日に加算して新しい変数を作成します。

```
Return = DepartureDate + Nights;
```

その結果は帰宅日の SAS 日付値で、次のように DATE9.出力形式を割り当てると表示できます。

```
options yearcutoff=1920;
data home;
   set mylib.tourdates;
   Return = DepartureDate + Nights;
   format Return date9.;
run;

proc print data=home;
   title 'Dates of Departure and Return';
run;
```

次の出力は結果を示しています。

図 15.11 *日付値に日数を加算*

Dates of Departure and Return

Obs	Country	DepartureDate	Nights	Return
1	Japan	14743	8	21MAY2000
2	Greece	14534	12	29OCT1999
3	New Zealand	15009	16	19FEB2001
4	Brazil	15034	8	08MAR2001
5	Venezuela	14924	9	19NOV2000
6	Italy	15090	8	03MAY2001
7	Russia	13668	14	17JUN1997
8	Switzerland	14989	9	23JAN2001
9	Australia	14176	12	05NOV1998
10	Ireland	14849	7	03SEP2000

データセット MYLIB.TOURDATES の変数 DepartureDate には永久出力形式がないので、その変数については判読できるカレンダ日付のかわりに数値が表示されます。

SAS 日付関数の使用

曜日の検索

SAS には SAS 日付値からカレンダ日付を生成するさまざまな関数があります。SAS 日付関数を使用すると、一部の日付情報を取得したり、現在の日付を計算に使用したりするなどの処理を行えます。

ツアーの最終支払い期限がツアーの出発 30 日前の場合、減算によって最終支払い日を計算できます。ただし、Tradewinds Travel 社は日曜は休業します。支払い期限が

日曜の場合、さらに 1 日を減算して支払い期限を土曜にする必要があります。曜日を 1 から 7(日曜から土曜)までの数字として返す WEEKDAY 関数を使用すると、返された日が日曜かどうか判断できます。

次のステートメントでは、最終支払い日を決定するために次の処理を行います。

- 出発日から 30 を減算
- WEEKDAY 関数で返された値のチェック
- 必要に応じてさらに 1 日を減算

```
DueDate=DepartureDate - 30;
if Weekday(DueDate)=1 then DueDate=DueDate - 1;
```

これらのステートメントでデータセットを作成すると、支払期日リストが生成されます。次のプログラムでは、これらのステートメントが含まれ、出力形式 WEEKDATE29.が新しい変数 DueDate に割り当てられます。

```
options yearcutoff=1920;
data pay;
   set mylib.tourdates;
   DueDate = DepartureDate - 30;
   if Weekday(DueDate) = 1 then DueDate = DueDate - 1;
   format DueDate weekdate29.;
run;

proc print data=pay;
   var Country DueDate;
   title 'Date and Day of Week Payment Is Due';
run;
```

次の出力は結果を示しています。

図 15.12 *WEEKDAY 関数の使用*

Date and Day of Week Payment Is Due

Obs	Country	DueDate
1	Japan	Thursday, April 13, 2000
2	Greece	Friday, September 17, 1999
3	New Zealand	Thursday, January 4, 2001
4	Brazil	Monday, January 29, 2001
5	Venezuela	Wednesday, October 11, 2000
6	Italy	Monday, March 26, 2001
7	Russia	Saturday, May 3, 1997
8	Switzerland	Friday, December 15, 2000
9	Australia	Thursday, September 24, 1998
10	Ireland	Friday, July 28, 2000

期間と SAS 日付値の比較

SAS 日付値を使用すると、日付間の時間の単位を調べられます。Tradewinds Travel 社は 1982 年 2 月 8 日に設立されました。1999 年 11 月 23 日に Tradewinds Travel 社の設立後年数を調べることになりました。

```
options yearcutoff=1920;
   /* Calculating a duration in days */
data ttage;
   Start = '08feb82'd;
   RightNow = today();
   Age = RightNow - Start;
   format Start RightNow date9.;
run;

proc print data=ttage;
   title 'Age of Tradewinds Travel';
run;
```

次の出力は結果を示しています。

図 15.13 期間の日数計算

Age of Tradewinds Travel

Obs	Start	RightNow	Age
1	08FEB1982	30APR2013	11404

Age の値は 11404 です。この数字は未フォーマットの SAS 日付値のように見えます。ただし、Age は実際には 1982 年 2 月 8 日と 2013 年 4 月 30 日との差で、SAS 日付値ではなく日数で期間を表しています。Age の値を理解しやすくするには、日数を 365(より正確には 365.25)で割って年数を求めます。次の DATA ステップでは、Tradewinds Travel 社の設立後年数が計算されます。

```
options yearcutoff=1920;
   /* Calculating a duration in years */
data ttage2;
   Start = '08feb82'd;
   RightNow = today();
   AgeInDays = RightNow - Start;
   AgeInYears = AgeInDays / 365.25;
   format AgeInYears 4.1 Start RightNow date9.;
run;

proc print data=ttage2;
   title 'Age in Years of Tradewinds Travel';
run;
```

次の出力は結果を示しています。

図 15.14 期間の年数計算

Age in Years of Tradewinds Travel

Obs	Start	RightNow	AgeInDays	AgeInYears
1	08FEB1982	30APR2013	11404	31.2

年の一部を示すために、DATA ステップの FORMAT ステートメントで AgeInYears の値に数値出力形式 4.1 が割り当てられます。4 によって数字に含まれる文字数が 4 以下になります。1 によって数字に含まれる小数点以下桁数が 1 になります。

要約

ステートメント

*date-variable='ddMMMyy'*D;
引用符内の日付を SAS 日付値に変換し、*date-variable* に割り当てるよう指示する割り当てステートメントです。SAS 日付定数'*ddMMMyy*'D には特定の日付(例: '23NOV00'D)を指定し、割り当てステートメントだけでなく多くの SAS ステートメントや式で使用します。

FORMAT *date-variabledate-format*;
date-variable の値を *date-format* を使用してフォーマットするよう指示します。DATA ステップ内の FORMAT ステートメントでは、出力形式と *date-variable* が永久的に関連付けられます。

INPUT *date-variabledate-informat*;
外部ファイルからの date-variable の値の読み取り方法を指示します。*date-informat* は外部ファイルの日付形式を指示する命令です。

日付の出力形式と入力形式

DATE9.
date-variable の形式は ddMMMyyyy です(例: 30APR2013)。

DATE7.
date-variable の形式は ddMMMyy です(例: 23NOV00)。

MMDDYY10.
date-variable の形式は mm/dd/yyyy です(例: 11/23/2012)。

MMDDYY8.
date-variable の形式は mm/dd/yy です(例: 11/23/00)。

WORDDATE18.
date-variable の形式は month-name dd, yyyy です(例: November 23, 2013)。

WEEKDATE29.
date-variable の形式は day-of-the-week, month-name dd, yyyy です(例: Thursday, November 23, 2013)。

関数

WEEKDAY (*SAS-date-value*)
SAS 日付値の曜日を返す関数です。値は 1 から 7 までの数字で、日曜には値 1 が割り当てられます。

TODAY()
SAS プログラムの開始日に対応する SAS 日付値を返す関数です。

システムオプション

YEARCUTOFF=
100 年間の最初の年を指定して、入力形式と関数では 2 桁の年の読み取りに使用し、出力形式では 2 桁の年の表示に使用します。YEARCUTOFF=に値を指定した結果、年範囲が 2 世紀にわたる場合があります。YEARCUTOFF=1950 の場合、50 から 99 までの 2 桁の値は 100 年間の前半を指し、1900 年代になります。00 から 49 までの 2 桁の値は 100 年間の後半を指し、2000 年代になります。YEARCUTOFF=は、既存の SAS 日付、または入力データから読み取られた 4 桁の年を含む日付には影響ありません。

詳細情報

ATTRIB ステートメント
ATTRIB ステートメントを使用した永久出力形式の割り当てや変更については *SAS Statements: Reference* で確認できます。

DATASETS プロシジャ
変数を永久出力形式に割り当てたり変更したりするには、36 章, “SAS ライブラリの管理” (675 ページ)の DATASETS プロシジャを参照してください。

PUT 関数と INPUT 関数
PUT 関数と INPUT 関数は、SAS 日付の処理時によく発生するエラーを修正するために使用します。このエラーは、英文字や記号を含む日付値を文字変数として処理することや、数字として書き込まれた日付を通常の数値変数として格納することです。どちらの方法でも日付を計算に使用することはできません。詳細については、“PUT Statement” (*SAS Statements: Reference*)および “INPUT Statement” (*SAS Statements: Reference*)を参照してください。

SAS 日付値
SAS 日付値、SAS 時間および SAS 日時値を処理するための入力形式、出力形式および関数についてのドキュメントは *SAS Language Reference: Concepts* で確認できます。このドキュメントには次の日付と時間の情報が含まれます。

- SAS では当日午前 0 時からの経過秒数として時間を格納します。たとえば、午前 9 時 30 分は 34200 です。この種の数字は SAS 時間値と呼ばれます。SAS 時間値は日付とは無関係で、毎日午前 0 時に 0 から始まります。
- 日付と時間の両方が存在する場合は、1960 年 1 月 1 日午前 0 時からの経過秒数として値を格納します。たとえば、2000 年 11 月 23 日午前 9 時 30 分は 1290591000 になります。この種の数字は SAS 日時値と呼ばれます。
- SAS 日時の入力形式ではさまざまな幅のフィールドが読み取られます。SAS 日時の出力形式では、出力形式名で指定する幅に従ってさまざまな方法で日付変数を表示できます。出力形式や入力形式の末尾の数字は使用可能なカラ

ム数を示しています。たとえば、DATE9.入力形式では 9 カラムまで読み取られます(23NOV2013 など)。WEEKDATE8.出力形式では 8 カラム(Thursday など)、WEEKDATE27.では 27 カラム(Thursday, November 23, 2013 など)が表示されます。

- SAS では、さまざまな経過期間をカウントするために、MONTH などの日付、時間、日時の間隔が提供されています。MONTH は特定の月の 1 日から翌月の 1 日までの間隔を表します。30 日間、または 31 日間ということではありません。
- 各国の日付、時間および日時の出力形式。

SYSDATE9
タイトルに現在の日付を含めるには、マクロ変数 SYSDATE9 を使用します。これについては 27 章, “PRINT プロシジャを使用した詳細レポートの作成” (421 ページ)で説明されています。

4 部

SAS データセットの結合

16 章
SAS データセットの結合方法 245

17 章
SAS データセットの連結 253

18 章
SAS データセットのインタリーブ 279

19 章
SAS データセットのマージ 287

20 章
SAS データセットの更新 315

21 章
SAS データセットの変更 333

22 章
複数の SAS データセットのオブザベーションの条件付き処理 347

16 章

SAS データセットの結合方法

SAS データセットの結合について **245**
目的 245
前提条件 245
連結の定義 **246**
インタリーブの定義 **246**
マージの定義 **247**
更新の定義 **248**
変更の定義 **249**
データセットの変更、マージ、更新の比較 **250**
詳細情報 **251**

SAS データセットの結合について

目的

SAS では、複数の SAS データセット結合方法が提供されています。このセクションでは、5 つのデータセット結合方法について説明します。

- 連結
- インタリーブ
- マージ
- 更新
- 変更

後続のセクションでは、これらの方法の使用について説明します。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)

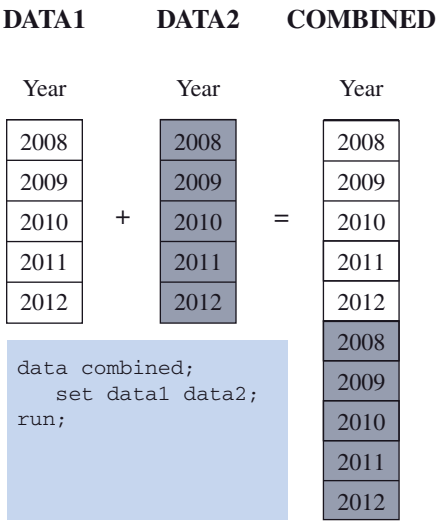
- 6 章, “SAS データセットから作成する” (91 ページ)
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)

連結の定義

連結とは、複数の SAS データセットを結合して、1 つの SAS データセットを作成する処理です。データセットを連結するには、DATA ステップの SET ステートメントか、APPEND プロシジャのどちらかを使用します。

次の図は、2 つの SAS データセットの連結結果と、その結果をもたらす DATA ステップを示しています。

図 16.1 *2 つの SAS データセットの連結*

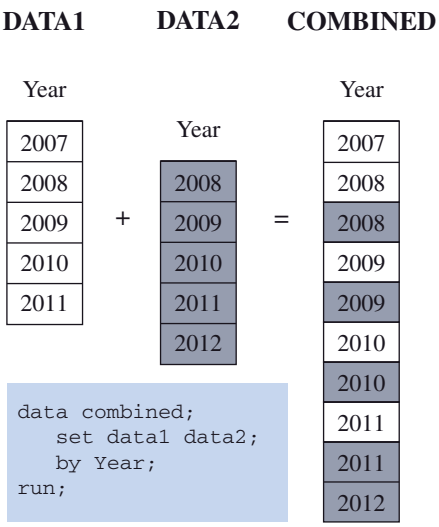


インタリーブの定義

インタリーブとは、並べ替えられた個々の SAS データセットを結合して 1 つの並べ替えられた SAS データセットにする処理です。次の図は、オブザベーションごとにデータセットの並べ替えの基準となる変数の値を示しています。SET ステートメントを BY ステートメントとあわせて使用してデータセットをインタリーブします。

次の例では、データセットは変数 Year を基準にして並べ替えられます。

図 16.2 SAS データセットのインタリーブ

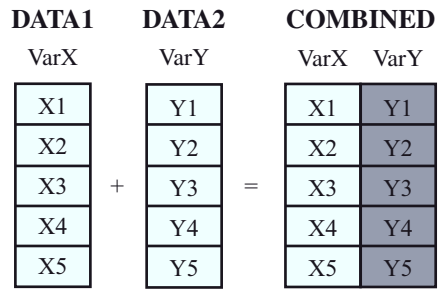


マージの定義

マージとは、複数の SAS データセットのオブザベーションを結合し、新しいデータセットの単一のオブザベーションにする処理です。

1 対 1 のマージでは、次の図に示すように、オブザベーションがデータセット内の位置に基づいて結合されます。1 対 1 のマージには MERGE ステートメントを使用します。

図 16.3 1 対 1 のマージ



マッチマージでは、次の図に示すように、オブザベーションが 1 つ以上の共通変数の値に基づいて結合されます。マッチマージを実行する場合は、MERGE ステートメントを BY ステートメントとあわせて使用します。

次の例では、2 つのデータセットが変数 Year の値を基準にしてマッチマージされます。

図 16.4 2 つの SAS データセットのマッチマージ

DATA1

Year	VarX
2008	X1
2009	X2
2010	X3
2011	X4
2012	X5

+

DATA2

Year	VarY
2008	Y1
2008	Y2
2010	Y3
2011	Y4
2012	Y5

=

COMBINED

Year	VarX	VarY
2008	X1	Y1
2008	X1	Y2
2009	X2	.
2010	X3	Y3
2011	X4	Y4
2012	X5	Y5

```
data combined;
   merge data1 data2;
   by Year;
run;
```

COMBINED データセットの変数 VarY に欠損値があることに注意してください。DATA2 データセットに 2009 年のオブザベーションがないため、値が欠損しています。

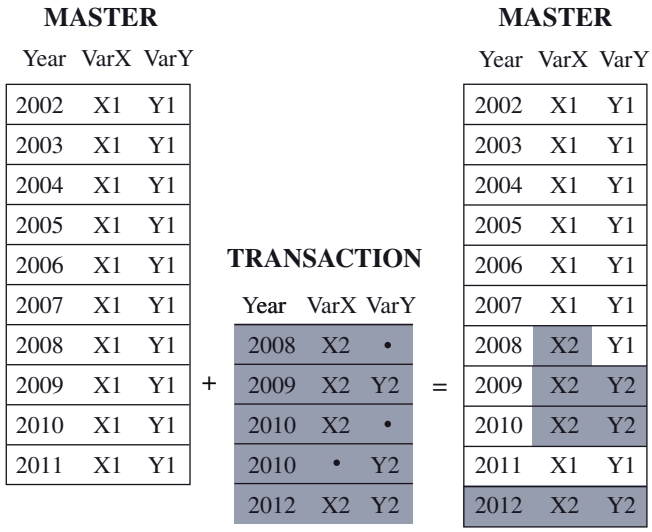
更新の定義

SAS データセットを更新すると、1 つのデータセット(マスタデータセット)の変数値が別のデータセット(トランザクションデータセット)の値に置換されます。UPDATE ステートメントで UPDATEMODE=オプションを MISSINGCHECK に設定すると、トランザクションデータセット内の欠損値はマスタデータセット内の既存値と置き換えられません。UPDATEMODE=オプションを NOMISSINGCHECK に設定すると、トランザクションデータセット内の欠損値はマスタデータセット内の既存値と置き換えられます。デフォルト設定は MISSINGCHECK です。

データセットを更新するには、UPDATE ステートメントを BY ステートメントとあわせて使用します。入力データセットは両方とも BY ステートメントで使用する変数順に並べ替えておく必要があります。

次の図は、SAS データセットを更新した結果を示しています。

図 16.5 マスタデータセットの更新




```
data master;
   update master transaction;
   by Year;
run;
```

TRANSACTION データセットに欠損値が含まれていることに注意してください。更新が発生した場合、新しい MASTER データセットでは、元の MASTER データセットからの値が保持され、欠損値は表示されません。

変更の定義

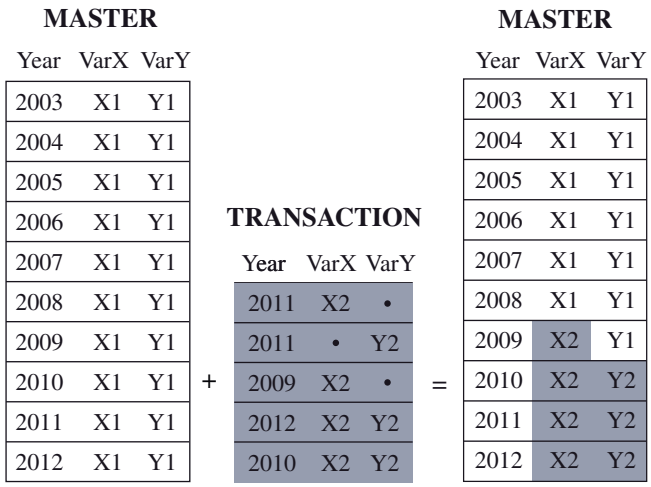
SAS データセットを変更すると、既存のデータセットにあるオブザベーションの置換、削除、追加が行われます。SAS データセットの変更は SAS データセットの更新と似ていますが、次の相違点があります。

- 変更では新しいデータセットは作成できませんが、更新では作成できます。
- 更新と違って、変更ではマスタデータセットやトランザクションデータセットを並べ替える必要はありません。

既存のデータセットを変更するには、MODIFY ステートメントを BY ステートメントとあわせて使用します。

次の例では、MASTER データセットが YEAR によって更新されます。

図 16.6 データセットの変更




```
data master;
   modify master transaction;
   by Year;
run;
```

TRANSACTION データセットに欠損値が含まれていることに注意してください。MASTER データセットが変更された場合、新しい MASTER データセットでは、元の MASTER データセットからの値が保持され、欠損値は表示されません。

データセットの変更、マージ、更新の比較

次の表に、MERGE ステートメント、UPDATE ステートメント、MODIFY ステートメントの相違点をいくつかまとめています。

表 16.1 MERGE ステートメント、UPDATE ステートメント、MODIFY ステートメントの相違点

基準	MERGE ステートメント	UPDATE ステートメント	MODIFY ステートメント
データセットの並べ替えかインデックス付けが必要	マッチマージ: はい 1 対 1 のマージ: いいえ	はい	いいえ
BY 値が一意になることが必要	いいえ	マスタデータセット: はい トランザクションデータセット: いいえ	いいえ
変数の作成または削除が可能	はい	はい	いいえ

基準	MERGE ステートメント	UPDATE ステートメント	MODIFY ステートメント
結合されるデータセットの数	任意の数	2	2
欠損値の処理	最初のデータセットの非欠損値を 2 番目のデータセットの欠損値で上書き	デフォルト動作: トランザクションデータセットの欠損値をマスタデータセットの値と置換しない	UPDATEMODE= オプションの値に依存(“欠損値の処理” (343 ページ)を参照) デフォルト: MISSINGCHECK

詳細情報

データセットの連結
: 詳細については、17 章, “SAS データセットの連結” (253 ページ)を参照してください。

データセットのインタリーブ
: 詳細については、18 章, “SAS データセットのインタリーブ” (279 ページ)を参照してください。

データセットの操作
: データセットは結合時に操作できます。たとえば、各データセットから特定のオブザベーションを選択したり、オブザベーションの元のデータセットを特定したりできます。詳細については、22 章, “複数の SAS データセットのオブザベーションの条件付き処理” (347 ページ)を参照してください。

MERGE ステートメント、MODIFY ステートメント、UPDATE ステートメント
: 詳細については、“MERGE Statement” (*SAS Statements: Reference*)、“MODIFY Statement” (*SAS Statements: Reference*)および“UPDATE Statement” (*SAS Statements: Reference*)を参照してください。

データセットのマージ
: 詳細については、19 章, “SAS データセットのマージ” (287 ページ)を参照してください。

データセットの変更
: 詳細については、21 章, “SAS データセットの変更” (333 ページ)、および 22 章, “複数の SAS データセットのオブザベーションの条件付き処理” (347 ページ)を参照してください。

データセットの更新
: 詳細については、20 章, “SAS データセットの更新” (315 ページ)を参照してください。

17 章

SAS データセットの連結

SAS データセットの連結について **253**
目的 253
前提条件 254
SET ステートメントを使用したデータセットの連結 **254**
SET ステートメントについて 254
SET ステートメントの使用: 最も単純な場合 254
データセットに異なる変数が含まれる場合の SET ステートメントの使用 257
変数の属性が異なる場合の SET ステートメントの使用 258
APPEND プロシジャを使用したデータセットの連結 **271**
APPEND プロシジャについて 271
APPEND プロシジャの使用: 最も単純な場合 271
データセットに異なる変数が含まれる場合の APPEND プロシジャの使用 272
変数の属性が異なる場合の APPEND プロシジャの使用 274
SET ステートメントまたは APPEND プロシジャの選択 **275**
要約 **276**
ステートメント 276
プロシジャ 276
詳細情報 **276**

SAS データセットの連結について

目的

連結とは、複数の SAS データセットを結合して、1 つのデータセットを作成する処理です。新しいデータセットに含まれるオブザベーションの数は、元のデータセットに含まれるオブザベーションの数の合計になります。

SAS データセットは次のいずれかの方法で連結できます。

- DATA ステップの SET ステートメント
- APPEND プロシジャ

連結するデータセットに含まれる変数が同じで、なおかつ各変数の属性がすべてのデータセットで同じ場合、SET ステートメントと PROC APPEND の結果は同じになります。その他の場合は、結果が異なります。このセクションでは、この両方の方法とその相違点を学習し、どちらを使用するか判断できるようにします。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 6 章, “SAS データセットから作成する” (91 ページ)
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)
- 8 章, “数値変数の操作” (119 ページ)
- 9 章, “文字変数の操作” (131 ページ)

SET ステートメントを使用したデータセットの連結

SET ステートメントについて

SET ステートメントは、1 つ以上の SAS データセットからオブザベーションを読み取り、それを使用して新しいデータセットを作成します。

データセットを連結する SET ステートメントの形式は次のとおりです。

SET *SAS-data-sets*;

SAS-data-sets は、連結する 2 つ以上の SAS データセットを示します。新しいデータセットでは、SET ステートメントで最初に名前を指定したデータセットのオブザベーションが最初に表示されます。最初のデータセットに続けて 2 番目のデータセットのオブザベーションが表示されます。以降も同様に表示されます。リストには任意の数のデータセットを含めることができます。

SET ステートメントの使用: 最も単純な場合

単純な状況では、連結するデータセットに同じ変数(同じ名前の変数)が含まれます。さらに、各変数の種類、長さ、入力形式、出力形式、ラベルがすべてのデータセットで一致します。この場合、最初のデータセットのオブザベーションがすべて新しいデータセットにコピーされます。次に 2 番目のデータセットのオブザベーションがすべて新しいデータセットにコピーされます。以降も同様にコピーされます。各オブザベーションは元の正確なコピーです。

次の例では、SAS を使用して別々の 6 部署の人事記録を管理している企業が、すべての人事記録の結合を決定したとします。営業とカスタマサポートの 2 部署では、データが同じ形式で格納されています。両データセットの各オブザベーションには次の変数の値が含まれています。

EmployeeID
: 従業員の ID 番号を含む文字変数を示します。

Name
: 姓、カンマ、名の形式で従業員の名前を含む文字変数を示します。

HireDate
: 従業員が雇用された日付を含む数値変数を示します。この変数の出力形式は DATE9.です。

Salary
: 従業員の米ドルでの年間給与を含む数値変数を示します。

HomePhone
　従業員の自宅電話番号を含む文字変数を示します。

次のプログラムでは、SALES データセットと CUSTOMER_SUPPORT データセットが作成されます。

```
data sales;
   input EmployeeID $ 1-9 Name $ 11-29 @30 HireDate date9.
         Salary HomePhone $;
   format HireDate date9.;
   datalines;
429685482 Martin, Virginia   09aug2002 45000 493-0824
244967839 Singleton, MaryAnn 24apr2004 34000 929-2623
996740216 Leighton, Maurice  16dec2001 57000 933-6908
675443925 Freuler, Carl      15feb2010 54500 493-3993
845729308 Cage, Merce        19oct2009 64000 286-0519
;
run;

proc print data=sales;
   title 'Sales Department Employees';
run;

data customer_support;
   input EmployeeID $ 1-9 Name $ 11-29 @30 HireDate date9.
         Salary HomePhone $;
   format HireDate date9.;
   datalines;
324987451 Sayre, Jay         15nov2005 66000 933-2998
596771321 Tolson, Andrew     18mar2000 54000 929-4800
477562122 Jensen, Helga      01feb2004 70300 286-2816
894724859 Kulenic, Marie     24jun2004 54800 493-1472
988427431 Zweerink, Anna     07jul2011 59000 929-3885
;
run;

proc print data=customer_support;
   title 'Customer Support Department Employees';
run;
```

次の出力は、両方の DATA ステップの結果を示しています。

図 17.1 *SALES データセット*

Sales Department Employees

Obs	EmployeeID	Name	HireDate	Salary	HomePhone
1	429685482	Martin, Virginia	09AUG2002	45000	493-0824
2	244967839	Singleton, MaryAnn	24APR2004	34000	929-2623
3	996740216	Leighton, Maurice	16DEC2001	57000	933-6908
4	675443925	Freuler, Carl	15FEB2010	54500	493-3993
5	845729308	Cage, Merce	19OCT2009	64000	286-0519

図 17.2 CUSTOMER_SUPPORT データセット

Customer Support Department Employees

Obs	EmployeeID	Name	HireDate	Salary	HomePhone
1	324987451	Sayre, Jay	15NOV2005	66000	933-2998
2	596771321	Tolson, Andrew	18MAR2000	54000	929-4800
3	477562122	Jensen, Helga	01FEB2004	70300	286-2816
4	894724859	Kulenic, Marie	24JUN2004	54800	493-1472
5	988427431	Zweerink, Anna	07JUL2011	59000	929-3885

2 つのデータセットを連結するには、SET ステートメントでリストします。結果の DEPT1_2 データセットを表示するには PRINT プロシジャを使用します。

```
data dept1_2;
   set sales customer_support;
run;

proc print data=dept1_2;
   title 'Employees in Sales and Customer Support Departments';
run;
```

次の出力には、新しい DEPT1_2 データセットが表示されています。データセットには、SALES のすべてのオブザベーションに続いて、CUSTOMER_SUPPORT のすべてのオブザベーションが含まれます。

図 17.3 連結された DEPT1_2 データセット

Employees in Sales and Customer Support Departments

Obs	EmployeeID	Name	HireDate	Salary	HomePhone
1	429685482	Martin, Virginia	09AUG2002	45000	493-0824
2	244967839	Singleton, MaryAnn	24APR2004	34000	929-2623
3	996740216	Leighton, Maurice	16DEC2001	57000	933-6908
4	675443925	Freuler, Carl	15FEB2010	54500	493-3993
5	845729308	Cage, Merce	19OCT2009	64000	286-0519
6	324987451	Sayre, Jay	15NOV2005	66000	933-2998
7	596771321	Tolson, Andrew	18MAR2000	54000	929-4800
8	477562122	Jensen, Helga	01FEB2004	70300	286-2816
9	894724859	Kulenic, Marie	24JUN2004	54800	493-1472
10	988427431	Zweerink, Anna	07JUL2011	59000	929-3885

データセットに異なる変数が含まれる場合の SET ステートメントの使用

前の例では 2 つのデータセットに同じ変数が含まれ、各変数が両データセットで同じように定義されています。しかし、すべての変数が SET ステートメントで指定したデータセットに共通しているわけではない場合、データセットの連結が必要になることがあります。この場合、新しいデータセットの各オブザベーションに、SET ステートメントで指定した SAS データセットの変数がすべて含まれます。

このセクションの例では、SECURITY データセット、ならびにこのデータセットの SALES データセットおよび CUSTOMER_SUPPORT データセットへの連結を示します。すべての変数が 3 つのデータセットに共通しているわけではありません。保安部の人事記録には、変数 HomePhone が含まれていません。変数 Gender は SALES データセットにも CUSTOMER_SUPPORT データセットにも出現しません。

次のプログラムは、SECURITY データセットを作成します。

```
data security;
   input EmployeeID $ 1-9 Name $ 11-29 Gender $ 30
         @32 HireDate date9. Salary;
   format HireDate date9.;
   datalines;
744289612 Saparilas, Theresa F 09may2005 45000
824904032 Brosnihan, Dylan   M 04jan2009 49000
242779184 Chao, Daeyong      M 28sep2004 48500
544382887 Slifkin, Leah      F 24jul2011 54000
933476520 Perry, Marguerite  F 19apr2010 49500
;
run;

proc print data=security;
   title 'Security Department Employees';
run;
```

次の出力には、SECURITY データセットが表示されています。

図 17.4 SECURITY データセット

Security Department Employees

Obs	EmployeeID	Name	Gender	HireDate	Salary
1	744289612	Saparilas, Theresa	F	09MAY2005	45000
2	824904032	Brosnihan, Dylan	M	04JAN2009	49000
3	242779184	Chao, Daeyong	M	28SEP2004	48500
4	544382887	Slifkin, Leah	F	24JUL2011	54000
5	933476520	Perry, Marguerite	F	19APR2010	49500

次のプログラムでは、SALES、CUSTOMER_SUPPORT、および SECURITY データセットが連結され、新しいデータセット DEPT1_3 が作成されます。

```
data dept1_3;
   set sales customer_support security;
run;
```

```
proc print data=dept1_3;
   title 'Employees in Sales, Customer Support,';
   title2 'and Security Departments';
run;
```

次の出力には、連結された DEPT1_3 データセットが表示されています。

図 17.5 *連結された DEPT1_3 データセット*

Employees in Sales, Customer Support, and Security Departments

Obs	EmployeeID	Name	HireDate	Salary	HomePhone	Gender
1	429685482	Martin, Virginia	09AUG2002	45000	493-0824	
2	244967839	Singleton, MaryAnn	24APR2004	34000	929-2623	
3	996740216	Leighton, Maurice	16DEC2001	57000	933-6908	
4	675443925	Freuler, Carl	15FEB2010	54500	493-3993	
5	845729308	Cage, Merce	19OCT2009	64000	286-0519	
6	324987451	Sayre, Jay	15NOV2005	66000	933-2998	
7	596771321	Tolson, Andrew	18MAR2000	54000	929-4800	
8	477562122	Jensen, Helga	01FEB2004	70300	286-2816	
9	894724859	Kulenic, Marie	24JUN2004	54800	493-1472	
10	988427431	Zweerink, Anna	07JUL2011	59000	929-3885	
11	744289612	Saparilas, Theresa	09MAY2005	45000		F
12	824904032	Brosnihan, Dylan	04JAN2009	49000		M
13	242779184	Chao, Daeyong	28SEP2004	48500		M
14	544382887	Slifkin, Leah	24JUL2011	54000		F
15	933476520	Perry, Marguerite	19APR2010	49500		F

データセット DEPT1_3 内のすべてのオブザベーションに、変数 Gender と変数 HomePhone の両方の値があります。変数 Gender を含まないデータセット SALES および CUSTOMER_SUPPORT からのオブザベーションでは、Gender が欠損値です(変数名の下のブランクで示されます)。変数 HomePhone を含まないデータセット SECURITY からのオブザベーションでは、HomePhone が欠損値です(変数名の下のブランクで示されます)。

変数の属性が異なる場合の SET ステートメントの使用

属性について

SAS データセット内の各変数には 6 つもの属性を関連付けられます。属性のリストを次に示します。

名前
　変数を識別します。2 つ以上のデータセットが参照される場合、同じ名前の変数は同じ変数であると見なされます。

種類
　変数の種類は、文字か数値のどちらかになります。

長さ
　変数の各値を SAS データセット内に格納するときに使用するバイト数を示します。文字変数のデフォルト長は 8 バイトなので、文字変数の使用時は特に長さを考慮することが重要になります。データ値が 8 バイトより大きい場合は、LENGTH ステートメントを使用して必要な記憶域のバイト数を指定し、データが切り捨てられないようにします。

入力形式
　データ値の読み取り時に使用される命令を示します。これらの命令によって入力値の形式が指定されます。

出力形式
　データ値の書き込み時に使用される命令を示します。これらの命令によって出力値の形式が指定されます。

ラベル
　特定の変数に関連付けられた説明テキストを示します。

SET ステートメントで指定したデータセットに名前と種類が同じ変数が含まれている場合は、変更なしでデータセットを連結できます。ただし、変数の種類が異なる場合は、連結の前に、1 つ以上のデータセットを変更する必要があります。長さ、出力形式、入力形式、ラベルのいずれかが異なる場合は、続行する前に 1 つ以上のデータセットの変更が必要になる可能性があります。

変数の種類が異なる場合の SET ステートメントの使用

変数が、SET ステートメントで指定した 1 つのデータセットでは文字変数、別のデータセットでは数値変数として定義されている場合、エラーメッセージが発行され、データセットは連結されません。

次の例では、社内の経理部が従業員 ID 番号(EmployeeID)を数値変数として扱っているのに対して、その他すべての部署では文字変数として扱っています。

次のプログラムでは、ACCOUNTING データセットが作成され、他のデータセットとともに連結されます。

```
data accounting;
   input EmployeeID 1-9 Name $ 11-29 Gender $ 30
         @32 HireDate date9. Salary;
   format HireDate date9.;
   datalines;
634875680 Gardinski, Barbara F 29may2001 59000
824576630 Robertson, Hannah  F 14mar2010 65500
744826703 Gresham, Jean      F 28apr1999 67000
824447605 Kruize, Ronald     M 23may2001 58000
988674342 Linzer, Fritz      M 23jul2007 63500
;
run;

proc print data=accounting;
   title 'Accounting Department Employees';
run;
```

次の出力には、ACCOUNTING データセットが表示されています。

図 17.6 ACCOUNTING データセット

Accounting Department Employees

Obs	EmployeeID	Name	Gender	HireDate	Salary
1	634875680	Gardinski, Barbara	F	29MAY2001	59000
2	824576630	Robertson, Hannah	F	14MAR2010	65500
3	744826703	Gresham, Jean	F	28APR1999	67000
4	824447605	Kruize, Ronald	M	23MAY2001	58000
5	988674342	Linzer, Fritz	M	23JUL2007	63500

次のプログラムでは、4 部署すべてのデータセットの連結が試行されます。

```
data dept1_4;
   set sales customer_support security accounting;
run;
```

4 部署内の変数の種類が異なるため、プログロムは失敗します。次のエラーメッセージがログに書き込まれます。

```
ERROR: Variable EmployeeID has been defined as both character
       and numeric.
```

変数の種類の変更

前の例のエラーを修正する 1 つの方法は、ACCOUNTING の変数 EmployeeID の種類を数値から文字に変更することです。従業員 ID 番号で計算を実行することはありそうにないので、EmployeeID を文字変数にできます。

次に示す方法で変数 EmployeeID の種類を変更できます。

- EmployeeID を文字変数として識別するように INPUT ステートメントを変更して、データセットを再作成する
- PUT 関数で新しい変数を作成し、データセットオプションで変数の名前変更と削除を行う

次のプログラムでは、PUT 関数とデータセットオプションを使用して、変数 EmployeeID の種類を数値から文字に変更しています。

```
data new_accounting (rename=(TempVar=EmployeeID)drop=EmployeeID);  1
   set accounting;  2
   TempVar=put(EmployeeID, 9.);  3
run;

proc datasets library=work;  4
   contents data=new_accounting;
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 RENAME=データセットオプションでは、オブザベーションが出力データセットに書き込まれる際に、変数 TempVar が EmployeeID に名前変更されます。DROP=デ

ータセットオプションは RENAME=オプションの前に適用されます。その結果、EmployeeID の変数の種類が数値から文字に変更されます。

注: この例では NEW_ACCOUNTING というデータセットが新規作成されますが、SET ステートメントでリストされているデータセットと同じ名前のデータセットを作成することもできます。これを実行する場合、ACCOUNTING データセットで EmployeeID の種類の属性が永久的に変更されます。

2 SET ステートメントでは、ACCOUNTING データセットからオブザベーションが読み取られます。

3 PUT 関数で数値が文字値に変換され、変数 EmployeeID に出力形式が適用されます。割り当てステートメントで PUT 関数の結果が変数 TempVar に割り当てられます。

4 DATASETS プロシジャでは、EmployeeID の新しい属性の種類を確認できます。

次の出力には、PROC DATASETS による一部のリストが表示されています。ここでは EmployeeID が文字変数であることに注意してください。

図 17.7 *NEW_ACCOUNTING データセットの PROC DATASETS 出力*

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
5	EmployeeID	Char	9	
2	Gender	Char	1	
3	HireDate	Num	8	DATE9.
1	Name	Char	19	
4	Salary	Num	8	

すべての変数の種類が一致するようになったので、次のプログラムを使用して 4 つのデータセットすべてを簡単に連結できます。

```
data dept1_4;
   set sales customer_support security new_accounting;
run;

proc print data=dept1_4;
   title 'Employees in Sales, Customer Support, Security,';
   title2 'and Accounting Departments';
run;
```

次の出力には、連結された DEPT1_4 データセットが表示されています。

図 17.8 連結された Dept1_4 データセット

Employees in Sales, Customer Support, Security, and Accounting Departments

Obs	EmployeeID	Name	HireDate	Salary	HomePhone	Gender
1	429685482	Martin, Virginia	09AUG2002	45000	493-0824	
2	244967839	Singleton, MaryAnn	24APR2004	34000	929-2623	
3	996740216	Leighton, Maurice	16DEC2001	57000	933-6908	
4	675443925	Freuler, Carl	15FEB2010	54500	493-3993	
5	845729308	Cage, Merce	19OCT2009	64000	286-0519	
6	324987451	Sayre, Jay	15NOV2005	66000	933-2998	
7	596771321	Tolson, Andrew	18MAR2000	54000	929-4800	
8	477562122	Jensen, Helga	01FEB2004	70300	286-2816	
9	894724859	Kulenic, Marie	24JUN2004	54800	493-1472	
10	988427431	Zweerink, Anna	07JUL2011	59000	929-3885	
11	744289612	Saparilas, Theresa	09MAY2005	45000		F
12	824904032	Brosnihan, Dylan	04JAN2009	49000		M
13	242779184	Chao, Daeyong	28SEP2004	48500		M
14	544382887	Slifkin, Leah	24JUL2011	54000		F
15	933476520	Perry, Marguerite	19APR2010	49500		F
16	634875680	Gardinski, Barbara	29MAY2001	59000		F
17	824576630	Robertson, Hannah	14MAR2010	65500		F
18	744826703	Gresham, Jean	28APR1999	67000		F
19	824447605	Kruize, Ronald	23MAY2001	58000		M
20	988674342	Linzer, Fritz	23JUL2007	63500		M

変数の出力形式、入力形式またはラベルが異なる場合の SET ステートメントの使用

SET ステートメントでデータセットを連結する場合は、次のルールによって、新しいデータセット内の変数にどの出力形式、入力形式およびラベルを関連付けるかが決定されます。

- 明示的に定義された出力形式、入力形式またはラベルは、SET ステートメントでのデータセットの位置にかかわらず、デフォルトより優先されます。
- 2 つ以上のデータセットで、同じ変数に対して異なる出力形式、入力形式またはラベルが明示的に定義されている場合、新しいデータセットの変数では、その属性を明示的に定義する SET ステートメントの最初のデータセットの属性が採用されます。

例に戻ると、SALES データセット、CUSTOMER_SUPPORT データセット、SECURITY データセット、ACCOUNTING データセットを作成した DATA ステップでは、FORMAT ステートメントを使用して変数 HireDate に DATE9.の出力形式を明示的に割り当てていることがわかります。したがって、HireDate は数値変数ですが、常に DDMMMYYYY として表示されます(例: 13DEC2000)。しかし、次の例で作成される SHIPPING データセットでは、HireDate に対して DATE11.の出力形式が使用されます。DATE11.出力形式は DD-MMM-YYYY のように表示されます(例: 13–DEC–2012)。

さらに、SALES データセット、CUSTOMER_SUPPORT データセット、SECURITY データセット、ACCOUNTING データセットには Salary に対するデフォルト出力形式が含まれていますが、SHIPPING データセットには同じ変数に対して明示的に定義された出力形式 COMMA6.が含まれています。COMMA6.出力形式では、数値変数 Salary の表示時に、カンマが適切な場所に挿入されます。

次のプログラムは、出荷部のデータセットを作成します。

```
data shipping;
   input employeeID $ 1-9 Name $ 11-29 Gender $ 30
         @32 HireDate date11.
         @42 Salary;
   format HireDate date11.
          Salary comma6.;
   datalines;
688774609 Carlton, Susan     F 28jan2012 41000
922448328 Hoffmann, Gerald   M 12oct2012 40500
544909752 DePuis, David      M 23aug2011 43500
745609821 Hahn, Kenneth      M 23aug2011 45500
634774295 Landau, Jennifer   F 30apr2012 43500
;
run;

proc print data=shipping;
   title 'Shipping Department Employees';
run;
```

次の出力には、SHIPPING データセットが表示されています。

図 17.9 *SHIPPING データセット*

Shipping Department Employees

Obs	employeeID	Name	Gender	HireDate	Salary
1	688774609	Carlton, Susan	F	28-JAN-2012	41,000
2	922448328	Hoffmann, Gerald	M	12-OCT-2012	40,500
3	544909752	DePuis, David	M	23-AUG-2011	43,500
4	745609821	Hahn, Kenneth	M	23-AUG-2011	45,500
5	634774295	Landau, Jennifer	F	30-APR-2012	43,500

次に SHIPPING を前の 4 つのデータセットと連結するとどうなるか考えて見ましょう。

```
data dept1_5;
   set sales customer_support security new_accounting shipping;
```

```
run;

proc print data=dept1_5;
   title 'Employees in Sales, Customer Support, Security,';
   title2 'Accounting, and Shipping Departments';
run;
```

次の出力は、5 つのデータセットの連結を示しています。

図 17.10 DEPT1_5 データセット 5 つのデータセットの連結

Employees in Sales, Customer Support, Security, Accounting, and Shipping Departments

Obs	EmployeeID	Name	HireDate	Salary	HomePhone	Gender
1	429685482	Martin, Virginia	09AUG2002	45,000	493-0824	
2	244967839	Singleton, MaryAnn	24APR2004	34,000	929-2623	
3	996740216	Leighton, Maurice	16DEC2001	57,000	933-6908	
4	675443925	Freuler, Carl	15FEB2010	54,500	493-3993	
5	845729308	Cage, Merce	19OCT2009	64,000	286-0519	
6	324987451	Sayre, Jay	15NOV2005	66,000	933-2998	
7	596771321	Tolson, Andrew	18MAR2000	54,000	929-4800	
8	477562122	Jensen, Helga	01FEB2004	70,300	286-2816	
9	894724859	Kulenic, Marie	24JUN2004	54,800	493-1472	
10	988427431	Zweerink, Anna	07JUL2011	59,000	929-3885	
11	744289612	Saparilas, Theresa	09MAY2005	45,000		F
12	824904032	Brosnihan, Dylan	04JAN2009	49,000		M
13	242779184	Chao, Daeyong	28SEP2004	48,500		M
14	544382887	Slifkin, Leah	24JUL2011	54,000		F
15	933476520	Perry, Marguerite	19APR2010	49,500		F
16	634875680	Gardinski, Barbara	29MAY2001	59,000		F
17	824576630	Robertson, Hannah	14MAR2010	65,500		F
18	744826703	Gresham, Jean	28APR1999	67,000		F
19	824447605	Kruize, Ronald	23MAY2001	58,000		M
20	988674342	Linzer, Fritz	23JUL2007	63,500		M
21	688774609	Carlton, Susan	28JAN2012	41,000		F
22	922448328	Hoffmann, Gerald	12OCT2012	40,500		M
23	544909752	DePuis, David	23AUG2011	43,500		M
24	745609821	Hahn, Kenneth	23AUG2011	45,500		M
25	634774295	Landau, Jennifer	30APR2012	43,500		F

この連結では、2 つの異なる出力形式を使用して明示的に定義された変数 HireDate が入力データセットに含まれています。データセットには、デフォルト出力形式と明示的な出力形式の両方がある変数 Salary も含まれています。この出力から、データセットが前述のルールに従って新規作成されていることがわかります。

- HireDate の場合、SET ステートメントで最初に指定したデータセットに定義されている出力形式(SALES の DATE9.)が使用されます。
- Salary の場合、SHIPPING データセットに定義されている明示的な出力形式(COMMA6.)が使用されます。この場合、デフォルト出力形式は使用されません。

SET ステートメントのデータセットの順序だけを逆にして類似の連結を実行した場合の相違点に注意してください。

```
data dept5_1;
   set shipping new_accounting security customer_support sales;
run;

proc print data=dept5_1;
   title 'Employees in Shipping, Accounting, Security,';
   title2 'Customer Support, and Sales Departments';
run;
```

次の出力には、DEPT5_1 データセットが表示されていますが、連結の順序は異なります。

図 17.11 DEPT5_1 データセット: 連結順序の変更

Employees in Shipping, Accounting, Security, Customer Support, and Sales Departments

Obs	employeeID	Name	Gender	HireDate	Salary	HomePhone
1	688774609	Carlton, Susan	F	28-JAN-2012	41,000	
2	922448328	Hoffmann, Gerald	M	12-OCT-2012	40,500	
3	544909752	DePuis, David	M	23-AUG-2011	43,500	
4	745609821	Hahn, Kenneth	M	23-AUG-2011	45,500	
5	634774295	Landau, Jennifer	F	30-APR-2012	43,500	
6	634875680	Gardinski, Barbara	F	29-MAY-2001	59,000	
7	824576630	Robertson, Hannah	F	14-MAR-2010	65,500	
8	744826703	Gresham, Jean	F	28-APR-1999	67,000	
9	824447605	Kruize, Ronald	M	23-MAY-2001	58,000	
10	988674342	Linzer, Fritz	M	23-JUL-2007	63,500	
11	744289612	Saparilas, Theresa	F	09-MAY-2005	45,000	
12	824904032	Brosnihan, Dylan	M	04-JAN-2009	49,000	
13	242779184	Chao, Daeyong	M	28-SEP-2004	48,500	
14	544382887	Slifkin, Leah	F	24-JUL-2011	54,000	
15	933476520	Perry, Marguerite	F	19-APR-2010	49,500	
16	324987451	Sayre, Jay		15-NOV-2005	66,000	933-2998
17	596771321	Tolson, Andrew		18-MAR-2000	54,000	929-4800
18	477562122	Jensen, Helga		01-FEB-2004	70,300	286-2816
19	894724859	Kulenic, Marie		24-JUN-2004	54,800	493-1472
20	988427431	Zweerink, Anna		07-JUL-2011	59,000	929-3885
21	429685482	Martin, Virginia		09-AUG-2002	45,000	493-0824
22	244967839	Singleton, MaryAnn		24-APR-2004	34,000	929-2623
23	996740216	Leighton, Maurice		16-DEC-2001	57,000	933-6908
24	675443925	Freuler, Carl		15-FEB-2010	54,500	493-3993
25	845729308	Cage, Merce		19-OCT-2009	64,000	286-0519

の出力と比較すると、図 17.10 (265 ページ)この例ではオブザベーションの順序が変更されただけでなく、HireDate に関して SHIPPING に指定した DATE7.出力形式が優先されています。これはそのデータセットが SET ステートメントで最初に出現するよう

になったためです。変数 Salary では COMMA6.出力形式が優先されます。これは SHIPPING が変数の出力形式を明示的に指定する唯一のデータセットであるためです。

変数の長さが異なる場合の SET ステートメントの使用

SET ステートメントを使用して、長さの異なる同じ変数を含むデータセットを連結する場合、連結の結果は変数が文字か数字かによって異なります。SET ステートメントでは、次のように変数の長さが決定されます。

- 文字変数でも数値変数でも、SET ステートメントでのデータセットの位置にかかわらず、明示的に定義された長さによってデフォルトが上書きされます。
- 2 つ以上のデータセットで、同じ数値変数に対して異なる長さが明示的に定義されている場合、新しいデータセットの変数は、SET ステートメントに最初に出現するデータセットの変数と同じ長さになります。
- データセットによって文字変数の長さが異なる場合、その違いが明示的であってもそうでなくても、新しいデータセットの変数は、SET ステートメントに最初に出現するデータセットの変数と同じ長さになります。

次のプログラムは、6 番目の部署である研究部の RESEARCH データセットを作成します。このデータセットの INPUT ステートメントでは長さ 27 の変数 Name が作成されることに注意してください。その他のデータセットでは Name の長さは 19 です。この例では、Name の長さが 19 の場合、Name の値は切り捨てられます。

```
data research;
   input EmployeeID $ 1-9 Name $ 11-37 Gender $ 38
         @40 HireDate date9. Salary;
   format HireDate date9.;
   datalines;
922854076 Schoenberg, Marguerite     F 19nov2004 60500
770434994 Addison-Hardy, Jonathon    M 23feb2011 63500
242784883 McNaughton, Elizabeth      F 24jul2001 65000
377882806 Tharrington, Catherine     F 28sep2004 60000
292450691 Frangipani, Christopher    M 12aug2008 63000
;
run;

proc print data=research;
   title 'Research Department Employees';
run;
```

次の出力には、RESEARCH データセットが表示されています。

図 17.12 *RESEARCH データセット*

Research Department Employees

Obs	EmployeeID	Name	Gender	HireDate	Salary
1	922854076	Schoenberg, Marguerite	F	19NOV2004	60500
2	770434994	Addison-Hardy, Jonathon	M	23FEB2011	63500
3	242784883	McNaughton, Elizabeth	F	24JUL2001	65000
4	377882806	Tharrington, Catherine	F	28SEP2004	60000
5	292450691	Frangipani, Christopher	M	12AUG2008	63000

6 つのデータセットをすべて連結する場合、SET ステートメントで RESEARCH を 1 番目以外の位置に指定すると、長さ 19 の Name が定義されます。

プログラムで長さ 27 の Name 変数を使用する場合は、次の 2 つの選択肢があります。

- SET ステートメントのデータセットの順序を変更
- 新しいデータセットの Name の長さを変更

最初の選択肢では、長い方のデータセット(RESEARCH)を先にリストします。

```
data dept6_1;
   set research shipping new_accounting
       security customer_support sales;
run;
```

2 番目の選択肢では、データセットを新規作成する DATA ステップに LENGTH ステートメントを含めます。数値変数の長さを変更する場合、LENGTH ステートメントは DATA ステップの任意の場所に記述できます。ただし、文字変数の長さを変更する場合、LENGTH ステートメントは SET ステートメントより先に記述する必要があります。

次のプログラムは、データセット DEPT1_6A を作成します。LENGTH ステートメントで文字変数 Name に長さ 27 が指定されていますが、SET ステートメントの最初のデータセット(SALES)では長さ 19 が割り当てられています。

```
data dept1_6a;
   length Name $ 27;
   set sales customer_support security
       new_accounting shipping research;
run;

proc print data=dept1_6a;
   title 'Employees in All Departments';
run;
```

次の出力は、すべての Name 値が完全に表示されていることを示しています。Name が DATA ステップで最初に発生する変数なので、新しいデータセットの変数順序が変更されることに注意してください。

図 17.13 *DEPT1_6A データセット: Name 変数に対する LENGTH ステートメントの使用*

Employees in All Departments

Obs	Name	EmployeeID	HireDate	Salary	HomePhone	Gender
1	Martin, Virginia	429685482	09AUG2002	45,000	493-0824	
2	Singleton, MaryAnn	244967839	24APR2004	34,000	929-2623	
3	Leighton, Maurice	996740216	16DEC2001	57,000	933-6908	
4	Freuler, Carl	675443925	15FEB2010	54,500	493-3993	
5	Cage, Merce	845729308	19OCT2009	64,000	286-0519	
6	Sayre, Jay	324987451	15NOV2005	66,000	933-2998	
7	Tolson, Andrew	596771321	18MAR2000	54,000	929-4800	
8	Jensen, Helga	477562122	01FEB2004	70,300	286-2816	
9	Kulenic, Marie	894724859	24JUN2004	54,800	493-1472	
10	Zweerink, Anna	988427431	07JUL2011	59,000	929-3885	
11	Saparilas, Theresa	744289612	09MAY2005	45,000		F
12	Brosnihan, Dylan	824904032	04JAN2009	49,000		M
13	Chao, Daeyong	242779184	28SEP2004	48,500		M
14	Slifkin, Leah	544382887	24JUL2011	54,000		F
15	Perry, Marguerite	933476520	19APR2010	49,500		F
16	Gardinski, Barbara	634875680	29MAY2001	59,000		F
17	Robertson, Hannah	824576630	14MAR2010	65,500		F
18	Gresham, Jean	744826703	28APR1999	67,000		F
19	Kruize, Ronald	824447605	23MAY2001	58,000		M
20	Linzer, Fritz	988674342	23JUL2007	63,500		M
21	Carlton, Susan	688774609	28JAN2012	41,000		F
22	Hoffmann, Gerald	922448328	12OCT2012	40,500		M
23	DePuis, David	544909752	23AUG2011	43,500		M
24	Hahn, Kenneth	745609821	23AUG2011	45,500		M
25	Landau, Jennifer	634774295	30APR2012	43,500		F
26	Schoenberg, Marguerite	922854076	19NOV2004	60,500		F
27	Addison-Hardy, Jonathon	770434994	23FEB2011	63,500		M
28	McNaughton, Elizabeth	242784883	24JUL2001	65,000		F
29	Tharrington, Catherine	377882806	28SEP2004	60,000		F
30	Frangipani, Christopher	292450691	12AUG2008	63,000		M

APPEND プロシジャを使用したデータセットの連結

APPEND プロシジャについて

APPEND プロシジャでは、一方の SAS データセットに含まれているオブザベーションが、もう一方の SAS データセットの末尾に追加されます。PROC APPEND では、最初のデータセットのオブザベーションは処理されません。2 番目のデータセットのオブザベーションが元のデータセットの末尾に直接追加されます。

APPEND プロシジャの形式は次のとおりです。

PROC APPEND BASE=*base-SAS-data-set* <DATA=*SAS-data-set-to-append*> <FORCE>;

base-SAS-data-set
オブザベーションの追加先の SAS データセット名を指定します。このデータセットが存在しない場合は作成されます。PROC APPEND の完了時には、*base-SAS-data-set* の値が現在の(最後に作成された)SAS データセットになります。

SAS-data-set-to-append
基本データセットの末尾に追加するオブザベーションを含む SAS データセットの名前を指定します。このオプションを省略すると、PROC APPEND では、最後に作成された SAS データセットのオブザベーションが基本データセットの末尾に追加されます。

FORCE
PROC APPEND で、通常はプロシジャが失敗するような状況においてファイルを強制連結します。

APPEND プロシジャの使用: 最も単純な場合

次のプログラムは、データセット CUSTOMER_SUPPORT をデータセット SALES に追加します。どちらのデータセットにも同じ変数が含まれ、各変数の属性はどちらのデータセットでも同じです。

```
proc append base=sales data=customer_support;
run;

proc print data=sales;
   title 'Employees in Sales and Customer Support Departments';
run;
```

次の出力は結果を示しています。

図 17.14 PROC APPEND による出力

Employees in Sales and Customer Support Departments

Obs	EmployeeID	Name	HireDate	Salary	HomePhone
1	429685482	Martin, Virginia	09AUG2002	45000	493-0824
2	244967839	Singleton, MaryAnn	24APR2004	34000	929-2623
3	996740216	Leighton, Maurice	16DEC2001	57000	933-6908
4	675443925	Freuler, Carl	15FEB2010	54500	493-3993
5	845729308	Cage, Merce	19OCT2009	64000	286-0519
6	324987451	Sayre, Jay	15NOV2005	66000	933-2998
7	596771321	Tolson, Andrew	18MAR2000	54000	929-4800
8	477562122	Jensen, Helga	01FEB2004	70300	286-2816
9	894724859	Kulenic, Marie	24JUN2004	54800	493-1472
10	988427431	Zweerink, Anna	07JUL2011	59000	929-3885

結果データセットは、SET ステートメントで SALES と CUSTOMER_SUPPORT を指定したことにより作成されたデータセットと同一です。(図 17.3 (256 ページ)を参照してください。)PROC APPEND によって BASE=オプションのデータセットである SALES データセットが永久的に変更されることを理解していることが重要です。現在は SALES に営業部とカスタマサポート部の両方のオブザベーションが含まれています。

データセットに異なる変数が含まれる場合の APPEND プロシジャの使用

SECURITY データセットには、SALES データセットにはない変数 Gender が含まれ、SALES データセットには存在する変数 HomePhone が欠けていることを思い出してください。PROC APPEND を使用して異なる変数を含むデータセットを連結しようとするとどうなるでしょうか。

次のプログラムを使用して SECURITY を SALES に追加しようとすると、連結は失敗します。

```
proc append base=sales data=security;
run;
```

次のメッセージがログに書き込まれます。

ログ 17.1 SAS ログ: PROC APPEND エラー

```
338  proc append base=sales data=security; 339  run; NOTE: Appending
WORK.SECURITY to WORK.SALES.WARNING: Variable Gender was not found on BASE
file.The variable will not be added to the BASE file.WARNING: Variable HomePhone
was not found on DATA file.Error: No appending done because of anomalies listed
above.Use FORCE option to append these files.NOTE: 0 observations added.NOTE:
The data set WORK.SALES has 5 observations and 5 variables.NOTE: Statements not
processed because of errors noted above.NOTE: PROCEDURE APPEND used (Total
process time): real time           0.01 seconds cpu time            0.03 seconds
NOTE: The SAS System stopped processing this step because of errors.
```

BASE=データセットにない変数が DATA=データセットに含まれている場合は、PROC APPEND で FORCE オプションを使用する必要があります。プログラムを変更して FORCE オプションを含めると、ファイルは正常に連結されます。

```
data sales;
    input EmployeeID $ 1-9 Name $ 11-29 @30 HireDate date9.
          Salary HomePhone $;
    format HireDate date9.;
    datalines;
429685482 Martin, Virginia   09aug2002 45000 493-0824
244967839 Singleton, MaryAnn 24apr2004 34000 929-2623
996740216 Leighton, Maurice  16dec2001 57000 933-6908
675443925 Freuler, Carl      15feb2010 54500 493-3993
845729308 Cage, Merce        19oct2009 64000 286-0519
;
run;

data security;
   input EmployeeID $ 1-9 Name $ 11-29 Gender $ 30
         @32 HireDate date9. Salary;
   format HireDate date9.;
   datalines;
744289612 Saparilas, Theresa F 09may2005 45000
824904032 Brosnihan, Dylan   M 04jan2009 49000
242779184 Chao, Daeyong      M 28sep2004 48500
544382887 Slifkin, Leah      F 24jul2011 54000
933476520 Perry, Marguerite  F 19apr2010 49500
;
run;

proc append base=sales data=security force;
run;

proc print data=sales;
   title 'Employees in the Sales and the Security Departments';
run;
```

次の出力は結果を示しています。

図 17.15 PROC APPEND での FORCE の使用結果

Employees in the Sales and the Security Departments

Obs	EmployeeID	Name	HireDate	Salary	HomePhone
1	429685482	Martin, Virginia	09AUG2002	45000	493-0824
2	244967839	Singleton, MaryAnn	24APR2004	34000	929-2623
3	996740216	Leighton, Maurice	16DEC2001	57000	933-6908
4	675443925	Freuler, Carl	15FEB2010	54500	493-3993
5	845729308	Cage, Merce	19OCT2009	64000	286-0519
6	744289612	Saparilas, Theresa	09MAY2005	45000	
7	824904032	Brosnihan, Dylan	04JAN2009	49000	
8	242779184	Chao, Daeyong	28SEP2004	48500	
9	544382887	Slifkin, Leah	24JUL2011	54000	
10	933476520	Perry, Marguerite	19APR2010	49500	

この出力は、PROC APPEND を使用した異なる変数を含むデータセットの連結について 2 つの重要点を示しています。

- DATA=データセットにはない変数(例: HomePhone)が BASE=データセットに含まれている場合、PROC APPEND では、データセットが連結され、DATA=データセットから取得されたオブザベーション内の該当変数に欠損値が割り当てられます。
- BASE=データセットにはない変数(例: Gender)が DATA=データセットに含まれている場合、PROC APPEND の FORCE オプションで、プロシジャによる 2 つのデータセットの連結が強制されます。ただし、その変数が BASE=データセットのディスクリプタ部にないため、プロシジャでは連結されたデータセットにその変数を含めることができません。

注: 現在の例では、各データセットに、他方にはない変数が含まれています。FORCE オプションの使用が必要になるのは、変数が DATA=データセットにはあって BASE=データセットにはない場合のみです。ただし、どちらの場合もログに警告が表示されます。

変数の属性が異なる場合の APPEND プロシジャの使用

PROC APPEND で属性の異なる変数を使用する場合、次のルールが適用されます。

- 変数の属性が BASE=データセットと DATA=データセットで異なる場合、BASE=データセットの属性が優先されます。出力形式、入力形式およびラベルが異なる場合、連結は成功します。
- DATA=データセットよりも BASE=データセットの変数長の方が長い場合、連結は成功します。

- BASE=データセットよりも DATA=データセットの変数長の方が長い場合、または同じ変数が一方のデータセットでは文字変数で他方では数値変数の場合、PROC APPEND は、FORCE オプションを指定しない限りファイルの連結に失敗します。

FORCE オプションを使用すると次の結果が出ます。

- BASE=データセットに指定した長さが優先されます。したがって、BASE=データセットに指定した長さに合わせて DATA=データセットからの値が切り捨てられます。
- BASE=データセットに指定した種類が優先されます。プロシジャでは、間違った種類の値(DATA=データセットの変数の値すべて)が欠損値に置換されます。

SET ステートメントまたは APPEND プロシジャの選択

2 つのデータセットに同じ変数が含まれ、変数の属性が同じ場合、SET ステートメントでの連結結果のファイルは、APPEND プロシジャでの連結結果のファイルと同じです。BASE=データセットが大きい場合は時に、APPEND プロシジャの方が SET ステートメントよりもずっと迅速に連結を行えます。これは APPEND プロシジャでは BASE=データセットのオブザベーションが処理されないためです。ただし、変数またはその属性がデータセット間で異なる場合、2 つの連結方法には非常に違いがあります。この場合、どの方法を使用するか決定する前に、動作の相違点を考慮する必要があります。

ファイルの連結に SET ステートメントを使用する場合と APPENND プロシジャを使用する場合の主な相違点を次の表にまとめています。

表 17.1 *SET ステートメントと APPEND プロシジャの相違点*

基準	SET ステートメント	APPEND プロシジャ
連結可能なデータセット数	任意の数のデータセットを使用します。	2 つのデータセットを使用します。
異なる変数を含むデータセットの処理	すべての変数を使用し、適切な場合には欠損値を割り当てます。	BASE=データセットの変数をすべて使用し、適切な場合には DATA=データセットのオブザベーションに欠損値を割り当てます。BASE=データセットにない変数が DATA=データセットに含まれている場合、データセットを連結するには FORCE オプションが必要です。データセットの連結時には、DATA=データセットにしかない変数は含められません。
異なる出力形式、入力形式またはラベルの処理	デフォルトではなく明示的に定義された出力形式、入力形式およびラベルを使用します。2 つ以上のデータセットで出力形式、入力形式またはラベルが明示的に定義されている場合は、SET ステートメントで最初に指定したデータセットの定義が使用されます。	BASE=データセットの出力形式、入力形式およびラベルを使用します。

基準	SET ステートメント	APPEND プロシジャ
異なる変数の長さの処理	2 つ以上のデータセット内の同じ変数で長さが異なる場合、SET ステートメントで最初に指定したデータセットの長さが使用されます。	DATA=データセットの変数長の方が長い場合は、FORCE オプションが必要です。BASE=データセットの長さに合わせて変数値が切り捨てられます。
異なる変数の種類の処理	データセットは連結されません。	データセットの連結には FORCE オプションが必要です。BASE=データセットの種類の属性を使用して、DATA=データセットのオブザベーション内の変数に欠損値を割り当てます。

要約

ステートメント

LENGTH *variables* <$> *length*;
変数の格納に使用するバイト数を指定します。

SET *SAS-data-sets*;
SAS データセットを 1 つ以上読み取り、DATA ステートメントで指定した 1 つの SAS データセットを作成します。

プロシジャ

PROC APPEND BASE=*base-SAS-data-set* <DATA=*SAS-data-set-to-append*> <FORCE>;
DATA=データセットを BASE=データセットに追加します。*Base-SAS-data-set* には、オブザベーションの追加先の SAS データセット名を指定します。このデータセットが存在しない場合は作成されます。PROC APPEND の完了時には、基本データセットが現在の(最後に作成された)SAS データセットになります。*SAS-data-set-to-append* には、基本データセットの末尾に追加するオブザベーションを含む SAS データセットの名前を指定します。このオプションを省略すると、PROC APPEND では、現在の SAS データセットのオブザベーションが基本データセットの末尾に追加されます。FORCE オプションでは、別の方法ではプロシジャが失敗するような状況において PROC APPEND でファイルを強制連結します。

詳細情報

CONTENTS ステートメント
すべての変数の名前と属性を含む、データセットについての情報が表示されます。この情報によって、データセットを連結しようとしたときに発生の可能性がある問題がすべて明らかになるので、SET ステートメントと PROC APPEND のどちらを使用

するか決定するのに役立ちます。DATASETS プロシジャでの CONTENTS ステートメントの使用の詳細については、“DATASETS” (*Base SAS Procedures Guide*)を参照してください。

END=ステートメントオプション
DATA ステップで最後のオブザベーションをいつ処理するかを決定できます。詳細については、22 章, “複数の SAS データセットのオブザベーションの条件付き処理” (347 ページ)を参照してください。

IN=データセットオプション
各データセットのオブザベーションを別々に処理できます。SET ステートメントでの IN=オプションの使用の詳細については、22 章, “複数の SAS データセットのオブザベーションの条件付き処理” (347 ページ)を参照してください。

変数属性
詳細については、“SAS Variable Attributes” (*SAS Language Reference: Concepts*)を参照してください。

18 章

SAS データセットのインタリーブ

SAS データセットのインタリーブについて **279**
目的 279
前提条件 279
BY グループ処理の概念について **280**
データセットのインタリーブ **280**
データセットのインタリーブの準備 280
インタリーブプロセスについて 282
インタリーブプロセスの使用 283
要約 **284**
ステートメント 284
詳細情報 **285**

SAS データセットのインタリーブについて

目的

インタリーブとは、並べ替えられた個々の SAS データセットを結合して 1 つの並べ替えられたデータセットにする処理です。DATA ステップで SET ステートメントと BY ステートメントを使用してデータセットをインタリーブします。新しいデータセットに含まれるオブザベーションの数は、元のデータセットに含まれるオブザベーションの数の合計になります。

このセクションでは、BY ステートメントの使用方法、データセットを並べ替えてインタリーブの準備をする方法、SET ステートメントと BY ステートメントを一緒に使用してオブザベーションをインタリーブする方法を学習します。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 4 章, “生データから作成する: 基本” (51 ページ)
- 6 章, “SAS データセットから作成する” (91 ページ)

BY グループ処理の概念について

BY ステートメントでは、データセットのインタリーブの基準にする 1 つまたは複数の変数が指定されます。インタリーブを理解するためには、BY 変数、BY 値および BY グループを理解する必要があります。

BY 変数
: BY ステートメントで指定された変数を示し、この変数を基準にしてデータが並べ替えられるか、並べ替えが必要になります。

BY 値
: BY 変数の値を示します。

BY グループ
: BY 変数の値が同じすべてのオブザベーションのセットを示します(BY 変数を 1 つだけ指定している場合)。BY ステートメント内で複数の変数を使用する場合、BY グループはそれらの変数の値の組み合わせが一意であるオブザベーションの集まりになります。インタリーブの説明では、通常、BY グループは複数のデータセットにわたります。

データセットのインタリーブ

データセットのインタリーブの準備

データセットをインタリーブする前に、SET ステートメントに伴う BY ステートメントで使用されるものと同じ 1 つまたは複数の変数順にデータを並べ替える必要があります。

たとえば、企業の研究開発部と出版部の両方で、現在進行中の各プロジェクトについての情報を含むデータセットが管理されているとします。各データセットには次の変数が含まれています。

Project
: プロジェクトを識別する一意のコードを示します。

Department
: プロジェクトにかかわる部署の名前を示します。

Manager
: 部署のマネージャの姓を示します。

StaffCount
: このプロジェクトのマネージャの下で働く人数を示します。

企業の上級管理職者は、データセットを Project 別に結合して、新しいデータセットで両部署が各プロジェクトに当てているリソースを参照できるようにしたいと考えています。データセットは両方とも、インタリーブの前に Project 順に並べ替える必要があります。

次のプログラムでは、データセット RESEARCH_DEVELOPMENT が作成、表示されます。入力データはすでに Project 順に並べ替えられていることに注意してください。

```
data research_development;
   length Department Manager $ 10;
   input Project $ Department $ Manager $ StaffCount;
```

```
   datalines;
MP971 Designing Daugherty 10
MP971 Coding Newton 8
MP971 Testing Miller 7
SL827 Designing Ramirez 8
SL827 Coding Cho 10
SL827 Testing Baker 7
WP057 Designing Hascal 11
WP057 Coding Constant 13
WP057 Testing Slivko 10
;
run;

proc print data=research_development;
   title 'Research and Development Project Staffing';
run;
```

次の出力には、RESEARCH_DEVELOPMENT データセットが表示されています。

図 18.1 RESEARCH_DEVELOPMENT データセット

Research and Development Project Staffing

Obs	Department	Manager	Project	StaffCount
1	Designing	Daugherty	MP971	10
2	Coding	Newton	MP971	8
3	Testing	Miller	MP971	7
4	Designing	Ramirez	SL827	8
5	Coding	Cho	SL827	10
6	Testing	Baker	SL827	7
7	Designing	Hascal	WP057	11
8	Coding	Constant	WP057	13
9	Testing	Slivko	WP057	10

次のプログラムでは、2 番目のデータセット PUBLICATIONS の作成、並べ替えおよび表示が行われます。出力データセットが Project 順に並べ替えられていることに注意してください。

```
data publications;
   length Department Manager $ 10;
   input Manager $ Department $ Project $ StaffCount;
   datalines;
Cook Writing WP057 5
Deakins Writing SL827 7
Franscombe Editing MP971 4
Henry Editing WP057 3
King Production SL827 5
Krysonski Production WP057 3
Lassiter Graphics SL827 3
```

```
Miedema Editing SL827 5
Morard Writing MP971 6
Posey Production MP971 4
Spackle Graphics WP057 2
;
run;

proc sort data=publications;
   by Project;
run;

proc print data=publications;
   title 'Publications Project Staffing';
run;
```

次の出力には、PUBLICATIONS データセットが表示されています。

図 18.2 PUBLICATIONS データセット

Publications Project Staffing

Obs	Department	Manager	Project	StaffCount
1	Editing	Franscombe	MP971	4
2	Writing	Morard	MP971	6
3	Production	Posey	MP971	4
4	Writing	Deakins	SL827	7
5	Production	King	SL827	5
6	Graphics	Lassiter	SL827	3
7	Editing	Miedema	SL827	5
8	Writing	Cook	WP057	5
9	Editing	Henry	WP057	3
10	Production	Krysonski	WP057	3
11	Graphics	Spackle	WP057	2

インタリーブプロセスについて

データセットをインターリーブすると、次のようにデータセットが新規作成されます。

1. SET ステートメントの実行前に、SET ステートメントで名前を指定した各データセットのディスクリプタ部が読み取られます。次に、プログラムデータベクトル(PDV)が作成されます。デフォルトでは、この PDV には、DATA ステップによって作成された任意の変数とともに、すべてのデータセットのすべての変数が格納されます。各変数は欠損値に設定されます。
2. 新しいデータセットの先頭に配置される BY グループを決定するために、SET ステートメントの各データセットに含まれている最初の BY グループが参照されます。

3. BY グループ内のオブザベーションを含む各データセットから新しいデータセットに、その BY グループ内のすべてのオブザベーションがコピーされます。データセットからのコピーは SET ステートメントでの記述と同じ順序で行われます。
4. 各データセットに含まれている次のオブザベーションが参照され、新しいデータセットの 2 番目に配置される BY グループが決定されます。
5. プログラムデータベクトル内の各変数が欠損値に設定されます。
6. すべてのオブザベーションが新しいデータセットにコピーされるまで、ステップ 3 から 5 までが繰り返されます。

インタリーブプロセスの使用

次のプログラムでは、SET ステートメントと BY ステートメントを使用して、データセット RESEARCH_DEVELOPMENT とデータセット PUBLICATIONS をインタリーブします。

```
data rnd_pubs;
   set research_development publications;
   by Project;
run;

proc print data=rnd_pubs;
   title 'Project Participation by Research and Development';
   title2 'and Publications Departments';
   title3 'Sorted by Project';
run;
```

新しいデータセット RND_PUBS には、両方のデータセットのすべてのオブザベーションが格納されています。新しいデータセットの各 BY グループには、

RESEARCH_DEVELOPMENT のオブザベーションに続けて PUBLICATIONS のオブザベーションが含まれます。

図 18.3 *2 つのデータセットのインタリーブ*

Project Participation by Research and Development and Publications Departments Sorted by Project

Obs	Department	Manager	Project	StaffCount
1	Designing	Daugherty	MP971	10
2	Coding	Newton	MP971	8
3	Testing	Miller	MP971	7
4	Editing	Franscombe	MP971	4
5	Writing	Morard	MP971	6
6	Production	Posey	MP971	4
7	Designing	Ramirez	SL827	8
8	Coding	Cho	SL827	10
9	Testing	Baker	SL827	7
10	Writing	Deakins	SL827	7
11	Production	King	SL827	5
12	Graphics	Lassiter	SL827	3
13	Editing	Miedema	SL827	5
14	Designing	Hascal	WP057	11
15	Coding	Constant	WP057	13
16	Testing	Slivko	WP057	10
17	Writing	Cook	WP057	5
18	Editing	Henry	WP057	3
19	Production	Krysonski	WP057	3
20	Graphics	Spackle	WP057	2

要約

ステートメント

SET *SAS-data-sets*;
BY *variable-list*;

SET ステートメントでは、並べ替えられた SAS データセットが複数読み取られ、並べ替えられた SAS データセットが 1 つ作成されます。*SAS-data-sets* はインタリーブ対象の SAS データセットのリストです。

BY グループ処理を実行するには、BY ステートメントを SET ステートメントと一緒に使用します。*Variable-list* には、データセットのインターリーブの基準にする 1 つ以上の変数(BY 変数)の名前が含まれます。インターリーブの前に、すべてのデータセットを同じ変数で並べ替えておく必要があります。

詳細情報

インデックス
: インタリーブの基準にする 1 つまたは複数の変数のインデックスがデータセットにある場合は、インタリーブの前に、並べ替えられていないデータセットを並べ替える必要はありません。詳細については、“Understanding SAS Indexes” (*SAS Language Reference: Concepts*)を参照してください。

SAS データセットの結合
: データセットに異なる変数、属性、種類、長さのいずれかが含まれている場合、SET ステートメントを使用した SAS データセットの結合の詳細については、“SET ステートメントを使用したデータセットの連結” (254 ページ)を参照してください。

  APPEND プロシジャでの SAS データセットの結合の詳細については、“APPEND プロシジャを使用したデータセットの連結” (271 ページ)を参照してください。

  データセットのインタリーブには連結と同じルールが適用されます。

SORT プロシジャと BY ステートメント
: 詳細については、12 章, “グループ化されたオブザベーションや並べ替えられたオブザベーションの操作” (181 ページ)を参照してください。

19 章

SAS データセットのマージ

SAS データセットのマージについて **288**
目的 288
前提条件 288

MERGE ステートメントについて **288**

1 対 1 のマージ **288**
1 対 1 のマージの定義 288
単純な 1 対 1 のマージの実行 289
同じ変数を含むデータセットの 1 対 1 のマージを実行 291

マッチマージ **294**
BY ステートメントによるマージ 294
例で使用される入力 SAS データセット 295
プログラム 297
説明 298
BY グループに複数のオブザベーションがある場合のデータセットのマッチマージ 298
変数を削除したデータセットのマッチマージ 304
IN=データセットオプションでのデータセットのマッチマージ 305
同じ変数を含むデータセットのマッチマージ 306
共通変数のないデータセットのマッチマージ 307

1 対 1 のマージまたはマッチマージの選択 **309**
マッチマージ方法の比較 309
例で使用される入力 SAS データセット 309
1 対 1 のマージの使用例 310
マッチマージの使用例 312

要約 **314**
ステートメント 314

詳細情報 **314**

SAS データセットのマージについて

目的

マージとは、複数の SAS データセットのオブザベーションを結合し、新しい SAS データセットの単一のオブザベーションにする処理です。別の方法を指定しない限り、新しいデータセットには元のデータセットすべての変数がすべて格納されます。

このセクションでは、1 対 1 のマージとマッチマージという 2 種類のマージについて学習します。1 対 1 のマージでは、BY ステートメントは使用しません。オブザベーションは入力データセット内の位置に基づいて結合されます。マッチマージでは、BY ステートメントを使用して、データセットのマージの基準となる変数の共通値に基づいて入力データセットのオブザベーションを結合します。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 4 章, “生データから作成する: 基本” (51 ページ)
- 6 章, “SAS データセットから作成する” (91 ページ)

MERGE ステートメントについて

DATA ステップで MERGE ステートメントを使用してデータセットをマージします。このセクションでは MERGE ステートメントを次の形式で使用します。

MERGE *SAS-data-set-list*;

BY *variable-list*;

SAS-data-set-list
マージする 2 つ以上の SAS データセットの名前を示します。リストには任意の数のデータセットを含めることができます。

variable-list
データセットのマージの基準とする 1 つ以上の変数を示します。BY ステートメントを使用する場合は、マージする前に、データセットを同じ BY 変数で並べ替えておく必要があります。

1 対 1 のマージ

1 対 1 のマージの定義

BY ステートメントなしで MERGE ステートメントを使用すると、MERGE ステートメントに指定したすべてのデータセットの最初のオブザベーションが結合され、新しいデータセットの最初のオブザベーションになります。同様に、すべてのデータセットの 2 番目

のオブザベーションが結合され、新しいデータセットの 2 番目のオブザベーションになります。以降も同様に処理されます。1 対 1 のマージでは、新しいデータセットに格納されるオブザベーションの数は、MERGE ステートメントに指定したデータセットのうち、最も大きなデータセットに含まれているオブザベーションの数と同じになります。

単純な 1 対 1 のマージの実行

例で使用される入力 SAS データセット

大学の演劇クラスの講師が各学生との面談のスケジュールを立てる必要があるとします。1 番目のデータセット CLASS には次の変数が含まれています。

Name
学生の名前を示します。

Year
学年を示します(first、second、third、fourth)。

Major
学生の専門分野を示します。1 年生と 2 年生は勉強する専攻科目をまだ選択していないため、この値が常に欠損しています。

次のプログラムでは、CLASS データセットが作成、表示されます。

```
data class;
   input Name $ 1-25 Year $ 26-34 Major $ 36-50;
   datalines;
Abbott, Jennifer          first
Carter, Tom               third     Theater
Mendoza, Elissa           fourth    Mathematics
Tucker, Rachel            first
Uhl, Roland               second
Wacenske, Maurice         third     Theater
;
run;

proc print data=class;
   title 'Acting Class Roster';
run;
```

次の出力には、CLASS データセットが表示されています。

図 19.1 演劇クラスのデータセット

Acting Class Roster

Obs	Name	Year	Major
1	Abbott, Jennifer	first	
2	Carter, Tom	third	Theater
3	Mendoza, Elissa	fourth	Mathematics
4	Tucker, Rachel	first	
5	Uhl, Roland	second	
6	Wacenske, Maurice	third	Theater

2 番目のデータセットには、講師が面談をスケジュールした日付と時間、および面談を行う部屋のリストが含まれています。次のプログラムでは、TIME_SLOT データセットが作成、表示されます。日付の出力形式と入力形式の使用法に注意してください。

```
data time_slot;
   input Date date9.  @12 Time $ @19 Room $;
   format date date9.;
   datalines;
14sep2012  10:00  103
14sep2012  10:30  103
14sep2012  11:00  207
15sep2012  10:00  105
15sep2012  10:30  105
17sep2012  11:00  207
;
run;

proc print data=time_slot;
   title 'Dates, Times, and Locations of Conferences';
run;
```

次の出力には、TIME_SLOT データセットが表示されています。

図 19.2 *TIME_SLOT データセット*

Dates, Times, and Locations of Conferences

Obs	Date	Time	Room
1	14SEP2012	10:00	103
2	14SEP2012	10:30	103
3	14SEP2012	11:00	207
4	15SEP2012	10:00	105
5	15SEP2012	10:30	105
6	17SEP2012	11:00	207

プログラム

次のプログラムでは、これらのデータセットの 1 対 1 のマージが実行され、クラスの各学生に面談の時間帯が割り当てられます。

```
data schedule;
   merge class time_slot;
run;

proc print data=schedule;
   title 'Student Conference Assignments';
run;
```

次の出力には、SCHEDULE データセットが表示されています。

図 19.3 *1 対 1 のマージ: 面談スケジュールのデータセット*

Student Conference Assignments

Obs	Name	Year	Major	Date	Time	Room
1	Abbott, Jennifer	first		14SEP2012	10:00	103
2	Carter, Tom	third	Theater	14SEP2012	10:30	103
3	Mendoza, Elissa	fourth	Mathematics	14SEP2012	11:00	207
4	Tucker, Rachel	first		15SEP2012	10:00	105
5	Uhl, Roland	second		15SEP2012	10:30	105
6	Wacenske, Maurice	third	Theater	17SEP2012	11:00	207

説明

前述の出力(1 対 1 のマージの SCHEDULE データセット)は、新しいデータセットで、CLASS の最初のオブザベーションと TIME_SLOT の最初のオブザベーションが結合されることを示しています。その後、CLASS の 2 番目のオブザベーションと TIME_SLOT の 2 番目のオブザベーションが結合され、以降も同様に処理されます。

同じ変数を含むデータセットの 1 対 1 のマージを実行

例で使用される入力 SAS データセット

前の例は、1 対 1 のマージの最も単純な場合を示しています。データセットには同数のオブザベーションが含まれ、すべての変数の名前が一意で、新しいデータセットで両方のデータセットの変数をすべて保持する必要がありました。この例では、同じ名前が付いた変数を含むデータセットをマージします。また、この例の 2 番目のデータセットには、オブザベーションが、最初のデータセットよりも 1 つ多く含まれています。各データセットには、別々の演劇クラスについてのデータが含まれています。

講師は CLASS データセットに加えて CLASS2 データセットも使用します。これには CLASS と同じ変数が含まれていますが、オブザベーションが 1 つ多いです。次のプログラムでは、CLASS2 データセットが作成、表示されます。

```
data class2;
   input Name $ 1-25 Year $ 26-34 Major $ 36-50;
   datalines;
Hitchcock-Tyler, Erin    second
Keil, Deborah            third     Theater
Nacewicz, Chester        third     Theater
Norgaard, Rolf           second
Prism, Lindsay           fourth    Anthropology
Singh, Rajiv             second
Wittich, Stefan          third     Physics
;
run;

proc print data=class2;
   title 'Acting Class Roster';
```

```
   title2 '(second section)';
run;
```

次の出力には、CLASS2 データセットが表示されています。

図 19.4 *CLASS2 データセット*

Acting Class Roster
(second section)

Obs	Name	Year	Major
1	Hitchcock-Tyler, Erin	second	
2	Keil, Deborah	third	Theater
3	Nacewicz, Chester	third	Theater
4	Norgaard, Rolf	second	
5	Prism, Lindsay	fourth	Anthropology
6	Singh, Rajiv	second	
7	Wittich, Stefan	third	Physics

プログラム

講師が、1 クラスの面談をスケジュールするかわりに、各クラスから 1 人ずつ、2 人 1 組の学生に対する演技演習をスケジュールするとします。講師は、各クラスから 1 人ずつの学生の名前、ならびに演習の日付、時間および場所が各オブザベーションに含まれるデータセットを作成する必要があります。変数 Year と変数 Major は新しいデータセットでは不要です。

この新しいデータセットを作成するには、データセット CLASS、CLASS2 および TIME_SLOT をマージします。Year と Major は新しいデータセットでは不要なので、DROP=データセットオプションを使用して削除します。データセット CLASS とデータセット CLASS2 の両方に変数 Name が含まれていますが、Name の値がそれぞれのデータセットで異なることに注意してください。両方の値セットを保持するには、RENAME=データセットオプションを使用して、どちらかのデータセット内の変数を名前変更する必要があります。

次のプログラムでは、DROP および RENAME データセットオプションを使用して 3 つのデータセットがマージされます。

```
data exercise;
   merge class (drop=Year Major)
   class2 (drop=Year Major rename=(Name=Name2))
   time_slot;
run;

proc print data=exercise;
   title 'Acting Class Exercise Schedule';
run;
```

次の出力には、マージされたデータセットが表示されています。

図 19.5 3 つのデータセットのマージ

Acting Class Exercise Schedule

Obs	Name	Name2	Date	Time	Room
1	Abbott, Jennifer	Hitchcock-Tyler, Erin	14SEP2012	10:00	103
2	Carter, Tom	Keil, Deborah	14SEP2012	10:30	103
3	Mendoza, Elissa	Nacewicz, Chester	14SEP2012	11:00	207
4	Tucker, Rachel	Norgaard, Rolf	15SEP2012	10:00	105
5	Uhl, Roland	Prism, Lindsay	15SEP2012	10:30	105
6	Wacenske, Maurice	Singh, Rajiv	17SEP2012	11:00	207
7		Wittich, Stefan	.		

説明

次の手順でデータセットがマージされます。

1. DATA ステップの実行前に、MERGE ステートメントで指定した各データセットのディスクリプタ部が読み込まれます。次に、新しいデータセットのプログラムデータベクトルが作成されます。デフォルトでは、このプログラムデータベクトルには、すべてのデータセットのすべての変数が含まれます。これには DATA ステップで作成される変数も含まれます。ただし、この場合は、DROP=データセットオプションによって、変数 Year と変数 Major がプログラムデータベクトルから除外されます。RENAME=データセットオプションでは、変数 Name2 がプログラムデータベクトルに追加されます。したがって、プログラムデータベクトルには、変数 Name、Name2、Date、Time、Room が格納されます。
2. 次の図に示すとおり、プログラムデータベクトル内の各変数が欠損値に設定されます。

   図 19.6 データセットからの読み込み前のプログラム データベクトル

Name	Name2	Date	Time	Room
		.		

3. 次の図に示すとおり、次に、各データセットの最初のオブザベーションが読み込まれ、プログラムデータベクトルにコピーされます(データセットの読み込みは、MERGE ステートメントでの指定と同じ順序で行われます)。

図 19.7 各データセットからの読み込み後のプログラムデータベクトル

Name	Name2	Date	Time	Room
Abbott, Jennifer		.		

Name	Name2	Date	Time	Room
Abbott, Jennifer	Hitchcock-Tyler, Erin	.		

Name	Name2	Date	Time	Room
Abbott, Jennifer	Hitchcock-Tyler, Erin	14SEP2000	10:00	103

4. 最後のデータセットの最初のオブザベーションが処理され、DATA ステップ内の他のステートメントがすべて実行されると、プログラムデータベクトルのコンテンツが新しいデータセットに書き込まれます。DATA ステップがデータセットの終わりを越えて読み込もうとすると、プログラムデータベクトルでそのデータセットのすべての変数が欠損値に設定されます。

   この動作によって 2 つの重要な結果がもたらされます。

   - ある変数が複数のデータセットに存在する場合、最後に読み込まれるデータセットからの値が欠損値であっても、新しいデータセットに値が書き込まれます。異なるデータセットの同じような名前の変数の値をすべて保持する場合は、RENAME=データセットオプションで 1 つ以上の変数の名前変更を行い、各変数の名前が一意になるようにする必要があります。
   - あるデータセットのすべてのオブザベーションが処理された後、プログラムデータベクトルおよび新しいデータセットの以後すべてのオブザベーションで、そのデータセット固有の変数には、欠損値が格納されます。そのため、次の図に示すとおり、新しいデータセットの最後のオブザベーションのプログラムデータベクトルでは、Name2 以外のすべての変数に欠損値が格納されます。

   図 19.8 最後のオブザベーションのプログラムデータベクトル

Name	Name2	Date	Time	Room
	Wittich, Stefan	.		

5. すべてのデータセットからすべてのオブザベーションがコピーされるまで、オブザベーションのマージが続けられます。

マッチマージ

BY ステートメントによるマージ

BY ステートメントによるマージを実行することで、指定する BY 変数の値に従ってオブザベーションをマッチできます。マッチマージを実行する前に、すべてのデータセットをマージに使用する変数で並べ替えておく必要があります。

マッチマージを理解するためには、次に示す 3 つの主要な概念を理解している必要があります。

BY *variable*
BY ステートメントで指定される変数を示します。

BY *value*
BY 変数の値を示します。

BY *group*
BY 変数の値が同じすべてのオブザベーションのセットを示します(BY 変数が 1 つのみの場合)。BY ステートメントで複数の変数を使用する場合、BY グループはそれらの変数の値の組み合わせが一意であるオブザベーションのセットになります。マッチマージの説明では、通常、BY グループは複数のデータセットにわたって存在します。

例で使用される入力 SAS データセット

小さな劇場専属の劇団 Little Theater のディレクターが、劇団のレコードを、2 つの SAS データセット COMPANY と FINANCE に保持しているとします。

表 19.1 *COMPANY データセットと FINANCE データセットの変数*

データセット	変数	説明
COMPANY	Name	俳優の名前
	Age	俳優の年齢
	Gender	俳優の性別
FINANCE	Name	俳優の名前
	IdNumber	俳優の従業員 ID 番号
	Salary	俳優の年棒

次のプログラムでは、COMPANY および FINANCE データセットが作成、並べ替え、および表示されます。

```
data company;
   input Name $ 1-25 Age 27-28 Gender $ 30;
   datalines;
Vincent, Martina          34 F
Phillipon, Marie-Odile    28 F
Gunter, Thomas            27 M
Harbinger, Nicholas       36 M
Benito, Gisela            32 F
Rudelich, Herbert         39 M
Sirignano, Emily          12 F
Morrison, Michael         32 M
;
run;

proc sort data=company;
```

```
   by Name;
run;

data finance;
   input IdNumber $ 1-11 Name $ 13-37 Salary;
   datalines;
074-53-9892 Vincent, Martina         35000
776-84-5391 Phillipon, Marie-Odile   29750
929-75-0218 Gunter, Thomas           27500
446-93-2122 Harbinger, Nicholas      33900
228-88-9649 Benito, Gisela           28000
029-46-9261 Rudelich, Herbert        35000
442-21-8075 Sirignano, Emily         5000
;
run;

proc sort data=finance;
   by Name;
run;

proc print data=company;
   title 'Little Theater Company Roster';
run;

proc print data=finance;
   title 'Little Theater Employee Information';
run;
```

次の出力には、COMPANY および FINANCE データセットが表示されています。FINANCE データセットには、Michael Morrison のオブザベーションが含まれていないことに注意してください。

図 19.9 *COMPANY データセット*

Little Theater Company Roster

Obs	Name	Age	Gender
1	Benito, Gisela	32	F
2	Gunter, Thomas	27	M
3	Harbinger, Nicholas	36	M
4	Morrison, Michael	32	M
5	Phillipon, Marie-Odile	28	F
6	Rudelich, Herbert	39	M
7	Sirignano, Emily	12	F
8	Vincent, Martina	34	F

図 19.10 FINANCE データセット

Little Theater Employee Information

Obs	IdNumber	Name	Salary
1	228-88-9649	Benito, Gisela	28000
2	929-75-0218	Gunter, Thomas	27500
3	446-93-2122	Harbinger, Nicholas	33900
4	776-84-5391	Phillipon, Marie-Odile	29750
5	029-46-9261	Rudelich, Herbert	35000
6	442-21-8075	Sirignano, Emily	5000
7	074-53-9892	Vincent, Martina	35000

プログラム

2 つの別個のデータセットを管理する必要をなくすため、ディレクターは両方のデータセットからの各俳優のレコードを、すべての変数を含む 1 つの新しいデータセットにマージしようとしています。両方のデータセットに共通する変数は Name です。そのため、Name が BY 変数として適しています。

いずれのデータセットもすでに Name で並べ替え済みであるため、さらに並べ替えを行う必要はありません。次のプログラムでは、2 つのデータセットが Name でマージされます。

```
data employee_info;
   merge company finance;
   by name;
run;

proc print data=employee_info;
   title 'Little Theater Employee Information';
   title2 '(including personal and financial information)';
run;
```

次の出力には、マージされた EMPLOYEE_INFO データセットが表示されています。

図 19.11 マッチマージ: EMPLOYEE_INFO データセット

Little Theater Employee Information
(including personal and financial information)

Obs	Name	Age	Gender	IdNumber	Salary
1	Benito, Gisela	32	F	228-88-9649	28000
2	Gunter, Thomas	27	M	929-75-0218	27500
3	Harbinger, Nicholas	36	M	446-93-2122	33900
4	Morrison, Michael	32	M		.
5	Phillipon, Marie-Odile	28	F	776-84-5391	29750
6	Rudelich, Herbert	39	M	029-46-9261	35000
7	Sirignano, Emily	12	F	442-21-8075	5000
8	Vincent, Martina	34	F	074-53-9892	35000

説明

新しいデータセットには、劇団の俳優ごとに 1 つのオブザベーションが含まれています。各オブザベーションには、両方のデータセットからのすべての変数が格納されています。4 番目のオブザベーションに特に注意してください。データセット FINANCE には、Michael Morrison のオブザベーションがありません。この場合、FINANCE に固有の変数(IdNumber と Salary)の値が欠損しています。

BY グループに複数のオブザベーションがある場合のデータセットのマッチマージ

例で使用される入力 SAS データセット

Little Theater には、季節ごとの劇の配役の割り当てを追跡する、3 番目のデータセット REPERTORY があります。REPERTORY には、次に示す 3 つの変数があります。

Play
　レパートリーにある劇の 1 つの名前を示します。

Role
　劇の役の名前を示します。

IdNumber
　Role を演じている俳優の従業員 ID 番号を示します。

次のプログラムでは、REPERTORY データセットが作成、表示されます。

```
data repertory;
   input Play $ 1-23 Role $ 25-48 IdNumber $ 50-60;
   datalines;
No Exit                 Estelle                 074-53-9892
No Exit                 Inez                    776-84-5391
```

```
No Exit                 Valet                   929-75-0218
No Exit                 Garcin                  446-93-2122
Happy Days              Winnie                  074-53-9892
Happy Days              Willie                  446-93-2122
The Glass Menagerie     Amanda Wingfield        228-88-9649
The Glass Menagerie     Laura Wingfield         776-84-5391
The Glass Menagerie     Tom Wingfield           929-75-0218
The Glass Menagerie     Jim O'Connor            029-46-9261
The Dear Departed       Mrs. Slater             228-88-9649
The Dear Departed       Mrs. Jordan             074-53-9892
The Dear Departed       Henry Slater            029-46-9261
The Dear Departed       Ben Jordan              446-93-2122
The Dear Departed       Victoria Slater         442-21-8075
The Dear Departed       Abel Merryweather       929-75-0218
;
run;

proc print data=repertory;
   title 'Little Theater Season Casting Assignments';
run;
```

次の出力には、REPERTORY データセットが表示されています。

図 19.12　REPERTORY データセット

Little Theater Season Casting Assignments

Obs	Play	Role	IdNumber
1	No Exit	Estelle	074-53-9892
2	No Exit	Inez	776-84-5391
3	No Exit	Valet	929-75-0218
4	No Exit	Garcin	446-93-2122
5	Happy Days	Winnie	074-53-9892
6	Happy Days	Willie	446-93-2122
7	The Glass Menagerie	Amanda Wingfield	228-88-9649
8	The Glass Menagerie	Laura Wingfield	776-84-5391
9	The Glass Menagerie	Tom Wingfield	929-75-0218
10	The Glass Menagerie	Jim O'Connor	029-46-9261
11	The Dear Departed	Mrs. Slater	228-88-9649
12	The Dear Departed	Mrs. Jordan	074-53-9892
13	The Dear Departed	Henry Slater	029-46-9261
14	The Dear Departed	Ben Jordan	446-93-2122
15	The Dear Departed	Victoria Slater	442-21-8075
16	The Dear Departed	Abel Merryweather	929-75-0218

予備配役期間の機密性を保持するために、このデータセットでは、俳優が従業員 ID 番号で示されています。ただし、配役が最終決定されるとマネージャは各従業員 ID 番号を俳優の名前で置換します。もちろん、従業員 ID 番号のかわりに各俳優の名前を生データに入力して、データセットを再作成することも可能です。ただし、すべての俳優の名前と従業員 ID 番号がすでに格納されている FINANCE データセットを利用する方がより効率的です。

データセットがマージされると、データセットに俳優の名前が追加されます。当然ながら、データセットをマージする前に、IdNumber で並べ替える必要があります。

```
proc sort data=finance;
   by IdNumber;
run;

proc sort data=repertory;
   by IdNumber;
run;

proc print data=finance;
   title 'Little Theater Employee Information';
   title2 '(sorted by employee ID number)';
run;

proc print data=repertory;
   title 'Little Theater Season Casting Assignments';
   title2 '(sorted by employee ID number)';
run;
```

次の出力は、IdNumber で並べ替えられた FINANCE データセットと REPERTORY データセットを示しています。

図 19.13 *IdNumber で並べ替えられた FINANCE データセット*

Little Theater Employee Information
(sorted by employee ID number)

Obs	IdNumber	Name	Salary
1	029-46-9261	Rudelich, Herbert	35000
2	074-53-9892	Vincent, Martina	35000
3	228-88-9649	Benito, Gisela	28000
4	442-21-8075	Sirignano, Emily	5000
5	446-93-2122	Harbinger, Nicholas	33900
6	776-84-5391	Phillipon, Marie-Odile	29750
7	929-75-0218	Gunter, Thomas	27500

図 19.14 IdNumber で並べ替えられた REPERTORY データセット

Little Theater Season Casting Assignments
(sorted by employee ID number)

Obs	Play	Role	IdNumber
1	The Glass Menagerie	Jim O'Connor	029-46-9261
2	The Dear Departed	Henry Slater	029-46-9261
3	No Exit	Estelle	074-53-9892
4	Happy Days	Winnie	074-53-9892
5	The Dear Departed	Mrs. Jordan	074-53-9892
6	The Glass Menagerie	Amanda Wingfield	228-88-9649
7	The Dear Departed	Mrs. Slater	228-88-9649
8	The Dear Departed	Victoria Slater	442-21-8075
9	No Exit	Garcin	446-93-2122
10	Happy Days	Willie	446-93-2122
11	The Dear Departed	Ben Jordan	446-93-2122
12	No Exit	Inez	776-84-5391
13	The Glass Menagerie	Laura Wingfield	776-84-5391
14	No Exit	Valet	929-75-0218
15	The Glass Menagerie	Tom Wingfield	929-75-0218
16	The Dear Departed	Abel Merryweather	929-75-0218

これら 2 つのデータセットには、7 つの BY グループが含まれています。つまり、23 オブザベーション内に、BY 変数 IdNumber の 7 つの異なる値があります。最初の BY グループの IdNumber の値は 029-46-9261 です。この BY グループに、FINANCE は 1 つのオブザベーションがあり、REPERTORY は 2 つのオブザベーションがあります。最後の BY グループの IdNumber の値は 929-75-0218 です。この BY グループに、FINANCE は 1 つのオブザベーション、REPERTORY は 3 つのオブザベーションがあります。

プログラム

次のプログラムでは、データセット FINANCE とデータセット REPERTORY がマージされます。また、1 つのデータセットの BY グループに、もう一方のデータセットの同じ BY グループより多くのオブザベーションが含まれている場合の結果も示されます。

結果のデータセットには、両方のデータセットからのすべての変数が含まれています。

```
data repertory_name;
   merge finance repertory;
   by IdNumber;
run;

proc print data=repertory_name;
   title 'Little Theater Season Casting Assignments';
```

```
    title2 'with employee financial information';
run;
```

次の出力には、マージされたデータセットが表示されています。

図 19.15 BY グループに複数のオブザベーションがある場合のマッチマージ

Little Theater Season Casting Assignments with employee financial information

Obs	IdNumber	Name	Salary	Play	Role
1	029-46-9261	Rudelich, Herbert	35000	The Glass Menagerie	Jim O'Connor
2	029-46-9261	Rudelich, Herbert	35000	The Dear Departed	Henry Slater
3	074-53-9892	Vincent, Martina	35000	No Exit	Estelle
4	074-53-9892	Vincent, Martina	35000	Happy Days	Winnie
5	074-53-9892	Vincent, Martina	35000	The Dear Departed	Mrs. Jordan
6	228-88-9649	Benito, Gisela	28000	The Glass Menagerie	Amanda Wingfield
7	228-88-9649	Benito, Gisela	28000	The Dear Departed	Mrs. Slater
8	442-21-8075	Sirignano, Emily	5000	The Dear Departed	Victoria Slater
9	446-93-2122	Harbinger, Nicholas	33900	No Exit	Garcin
10	446-93-2122	Harbinger, Nicholas	33900	Happy Days	Willie
11	446-93-2122	Harbinger, Nicholas	33900	The Dear Departed	Ben Jordan
12	776-84-5391	Phillipon, Marie-Odile	29750	No Exit	Inez
13	776-84-5391	Phillipon, Marie-Odile	29750	The Glass Menagerie	Laura Wingfield
14	929-75-0218	Gunter, Thomas	27500	No Exit	Valet
15	929-75-0218	Gunter, Thomas	27500	The Glass Menagerie	Tom Wingfield
16	929-75-0218	Gunter, Thomas	27500	The Dear Departed	Abel Merryweather

説明

新しいデータセットの最初のいくつかのオブザベーションを観察して、データセットがどのように SAS によって作成されたか考えてみましょう。

1. DATA ステップを実行する前に、SAS は 2 つのデータセットのディスクリプタ部分を読み込んで、次に示す両方のデータセットからのすべての変数を含むプログラムデータベクトルを作成します。

   - IdNumber、Name および Salary (FINANCE より)。
   - Play と Role (REPERTORY より)。

   IdNumber は FINANCE データセットに含まれているため、すでにプログラムデータベクトルに存在します。次の図に示すように、SAS によりすべての変数の値が欠損値として設定されます。

図 19.16 データセットからの読み込み前のプログラムデータベクトル

IdNumber	Name	Salary	Play	Role
		.		

2. 各データセットに含まれている最初の BY グループが参照され、最初に配置される BY グループが決定されます。この場合、最初の BY グループ(IdNumber の値が 029-46-9261 であるオブザベーション)は、両方のデータセットで同じです。
3. 次の図に示すように、FINANCE からの最初のオブザベーションが読み込まれて、プログラムデータベクトルにコピーされます。

図 19.17 FINANCE データセットからの読み込み後のプログラムデータベクトル

IdNumber	Name	Salary	Play	Role
029-46-9261	Rudelich, Herbert	35000		

4. 次の図に示すように、REPERTORY からの最初のオブザベーションが読み込まれて、プログラムデータベクトルにコピーされます。BY グループにデータセットのオブザベーションが含まれていない場合、そのデータセットに固有の変数については、プログラムデータベクトルに欠損値が格納されます。

図 19.18 REPERTORY データセットからの読み込み後のプログラムデータベクトル

IdNumber	Name	Salary	Play	Role
029-46-9261	Rudelich, Herbert	35000	The Glass Menagerie	Jim O'Connor

5. 新しいデータセットにオブザベーションが書き込まれ、プログラムデータベクトルに値が保持されます。(プログラムデータベクトルに DATA ステップによって作成された変数が含まれている場合、新しいデータセットへの書き込みの後、それらが欠損値に設定されます。)
6. 各データセットに含まれている、BY グループの 2 番目のオブザベーションが検索されます。REPERTORY にはありますが、FINANCE にはありません。MERGE ステートメントにより、REPERTORY から BY グループの 2 番目のオブザベーションが読み込まれます。FINANCE には、その BY グループのオブザベーションが 1 つのみ存在するため、ステートメントでは、プログラムデータベクトルに保持されている Name の値(**Rudelich** , **Herbert**)と Salary の値(**35000**)が、新しいデータセットの 2 番目のオブザベーションに使用されます。次の図にこの動作を示します。

図 19.19 BY グループの 2 番目のオブザベーションのプログラムデータベクトル

IdNumber	Name	Salary	Play	Role
029-46-9261	Rudelich, Herbert	35000	The Dear Departed	Henry Slater

7. 新しいデータセットにオブザベーションが書き込まれます。いずれのデータセットにも、この BY グループのオブザベーションはこれ以上存在しません。そのため、最後の図に示すとおり、プログラムデータベクトルのすべての値が欠損値に設定され、次の BY グループの処理が開始されます。両方のデータセットのすべてのオブザベーションが終了するまで、オブザベーションの処理は続行されます。

図 19.20 新しい BY グループの前のプログラムデータベクトル

IdNumber	Name	Salary	Play	Role
		.		

変数を削除したデータセットのマッチマージ

配役が最終決定されたため、ディレクターは配役リストをポストするが、給与や従業員 ID の情報を含めないようにするとします。次のプログラムに示すとおり、新しいデータセットの作成時に DROP=データセットオプションを使用すると、Salary と IdNumber を削除できます。

```
data newrep (drop=IdNumber);
   merge finance (drop=Salary) repertory;
   by IdNumber;
run;

proc print data=newrep;
   title 'Final Little Theater Season Casting Assignments';
run;
```

注: 2 つの DROP=データセットオプションの配置の違いは重要です。DATA ステートメントの IdNumber の削除は、その変数が MERGE ステートメントと BY ステートメント(変数が必須)で使用可能ですが、新しいデータセットには追加されないことを意味します。MERGE ステートメントの Salary の削除は、MERGE ステートメントではこの変数の読み込みすら行われないため、Salary がプログラムステートメントで使用できないことを意味します。変数 Salary は処理に必要ないため、最初から PDV へ読み込まない方がより効率的です。

次の出力には、IdNumber 変数と Salary 変数のないマージされたデータセットが表示されています。

図 19.21　変数を削除したデータセットのマッチマージ

Final Little Theater Season Casting Assignments

Obs	Name	Play	Role
1	Rudelich, Herbert	The Glass Menagerie	Jim O'Connor
2	Rudelich, Herbert	The Dear Departed	Henry Slater
3	Vincent, Martina	No Exit	Estelle
4	Vincent, Martina	Happy Days	Winnie
5	Vincent, Martina	The Dear Departed	Mrs. Jordan
6	Benito, Gisela	The Glass Menagerie	Amanda Wingfield
7	Benito, Gisela	The Dear Departed	Mrs. Slater
8	Sirignano, Emily	The Dear Departed	Victoria Slater
9	Harbinger, Nicholas	No Exit	Garcin
10	Harbinger, Nicholas	Happy Days	Willie
11	Harbinger, Nicholas	The Dear Departed	Ben Jordan
12	Phillipon, Marie-Odile	No Exit	Inez
13	Phillipon, Marie-Odile	The Glass Menagerie	Laura Wingfield
14	Gunter, Thomas	No Exit	Valet
15	Gunter, Thomas	The Glass Menagerie	Tom Wingfield
16	Gunter, Thomas	The Dear Departed	Abel Merryweather

IN=データセットオプションでのデータセットのマッチマージ

IN=データセットオプションを使用して、オブザベーションが 1 つ以上のデータセットに出現するかどうか確認した後で、データセットをマージできます。給料が$28,000 を超える劇団員のみを含める HIGH_SALARY というデータセットを作成するとします。その給料が高い俳優の役の全リストを参照する必要があります。REPERTORY データセットと HIGH_SALARY データセットをマージすると、このリストを作成できます。IN=データセットオプションを使用して、給料が高い俳優が演じた役のみを含めます。すなわち、俳優が HIGH_SALARY データセットにも存在する場合のみ、マージされたデータセットに役を含める必要があります。

まず、HIGH_SALARY データセットを作成します。

```
data high_salary;
  set finance;
  if (salary > 28000);
run;
```

データセットを IdNumber 順に並べ替えて、マージします。IN=データセットオプションを使用して、HIGH_SALARY データセット内の俳優が演じた役のみを含めます。HIGH_PAID_ROLES のリストを出力して、結果データセットを参照します。

```
proc sort data=high_salary;
  by IdNumber;
run;

proc sort data=repertory;
  by IdNumber;
run;

data high_paid_roles(drop=IdNumber);
  merge high_salary(in=in_high_sal) repertory;
  by IdNumber;
  if in_high_sal then output;
run;

proc print data=high_paid_roles;
title 'Roles of Highest Paid Performers';
run;
```

次の出力は、給料が高い俳優の役のリストを示しています。リストには 10 オブザベーションが含まれています。これは REPERTORY データセットの全 16 オブザベーションのリストよりも短いです(図 19.14 (301 ページ)を参照)。

図 19.22 *HIGH_SALARY の俳優の役*

Roles of Highest Paid Performers

Obs	Name	Salary	Play	Role
1	Rudelich, Herbert	35000	The Glass Menagerie	Jim O'Connor
2	Rudelich, Herbert	35000	The Dear Departed	Henry Slater
3	Vincent, Martina	35000	No Exit	Estelle
4	Vincent, Martina	35000	Happy Days	Winnie
5	Vincent, Martina	35000	The Dear Departed	Mrs. Jordan
6	Harbinger, Nicholas	33900	No Exit	Garcin
7	Harbinger, Nicholas	33900	Happy Days	Willie
8	Harbinger, Nicholas	33900	The Dear Departed	Ben Jordan
9	Phillipon, Marie-Odile	29750	No Exit	Inez
10	Phillipon, Marie-Odile	29750	The Glass Menagerie	Laura Wingfield

同じ変数を含むデータセットのマッチマージ

1 対 1 のマージを実行する場合と同様に、RENAME=データセットオプションを使用して、同じ変数(同じ名前を持つ変数)を含む複数のデータセットをマッチマージできます(“同じ変数を含むデータセットの 1 対 1 のマージを実行” (291 ページ)を参照)。

RENAME=オプションを使用しない場合、変数が複数のデータセットに存在すると、最後に読み込まれるデータセットのその変数の値が新しいデータセットに書き込まれる値になります。

共通変数のないデータセットのマッチマージ

MERGE ステートメントでは任意の数のデータセット名を指定できます。ただし、複数のデータセットをマッチマージしようとしている場合、それらのデータセットすべてに共通変数があり、その変数で並べ替えられている必要があります。データセットに共通変数がない場合、マッチマージしようとしているデータセットと共通の変数がある別のデータセットを使用して、それらをマージできます。

たとえば、マッチマージの例で使用されるデータセットについて考えてみます。次の表は、データセットの名前とそれぞれのデータセットの変数の名前を示しています。

表 19.2　マッチマージの例で使用されるデータセットと変数

データセット	変数
COMPANY	Name、Age、Gender
FINANCE	Name、IdNumber、Salary
REPERTORY	Play、Role、IdNumber

これらのデータセットは、共通変数を共有していません。ただし、COMPANY と FINANCE は、変数 Name を共有しています。同様に、FINANCE と REPERTORY は変数 IdNumber を共有しています。そのため、次のプログラムに示すように、2 つの個別の DATA ステップを使用して、これらのデータセットを 1 つにマージできます。通常どおり、データセットを適切な BY 変数で並べ替えておく必要があります。(REPERTORY はすでに IdNumber で並べ替えられています。)

```
   /* Sort FINANCE and COMPANY by Name */
 proc sort data=finance;
   by Name;
run;

proc sort data=company;
   by Name;
run;

   /* Merge COMPANY and FINANCE into a */
   /* temporary data set.              */
data temp;
   merge company finance;
   by Name;
run;

proc sort data=temp;
   by IdNumber;
run;

   /* Merge the temporary data set with REPERTORY */
data all;
   merge temp repertory;
   by IdNumber;
run;
```

```
proc print data=all;
   title 'Little Theater Complete Casting Information';
run;
```

3 つのデータセットをマージするために、このプログラムは次のタスクを実行します。

- FINANCE と COMPANY を Name で並べ替え
- COMPANY と FINANCE を一時データセット TEMP にマージ
- TEMP を IdNumber で並べ替え
- TEMP と REPERTORY を IdNumber でマージ

次の出力には結果のデータセット ALL が表示されています。

図 19.23 *共通変数のないデータセットのマッチマージ*

Little Theater Complete Casting Information

Obs	Name	Age	Gender	IdNumber	Salary	Play	Role
1	Morrison, Michael	32	M		.		
2	Rudelich, Herbert	39	M	029-46-9261	35000	The Glass Menagerie	Jim O'Connor
3	Rudelich, Herbert	39	M	029-46-9261	35000	The Dear Departed	Henry Slater
4	Vincent, Martina	34	F	074-53-9892	35000	No Exit	Estelle
5	Vincent, Martina	34	F	074-53-9892	35000	Happy Days	Winnie
6	Vincent, Martina	34	F	074-53-9892	35000	The Dear Departed	Mrs. Jordan
7	Benito, Gisela	32	F	228-88-9649	28000	The Glass Menagerie	Amanda Wingfield
8	Benito, Gisela	32	F	228-88-9649	28000	The Dear Departed	Mrs. Slater
9	Sirignano, Emily	12	F	442-21-8075	5000	The Dear Departed	Victoria Slater
10	Harbinger, Nicholas	36	M	446-93-2122	33900	No Exit	Garcin
11	Harbinger, Nicholas	36	M	446-93-2122	33900	Happy Days	Willie
12	Harbinger, Nicholas	36	M	446-93-2122	33900	The Dear Departed	Ben Jordan
13	Phillipon, Marie-Odile	28	F	776-84-5391	29750	No Exit	Inez
14	Phillipon, Marie-Odile	28	F	776-84-5391	29750	The Glass Menagerie	Laura Wingfield
15	Gunter, Thomas	27	M	929-75-0218	27500	No Exit	Valet
16	Gunter, Thomas	27	M	929-75-0218	27500	The Glass Menagerie	Tom Wingfield
17	Gunter, Thomas	27	M	929-75-0218	27500	The Dear Departed	Abel Merryweather

1 対 1 のマージまたはマッチマージの選択

マッチマージ方法の比較

それぞれのデータセットから 1 つのオブザベーションを結合しますが、オブザベーションの照合が重要でない場合、1 対 1 のマージを使用します。たとえば、学生の名前、学年および専攻を含むオブザベーションと、面談の日付、時間および場所を含むオブザベーションをマージする場合、どの学生がどの時間帯を取得するかは重要ではありません。そのため、1 対 1 のマージが適しています。

特定のオブザベーションをマージする必要がある場合は、マッチマージを使用します。たとえば、2 つの異なるデータセットからの従業員情報をマージする場合、同じ従業員についてのオブザベーションをマージすることが重要です。そのため、マッチマージを使用する必要があります。

特定の変数でマージする際、データが適切な形式で配置されているために 1 対 1 のマージで処理可能な場合もあります。次の例では、使用するデータが適切な順番で並べ替えられていることが明らかであるために、オブザベーションのマッチに 1 対 1 のマージが使用可能なケースを示します。ただし、後続の例で示すようにそのような状況で 1 対 1 のマージを使用するのにはリスクが伴います。

例で使用される入力 SAS データセット

COMPANY2 データセットで考えてみます。このデータセットの各オブザベーションは、FINANCE の Name の値が同じであるオブザベーションに対応しています。次のプログラムでは、COMPANY2 データセットが作成、表示されます。また、FINANCE データセットも比較のために表示されます。

```
data company2;
   input name $ 1-25 age 27-28 gender $ 30;
   datalines;
Benito, Gisela            32 F
Gunter, Thomas            27 M
Harbinger, Nicholas       36 M
Phillipon, Marie-Odile    28 F
Rudelich, Herbert         39 M
Sirignano, Emily          12 F
Vincent, Martina          34 F
;
run;

proc print data=company2;
   title 'Little Theater Company Roster';
run;

proc print data=finance;
   title 'Little Theater Employee Information';
run;
```

次の出力はその 2 つのデータセットを示しています。

図 19.24 COMPANY2 データセット

Little Theater Company Roster

Obs	name	age	gender
1	Benito, Gisela	32	F
2	Gunter, Thomas	27	M
3	Harbinger, Nicholas	36	M
4	Phillipon, Marie-Odile	28	F
5	Rudelich, Herbert	39	M
6	Sirignano, Emily	12	F
7	Vincent, Martina	34	F

図 19.25 FINANCE データセット

Little Theater Employee Information

Obs	IdNumber	Name	Salary
1	228-88-9649	Benito, Gisela	28000
2	929-75-0218	Gunter, Thomas	27500
3	446-93-2122	Harbinger, Nicholas	33900
4	776-84-5391	Phillipon, Marie-Odile	29750
5	029-46-9261	Rudelich, Herbert	35000
6	442-21-8075	Sirignano, Emily	5000
7	074-53-9892	Vincent, Martina	35000

1 対 1 のマージの使用例

次のプログラムでは、両方のデータセットが NAME で並べ替えられているため、また 1 つのデータセットの各オブザベーションに他方のデータセットの対応するオブザベーションが存在するため、1 対 1 のマージの結果と、Name でのマージの結果とが同じであることが示されています。

```
   /* One-to-one merge */
data one_to_one;
   merge company2 finance;
run;

proc print data=one_to_one;
   title 'Using a One-to-One Merge to Combine';
```

```
   title2 'COMPANY2 and FINANCE';
run;

   /* Match-merge */
data match;
   merge company2 finance;
   by name;
run;

proc print data=match;
   title 'Using a Match-Merge to Combine';
   title2 'COMPANY2 and FINANCE';
run;
```

次の出力はその 2 つのマージの結果を示しています。2 つは同一であることがわかります。

図 19.26 オブザベーションが対応する場合に 1 対 1 のマージによりオブザベーションを結合

Using a One-to-One Merge to Combine
COMPANY2 and FINANCE

Obs	name	age	gender	IdNumber	Salary
1	Benito, Gisela	32	F	228-88-9649	28000
2	Gunter, Thomas	27	M	929-75-0218	27500
3	Harbinger, Nicholas	36	M	446-93-2122	33900
4	Phillipon, Marie-Odile	28	F	776-84-5391	29750
5	Rudelich, Herbert	39	M	029-46-9261	35000
6	Sirignano, Emily	12	F	442-21-8075	5000
7	Vincent, Martina	34	F	074-53-9892	35000

図 19.27 オブザベーションが対応する場合にマッチマージによりオブザベーションを結合

Using a Match-Merge to Combine COMPANY2 and FINANCE

Obs	name	age	gender	IdNumber	Salary
1	Benito, Gisela	32	F	228-88-9649	28000
2	Gunter, Thomas	27	M	929-75-0218	27500
3	Harbinger, Nicholas	36	M	446-93-2122	33900
4	Phillipon, Marie-Odile	28	F	776-84-5391	29750
5	Rudelich, Herbert	39	M	029-46-9261	35000
6	Sirignano, Emily	12	F	442-21-8075	5000
7	Vincent, Martina	34	F	074-53-9892	35000

結果のデータセットが同一でも、1 つのデータセットからの特定のオブザベーションと別のデータセットからの特定のオブザベーションをマージすることが重要である場合、1 対 1 のマージの使用は推奨されません。

マッチマージの使用例

前の例では、両方のデータセットに Name の同じ値があり、値が同じ順に配置されていることが容易に判断できます。ただし、データセットに数百のオブザベーションがある場合、すべての値がマッチしているか判断するのは困難です。オブザベーションがマッチしない場合、深刻な問題が発生します。次の例では、オブザベーションのマッチに 1 対 1 のマージを使用すべきでない理由を説明します。

Michael Morrison のオブザベーションを含む、元のデータセット COMPANY について考えてみます(図 19.9 (296 ページ)を参照)。FINANCE データセットには、対応するオブザベーションがありません。この違いに気付かず、次のプログラムを使用して FINANCE で 1 対 1 のマージを実行しようとすると、いくつかの問題が発生します。

```
data badmerge;
   merge company finance;
run;

proc print data=badmerge;
   title 'Using a One-to-One Merge Instead of a Match-Merge';
run;
```

次の出力は潜在的な問題を示しています。

図 19.28 1 対 1 のマージ: 各データセット内にあるオブザベーションの数が異なる場合

Using a One-to-One Merge Instead of a Match-Merge

Obs	Name	Age	Gender	IdNumber	Salary
1	Benito, Gisela	32	F	228-88-9649	28000
2	Gunter, Thomas	27	M	929-75-0218	27500
3	Harbinger, Nicholas	36	M	446-93-2122	33900
4	Phillipon, Marie-Odile	32	M	776-84-5391	29750
5	Rudelich, Herbert	28	F	029-46-9261	35000
6	Sirignano, Emily	39	M	442-21-8075	5000
7	Vincent, Martina	12	F	074-53-9892	35000
8	Vincent, Martina	34	F		.

最初の 3 つのオブザベーションは正しくマージされています。ただし、データセット FINANCE には、Michael Morrison のオブザベーションがありません。1 対 1 のマージでは、それぞれのデータセットからのオブザベーションの部分のマッチは試みられません。オブザベーションは、単純に MERGE ステートメントで指定されるデータセット内での位置に基づいて結合されます。そのため、BADMERGE の 4 番目のオブザベーションでは、COMPANY の 4 番目のオブザベーション(Michael の名前、年齢、性別)と、FINANCE の 4 番目のオブザベーション(Marie-Odile の名前、従業員 ID 番号、給与)とが結合されます。オブザベーションの結合により、Marie-Odile の名前で Michael の名前が上書きされます。このオブザベーションの新しいデータセットへの書き込みの完了後、それぞれのデータセットの次のオブザベーションが処理されます。それらのオブザベーションも同様に一致しません。

この種類の不一致が、MERGE ステートメントでサイズがより小さいデータセット FINANCE の最後のオブザベーションが処理される、7 番目のオブザベーションまで続きます。新しいデータセットへの 7 番目のオブザベーションへの書き込みの完了後、DATA ステップの次の反復が開始されます。FINANCE のすべてのオブザベーションはすでに読み込みまれているため、FINANCE データセットからの変数の値はプログラムデータベクトルで欠損値に設定されます。次に、COMPANY からの Name、Age、Gender の値が読み込まれ、プログラムデータベクトルのコンテンツが新しいデータセットに書き込まれます。そのため、最後のオブザベーションの NAME の値は前のオブザベーションと同じで、IdNumber と Salary には欠損値が含まれます。

これらの欠損値と Name の値の重複のため、オブザベーションが意図したとおりにマージされていないことが疑われます。しかしながら、Michael Morrison のオブザベーションが、COMPANY2 に追加されるのではなく別のオブザベーションを置き換える場合は、欠損値を持つオブザベーションがなくなるため、問題の検出が難しくなります。そのため、データが整理されており 1 対 1 のマージで同じ結果が得られると思われる場合でも、マッチマージを使用できる状況では、マッチマージを使用する方が安全です。

要約

ステートメント

MERGE *SAS-data-set-list*;
BY *variable-list*;

MERGE ステートメントでは、複数の SAS データセットのオブザベーションが読み込まれ、1 つの新しい SAS データセットの 1 つのオブザベーションに結合されます。*SAS-data-set-list* はマージ対象の SAS データセットのリストです。リストには任意の数のデータセットを含めることができます。

Variable-list はデータセットのマージの基準とする 1 つ以上の変数の名前です。BY ステートメントを使用する場合は、マージする前に、データセットを同じ BY 変数で並べ替えておく必要があります。BY ステートメントを使用しない場合は、オブザベーションは元のデータセット内での位置に基づいてマージされます。

詳細情報

インデックス

MERGE ステートメントとともに指定する BY ステートメントで指定されたデータセットの変数にインデックスが設定されている場合、データセットを並べ替える必要はありません。詳細については、“Understanding SAS Indexes” (*SAS Language Reference: Concepts*)を参照してください。

SAS 日時の出力形式と入力形式

このセクションの例では、Time を文字変数として読み込み、Date を SAS 日付の入力形式を使用して読み込んでいます。Time を、特殊な SAS 時間の入力形式を使用して読み込むこともできます。SAS 日時の入力形式の詳細については、“Working with SAS Dates and Times” (*SAS Language Reference: Concepts*)を参照してください。

20 章

SAS データセットの更新

SAS データセットの更新について **315**
目的 315
前提条件 315
UPDATE ステートメントについて **316**
BY 変数の選択方法について **316**
データセットの更新 **317**
増分値での更新 **322**
更新とマージの違いについて **324**
更新とマージの一般的な比較 324
UPDATE ステートメントと MERGE ステートメントの欠損値処理方法の違い . . 326
UPDATE ステートメントと MERGE ステートメントの BY グループにおける複数オブザベーション処理方法の違い 327
欠損値の処理 **327**
要約 **331**
ステートメント 331
詳細情報 **331**

SAS データセットの更新について

目的

更新では、データセットにある変数の値を別のデータセットからの非欠損値で置き換えます。このセクションでは、次について学習します。

- マスタデータセットとトランザクションデータセットの相違点
- UPDATE ステートメントの使用
- 更新とマージの選択方法

前提条件

このセクションを使用する前に、次のセクションで説明した概念を理解している必要があります。

- 4 章, “生データから作成する: 基本” (51 ページ)
- 6 章, “SAS データセットから作成する” (91 ページ)
- 19 章, “SAS データセットのマージ” (287 ページ)

UPDATE ステートメントについて

更新のときは、2 つの SAS データセットが操作されます。元の情報を格納しているデータセットをマスタデータセットといいます。新しい情報を格納しているデータセットをトランザクションデータセットといいます。メーリングリストや在庫の管理などの多くのアプリケーションで、情報の定期的な更新が必要とされます。

DATA ステップで UPDATE ステートメントを使用して、トランザクションデータセットのオブザベーションを読み込み、マスタデータセットの対応するオブザベーション(すべての BY 変数の値が同じオブザベーション)を更新します。トランザクションデータセットにある変数のすべての非欠損値が、マスタデータセットから読み込まれた対応する値を置き換えます。SAS は、マスタデータセットやトランザクションデータセットを変更せず、DATA ステートメントで指定されたデータセットへ変更されたオブザベーションを書き込みます。

次に UPDATE ステーメントの一般的な形式を示します。

UPDATE *master-SAS-data-set transaction-SAS-data-set*;

BY *identifier-list*;

master-SAS-data-set
更新が必要な情報を含む SAS データセットを指定します。

transaction-SAS-data-set
マスタデータセットを更新するための情報を含む SAS データセットを指定します。

identifier-list
対応するオブザベーションを識別するための BY 変数のリストを指定します。

マスタデータセットに、トランザクションデータセットのオブザベーションに対応付けられないオブザベーションがある場合、DATA ステップでは、そのオブザベーションを変更せずに新しいデータセットへ書き込みます。マスタデータセットのオブザベーションに対応付けられないトランザクションデータセットのオブザベーションは、新しいオブザベーションの基礎になります。新しいオブザベーションは、新しいデータセットに書き込まれるまでは、トランザクションデータセットの他のオブザベーションによって変更される可能性があります。

BY 変数の選択方法について

1 つ以上の同一変数に基づいてマスタデータセットとトランザクションデータセットを並べ替える必要があります。使用する変数は、BY ステートメントで指定します。これらの基準に合う変数を選択してください。

- 変数の値は、マスタデータセットのオブザベーションごとに一意です。複数の BY 変数を使用する場合は、すべての BY 変数が同じ値を持つオブザベーションをマスタデータセット内で重複させないでください。
- 更新する必要のない変数。

BY ステートメントで使用できる変数の例としては、従業員や学生の識別番号、在庫番号、および在庫のオブジェクト名などがあります。

データセットを更新するときには、マスタデータセットの BY 変数の値が重複しないようにします。たとえば、NAME で更新する場合、マスタデータセットの各オブザベーションで NAME が一意の値であることが必要です。NAME と AGE で更新する場合、複数のオブザベーションで NAME または AGE のいずれかが同じ値になる可能性はありますが、両方を同じ値にはしないでください。SAS が重複を検出した場合、更新が続行されると警告を出します。すべてのトランザクションが、マスタデータセットの BY グループで最初のオブザベーションにのみ適用されます。

データセットの更新

この例では、雑誌の配信部が何万名かのメーリングリストを保守しているとします。雑誌の発刊号ごとに、読者が名前や住所を変更するときに記入するフォームがあります。リストの保守作業を簡単にするため、そのフォームでは読者に新しい情報のみを送ってもらうようにします。新しい購読者はフォーム全体を完成すれば購読を始めることができます。フォームが受信されると、データ入力オペレータはフォームの情報を生データファイルへ入力します。メーリングリストは生データファイルから 1 か月に 1 回更新されます。

メーリングリストには、購読者ごとにこれらの変数があります。

SubscriberId
: 購読の開始時に購読者へ割り当てられる一意の番号。購読者の SubscriberId は変更不可です。

Name
: 購読者の名前。姓が最初に表示され、カンマの後に名前が続きます。

StreetAddress
: 購読者の番地住所。

City
: 購読者の市町村。

StateProv
: 購読者の州。この変数は合衆国およびカナダ以外の住所には該当しません。

PostalCode
: 購読者の郵便番号(合衆国の住所では ZIP コード)。

Country
: 購読者の国。

次のプログラムが、このデータセットの最初の部分を作成し表示します。生データはすでに SubscriberId で並べ替えられています。

```
data mail_list;
   input SubscriberId 1-8 Name $ 9-27 StreetAddress $ 28-47 City $ 48-62
         StateProv $ 63-64 PostalCode $ 67-73 Country $ ;
   datalines;
1001    Ericson, Jane      111 Clancey Court   Chapel Hill    NC  27514   USA
1002    Dix, Martin        4 Shepherd St.      Vancouver      BC  V6C 3E8 Canada
1003    Gabrielli, Theresa Via Pisanelli, 25   Roma               00196   Italy
1004    Clayton, Aria      14 Bridge St.       San Francisco  CA  94124   USA
1005    Archuleta, Ruby    Box 108             Milagro        NM  87429   USA
1006    Misiewicz, Jeremy  43-C Lakeview Apts. Madison        WI  53704   USA
```

```
1007    Ahmadi, Hafez      52 Rue Marston      Paris              75019   France
1008    Jacobson, Becky    1 Lincoln St.       Tallahassee    FL  32312   USA
1009    An, Ing            95 Willow Dr.       Toronto        ON  M5J 2T3 Canada
1010    Slater, Emily      1009 Cherry St.     York           PA  17407   USA
;
run;

proc print data=mail_list;
   title 'Magazine Master Mailing List';
run;
```

次の出力はマスタ MAIL_LIST データセットを示します。

図 20.1 *雑誌のマスタメーリングリスト*

Magazine Master Mailing List

Obs	SubscriberId	Name	StreetAddress	City	StateProv	PostalCode	Country
1	1001	Ericson, Jane	111 Clancey Court	Chapel Hill	NC	27514	USA
2	1002	Dix, Martin	4 Shepherd St.	Vancouver	BC	V6C 3E8	Canada
3	1003	Gabrielli, Theresa	Via Pisanelli, 25	Roma		00196	Italy
4	1004	Clayton, Aria	14 Bridge St.	San Francisco	CA	94124	USA
5	1005	Archuleta, Ruby	Box 108	Milagro	NM	87429	USA
6	1006	Misiewicz, Jeremy	43-C Lakeview Apts.	Madison	WI	53704	USA
7	1007	Ahmadi, Hafez	52 Rue Marston	Paris		75019	France
8	1008	Jacobson, Becky	1 Lincoln St.	Tallahassee	FL	32312	USA
9	1009	An, Ing	95 Willow Dr.	Toronto	ON	M5J 2T3	Canada
10	1010	Slater, Emily	1009 Cherry St.	York	PA	17407	USA

今月は、メーリングリスト更新のために次の情報が受信されます。

- Martin Dix は名前を Martin Dix-Rosen に変更。
- Jane Ericson は郵便番号を変更。
- Jeremy Misiewicz は番地住所を移動。市町村住所、郵便番号は同じ。
- Ing An は Toronto, Ontario から Calgary, Alberta へ移動。
- Martin Dix-Rosen は名前の変更後すぐに、Vancouver, British Columbia から Seattle, Washington へ移動。
- 2 人の新しい購読者をリストに追加。SubscriberID 番号 1011 と 1012 を付与。

受信されるとすぐに、各変更は生データファイルに入力されます。どの場合も、顧客の SubscriberId および新しい情報だけが入力されます。生データファイルは次のようになります。

```
1002    Dix-Rosen, Martin
1001                                                       27516
1006                       932 Webster St.
1009                       2540 Pleasant St.   Calgary        AB  T2P 4H2
1011    Mitchell, Wayne    28 Morningside Dr.  New York       NY  10017   USA
1002                       P.O. Box 1850       Seattle        WA  98101   USA
1012    Stavros, Gloria    212 Northampton Rd. South Hadley   MA  01075   USA
```

データは、MAIL_LIST を作成した INPUT ステートメントと一致する固定カラムにあります。

最初に、生データを SAS データセットに変換し、そのデータセットを SubscriberId で並べ替える必要があります。その後で、マスタリストを更新するために使用できるようになります。

```
data mail_trans;
   infile datalines missover;
   input SubscriberId 1-8 Name $ 9-27 StreetAddress $ 28-47 City $ 48-62
         StateProv $ 63-64 PostalCode $ 67-73 Country $ 75-80;
   datalines;
1002    Dix-Rosen, Martin
1001                                                        27516
1006                      932 Webster St.
1009                      2540 Pleasant St.   Calgary       AB  T2P 4H2
1011    Mitchell, Wayne   28 Morningside Dr.  New York      NY  10017   USA
1002                      P.O. Box 1850       Seattle       WA  98101   USA
1012    Stavros, Gloria   212 Northampton Rd. South Hadley  MA  01075   USA
;
run;

proc sort data=mail_trans;
   by SubscriberId;
run;

proc print data=mail_trans;
   title 'Magazine Mailing List Changes';
   title2 '(for current month)';
run;
```

INFILE ステートメントの MISSOVER オプションに注意してください。MISSOVER オプションは、INPUT ステートメントで、値を受け取らなかった変数に対する値の検索が新しい行へ進行しないようにします。かわりに、値を受け取っていない変数のすべてを欠損と設定します。たとえば、最初のレコードが読み込まれるとき、任意の値が Country 変数に割り当てられる前にレコードの終わりが検出された場合は、Country の値を次のレコードへ検索しに行くかわりに、Country 変数には欠損値が割り当てられます。MISSOVER オプションの詳細については、“制御方法: オプション” (86 ページ)を参照してください。

次の出力は、並べ替えされた MAIL_TRANS トランザクションデータセットを示し、そこには現在の月に受けた雑誌購読者の変更がリストされています。

図 20.2 雑誌購読者のメーリングリストの変更: 並べ替え済み

Magazine Mailing List Changes
(for current month)

Obs	SubscriberId	Name	StreetAddress	City	StateProv	PostalCode	Country
1	1001					27516	
2	1002	Dix-Rosen, Martin					
3	1002		P.O. Box 1850	Seattle	WA	98101	USA
4	1006		932 Webster St.				
5	1009		2540 Pleasant St.	Calgary	AB	T2P 4H2	
6	1011	Mitchell, Wayne	28 Morningside Dr.	New York	NY	10017	USA
7	1012	Stavros, Gloria	212 Northampton Rd.	South Hadley	MA	01075	USA

これで、新しいデータは並べ替えられた SAS データセットにあり、次のプログラムがメーリングリストを更新します。

```
data mail_newlist;
   update mail_list mail_trans;
   by SubscriberId;
run;

proc print data=mail_newlist;
   title 'Magazine Mailing List';
   title2 '(updated for current month)';
run;
```

次の出力は MAIL_NEWLIST データセットを示しています。

図 20.3 更新されたメーリングリスト

Magazine Mailing List
(updated for current month)

Obs	SubscriberId	Name	StreetAddress	City	StateProv	PostalCode	Country
1	1001	Ericson, Jane	111 Clancey Court	Chapel Hill	NC	27516	USA
2	1002	Dix-Rosen, Martin	P.O. Box 1850	Seattle	WA	98101	USA
3	1003	Gabrielli, Theresa	Via Pisanelli, 25	Roma		00196	Italy
4	1004	Clayton, Aria	14 Bridge St.	San Francisco	CA	94124	USA
5	1005	Archuleta, Ruby	Box 108	Milagro	NM	87429	USA
6	1006	Misiewicz, Jeremy	932 Webster St.	Madison	WI	53704	USA
7	1007	Ahmadi, Hafez	52 Rue Marston	Paris		75019	France
8	1008	Jacobson, Becky	1 Lincoln St.	Tallahassee	FL	32312	USA
9	1009	An, Ing	2540 Pleasant St.	Calgary	AB	T2P 4H2	Canada
10	1010	Slater, Emily	1009 Cherry St.	York	PA	17407	USA
11	1011	Mitchell, Wayne	28 Morningside Dr.	New York	NY	10017	USA
12	1012	Stavros, Gloria	212 Northampton Rd.	South Hadley	MA	01075	USA

2 つの更新トランザクションがある購読者 1002(MAIL_TRANS データセットを参照)のデータを使用して、マスタデータセットのオブザベーションをトランザクションデータセットの対応するオブザベーションで更新するときに何が起こるかを示します。

1. DATA ステップの実行前に、SAS は、UPDATE ステートメントで指定された各データセットのディスクリプタ部を読み込み、デフォルトでは、すべてのデータセットの変数すべてを含んでいるプログラムデータベクトルを作成します。次の図が示すように、SAS は各変数の値を欠損と設定します。(1 つ以上の変数を除外するためには、DROP=または KEEP=データセットオプションを使用します。)

図 20.4 *DATA ステップ実行前のプログラム データベクトル*

SubscriberId	Name	Street Address	City	StateProv	PostalCode	Country
.						

2. 次に、下の図に示すように、SAS はマスタデータセットから最初のオブザベーションを読み込み、プログラムデータベクトルにそれをコピーします。

図 20.5 *マスタデータセットから最初のオブザベーションを読み込んだ後のプログラム データベクトル*

SubscriberId	Name	Street Address	City	StateProv	PostalCode	Country
1002	Dix, Martin	4 Shepherd St.	Vancouver	BC	V6C 3E8	Canada

3. 次の図で示すように、SAS は最初のトランザクションを適用し、この BY グループ(ID=1002)における最初のオブザベーションから非欠損値(Name の値)をすべてプログラムデータベクトルにコピーします。

図 20.6 *最初のトランザクション適用後のプログラム データベクトル*

SubscriberId	Name	Street Address	City	StateProv	PostalCode	Country
1002	Dix-Rosen, Martin	4 Shepherd St.	Vancouver	BC	V6C 3E8	Canada

4. このトランザクション完了後、SAS は、トランザクションデータセットの同一 BY グループにある別のオブザベーションを検索します。ID に対して同じ値を持つ 2 番目のオブザベーションを検出した場合、そのトランザクションも適用します(StreetAddress、City、StateProv、PostalCode、Country に対する新しい値)。次の図が示すように、これで、そのオブザベーションは両方のトランザクションからの新しい値を含みます。

図 20.7 *2 番目のトランザクション適用後のプログラム データベクトル*

SubscriberId	Name	Street Address	City	StateProv	Postal Code	Country
1002	Dix-Rosen, Martin	P.O. Box 1850	Seattle	WA	98101	USA

5. 2 番目のトランザクション完了後、SAS は、同一 BY グループの 3 番目のオブザベーションを検索します。そのようなオブザベーションは存在しないので、オブザベーションを現在の形式で新しいデータセットに書き込み、プログラムデータベクトルの値を欠損値に設定します。

DATA ステップが反復される間、マスタデータセットとトランザクションデータセットの終わりに達するまで、UPDATE ステートメントでオブザベーションの処理がこのように続行します。新しい購読者を記述する(このため、対応するオブザベーションがマスタデータセットにない)トランザクションデータセットの 2 つのオブザベーションは、新しいデータセットのオブザベーションとなります。

マスタデータセットに重複するオブザベーションがある場合、トランザクションデータセットの一致するオブザベーションすべてが、マスタデータセットにある重複オブザベーションのうち最初のものだけに適用されることに注意してください。

増分値での更新

アプリケーションによっては、トランザクションデータセットの新しい値でマスタデータセットの値を上書きすることでは、データセットを更新しません。かわりに、トランザクションデータセットの変数の値に基づいてその値を算術操作し、変数を更新します。

この例では、書店が毎週の売り上げおよび過去 1 年間の売り上げを追跡するために SAS を使用するとします。次のプログラムは、データセット YEAR_SALES(これには過去 1 年間の情報が含まれます)の作成、タイトルでの並べ替え、表示を行います。

```
data year_sales;
   input Title $ 1-25 Author $ 27-50 Sales;
   datalines;
The Milagro Beanfield War Nichols, John            303
The Stranger              Camus, Albert            150
Always Coming Home        LeGuin, Ursula            79
Falling through Space     Gilchrist, Ellen         128
Don Quixote               Cervantes, Miguel de      87
The Handmaid's Tale       Atwood, Margaret          64
;

proc sort data=year_sales;
   by title;
run;

proc print data=year_sales;
   title 'Bookstore Sales, Year-to-Date';
   title2 'By Title';
run;
```

次の出力は、YEAR_SALES データセットを示しています。

図 20.8 *タイトルで並べ替えた YEAR_SALES データセット*

Bookstore Sales, Year-to-Date
By Title

Obs	Title	Author	Sales
1	Always Coming Home	LeGuin, Ursula	79
2	Don Quixote	Cervantes, Miguel de	87
3	Falling through Space	Gilchrist, Ellen	128
4	The Handmaid's Tale	Atwood, Margaret	64
5	The Milagro Beanfield War	Nichols, John	303
6	The Stranger	Camus, Albert	150

毎週土曜日、前週に販売された本すべてについての情報を含む SAS データセットが作成されます。次のプログラムは、データセット WEEK_SALES(これには現在の週の情報が含まれます)の作成、タイトルでの並べ替え、表示を行います。

```
data week_sales;
   input Title $ 1-25 Author $ 27-50 Sales;
   datalines;
The Milagro Beanfield War Nichols, John            32
The Stranger              Camus, Albert            17
Always Coming Home        LeGuin, Ursula           10
Falling through Space     Gilchrist, Ellen         12
The Accidental Tourist    Tyler, Anne              15
The Handmaid's Tale       Atwood, Margaret         8
;
proc sort data=week_sales;
   by title;
run;

proc print data=week_sales;
   title 'Bookstore Sales for Current Week';
   title2 'By Title';
run;
```

次の出力は WEEK_SALES データセットを示し、そこには YEAR_SALES データセットと同じ変数が含まれますが、変数 Sales は 1 週間分のみの売り上げを表します。

図 20.9 *タイトルで並べ替えた WEEK_SALES データセット*

Bookstore Sales for Current Week
By Title

Obs	Title	Author	Sales
1	Always Coming Home	LeGuin, Ursula	10
2	Falling through Space	Gilchrist, Ellen	12
3	The Accidental Tourist	Tyler, Anne	15
4	The Handmaid's Tale	Atwood, Margaret	8
5	The Milagro Beanfield War	Nichols, John	32
6	The Stranger	Camus, Albert	17

注: トランザクションデータセットが YEAR_SALES の既存タイトルのみを更新している場合は、変数 Author を含んでいる必要はありません。ただし、この変数がそこにあることで、トランザクションデータセットを使用して、完全なオブザベーションをマスタデータセットに追加できます。

次のプログラムは、週の情報を使用して YEAR_SALES データセットを更新し、新しいデータセットを表示します。

```
data total_sales;
   drop NewSales;  1
   update year_sales week_sales (rename=(Sales=NewSales));  2
   by Title;
   sales=sum(Sales,NewSales);  3
```

```
run;

proc print data=total_sales;
   title 'Updated Year-to-Date Sales';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 新しいデータセットには必要ないので、プログラムは変数 NewSales を削除します。

2 UPDATE ステートメントの RENAME=データセットオプションは、トランザクションデータセット(WEEK_SALES)の変数 Sales の名前を NewSales に変更します。その結果、これらの値はマスタデータセット(YEAR_SALES)から読み込まれた Sales の値を置き換えません。

3 更新されたデータセット(TOTAL_SALES)にある Sales の値は、過去 1 年間の売り上げと毎週の売り上げの合計です。

次の出力は、TOTAL_SALES データセットを示しています。マスタデータセットにすでにあるタイトルの売り上げ情報の更新に加えて、UPDATE ステートメントで新しいタイトル(The Accidental Tourist)が追加されました。

図 20.10 *過去 1 年間の売り上げを週の売り上げで更新*

Updated Year-to-Date Sales

Obs	Title	Author	Sales
1	Always Coming Home	LeGuin, Ursula	89
2	Don Quixote	Cervantes, Miguel de	87
3	Falling through Space	Gilchrist, Ellen	140
4	The Accidental Tourist	Tyler, Anne	15
5	The Handmaid's Tale	Atwood, Margaret	72
6	The Milagro Beanfield War	Nichols, John	335
7	The Stranger	Camus, Albert	167

更新とマージの違いについて

更新とマージの一般的な比較

MERGE ステートメントと UPDATE ステートメントの両方とも、2 つの SAS データセットの一致するオブザベーションを処理するものです。しかしながら、2 つのステートメントは著しく異っています。2 つのプロセスを区別し、アプリケーションに対して適切なものを選ぶことが重要です。

最も明確な違いは次のとおりです。

- UPDATE ステートメントは 2 つのデータセットのみを使用します。MERGE ステートメントが使用できるデータセットの数は、メモリやディスクスペースなどのマシン依存の要因だけで制限されます。

- UPDATE ステートメントは、BY ステートメントと一緒に使う必要があります。MERGE ステートメントは、BY ステートメントが次に続かない場合は 1 対 1 のマージを実行します。
- データセットが BY グループに欠損値や複数のオブザベーションを含む場合も、2 つのステートメントでのオブザベーションの処理は異なります。

違いを示すために、SAS データセット MAIL_LIST とデータセット MAIL_TRANS の更新とマージの結果を比較します。データセット MAIL_NEWLIST を作成した例で、更新の結果はすでに記載しました。比較しやすいように、その出力をもう一度示します。

次の出力は MAIL_NEWLIST データセットを示しています。

図 20.11 更新された雑誌購読者のメーリングリスト

Magazine Mailing List
(updated for current month)

Obs	SubscriberId	Name	StreetAddress	City	StateProv	PostalCode	Country
1	1001	Ericson, Jane	111 Clancey Court	Chapel Hill	NC	27516	USA
2	1002	Dix-Rosen, Martin	P.O. Box 1850	Seattle	WA	98101	USA
3	1003	Gabrielli, Theresa	Via Pisanelli, 25	Roma		00196	Italy
4	1004	Clayton, Aria	14 Bridge St.	San Francisco	CA	94124	USA
5	1005	Archuleta, Ruby	Box 108	Milagro	NM	87429	USA
6	1006	Misiewicz, Jeremy	932 Webster St.	Madison	WI	53704	USA
7	1007	Ahmadi, Hafez	52 Rue Marston	Paris		75019	France
8	1008	Jacobson, Becky	1 Lincoln St.	Tallahassee	FL	32312	USA
9	1009	An, Ing	2540 Pleasant St.	Calgary	AB	T2P 4H2	Canada
10	1010	Slater, Emily	1009 Cherry St.	York	PA	17407	USA
11	1011	Mitchell, Wayne	28 Morningside Dr.	New York	NY	10017	USA
12	1012	Stavros, Gloria	212 Northampton Rd.	South Hadley	MA	01075	USA

一方、次のプログラムは 2 つのデータセットをマージします。

```
data mail_merged;
   merge mail_list mail_trans;
   by SubscriberId;
run;

proc print data=mail_merged;
   title 'Magazine Mailing List';
run;
```

次の出力は、MAIL_MERGED データセットを示しています。

図 20.12 雑誌購読者のメーリングリスト: マスタデータセットとトランザクションデータセットのマージ

Magazine Mailing List

Obs	SubscriberId	Name	StreetAddress	City	StateProv	PostalCode	Country
1	1001					27516	
2	1002	Dix-Rosen, Martin					
3	1002		P.O. Box 1850	Seattle	WA	98101	USA
4	1003	Gabrielli, Theresa	Via Pisanelli, 25	Roma		00196	Italy
5	1004	Clayton, Aria	14 Bridge St.	San Francisco	CA	94124	USA
6	1005	Archuleta, Ruby	Box 108	Milagro	NM	87429	USA
7	1006		932 Webster St.				
8	1007	Ahmadi, Hafez	52 Rue Marston	Paris		75019	France
9	1008	Jacobson, Becky	1 Lincoln St.	Tallahassee	FL	32312	USA
10	1009		2540 Pleasant St.	Calgary	AB	T2P 4H2	
11	1010	Slater, Emily	1009 Cherry St.	York	PA	17407	USA
12	1011	Mitchell, Wayne	28 Morningside Dr.	New York	NY	10017	USA
13	1012	Stavros, Gloria	212 Northampton Rd.	South Hadley	MA	01075	USA

MERGE ステートメントは、13 オブザベーションを含むデータセットを生成します。一方、UPDATE ステートメントは、12 オブザベーションを含むデータセットを生成します。また、データセットのマージではいくつかの欠損値が生じますが、更新では生じません。明らかに、間違ったステートメントの使用で正しくないデータを生じる場合はあります。マージしたデータセットと更新したデータセットに相違が生じるのは、2 つのステートメントで欠損値や BY グループの複数オブザベーションを処理する方法が異なるためです。

UPDATE ステートメントと MERGE ステートメントの欠損値処理方法の違い

更新時、その変数に対する値がトランザクションデータセットで欠損している場合、新しいデータセットへそのオブザベーションを書き込むとき、SAS はマスタデータセットの値を使用します。同じオブザベーションをマージするときは、SAS はプログラムデータベクトルの値を欠損値で上書きします。たとえば、次のオブザベーションがデータセット MAILING_MASTER に存在するとします。

```
1001    ERICSON, JANE      111 CLANCEY COURT   CHAPEL HILL    NC  27514
```

次の、対応するオブザベーションが MAILING_TRANS に存在します。

```
1001                                                                27516
```

更新では 2 つのオブザベーションを結合して、次のオブザベーションを作成します。

```
1001    ERICSON, JANE      111 CLANCEY COURT   CHAPEL HILL    NC  27516
```

マージでは 2 つのオブザベーションを結合して、このオブザベーションを作成します。

```
1001                                                                27516
```

UPDATE ステートメントと MERGE ステートメントの BY グループにおける複数オブザベーション処理方法の違い

BY グループのトランザクションをすべて適用するまで、SAS は更新したオブザベーションを新しいデータセットに書き込みません。データセットをマージする場合、SAS は、データセットのオブザベーションごとに新しいオブザベーションを書き込むため、BY グループのオブザベーション数が最大になります。たとえば、MAILING_MASTER のこのオブザベーションについて考えます。

```
1002    DIX, MARTIN        4 SHEPHERD ST.      NORWICH        VT  05055
```

MAILING_TRANS の対応するオブザベーションは次のとおりです。

```
1002    DIX-ROSEN, MARTIN
1002                       R.R. 2, BOX 1850    HANOVER        NH  03755
```

UPDATE ステートメントは両方のトランザクションを適用してこれらのオブザベーションを結合し、単一のオブザベーションにします。

```
1002    DIX-ROSEN, MARTIN  R.R. 2, BOX 1850    HANOVER        NH  03755
```

これに対して、MERGE ステートメントでは、まず、MAILING_MASTER のオブザベーションを、MAILING_TRANS での対応する BY グループに含まれる最初のオブザベーションとマージします。欠損している場合でも、MAILING_TRANS のオブザベーションからの変数の値がすべて使用されます。SAS はそのオブザベーションを新しいデータセットに書き込みます。

```
1002    DIX-ROSEN, MARTIN
```

次に、SAS は、各データセットの同一 BY グループにある他のオブザベーションを検索します。MAILING_TRANS の BY グループにはさらにオブザベーションがあるため、プログラムデータベクトルのすべての値が保持されます。SAS は、それらを MAILING_TRANS の BY グループにある 2 番目のオブザベーションとマージし、新しいデータセットに結果を書き込みます。

```
1002                       R.R. 2, BOX 1850    HANOVER        NH  03755
```

したがって、マージでは新しいデータセットに 2 つのオブザベーションが作成されますが、更新では 1 つ作成されるのみです。

欠損値の処理

マスタデータセットをトランザクションデータセットで更新するときにトランザクションデータセットが欠損値を含んでいる場合、UPDATE ステートメントの UPDATEMODE=オプションを使用して欠損値の処理方法を SAS に指示できます。UPDATEMODE=オプションでは、トランザクションデータセットの欠損値がマスタデータセットの既存の値を置き換えるかどうかを指定します。

UPDATE ステートメントの UPDATEMODE=オプションを使用するための構文は、次のとおりです。

UPDATE *master-SAS-data-set transaction-SAS-data-set*
<UPDATEMODE=MISSINGCHECK | NOMISSINGCHECK>;

BY *by-variable*;

UPDATEMODE=オプションの値が MISSINGCHECK の場合は、トランザクションデータセットの欠損値でマスタデータセットの値が置き換えられるのを防げます。これがデフォルトです。UPDATEMODE=オプションの値が NOMISSINGCHECK の場合は、欠

損データのチェックは実行されず、トランザクションデータセットの欠損値でマスタデータセットの値が置き換えられます。

次の例では、UPDATE ステートメントで UPDATEMODE=オプションを使用する場合の、SAS による欠損値の処理方法を示します。

次の例で、マスタデータセットを作成して並べ替えます。

```
data inventory;
   input PartNumber $ Description $ Stock @17
   ReceivedDate date9. @27 Price;
   format  ReceivedDate date9.;
   datalines;
K89R seal   34  27jul2004 245.00
M4J7 sander 98  20jun2004 45.88
LK43 filter 121 19may2005 10.99
MN21 brace 43   10aug2005 27.87
BC85 clamp 80   16aug2005 9.55
NCF3 valve 198  20mar2005 24.50
;
run;

proc sort data=inventory;
   by PartNumber;
run;

proc print data=inventory;
   title 'Master Data Set';
   title2 'Tool Warehouse Inventory';
run;
```

次の出力には、マスタ INVENTORY データセットが表示されています。

図 20.13 *マスタ INVENTORY データセット*

Master Data Set
Tool Warehouse Inventory

Obs	PartNumber	Description	Stock	ReceivedDate	Price
1	BC85	clamp	80	16AUG2005	9.55
2	K89R	seal	34	27JUL2004	245.00
3	LK43	filter	121	19MAY2005	10.99
4	M4J7	sander	98	20JUN2004	45.88
5	MN21	brace	43	10AUG2005	27.87
6	NCF3	valve	198	20MAR2005	24.50

次の例では、トランザクションデータセットを作成して並べ替えます。

```
data add_inventory;
   input PartNumber $ 1-4 Description $ 6-11 Stock 13-15 @17 Price;
   datalines;
K89R seal       245.00
```

```
M4J7 sander 121  45.88
LK43 filter 34  10.99
MN21 brace      28.87
BC85 clamp  57  11.64
NCF3 valve  121    .
;
run;

proc sort data=add_inventory;
   by PartNumber;
run;

proc print data=add_inventory;
   title 'Transaction Data Set';
   title2 'Tool Warehouse Inventory';
run;
```

次の出力には、トランザクション ADD_INVENTORY データセットが表示されています。

図 20.14 *トランザクションデータセット*

Transaction Data Set
Tool Warehouse Inventory

Obs	PartNumber	Description	Stock	Price
1	BC85	clamp	57	11.64
2	K89R	seal	.	245.00
3	LK43	filter	34	10.99
4	M4J7	sander	121	45.88
5	MN21	brace	.	28.87
6	NCF3	valve	121	.

次の例で、SAS は、UPDATE ステートメントで UPDATEMODE=オプションの NOMISSINGCHECK 値を使用します。

```
data new_inventory;
   update inventory add_inventory updatemode=nomissingcheck;
   by PartNumber;
   ReceivedDate=today();
run;

proc print data=new_inventory;
   title 'Updated Master Data Set';
   title2 'Tool Warehouse Inventory';
run;
```

次の出力は、NOMISSINGCHECK 値の使用結果を示しています。トランザクションデータセットではこれらの項目に対して STOCK に欠損値を含んでいるので、オブザベーション 2 と 5 は STOCK が欠損値になっています。トランザクションデータセットの欠損

値チェックは実行されないため、STOCK の元の値が欠損値で置き換えられます。6 番目のオブザベーションでは、PRICE の元の値が欠損値で置き換えられています。

図 20.15 *更新されたマスタデータセット: UPDATEMODE=NOMISSINGCHECK*

Updated Master Data Set
Tool Warehouse Inventory

Obs	PartNumber	Description	Stock	ReceivedDate	Price
1	BC85	clamp	57	09APR2013	11.64
2	K89R	seal	.	09APR2013	245.00
3	LK43	filter	34	09APR2013	10.99
4	M4J7	sander	121	09APR2013	45.88
5	MN21	brace	.	09APR2013	28.87
6	NCF3	valve	121	09APR2013	.

次の出力は、MISSINGCHECK 値の使用結果を示しています。更新されたマスタデータセットには欠損値が書き込まれないことに注意してください。トランザクションデータセットのオブザベーション 2、5 および 6 の欠損値は無視され、マスタデータセットの元のデータがそのまま保持されます。

図 20.16 *更新されたマスタデータセット: UPDATEMODE=MISSINGCHECK*

Updated Master Data Set
Tool Warehouse Inventory

Obs	PartNumber	Description	Stock	ReceivedDate	Price
1	BC85	clamp	57	09APR2013	11.64
2	K89R	seal	34	09APR2013	245.00
3	LK43	filter	34	09APR2013	10.99
4	M4J7	sander	121	09APR2013	45.88
5	MN21	brace	43	09APR2013	28.87
6	NCF3	valve	121	09APR2013	24.50

詳細については、“UPDATE Statement” (*SAS Statements: Reference*)を参照してください。

要約

ステートメント

UPDATE *master-SAS-data-set transaction-SAS-data-set*;
BY *identifier-list*;

UPDATE ステートメントは、1 つの SAS データセットにある変数の値を別の SAS データセットの非欠損値に置き換えます。*Master-SAS-data-set* は、更新が必要な情報を含む SAS データセットです。*Transaction-SAS-data-set* は、マスタデータセットを更新するための情報を含む SAS データセットです。

Identifier-list は、BY ステートメントで、対応するオブザベーションを識別するための BY 変数のリストです。

詳細情報

DATASETS プロシジャ

データセットを更新する場合は、更新された情報を含む新しいデータセットを作成します。通常は、古いマスタデータセットを削除し新しいものの名前を変更して、次回情報を更新するときに同じプログラムを使用できるように、PROC DATASETS を使用します。詳細については、“DATASETS プロシジャについて” (676 ページ)を参照してください。

インデックス

UPDATE ステートメントとともに指定する BY ステートメントで指定されたデータセットの変数にインデックスが設定されている場合、データセットを並べ替える必要はありません。詳細については、“Understanding SAS Indexes” (*SAS Language Reference: Concepts*)を参照してください。

MERGE ステートメント

詳細については、19 章, “SAS データセットのマージ” (287 ページ)を参照してください。

21 章

SAS データセットの変更

SAS データセットの変更について **333**
目的 333
前提条件 334
例で使用される入力 SAS データセット **334**
SAS データセットの変更: 最も単純な場合 **335**
マスタデータセットをトランザクションデータセットのオブザベーションで更新する . . **336**
MODIFY ステートメントについて 336
マスタデータセットへの新規オブザベーションの追加 337
プログラムエラーのチェック 337
プログラム 337
重複した BY 変数がファイル更新に与える影響について **340**
重複した BY 変数の DATA ステップでの処理方法 340
プログラム 341
欠損値の処理 **343**
要約 **345**
ステートメント 345
詳細情報 **346**

SAS データセットの変更について

目的

変更すると、元のマスタファイルでオブザベーションが直接変更されます。ファイルのコピーは作成しません。このセクションでは、DATA ステップで MODIFY ステートメントを使用して、次の操作を行う方法を学習します。

- データセット内の値を置き換え
- マスタデータセットにある値をトランザクションデータセットの値で置き換え
- 既存の SAS データセットへオブザベーションを追加
- 既存の SAS データセットからオブザベーションを削除

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 4 章, “生データから作成する: 基本” (51 ページ)
- 6 章, “SAS データセットから作成する” (91 ページ)
- 19 章, “SAS データセットのマージ” (287 ページ)
- 20 章, “SAS データセットの更新” (315 ページ)

例で使用される入力 SAS データセット

このセクションでは、工具販売会社が使用する在庫追跡システムからの例を見てみましょう。例では、SAS データセット INVENTORY を入力として使用します。データセットはこれらの変数を含みます。

PartNumber
　各品目を識別する一意の値を含む文字変数。

Description
　各品目のテキスト記述を含む文字変数。

InStock
　倉庫に保管されている各工具の単位数を示す値を含む数値変数。

ReceivedDate
　SAS 日付値を含む数値変数。これは現在の InStock 値の日付です。

Price
　各品目の価格を含む数値変数。

次のプログラムでは、INVENTORY_TOOL データセットが作成、表示されます。

```
data inventory_tool;
   input PartNumber $ Description $ InStock @17
         ReceivedDate date9. @27 Price;
   format ReceivedDate date9.;
   datalines;
K89R seal    34  27jul2010 245.00
M4J7 sander 98  20jun2011 45.88
LK43 filter 121 19may2011 10.99
MN21 brace 43   10aug2012 27.87
BC85 clamp 80   16aug2012 9.55
NCF3 valve 198  20mar2012 24.50
KJ66 cutter 6   18jun2010 19.77
UYN7 rod  211   09sep2010 11.55
JD03 switch 383 09jan2013 13.99
BV1E timer 26   03aug2013 34.50
;
run;

proc print data=inventory_tool;
   title 'Tool Warehouse Inventory';
run;
```

次の出力には、INVENTORY_TOOL データセットが表示されています。

図 21.1 INVENTORY_TOOL データセット

Tool Warehouse Inventory

Obs	PartNumber	Description	InStock	ReceivedDate	Price
1	K89R	seal	34	27JUL2010	245.00
2	M4J7	sander	98	20JUN2011	45.88
3	LK43	filter	121	19MAY2011	10.99
4	MN21	brace	43	10AUG2012	27.87
5	BC85	clamp	80	16AUG2012	9.55
6	NCF3	valve	198	20MAR2012	24.50
7	KJ66	cutter	6	18JUN2010	19.77
8	UYN7	rod	211	09SEP2010	11.55
9	JD03	switch	383	09JAN2013	13.99
10	BV1E	timer	26	03AUG2013	34.50

SAS データセットの変更: 最も単純な場合

MODIFY ステートメントを使用して、データセットにある特定の変数やいくつかの変数に対するすべての値を置き換えられます。この目的で MODIFY ステートメントを使用する構文は次のとおりです。

MODIFY *SAS-data-set*;

次のプログラムでは、在庫での各部品の価格が 15%値上げされます。元の INVENTORY_TOOL データセットにあるすべてのレコードの旧価格を PRICE の新価格で置き換えます。PRINT プロシジャの FORMAT ステートメントで、各品目の価格を 2 桁の 10 進数で書き込みます。

```
data inventory_tool;
   modify inventory_tool;
   price=price+(price*.15);
run;

proc print data=inventory_tool;
   title 'Tool Warehouse Inventory';
   title2 '(Price reflects 15% increase)';
   format price 8.2;
run;
```

次の出力には、変更された INVENTORY_TOOL データセットが表示されています。

図 21.2 価格を更新した INVENTORY_TOOL データセット

Tool Warehouse Inventory
(Price reflects 15% increase)

Obs	PartNumber	Description	InStock	ReceivedDate	Price
1	K89R	seal	34	27JUL2010	281.75
2	M4J7	sander	98	20JUN2011	52.76
3	LK43	filter	121	19MAY2011	12.64
4	MN21	brace	43	10AUG2012	32.05
5	BC85	clamp	80	16AUG2012	10.98
6	NCF3	valve	198	20MAR2012	28.18
7	KJ66	cutter	6	18JUN2010	22.74
8	UYN7	rod	211	09SEP2010	13.28
9	JD03	switch	383	09JAN2013	16.09
10	BV1E	timer	26	03AUG2013	39.68

マスタデータセットをトランザクションデータセットのオブザベーションで更新する

MODIFY ステートメントについて

MODIFY ステートメントは、マスタデータセットのデータをトランザクションデータセットのデータで置き換え、元のマスタデータセットを変更します。トランザクションデータセットのオブザベーションとマスタデータセットのオブザベーションを一致させるには、BY ステートメントを使用します。MODIFY ステートメントと BY ステートメントを使用する構文は次のとおりです。

MODIFY *master-SAS-data-set transaction-SAS-data-set*;

BY *by-variable*;

Master-SAS-data-set には、変更が必要な SAS データセットを指定します。*Transaction-SAS-data-set* には、マスタデータセットを更新する値を提供する SAS データセットを指定します。*By-variable* には、対応するオブザベーションを識別する 1 つ以上の変数を指定します。

MODIFY ステートメントと一緒に BY ステートメントを使用するとき、DATA ステップは動的 WHERE 処理を使用してマスタデータセットでオブザベーションを検索します。マスタデータセットとトランザクションデータセットの両方とも並べ替えは必要ありません。ただし、データセットが大きい場合には、変更前にデータを並べ替えておくと、パフォーマンスが著しく向上する可能性があります。

マスタデータセットへの新規オブザベーションの追加

MODIFY ステートメントを使用して、既存のマスタデータセットにオブザベーションを追加できます。トランザクションデータセットがマスタデータセットのオブザベーションと一致しないオブザベーションを含んでいる場合、プログラム内で明示的な OUTPUT ステートメントを使用すれば、SAS はマスタデータセットに新しいオブザベーションを書き込むことができます。明示的な OUTPUT ステートメントを指定する場合、オブザベーションを適切な場所に置き換えるためには、REPLACE ステートメントも指定する必要があります。新規オブザベーションのすべてがマスタデータセットの末尾に追加されます。

プログラムエラーのチェック

DATA ステッププログラムでは_IORC_自動変数をエラーチェックに使用することができます。_IORC_自動変数は、MODIFY ステートメントが実行しようとする各入出力操作のリターンコードを含んでいます。

_IORC_の値のテストには、SYSRC 自動呼び出しマクロによって提供されるニーモニックコードを使用するのが最良の方法です。それぞれのニーモニックコードには、特定の条件が記述されています。そのため、ニーモニックを使用すると、DATA ステッププログラムで発生した問題を簡単に検証するできるようになります。次はコードの部分的なリストです。

_DSENMR
: (MODIFY ステートメントと BY ステートメントでのみ使用される)マスタデータセットに、トランザクションデータセットのオブザベーションが存在しないことを示します。異なる BY 値を持つ連続したオブザベーションと、マスタデータセット内で一致するものが検出されない場合は、両方とも_DSENMR を返します。

_DSEMTR
: (MODIFY ステートメントと BY ステートメントでのみ使用される)マスタデータセットに、特定の BY 値を持つ複数のトランザクションデータセットのオブザベーションが存在しないことを示します。同一の BY 値を持つ連続したオブザベーションと、マスタデータセット内で一致するものが検出されない場合、最初のオブザベーションは_DSENMR を返し、以降のオブザベーションは_DSEMTR を返します。

_DSENOM
: 一致するオブザベーションが見つからなかったことを示します。

_SOK
: 該当するオブザベーションがマスタデータセットに格納されていたこと、また、MODIFY ステートメントが正常に実行されたことを示します。

プログラム

このセクションのプログラムは、トランザクションデータセットの値でマスタデータセットの値を更新します。トランザクションがマスタデータセットに存在しない場合、プログラムはそのトランザクションをマスタデータセットに追加します。

この例では、倉庫で新しい品目の出荷指示を受け取ったため、変更を反映するために INVENTORY_TOOL マスタデータセットを変更する必要があります。マスタデータセットは、在庫品目の完全なリストを含んでいます。トランザクションデータセットは、新しい在庫品目だけでなく、在庫マスタにある品目を含んでいます。

次のプログラムは、ADD_INVENTORY トランザクションデータセットを作成し、マスタデータセットを更新するための品目を格納します。PartNumber 変数は、その品目の部品番号を含み、INVENTORY_TOOL データセットの PartNumber に相当します。

Description 変数は、品目名を示します。NewStock 変数は、現在の出荷での各品目の数を含んでいます。NewPrice 変数は、品目の新しい価格を含んでいます。

プログラムは、ADD_INVENTORY トランザクションデータセットの値に従ってマスタデータセット INVENTORY_TOOL (図 21.1 (335 ページ)を参照)の更新を試みます。プログラムは、_IORC_自動変数をエラーの検出に使用します。

```
data inventory_tool;
   input PartNumber $ Description $ InStock @17
         ReceivedDate date9. @27 Price;
   format ReceivedDate date9.;
   datalines;
K89R seal   34  27jul2010 245.00
M4J7 sander 98  20jun2011 45.88
LK43 filter 121 19may2011 10.99
MN21 brace 43   10aug2012 27.87
BC85 clamp 80   16aug2012 9.55
NCF3 valve 198  20mar2012 24.50
KJ66 cutter 6   18jun2010 19.77
UYN7 rod  211   09sep2010 11.55
JD03 switch 383 09jan2013 13.99
BV1E timer 26   03aug2013 34.50
;
run;

data add_inventory;  1
   input PartNumber $ Description $ NewStock @16 NewPrice;
   datalines;
K89R seal   6  247.50
AA11 hammer 55 32.26
BB22 wrench 21 17.35
KJ66 cutter 10 24.50
CC33 socket 7  22.19
BV1E timer 30  36.50
;
run;

data inventory_tool;
   modify inventory_tool add_inventory;  2
   by PartNumber;
   select (_iorc_);  3
      /* The observation exists in the master data set. */
      when (%sysrc(_sok)) do;  4
         InStock=InStock+NewStock;
         ReceivedDate=today();
         Price=NewPrice;
         replace;  5
      end;
      /* The observation does not exist in the master data set. */
      when (%sysrc(_dsenmr)) do;  6
         InStock=NewStock;
         ReceivedDate=today();
         Price=NewPrice;
         output;  7
         _error_=0;
      end;
      otherwise do;  8
```

```
            put 'An unexpected I/O error has occurred.'/
              'Check you data and your program.';  8
          _error_=0;
          stop;
        end;
    end;
 run;

proc print data=inventory_tool;
   title 'Tool Warehouse Inventory';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 DATA ステートメントは、ADD_INVENTORY トランザクションデータセットを作成します。

2 MODIFY ステートメントは、INVENTORY_TOOL データセットと ADD_INVENTORY データセットのデータをロードします。

3 _IORC_自動変数がエラーチェックのために使用されます。_IORC_の値は数値のリターンコードで、最新の入出力操作の状態を示します。

4 SYSRC 自動呼び出しマクロは、_IORC_の値が_SOK かどうかをチェックします。値が_SOK の場合、トランザクションデータセットのオブザベーションはマスタデータセットのオブザベーションと一致しています。

5 REPLACE ステートメントは、マスタデータセットのオブザベーションをトランザクションデータセットのオブザベーションで置き換え、INVENTORY マスタデータセットを更新します。

6 SYSRC 自動呼び出しマクロは、_IORC_の値が_DSENMR どうかをチェックします。値が_DSENMR の場合、トランザクションデータセットのオブザベーションはマスタデータセットに存在しません。

7 OUTPUT ステートメントはマスタデータセットの末尾に現在のオブザベーションを書き込みます。

8 どの条件も満たされない場合、PUT ステートメントがメッセージをログへ書き込みます。

次の出力には、更新された INVENTORY_TOOL データセットが表示されています。

図 21.3 更新された INVENTORY_TOOL データセット

Tool Warehouse Inventory

Obs	PartNumber	Description	InStock	ReceivedDate	Price
1	K89R	seal	40	09APR2013	247.50
2	M4J7	sander	98	20JUN2011	45.88
3	LK43	filter	121	19MAY2011	10.99
4	MN21	brace	43	10AUG2012	27.87
5	BC85	clamp	80	16AUG2012	9.55
6	NCF3	valve	198	20MAR2012	24.50
7	KJ66	cutter	16	09APR2013	24.50
8	UYN7	rod	211	09SEP2010	11.55
9	JD03	switch	383	09JAN2013	13.99
10	BV1E	timer	56	09APR2013	36.50
11	AA11	hammer	55	09APR2013	32.26
12	BB22	wrench	21	09APR2013	17.35
13	CC33	socket	7	09APR2013	22.19

次のメッセージがログに書き込まれます。

```
NOTE: The data set WORK.INVENTORY_TOOL has been updated.There were 3
observations rewritten, 3 observations added and 0 observations deleted.
```

注意:

OUTPUT ステートメントと REPLACE ステートメントがないプログラムを実行すると、マスタファイルは正確に更新されない場合があります。 DATA ステップで OUTPUT ステートメントまたは REPLACE ステートメントを使用することで、オブザベーションのデフォルト置き換えが上書きされます。DATA ステップでこれらのステートメントを使用する場合、実施したいアクションごとに明示的にプログラミングする必要があります。

詳細については、“OUTPUT Statement” (*SAS Statements: Reference*) および “REPLACE Statement” (*SAS Statements: Reference*)を参照してください。

重複した BY 変数がファイル更新に与える影響について

重複した BY 変数の DATA ステップでの処理方法

MODIFY ステートメントと一緒に BY ステートメントを使用する場合、BY 変数の値が重複したオブザベーションをマスタデータセットとトランザクションデータセットの両方で持つことができます。BY グループ処理では動的 WHERE 処理を使用してマスタデー

タセットのオブザベーションを検索するため、マスタデータセットとトランザクションデータセットのどちらも並べ替える必要はありません。

DATA ステップでは、重複したオブザベーションを次の方法で処理します。

- 重複した BY 値がマスタデータセットに存在する場合、MODIFY では、マスタデータセットの最初に出現したオブザベーションへ現在のトランザクションを適用します。
- 重複した BY 値がトランザクションデータセットに存在する場合、一方のオブザベーションが他方の上に適用されることになり、その値が交互に上書きされます。最後のトランザクションでの値がマスタデータセットの最終の値になります。
- マスタデータセットとトランザクションデータセットの両方が重複した BY 値を含んでいる場合、MODIFY ステートメントはマスタデータセットのグループで最初に出現したものに各トランザクションを適用します。

プログラム

このセクションのプログラムは、ADD_INVENTORY_2 トランザクションデータセットのオブザベーションで INVENTORY_2 マスタデータセットを更新します。両方のデータセットが、BY 変数 PartNumber の連続した重複値と連続しない重複値を含んでいます。

次のプログラムは、マスタデータセット INVENTORY_2 を作成します。データセットが PartNumber M4J7 に対する 3 つのオブザベーションを含むことに注意してください。

```
data inventory_2;
   input PartNumber $ Description $ InStock @17
         ReceivedDate date9. @27 Price;
   format ReceivedDate date9.;
   datalines;
K89R seal   34  27jul1998 245.00
M4J7 sander 98  20jun2012 45.88
M4J7 sander 98  20jun2012 45.88
LK43 filter 121 19may2013 10.99
MN21 brace 43   10aug2013 27.87
M4J7 sander 98  20jun2012 45.88
BC85 clamp 80   16aug2013 9.55
NCF3 valve 198  20mar2013 24.50
KJ66 cutter 6   18jun2013 19.77
;
run;

proc print data=inventory_2;
   title 'INVENTORY_2 Data Set';
run;
```

次の出力には、INVENTORY_2 データセットが表示されています。

図 21.4 INVENTORY_2 データセット

INVENTORY_2 Data Set

Obs	PartNumber	Description	InStock	ReceivedDate	Price
1	K89R	seal	34	27JUL1998	245.00
2	M4J7	sander	98	20JUN2012	45.88
3	M4J7	sander	98	20JUN2012	45.88
4	LK43	filter	121	19MAY2013	10.99
5	MN21	brace	43	10AUG2013	27.87
6	M4J7	sander	98	20JUN2012	45.88
7	BC85	clamp	80	16AUG2013	9.55
8	NCF3	valve	198	20MAR2013	24.50
9	KJ66	cutter	6	18JUN2013	19.77

次のプログラムはトランザクションデータセット ADD_INVENTORY_2 を作成し、マスタデータセット INVENTORY_2 を変更します。データセット ADD_INVENTORY_2 は PartNumber M4J7 に対する 3 つのオブザベーションを含むことに注意してください。

```
data add_inventory_2;
   input PartNumber $ Description $ NewStock;
   datalines;
K89R abc 17
M4J7 def 72
M4J7 ghi 66
LK43 jkl 311
M4J7 mno 43
BC85 pqr 75
;
run;

data inventory_2;
   modify inventory_2 add_inventory_2;
   by PartNumber;
   ReceivedDate=today();
   InStock=InStock+NewStock;
run;

proc print data=inventory_2;
   title 'Tool Warehouse Inventory';
run;
```

次の出力には、更新された INVENTORY_2 データセットが表示されています。

図 21.5 更新された INVENTORY_2 データセット: 重複した BY 変数

Tool Warehouse Inventory

Obs	PartNumber	Description	InStock	ReceivedDate	Price
1	K89R	abc	51	09APR2013	245.00
2	M4J7	mno	279	09APR2013	45.88
3	M4J7	sander	98	20JUN2012	45.88
4	LK43	jkl	432	09APR2013	10.99
5	MN21	brace	43	10AUG2013	27.87
6	M4J7	sander	98	20JUN2012	45.88
7	BC85	pqr	155	09APR2013	9.55
8	NCF3	valve	198	20MAR2013	24.50
9	KJ66	cutter	6	18JUN2013	19.77

欠損値の処理

デフォルトでは、マスタデータセットとトランザクションデータセットの両方に共通する変数にトランザクションデータセットで欠損値が含まれている場合、MODIFY ステートメントがマスタデータセットの値を欠損値で置き換えることはありません。

マスタデータセットの値を欠損値で置き換えるには、MODIFY ステートメントで UPDATEMODE=オプションを使用します。UPDATEMODE=では、マスタデータセットの既存の値をトランザクションデータセットの欠損値で置き換えるかどうかを指定します。

MODIFY ステートメントの UPDATEMODE=オプションを使用するための構文は、次のとおりです。

MODIFY *master-SAS-data-set transaction-SAS-data-set*
<UPDATEMODE=MISSINGCHECK | NOMISSINGCHECK>;

BY *by-variable*;

MISSINGCHECK は、マスタデータセットの値をトランザクションデータセットの欠損値で置き換えないようにします。これがデフォルトです。NOMISSINGCHECK では、欠損データのチェックは行なわれないためトランザクションデータセットの欠損値がマスタデータセットの値を置き換えることができます。

次の例は、マスタデータセット Event_List を作成し、競技大会のためのスケジュールおよびコードをそこに格納します。例では、トランザクションデータセット Event_Change で Event_List を更新し、スケジュールに関する新しい情報が格納されます。MODIFY ステートメントが UPDATEMODE=オプションの NOMISSINGCHECK 値を使用するため、マスタデータセットの値はトランザクションデータセットの欠損値で置き換えられます。

次のプログラムは EVENT_LIST マスタデータセットを作成します。

```
data Event_List;
   input Event $ 1-10 Weekday $ 12-20 TimeofDay $ 22-30 Fee Code;
   datalines;
Basketball Monday    evening   10 58
Soccer     Tuesday   morning   5  33
Yoga       Wednesday afternoon 15 92
Swimming   Wednesday morning   10 63
;
run;

proc print data=Event_List;
   title 'EVENT_LIST Data Set';
run;
```

次の出力には、EVENT_LIST データセットが表示されています。

図 21.6 *EVENT_LIST データセット*

EVENT_LIST Data Set

Obs	Event	Weekday	TimeofDay	Fee	Code
1	Basketball	Monday	evening	10	58
2	Soccer	Tuesday	morning	5	33
3	Yoga	Wednesday	afternoon	15	92
4	Swimming	Wednesday	morning	10	63

次のプログラムは EVENT_CHANGE トランザクションデータセットを作成します。

```
data Event_Change;
   input Event $ 1-10 Weekday $ 12-20 Fee Code;
   datalines;
Basketball Wednesday 10 .
Yoga       Monday    . 63
Swimming             .  .
;
run;

proc print data=Event_Change;
   title 'EVENT_CHANGE Data Set';
run;
```

次の出力には、EVENT_CHANGE トランザクションデータセットが表示されています。

図 21.7 EVENT_CHANGE トランザクションデータセット

EVENT_CHANGE Data Set

Obs	Event	Weekday	Fee	Code
1	Basketball	Wednesday	10	.
2	Yoga	Monday	.	63
3	Swimming		.	.

次のプログラムはマスタデータセットを変更して書き込みます。

```
data Event_List;
    modify Event_List Event_Change updatemode=nomissingcheck;
    by Event;
 run;

proc print data=Event_List;
   title 'Schedule of Athletic Events';
run;
```

次の出力には、変更された EVENT_LIST データセットが表示されています。

図 21.8 EVENT_LIST マスタデータセット: 欠損値

Schedule of Athletic Events

Obs	Event	Weekday	TimeofDay	Fee	Code
1	Basketball	Wednesday	evening	10	.
2	Soccer	Tuesday	morning	5	33
3	Yoga	Monday	afternoon	.	63
4	Swimming		morning	.	.

要約

ステートメント

BY *by-variable*;
マスタデータセットとトランザクションデータセットの対応するオブザベーションを識別します。*By-variable* は、BY ステートメントと一緒に使用する 1 つ以上の変数を示します。

MODIFY *master-SAS-data-set transaction-SAS-data-set*
<UPDATEMODE=MISSINGCHECK|NOMISSINGCHECK>;
1 つの SAS データセットの変数の値を別の SAS データセットの値に置き換えます。*Master-SAS-data-set* は、更新する必要のあるデータを含んでいます。*Transaction-SAS-data-set* は、マスタデータセットを更新するためのオブザベーションを含んでいます。

UPDATEMODE=引数は、トランザクションデータセットの欠損値をマスタデータセットの値に上書きするかどうかを決定します。MISSINGCHECK オプションは、トランザクションデータセットの欠損値でマスタデータセットの値を置き換えないようにします。これがデフォルトです。NOMISSINGCHECK オプションでは、欠損データのチェックは行なわれないため、トランザクションデータセットの欠損値でマスタデータセットの値を置き換えることができます。

MODIFY *SAS-data-set*;
プログラムで指定する値にデータセットの変数の値を置き換えます。

OUTPUT;
MODIFY ステートメントが存在する場合、マスタデータセットの末尾に現在のオブザベーションを書き込みます。

REPLACE;
MODIFY ステートメントが存在する場合、DATA ステートメントで指定されたデータセットで読み込まれたのと物理的に同じ場所に現在のオブザベーションを書き込みます。

詳細情報

MERGE ステートメント
詳細については、“MERGE Statement” (*SAS Statements: Reference*)を参照してください。

MODIFY ステートメント
MODIFY ステートメントのさまざまな応用の詳細については、“MODIFY Statement” (*SAS Statements: Reference*)を参照してください。

UPDATE ステートメント
詳細については、“UPDATE Statement” (*SAS Statements: Reference*)を参照してください。

22 章

複数の SAS データセットのオブザベーションの条件付き処理

複数の SAS データセットの条件付き処理について **347**
目的 347
前提条件 348
例で使用される入力 SAS データセット **348**
オブザベーションが読み込まれたデータセットを特定する **351**
IN=データセットオプションについて 351
プログラム 352
複数のデータセットから選択したオブザベーションを結合する **356**
最後のオブザベーションに基づき計算を実行する **357**
最後のオブザベーションが処理されるタイミングについて 357
プログラム 358
要約 **359**
ステートメント 359
詳細情報 **360**

複数の SAS データセットの条件付き処理について

目的

SAS データセットを結合する場合、そのオブザベーションがどのデータセットから読み込まれたかに基づき、条件を付けてオブザベーションを処理できます。次の操作を行えます。

- 結合されたデータセットの各オブザベーションがどのデータセットから読み込まれたかを特定します。
- 結合するデータセットから選択したオブザベーションだけを含む新しいデータセットを作成します。
- DATA ステップで最後のオブザベーションを SAS が処理するタイミングを判別して、合計を作成するなどの条件付きの演算を実行することができるようにします。

初期のトピックでこれらの概念のうちのいくつかを紹介していますが、このセクションでは、複数データセットの処理に適用します。例では、SET ステートメントを使用しますが、ここで説明する機能のすべてを MERGE、MODIFY および UPDATE ステートメントと一緒に使用することもできます。

前提条件

このセクションを使用する前に、次のセクションで説明した概念を理解している必要があります。

- 4 章, “生データから作成する: 基本” (51 ページ)
- 6 章, “SAS データセットから作成する” (91 ページ)
- 18 章, “SAS データセットのインタリーブ” (279 ページ)

例で使用される入力 SAS データセット

次のプログラムは、2 つの SAS データセット、SOUTHAMERICAN および EUROPEAN を作成します。各データセットは次の変数を含みます。

Year
: 1954 年から 1998 年までのワールドカップ決勝で南アメリカとヨーロッパの国々が競った年。

Country
: 競争相手国の名前。

Score
: ゲームの最終スコア。

Result
: ゲームの結果。勝者の値は **won** で、敗者の値は **lost** です。

次の例の PROC SORT ステートメントは、BY 変数に従って昇順にデータセットを並べ替えます。次の例で、インタリーブされたデータセットを作成するには、データは昇順であることが必要です。

```
data southamerican;
   title 'South American World Cup Finalists from 1954 to 1998';
   input  Year $  Country $ 9-23 Score $ 25-28 Result $ 32-36;
   datalines;
1998    Brazil          0-3    lost
1994    Brazil          3-2    won
1990    Argentina       0-1    lost
1986    Argentina       3-2    won
1978    Argentina       3-1    won
1970    Brazil          4-1    won
1962    Brazil          3-1    won
1958    Brazil          5-2    won
;

data european;
   title 'European World Cup Finalists From 1954 to 1998';
   input  Year $  Country $ 9-23 Score $ 25-28 Result $ 32-36;
   datalines;
1998    France          3-0    won
1994    Italy           2-3    lost
1990    West Germany    1-0    won
1986    West Germany    2-3    lost
1982    Italy           3-1    won
```

```
1982    West Germany    1-3    lost
1978    Netherlands     1-3    lost
1974    West Germany    2-1    won
1974    Netherlands     1-2    lost
1970    Italy           1-4    lost
1966    England         4-2    won
1966    West Germany    2-4    lost
1962    Czechoslovakia  1-3    lost
1958    Sweden          2-5    lost
1954    West Germany    3-2    won
1954    Hungary         2-3    lost
;
run;

proc sort data=southamerican;
   by year;

proc print data=southamerican;
   title 'World Cup Finalists:';
   title2 'South American Countries';
   title3 'from 1954 to 1998';
run;

proc sort data=european;
   by year;
run;

proc print data=european;
   title 'World Cup Finalists:';
   title2 'European Countries';
   title3 'from 1954 to 1998';
run;
```

PROC SORT ステートメントは、BY 変数に従って昇順にデータセットを並べ替えます。次の例で、インタリーブされたデータセットを作成するには、データは昇順であることが必要です。

次の出力には、SOUTHAMERICAN データセットが表示されています。

図 22.1 大陸別ワールドカップ決勝戦出場者: 南アメリカ

World Cup Finalists:
South American Countries
from 1954 to 1998

Obs	Year	Country	Score	Result
1	1958	Brazil	5-2	won
2	1962	Brazil	3-1	won
3	1970	Brazil	4-1	won
4	1978	Argentina	3-1	won
5	1986	Argentina	3-2	won
6	1990	Argentina	0-1	lost
7	1994	Brazil	3-2	won
8	1998	Brazil	0-3	lost

次の出力には、EUROPEAN データセットが表示されています。

図 22.2 大陸別ワールドカップ決勝戦出場者: ヨーロッパ

World Cup Finalists:
European Countries
from 1954 to 1998

Obs	Year	Country	Score	Result
1	1954	West Germany	3-2	won
2	1954	Hungary	2-3	lost
3	1958	Sweden	2-5	lost
4	1962	Czechoslovakia	1-3	lost
5	1966	England	4-2	won
6	1966	West Germany	2-4	lost
7	1970	Italy	1-4	lost
8	1974	West Germany	2-1	won
9	1974	Netherlands	1-2	lost
10	1978	Netherlands	1-3	lost
11	1982	Italy	3-1	won
12	1982	West Germany	1-3	lost
13	1986	West Germany	2-3	lost
14	1990	West Germany	1-0	won
15	1994	Italy	2-3	lost
16	1998	France	3-0	won

オブザベーションが読み込まれたデータセットを特定する

IN=データセットオプションについて

2 つ以上のデータセットのオブザベーションを結合して新しいデータセットを作成する場合、そのオブザベーションがどのデータセットに由来するかがわかると有用な場合があります。たとえば、どのデータセットからオブザベーションが読み込まれたかに基づいて計算を実施したいときです。そうしないと、後の処理で必要となる重要なコンテキストの情報を失う場合があります。IN=データセットオプションを使用して、特定のオブザベーションが読み込まれたデータセットはどれかを判別できます。

プログラムデータベクトルに現在あるオブザベーションはどのデータセットから読み込まれたかが、IN=データセットオプションによって判別できます。SET ステートメントでのこのオプションの構文は次のとおりです。

SET *SAS-data-set-1* (IN=*variable*) *SAS-data-set-2*;

BY *a-common-variable*;

SET、MERGE、MODIFY、UPDATE ステートメントでデータセットに IN=オプションを使用する場合、SAS はそのデータセットに関連した一時変数を作成します。現在プログラムデータベクトルにあるオブザベーションが読み込まれた元のデータセットである場合、*variable* の値は 1 です。読み込まれたデータセットでない場合、値は 0 です。SET、MERGE、MODIFY、UPDATE ステートメントで指定するデータセットのいずれかまたはすべてに IN=オプションを使用できます。ただし、それぞれの場合で異なる変数名を使用する必要があります。

注: IN=変数は DATA ステップの実行中のみ存在します。作成される出力データセットには書き込まれません。

プログラム

SOUTHAMERICAN のすべてのオブザベーションが南アメリカの大陸に関連し、EUROPEAN のすべてのオブザベーションはヨーロッパ大陸に関連しているので、元のデータセット(SOUTHAMERICAN と EUROPEAN)では国々のある大陸を識別する変数を必要としません。データセットを結合する場合にはそのコンテキストが失われますが、このケースではそれは各オブザベーションの関連する大陸です。次の例では、SET ステートメントを BY ステートメントと一緒に使用し、2 つのデータセットを結合して年代順にすべてのオブザベーションを含む 1 つのデータセットにします。

```
data finalists;
   set southamerican european;
   by year;
run;

proc print data=finalists;
   title 'World Cup Finalists';
   title2 'from 1954 to 1998';
run;
```

次の出力には、FINALISTS データセットが表示されています。

図 22.3 年でグループ化したワールドカップ決勝戦出場者

World Cup Finalists from 1954 to 1998

Obs	Year	Country	Score	Result
1	1954	West Germany	3-2	won
2	1954	Hungary	2-3	lost
3	1958	Brazil	5-2	won
4	1958	Sweden	2-5	lost
5	1962	Brazil	3-1	won
6	1962	Czechoslovakia	1-3	lost
7	1966	England	4-2	won
8	1966	West Germany	2-4	lost
9	1970	Brazil	4-1	won
10	1970	Italy	1-4	lost
11	1974	West Germany	2-1	won
12	1974	Netherlands	1-2	lost
13	1978	Argentina	3-1	won
14	1978	Netherlands	1-3	lost
15	1982	Italy	3-1	won
16	1982	West Germany	1-3	lost
17	1986	Argentina	3-2	won
18	1986	West Germany	2-3	lost
19	1990	Argentina	0-1	lost
20	1990	West Germany	1-0	won
21	1994	Brazil	3-2	won
22	1994	Italy	2-3	lost
23	1998	Brazil	0-3	lost
24	1998	France	3-0	won

各オブザベーションがそれぞれどのデータセットに由来するかを示せば、この出力がより有用だろうことがわかります。この問題を解決するには、次のプログラムで IF-THEN/ELSE ステートメントとあわせて IN=データセットオプションを使用します。オブザベーションを読み込んだデータセットを特定することで、条件付きステートメントが実行され、新しいデータセット FINALISTS の各オブザベーションにある変数 Continent へ適切な値が割り当てられます。

```
data finalists;
   set southamerican (in=S) european;  1
   by Year;
   if S then Continent='South America';  2
   else Continent='Europe';
run;

proc print data=finalists;
   title 'World Cup Finalists';
   title2 'from 1954 to 1998';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 SET ステートメントの IN=オプションは、S という名前の変数を作成するように SAS に指示します。

2 現在のオブザベーションがデータセット SOUTHAMERICAN に由来する場合、S の値は 1 です。そうでない場合は、値は 0 です。IF-THEN/ELSE ステートメントは、S の値に応じて 2 つの割り当てステートメントのうちの 1 つを実行します。オブザベーションがデータセット SOUTHAMERICAN に由来する場合、Continent に割り当てられる値は South America です。オブザベーションがデータセット EUROPEAN に由来する場合、Continent に割り当てられる値は Europe です。

次の出力には、更新された FINALISTS データセットが表示されています。

図 22.4 大陸情報を付けたワールドカップ決勝戦出場者

World Cup Finalists from 1954 to 1998

Obs	Year	Country	Score	Result	Continent
1	1954	West Germany	3-2	won	Europe
2	1954	Hungary	2-3	lost	Europe
3	1958	Brazil	5-2	won	South America
4	1958	Sweden	2-5	lost	Europe
5	1962	Brazil	3-1	won	South America
6	1962	Czechoslovakia	1-3	lost	Europe
7	1966	England	4-2	won	Europe
8	1966	West Germany	2-4	lost	Europe
9	1970	Brazil	4-1	won	South America
10	1970	Italy	1-4	lost	Europe
11	1974	West Germany	2-1	won	Europe
12	1974	Netherlands	1-2	lost	Europe
13	1978	Argentina	3-1	won	South America
14	1978	Netherlands	1-3	lost	Europe
15	1982	Italy	3-1	won	Europe
16	1982	West Germany	1-3	lost	Europe
17	1986	Argentina	3-2	won	South America
18	1986	West Germany	2-3	lost	Europe
19	1990	Argentina	0-1	lost	South America
20	1990	West Germany	1-0	won	Europe
21	1994	Brazil	3-2	won	South America
22	1994	Italy	2-3	lost	Europe
23	1998	Brazil	0-3	lost	South America
24	1998	France	3-0	won	Europe

複数のデータセットから選択したオブザベーションを結合する

特定の基準に応じて選択されたオブザベーションだけを含むデータセットを作成するには、サブセット化 IF ステートメントおよび複数のデータセットを指定する SET ステートメントを使用できます。次の DATA ステップは、2 つの入力データセットを読み込んで優勝チームだけをリストする結合データセットを作成します。

```
data champions(drop=result); 1
   set southamerican (in=S) european; 2
   by Year;
   if result='won';  3
   if S then Continent='South America';  4
   else Continent='Europe';
run;

proc print data=champions;
   title 'World Cup Champions from 1954 to 1998';
   title2 'including Countries'' Continent';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 DROP=データセットオプションでは、この変数に対する値がすべて同じになるので、新しいデータセット CHAMPIONS から変数 Result を削除します。

2 SET ステートメントは 2 つのデータセット(SOUTHAMERICAN と EUROPEAN)からオブザベーションを読み込みます。S=データオプションでは、変数 S が作成されます。この変数は、オブザベーションが SOUTHAMERICAN データセットから読み込まれるたびごとに 1 に設定されます。

3 サブセット化 IF ステートメントは、Result 変数の値が **won** の場合にのみ、出力データセット CHAMPIONS にオブザベーションを書き込みます。

4 現在のオブザベーションがデータセット SOUTHAMERICAN に由来する場合、S の値は 1 です。そうでない場合は、値は 0 です。IF-THEN/ELSE ステートメントは、S の値に応じて 2 つの割り当てステートメントのうちの 1 つを実行します。オブザベーションがデータセット SOUTHAMERICAN に由来する場合、Continent に割り当てられる値は South America です。オブザベーションがデータセット EUROPEAN に由来する場合、Continent に割り当てられる値は Europe です。

次の出力には、CHAMPIONS データセットが表示されています。

図 22.5 選択したオブザベーションの結合

World Cup Champions from 1954 to 1998 including Countries' Continent

Obs	Year	Country	Score	Continent
1	1954	West Germany	3-2	Europe
2	1958	Brazil	5-2	South America
3	1962	Brazil	3-1	South America
4	1966	England	4-2	Europe
5	1970	Brazil	4-1	South America
6	1974	West Germany	2-1	Europe
7	1978	Argentina	3-1	South America
8	1982	Italy	3-1	Europe
9	1986	Argentina	3-2	South America
10	1990	West Germany	1-0	Europe
11	1994	Brazil	3-2	South America
12	1998	France	3-0	Europe

最後のオブザベーションに基づき計算を実行する

最後のオブザベーションが処理されるタイミングについて

多くのアプリケーションでは、DATA ステップが入力データセットにある最後のオブザベーションを処理するタイミングの特定が必要です。たとえば、データセットの最後のオブザベーションでのみ計算を行ないたい、あるいは、最後のオブザベーションが処理された後だけに、オブザベーションを書き込みたい場合があります。この目的で、SET、MERGE、MODIFY、UPDATE ステートメントと一緒に END=オプションを使用できます。このオプションの構文は次のとおりです。

SET *SAS-data-set-list* END=*variable*;

END=オプションは、DATA ステップが最後のオブザベーションを処理するとき値が 1 になる一時変数を定義します。それ以外のときはいつも、変数の値は 0 です。DATA ステップでは END=変数を使用できますが、SAS は出力結果のデータセットにそれを付加はしません。

注: 13 章, “計算に複数のオブザベーションを使用する” (197 ページ) で、単一データセットの SET ステートメントで END=オプションを使用する方法について説明します。END=オプションでは複数のデータセットが同じ方法で処理されます。ただし、すべての入力データセットで最後のオブザベーションが処理されるときに限り、END=が 1 に設定されることに注意することが重要です。

プログラム

この例では、SOUTHAMERICAN と EUROPEAN のデータを使用して、各大陸からのチームが 1954 年から 1998 年までにワールドカップで何年間勝利したかを計算します。

この計算を行うには、このプログラムで次のタスクを実行することが必要です。

1. 国がどの大陸に位置するかを識別します。
2. 各大陸からのチームがワールドカップで何回勝利したかの現在までの中間結果合計を保持します。
3. オブザベーションをすべて処理した後に、大陸がそれぞれワールドカップチャンピオンだった期間の長さを決定するために各大陸の総計に 4(ワールドカップ開催間隔の長さ)を掛けます。
4. 出力データセットに最終オブザベーションだけを書き込みます。最後のオブザベーションが処理されるまで、合計が含まれる変数には総計が含まれません。

次の DATA ステップでは、現在までの中間結果を計算し、それらの合計だけを含んでいる出力データセットを生成します。

```
data timespan (keep=YearsSouthAmerican keep=YearsEuropean);  1
   set southamerican (in=S) european end=LastYear;  2
   by Year;
   if result='won' then
      do;
         if S then SouthAmericanWins+1;  3
         else EuropeanWins+1;  3
      end;
   if lastyear then  4
      do;
         YearsSouthAmerican=SouthAmericanWins*4;
         YearsEuropean=EuropeanWins*4;
         output;  5
      end;
run;

proc print data=timespan;
   title 'Total Years as Reigning World Cup Champions';
   title2 'from 1954 to 1998';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 KEEP=オプションは YearsSouthAmerican 変数と YearsEuropean 変数のみを TIMESPAN データセットに書き込みます。

2 END=オプションは一時変数 LastYear を作成します。DATA ステップで最後のオブザベーションが処理され始めるまで、LastYear の値は 0 です。その時点で、LastYear の値が 1 に設定されます。

3 新しい 2 つの変数(SouthAmericanWins と EuropeanWins)は、それぞれの大陸が達成した勝利の数の現在までの中間結果を保持します。変数 Result の値が **won** である各オブザベーションの場合は、オブザベーションが由来する次のいずれかのデータセットに基づき、別の合計ステートメントが実行されます。

```
SouthAmericanWins+1;
```

または

```
EuropeanWins+1
```

4 DATA ステップで最後のオブザベーションが処理され始めるとき、LASTYEAR の値が 0 から 1 へ変更されます。この変更が起こると、条件付きステートメント **IF LastYear** は真になり、それに続くステートメントが実行されます。割り当てステートメントは、各大陸の勝利の総数に 4 を掛け、適切な変数 YearsSouthAmerican または YearsEuropean に結果を割り当てます。

5 OUTPUT ステートメントは新しく作成されたデータセットにオブザベーションを書き込みます。DATA ステップは各反復の終わりに自動的にオブザベーションを書き込むことに注意してください。ただし、OUTPUT ステートメントはこの自動機能を無効にします。DATA ステップは最後のオブザベーションだけを TIMESPAN に書き込みます。DATA ステップがプログラムデータベクトルから出力データセットにオブザベーションを書き込む場合、DATA ステートメントの KEEP=データセットオプションによる指示のとおり、2 つの変数、YearsSouthAmerican と YearsEuropean だけを書き込みます。

次の出力には、TIMESPAN データセットが表示されています。

図 22.6 *データセットの最後のオブザベーションに基づいて計算を実行するために END=オプションを使用*

Total Years as Reigning World Cup Champions from 1954 to 1998

Obs	YearsSouthAmerican	YearsEuropean
1	24	24

要約

ステートメント

IF *condition*;
条件が真かどうかが検証されます。それが真の場合、SAS は現在のオブザベーションの処理を続行します。それが偽の場合は、SAS はそのオブザベーションの処理を停止し、DATA ステップの始めに戻ります。このタイプの IF ステートメントは、元のオブザベーションのサブセットを生成するため、サブセット化 IF ステートメントと呼ばれます。

IF *condition* THEN *action*;
<ELSE *action*;>
条件が真かどうかが検証されます。真の場合、THEN 句の処理が実行されます。条件が偽で ELSE ステートメントが存在する場合、ELSE 処理が実行されます。条件が偽で ELSE ステートメントが存在しない場合、DATA ステップの次のステートメントへ実行が進みます。

SET *SAS-data-set* (IN=*variable*) *SAS-data-set-list*;
SAS データセットに関係する変数を作成します。現在プログラムデータベクトルにあるオブザベーションが読み込まれた元のデータセットである場合、*variable* の値

は 1 です。それ以外の場合、値は 0 です。IN=変数は DATA ステップを実行している間のみ存在します。出力データセットには書き込まれません。

SET、MERGE、MODIFY、UPDATE ステートメントで任意のデータセットを指定した IN=オプションを使用できますが、それぞれに異なる変数名を使用することが必要です。

SET *SAS-data-set-list* END=*variable*;
DATA ステップが最後のオブザベーションを処理し始めるまでその値が 0 である変数を作成します。最後のオブザベーションの処理が始まると、*variable* の値は 1 に変更されます。END=変数は DATA ステップを実行している間のみ存在します。出力データセットには書き込まれません。

MERGE、MODIFY および UPDATE ステートメントとも一緒に END=オプションを使用できます。

詳細情報

データセットオプション
データセットオプションについては、“選択した変数の読み込み” (96 ページ)を参照してください。

DO ステートメント
DO ループ処理の詳細については、14 章, “プログラミングを簡単にする” (213 ページ)を参照してください。

IF ステートメント
サブセット化 IF ステートメントと条件付き IF ステートメントの両方の詳細については、10 章, “選択したオブザベーションの操作” (147 ページ)を参照してください。

OUTPUT およびサブセット化 IF ステートメント
OUTPUT およびサブセット化 IF ステートメントの使用の詳細については、10 章, “選択したオブザベーションの操作” (147 ページ)を参照してください。

SUM ステートメントおよび END=オプション
合計の累計および END=オプションの使用の詳細については、“データセット全体の合計を累計する” (199 ページ)および “BY グループの合計の取得” (202 ページ)を参照してください。

5 部

SAS プログラムのデバッグ

23 章
SAS ログを使用した SAS セッションの分析 363

24 章
SAS 出力と SAS ログの出力指定 383

25 章
エラーの診断と回避 391

26 章
プログラムの論理エラーの検出 403

23 章

SAS ログを使用した SAS セッションの分析

SAS ログを使用した SAS セッションの分析について **364**
目的 364
前提条件 364

SAS ログについて **364**
SAS ログの役割について 364
ログでのエラーの解決 365

SAS ログの場所 **366**

ログ構造について **366**
SAS ログのコンポーネント 366
SAS ログのメッセージ 366
SAS ログの NOTE メッセージ 367
SAS ログの警告メッセージ 368
SAS ログのエラーメッセージ 369
構文エラーの検出 369

SAS ログへの書き込み **370**
SAS ログへのデフォルト出力 370
SAS ログへのメッセージの書き込み: PUT ステートメント 371
入力レコードのコンテンツの書き込み: LIST ステートメント 372
SAS ログへのメッセージの書き込み: %PUT マクロステートメント 374

SAS ログで情報を非表示にする **376**
SAS システムオプションを使用したログ出力の抑制 376
SAS ステートメントを非表示にする 376
システム情報を非表示にする 376
エラーメッセージ数の制限 377
SAS ログの詳細レベルの調整 377
SAS ステートメント、NOTE メッセージ、エラーメッセージを非表示にする 377

ログの表示形式の変更 **379**

要約 **379**
ステートメント 379
システムオプション 380

詳細情報 **381**

SAS ログを使用した SAS セッションの分析について

目的

SAS ログは、SAS セッションおよびプログラムを分析するための有用なツールです。このセクションでは、次について学習します。

- 出力に関するログ
- SAS ログでのメッセージの種類
- ログ構造
- SAS ログへの書き込み
- SAS ログで情報を非表示にする

前提条件

次のセクションで説明した SAS プログラミングの基本概念を理解している必要があります。

- 1 章, “SAS System について” (3 ページ)
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)
- 4 章, “生データから作成する: 基本” (51 ページ)

SAS ログについて

SAS ログの役割について

SAS ログは、SAS プログラムを実行した結果として出力されます。SAS ログは、作成したデータセットの名前からそれらのデータセットにあるオブザベーションや変数の数まで、SAS セッション内や SAS プログラムで実行したすべての記録を提供します。この記録では、どのステートメントが実行されたか、DATA ステップと PROC ステップにどれくらいの時間を必要としたか、およびそのプログラムがエラーを含むかどうかを示すことができます。

SAS 出力と同様に、SAS ログの出力先は、SAS の実行方法と動作環境に応じて異なります。SAS ログのコンテンツは、実行される DATA ステップと PROC ステップ、および使用されるオプションによって異なります。

次の例におけるサンプルログは、1 つの DATA ステップと 2 つの PROC ステップを含む SAS プログラムによって作成されました。DATA ステップでは、“UNIVERSITY_TEST_SCORES データセット” (793 ページ)にあるファイルが使用されています。LIBNAME ステートメントを使用して、ライブラリ参照名を作成し、データセットの場所を識別します。データセットは、このセクションではこれ以降、ライブラリ参照名 OUT で参照される SAS データライブラリに格納されます。

LIBNAME ステートメントは次の形式を使用します。

```
libname libref 'your-data-library';
```

次の SAS プログラムを実行すると、下に示すサンプルログが作成されます。

```
libname out 'your-data-library';

data out.university_test_scores;
   infile out 'your-input-file';
   input Test $ Gender $ Year TestScore;
run;

proc sort data=out.university_test_scores;
   by test;
run;

proc print data=out.university_test_scores;
   by test;
   label TestScore='Test Score';
   title1 'University Test Scores by Year, 2005-2011';
   title3 'Separate Statistics by Test Type';
run;
```

ログ 23.1 *SAS ログの例*

```
1    libname out 'your-data-library'; NOTE: Libref OUT was successfully assigned
as follows: Engine:        V9 Physical Name: your-data-library 2 3    data
out.university_test_scores; 4       infile 'your-input-file'; 5       input Test
$ Gender $ Year TestScore; 6    run; NOTE: The infile 'your-input-file' is:
Filename=your-input-file RECFM=V,LRECL=32767,File Size (bytes)=544, Last
Modified=03May2013:07:18:05, Create Time=01May2013:10:41:25 NOTE: 28 records
were read from the infile 'your-input-file'.The minimum record length was 17.The
maximum record length was 18.NOTE: The data set OUT.UNIVERSITY_TEST_SCORES has
28 observations and 4 variables.NOTE: DATA statement used (Total process time):
real time           0.03 seconds cpu time            0.03 seconds 7 8    proc
sort data=out.university_test_scores; 9       by test; 10   run; NOTE: There
were 28 observations read from the data set OUT.UNIVERSITY_TEST_SCORES.NOTE: The
data set OUT.UNIVERSITY_TEST_SCORES has 28 observations and 4 variables.NOTE:
PROCEDURE SORT used (Total process time): real time           0.15 seconds cpu
time            0.04 seconds 11 12   proc print data=out.university_test_scores;
NOTE: Writing HTML Body file: sashtml.htm 13      by test; 14      label
TestScore='Test Score'; 15      title1 'University Test Scores by Year,
2005-2011'; 16      title3 'Separate Statistics by Test Type'; 17   run; NOTE:
There were 28 observations read from the data set
OUT.UNIVERSITY_TEST_SCORES.NOTE: PROCEDURE PRINT used (Total process time): real
time           0.51 seconds cpu time            0.39 seconds
```

このログを作成した SAS プログラムは、エラーなく実行されました。NOTE メッセージは、SAS がどのようにプログラムを処理したかを示す情報メッセージです。

ログでのエラーの解決

前出のプログラムにエラーがあった場合、それらのエラーはセッションの一部として SAS ログに反映されます。SAS は、データエラー、構文エラーおよびプログラミングエラーのメッセージを生成します。それらのメッセージを参照してプログラムに必要な変更を加えると、プログラムが正常に実行できるようになります。

プログラムの初回実行時に、プログラムエラーのすべてを SAS が識別するとは限りません。1 つのエラーを解決することで、プログラムに別のエラーが発生する場合もあります。

SAS ログの場所

ログの出力先は、SAS の開始、実行、終了に使用する方法によって決まります。使用している動作環境および SAS システムオプションの設定によっても異なります。次の表に、動作方法ごとのデフォルトの出力先を示します。

表 23.1 SAS ログのデフォルト出力先

動作方法	SAS ログの出力先
SAS ウィンドウ環境(対話型フルスクリーン)	**ログ**ウィンドウ
対話型ラインモード	端末ディスプレイで、ステートメント入力時に使用
非対話型 SAS プログラム	動作環境によって異なる
バッチジョブ	プリンタまたはディスクファイル、動作環境によって異なる

ログ構造について

SAS ログのコンポーネント

SAS ログには、プログラムの実行時に問題を判別する助けとなる価値ある情報が提供されています。特に、何か疑問があって、オンサイトの SAS サポート要員や SAS テクニカルサポートに連絡する必要がある場合に役立ちます。ログのコンテンツが問題の分析を援助してくれます。

ログには、次の要素が含まれます。

- SAS ステートメント(DATA ステップおよび PROC ステップ)
- エラーメッセージ
- NOTE メッセージおよび警告メッセージ
- 作成される各データセットのオブザベーションおよび変数の数を含む NOTE メッセージ

SAS ログのメッセージ

SAS は、プログラムのコンパイルと実行フェーズでエラー処理を行い、NOTE メッセージ、警告メッセージ、およびエラーメッセージをログに書き込みます。プログラムにエラーがある場合、SAS はエラーの意味を解釈しようとします。SAS がエラーを修正できない場合、ログにエラーメッセージを書き込みます。いくつかのエラーは、ログにあるメッセージで十分に説明されます。エラーの発生箇所を SAS が常に正確に検出できると

は限らないため、解釈が容易ではないメッセージもあります。SAS ログで処理されているメッセージを理解し、コードを変更することで、SAS プログラムをデバッグできます。

SAS ログには、プログラムのデバッグを援助する、次の種類のメッセージが表示されます。

- NOTE メッセージ
- 警告メッセージ
- エラーメッセージ

これらのメッセージそれぞれが、プログラムの処理についてさまざまな情報を提供します。ログにあるすべてのメッセージを確認することが、ログを理解するにはよい練習になります。

SAS ログの NOTE メッセージ

SAS NOTE メッセージは情報メッセージで、プログラムの実行を停止しません。NOTE メッセージでは、コードの一部がプログラムとして正しくないことを示すことができます。

SAS NOTE メッセージでは、処理時間、DATA ステップや PROC ステップの正常/異常終了、ファイルから読み込まれたレコード数、およびプログラムで使用される入力ファイルの名前などの項目についての情報も提供できます。

次に SAS NOTE メッセージの例を示します。

- NOTE: The data set WORK.TEST has 50 observations and five variables.
- NOTE: 35 records were read from the INFILE *file-name*.
- NOTE: Variable ABCD is uninitialized.
- NOTE: The SAS System stopped processing this step because of errors.
- NOTE: No variables in data set WORK.TEST.
- NOTE: Invalid data for *variable-name* at line *n*.

次のプログラムは、2 番目の DATA ステップへの入力として使用される STOCK データセットを作成します。STOCK データセットには、Inventory に無効な値があります。INPUT ステートメントは Inventory を数値として識別しますが、4 番目のオブザベーションではこのフィールドに英文字が入っています。プログラムは PUT ステートメントを使用してユーザー指定のメッセージをログに書き込みます。(PUT ステートメントの詳細については、“SAS ログへのメッセージの書き込み: PUT ステートメント” (371 ページ)を参照してください。)最初のレコードは Product フィールドで欠損値になっていることに注意します。この欠損値では、プログラムの実行は停止しません。

```
data stock;
   input inventory QuantitySold Idnum 8-11 Product $ 13-18 cost 20-24;
   datalines;
100 52 1001        67.45
345 49 1020 saw    99.99
237 55 2003 wrench 34.97
abc 65 3015 shovel 25.99
932 38 4215 rake   22.50
;
run;

data stock2;
   set stock;
   if inventory < 300 and QuantitySold > 50 then
      put 'Time to order product: ' @27 product= @42 idnum=;
```

```
      else;
         if inventory > 300 then
            put 'No need to order product: ' @27 product= @42 idnum=;
   run;
```

ログ 23.2 *Stock Inventory プログラムからのログ出力*

```
94   data stock; 95      input inventory QuantitySold Idnum 8-11 Product
$ 13-18; 96      datalines; NOTE: Invalid data for inventory in line 100
1-3.RULE:      ----+----1----+----2----+----3----+----4----+----5----+----6----
+----7----+----8----+- 100        abc 65 3015 shovel inventory=.QuantitySold=65
Idnum=3015 Product=shovel _ERROR_=1 _N_=4 NOTE: The data set WORK.STOCK has 5
observations and 4 variables.NOTE: DATA statement used (Total process time):
real time           0.00 seconds cpu time            0.00 seconds 102  ; 103
run; 104 105  data stock2; 106     set stock; 107     if inventory < 300 and
QuantitySold > 50 then 108        put 'Time to order product: ' @27 product= @42
idnum=; 109     if inventory > 300 then 110        put 'No need to order
product: ' @27 product= @42 idnum=; 111  run; Time to order product:
Product=       Idnum=1001 No need to order product: Product=saw    Idnum=1020
Time to order product:    Product=wrench Idnum=2003 Time to order product:
Product=shovel Idnum=3015 No need to order product: Product=rake   Idnum=4215
NOTE: There were 5 observations read from the data set WORK.STOCK.NOTE: The data
set WORK.STOCK2 has 5 observations and 4 variables.NOTE: DATA statement used
(Total process time): real time           0.00 seconds cpu time            0.00
seconds
```

NOTE メッセージで、インベントリアイテムに無効なデータがあることが確認されます。SAS は、Inventory には数値を期待していましたが、4 番目のオブザベーションでは Inventory の値に文字がありました。SAS は引き続きプログラムを実行します。

SAS ログの警告メッセージ

SAS 警告メッセージは、コードに問題がある可能性について注意を促しますが、プログラムの実行は停止しません。たとえば、SAS は解釈できるけれど間違った単語を入力した場合、プログラムでは出力を生成しない場合などに、警告メッセージが出されます。

SAS ログの警告メッセージを表示して、プログラムが想定どおりに実行されたかどうかを確認することは重要です。次に SAS ログの警告メッセージの例を示します。

- WARNING: The data set WORK.TEST might be incomplete.When this step was stopped, there were 0 observations and 0 variables.
- WARNING: Data set WORK.TEST was not replaced because this step was stopped.
- WARNING: Assuming that the symbol DATA was misspelled as date.

次の DATA ステップを実行すると、エラーメッセージ、NOTE メッセージ、および警告メッセージが SAS ログに出力されます。この例では、SET ステートメントで入力データセットを識別します。しかしながら、SAS はそのデータセットを検出できませんでした。

```
data test;
   set mydataset;
run;
```

ログ 23.3 警告メッセージのあるログ出力

```
114  data test; 115     set mydataset; ERROR: File WORK.MYDATASET.DATA does not
exist.NOTE: The SAS System stopped processing this step because of
errors.WARNING: The data set WORK.TEST may be incomplete.When this step was
stopped there were 0 observations and 0 variables.WARNING: Data set WORK.TEST
was not replaced because this step was stopped.NOTE: DATA statement used (Total
process time): real time           0.03 seconds cpu time            0.00 seconds
```

ログの WARNING メッセージには、一時出力データセット WORK.TEST をプログラムが作成しようとしたという情報があります。

問題を修正するには、SET ステートメントの入力データセットがすでに存在しているか確認します。データセット名のミススペルと同程度の簡単なエラーの場合があります。プログラムを再度実行して、ログメッセージを確認します。

SAS ログのエラーメッセージ

SAS エラーメッセージは、コードの重要な問題について注意を促します。SAS は、プログラムの処理を停止するか、エラーフラグを書き込んでプログラムの処理を続行します。エラーメッセージがログに書き込まれます。

ステートメントの終わりにセミコロンがないのは、最も一般的なエラーの 1 つです。セミコロンがない場合、プログラムのどこでエラーが起こったかにより、SAS は複数のメッセージをログに書き込むことがあります。セミコロンがないエラーを識別する特定のメッセージはないので、エラー箇所の判別が困難な場合があります。

次に SAS ログのエラーメッセージの例を示します。

- Error: Variable ITEM2 not found.
- Error: Illegal reference to array ALL.
- Error: LIBNAME MYLIB is not assigned.
- Error: Syntax error, statement will be ignored.

SAS ログのエラーメッセージの例については、“SAS ログの警告メッセージ” (368 ページ)を参照してください。同一ログに複数種類のメッセージが出されることがよくみられます。

構文エラーの検出

次の SAS プログラムは 1 つの DATA ステップと 2 つの PROC ステップを含みます。ただし、DATA ステップには構文エラーがあります。セミコロンで終了していません。プログラムの次に示した SAS ログでは、プログラム処理について詳細に説明しています。

```
libname out 'your-data-library';

   /* omitted semicolon */
data out.university_test_scores2
   infile 'your-input-file';
   input test $ gender $ year TestScore;
run;

proc sort data=out.university_test_scores2;
   by test;
run;
```

```
proc print data=out.university_test_scores2;
   by test;
run;
```

次のログには、SAS がプログラムをどのように処理するのかを段階的に示します。

ログ 23.4 *構文エラーを識別するログ出力*

```
16  libname out 'your-data-library'; NOTE: Libref OUT was successfully assigned
as follows:  1 Engine:        V9 Physical Name: your-data-library 17 18
19      /* omitted semicolon */ 20   data out.university_test_scores2  2 21
infile 'your-input-file'; 22      input test $ gender $ year TestScore; 23
run; ERROR: No DATALINES or INFILE statement.  3 Error: Extension for physical
file name "your-input-file" does not correspond to a valid member type.  3 NOTE:
The SAS System stopped processing this step because of errors.  3 WARNING: The
data set OUT.UNIVERSITY_TEST_SCORES2 may be incomplete.When this step was
stopped there were 0 observations and 4 variables.  4 WARNING: The data set
WORK.INFILE may be incomplete.When this step was stopped there were 0
observations and 4 variables.  4 NOTE: DATA statement used (Total process time):
real time           0.01 seconds cpu time            0.01 seconds 24 25   proc
sort data=out.university_test_scores2;  5 26      by test; 27   run; NOTE: Input
data set is empty.  5 NOTE: The data set OUT.UNIVERSITY_TEST_SCORES2 has 0
observations and 4 variables.  5 NOTE: PROCEDURE SORT used (Total process time):
real time           0.01 seconds cpu time            0.01 seconds 28 29   proc
print data=out.university_test_scores2;  6 30      by test; 31   run; NOTE: No
observations in data set OUT.UNIVERSITY_TEST_SCORES2.  6 NOTE: PROCEDURE PRINT
used (Total process time): real time          0.00 seconds cpu time
0.00 seconds
```

次のリストは、前述のログの番号付き項目に対応しています。

1 LIBNAME ステートメントによって、ライブラリ参照名 OUT はユーザーのライブラリに正常に関連付けされました。

2 DATA ステートメントにセミコロンが欠損しているため、プログラムでエラーになります。

3 エラーメッセージによりエラーが判明し、プログラムが実行を停止したことが NOTE メッセージに示されています。

4 警告メッセージで、OUT.UNIVERSITY_TEST_SCORES2 データセットと WORK.INFILE データセットについての情報が提供されています。SAS は、前出の DATA ステートメントをセミコロンが出てくるまで読み込み、作成されるデータセットが WORK.INFILE であると解釈します。

5 SAS は出力データセットを並べ替えしようとします。

6 SAS は OUT.UNIVERSITY_TEST_SCORES2 データセットを書き込もうとします。

エラーを修正するには、DATA ステートメントの後にセミコロンを追加し、プログラムを再実行します。

SAS ログへの書き込み

SAS ログへのデフォルト出力

前述のサンプルログは、デフォルトでログに現われる情報を示します。DATA ステップで PUT ステートメントまたは LIST ステートメントを使用して、ログに書き込めます。また、%PUT マクロステートメントは、プログラム内の任意の場所で使用できます。これらのステートメントをユーザーの SAS プログラムのデバッグに使用できます。

SAS ログへのメッセージの書き込み: PUT ステートメント

PUT ステートメントについて

PUT ステートメントでは、テキスト文字列と変数値を含む指定の情報をログに書き込めます。DATA ステップが反復されるごとに PUT ステートメントが実行され、メッセージがログに書き込まれます。値は、カラム、リスト、フォーマット化、指定した出力形式で書き込めます。(詳細については、*SAS Statements: Reference* で INPUT ステートメントの別形式を参照してください。)PUT ステートメントは、有用なデバッグツールです。プログラムの他の部分からログにメッセージを書き込み、プログラムの実行をトラックできます。

これが PUT ステートメントの簡単な形式です。

PUT <'*message*'> | <*variable-name*>;

message には、SAS ログに表示したいメッセージを指定します。文字リテラルは、引用符で囲む必要があります。ユーザー指定のメッセージテキストには、"ここに注目"などの単語やフレーズを含むことができ、そのメッセージをより簡単に識別するための役に立ちます。

variable-name には、値をログに書き込む変数を指定します。変数名の後ろに等号を使用すると、変数名と値の両方がログに書き込まれます。

次の PUT ステートメントの使用によって、自動変数_ERROR_および_N_を含む、現在の DATA ステップに定義されているすべての変数の値を書き込めます。

```
PUT _ALL_;
```

PUT ステートメントとその使用法の詳細については、“PUT Statement” (*SAS Statements: Reference*)を参照してください。

例: PUT ステートメントでのログへの書き込み

次の例では、最初の DATA ステップで INVENTORY という一時データセットを作成します。このデータセットは、2 番目の DATA ステップの SET ステートメントへの入力として使用されます。PUT ステートメントはメッセージをログに書き込み、注文が必要なアイテムかどうかを示します。PUT ステートメントで変数名 Item の後ろに等号を続けて入力すると、PUT ステートメントは変数名とその値の両方をログに書き込みます。この例で PUT ステートメントのログ出力を示します。

```
data inventory;
   input InStock QuantitySold Idnum 8-11 Item $ 13-18;
   datalines;
100 52 1001 hammer
345 49 1020 saw
237 55 2003 wrench
864 65 3015 shovel
932 38 4215 rake
;
run;

data inventory2;
   set inventory;
   if InStock < 300 and QuantitySold > 50 then
      put 'Time to order product: ' Item=;
   if InStock > 300 then
      put 'No need to order product: ' Item=;
run;
```

ログ 23.5 *PUT ステートメントを使用したログ出力*

```
1    data inventory; 2       input InStock QuantitySold Idnum 8-11 Item $ 13-18;
3       datalines; NOTE: The data set WORK.INVENTORY has 5 observations and 4
variables.NOTE: DATA statement used (Total process time): real time
0.30 seconds cpu time            0.09 seconds 9    ; 10   run; 11 12   data
inventory2; 13      set inventory; 14      if InStock < 300 and QuantitySold >
50 then 15         put 'Time to order product: ' Item=; 16      if InStock > 300
then 17         put 'No need to order product: ' Item=; 18   run; Time to order
product: Item=hammer No need to order product: Item=saw Time to order product:
Item=wrench No need to order product: Item=shovel No need to order product:
Item=rake NOTE: There were 5 observations read from the data set
WORK.INVENTORY.NOTE: The data set WORK.INVENTORY2 has 5 observations and 4
variables.NOTE: DATA statement used (Total process time): real time
0.01 seconds cpu time            0.01 seconds
```

PUT ステートメントは 5 行をログに書き込みます。

プログラムコードには、特定のカラムに出力を書き込む指示を含むことができます。これを行うと、出力がさらに読みやすくなることがあります。上の例では、次の方法で IF ステートメントを変更できます。

```
if Inventory < 300 and QuantitySold > 50 then
   put 'Time to order product: ' @27 Item= @39 Idnum=;
if Inventory > 300 then
   putlog 'No need to order product: ' @27 Item= @39 Idnum=;
```

@27 は、変数 Item が 27 カラムで始まることを示します。@39 は、変数 Idnum が 39 カラムで始まることを示します。次に SAS ログ出力の一部を示します。

```
Time to order product:    Item=hammer Idnum=1001 No need to order product:
Item=saw    Idnum=1020 Time to order product:    Item=wrench Idnum=2003 No need
to order product: Item=shovel Idnum=3015 No need to order product: Item=rake
Idnum=4215
```

入力レコードのコンテンツの書き込み: LIST ステートメント

LIST ステートメントについて

DATA ステップで LIST ステートメントを使用して、現在の入力レコードをログに書き込みます。データを INPUT ステートメントから読み込む場合にのみ LIST ステートメントを使用できます。SET ステートメント、MERGE ステートメント、MODIFY ステートメント、UPDATE ステートメントを使用してデータを読み込む場合には、有効ではありません。LIST ステートメントは、DATA ステップの各反復の終わりにオブザベーションを書き込みます。

例: 現在の入力レコードのリスト

条件付き処理を組み合わせて選択された情報をログに書き込むには、PUT ステートメントのように、LIST ステートメントが非常に効果的であることを次のプログラムは示しています。

```
libname out 'your-data-library';

data out.university_test_scores3;
   infile 'your-data-file';
   input test $ gender $ year TestScore;
   if TestScore < 525 then delete;
   else list;
```

```
run;
```

ログ 23.6 レコードコンテンツのリスト

```
46  libname out 'your-data-library'; NOTE: Libref OUT was successfully assigned
as follows: Engine:        V9 Physical Name: your-data-library 47 48 49 50
data out.university_test_scores3; 51      infile 'your-input-file'; 52
input test $ gender $ year TestScore; 53      if TestScore < 525 then delete;
54      else list; 55   run; NOTE: The infile 'your-input-file' is:
Filename=your-input-file, RECFM=V,LRECL=32767,File Size (bytes)=544, Last
Modified=03May2013:07:18:05, Create Time=01May2013:10:41:25 RULE:     ----
+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----
+-- 21        Math   m 2008 525  18 23        Math   m 2009 527  18 25
Math   m 2010 530  18 27        Math   m 2011 531  18 NOTE: 28 records were read
from the infile 'your-input-file'.The minimum record length was 17.The maximum
record length was 18.NOTE: The data set OUT.UNIVERSITY_TEST_SCORES3 has 4
observations and 4 variables.NOTE: DATA statement used (Total process time):
real time           0.01 seconds cpu time            0.01 seconds
```

LIST ステートメントを実行するとき、SAS は現在の入力バッファをログに書き込みます。データの最初の行の前にカラムルーラがあることに注目してください。ルーラは、入力データがログに書き込まれたことを示します。入力バッファの参照カラムの位置へルーラを使用できます。LIST ステートメントを使用すると、各行の終わりにレコード長(この場合では各レコード長は 18)が出力されることに注意します。LIST ステートメントのこの機能は、可変長入力レコードをサポートする動作環境でのみ機能します。

例: 欠損データのあるレコードのリスト

LIST ステートメントを使用してレコードをログに書き込む場合の別の例を次に示します。この例では、EMPLOYEE データセットを作成します。プログラムは INPUT ステートメントを使用して、入力レコードにおける値の配置を記述します。各従業員の入力データは、3 行の長さで、従業員 ID 番号から始まります。3 行目には Salary の値が含まれます。ID 番号を読み込んだ後の#3 という表記は、SAS に対して、各レコードの 3 行目に行き Salary の値を読むことを指示しています。IF ステートメントでは、Salary が欠損値であるすべての従業員のレコードをリストするよう SAS に指示しています。

```
data employee;
   input IdNum 1-9 #3 Salary 1-8;
   if salary=. then list;
   datalines;
234567890
James Smith
70356.79
345678912
Jeffery Feldenstern
.
382623454
Sandy Lineman
75724.96
346521145
Jose Garcia
.
;
run;

proc print data=employee;
   format salary dollar10.2;
   title 'Employee Salary';
run;
```

2 つのオブザベーションで Salary が欠損値になっていることが分かります。

ログ 23.7 *欠損値のあるレコードコンテンツのリスト*

```
70   data employee; 71      input IdNum 1-9 #3 Salary 1-8; 72      if salary=.
then list; 73      datalines; RULE:      ----+----1----+----2----+----3----
+----4----+----5----+----6----+----7----+----8----+- 77         345678912
78         Jeffery Feldenstern 79         .83         346521145 84         Jose
Garcia 85         .NOTE: The data set WORK.EMPLOYEE has 4 observations and 2
variables.NOTE: DATA statement used (Total process time): real time
0.00 seconds cpu time            0.00 seconds 86   ; 87   run; 88 89   proc
print data=employee; 90      format salary dollar10.2; 91      title 'Employee
Salary'; 92   run; NOTE: There were 4 observations read from the data set
WORK.EMPLOYEE.NOTE: PROCEDURE PRINT used (Total process time): real
time           0.00 seconds cpu time            0.01 seconds
```

次の出力は欠損値を示しています。

図 23.1 *欠損値のある従業員給与*

Employee Salary

Obs	IdNum	Salary
1	234567890	$70,356.79
2	345678912	.
3	382623454	$75,724.96
4	346521145	.

SAS ログへのメッセージの書き込み: %PUT マクロステートメント

%PUT マクロステートメントについて

%PUT ステートメントは、DATA ステップに依存しないマクロステートメントです。プログラム内の任意の場所で使用して、メッセージをログに書き込めます。

例: ログへのメッセージの書き込み

次の例では、SAS プログラムでの%PUT マクロステートメントの使用方法を示します。

```
data professions;
   input Name $ 1-17 Gender $ 19 Occupation $ 21-33;
   datalines;
Shirley Grayson   F attorney
Kristen Hagshould F doctor
Matthew Rodriguez M
Michael Wu        M mathematician
Sophie Majkut     F physicist
;

%put Notice: This is the end of the first DATA step.;

data professions2;
```

```
   set professions;
   %put Notice: Testing for missing values.;
   if gender = 'M' and Occupation = ' ' then Occupation='MISSING';
run;

%put Notice: This is the end of the second DATA step.;

proc print data=professions2;
   title 'Staff Occupations';
run;
%put Notice: This is the end of the program.;
```

ログ 23.8　%PUT マクロステートメントを使用した出力

```
11   data professions; 12      input Name $ 1-17 Gender $ 19 Occupation $ 21-33;
13      datalines; NOTE: The data set WORK.PROFESSIONS has 5 observations and 3
variables.NOTE: DATA statement used (Total process time): real time
0.07 seconds cpu time            0.00 seconds 19   ; 20 21   %put Notice: This
is the end of the first DATA step.; Notice: This is the end of the first DATA
step.22 23   data professions2; 24      set professions; 25      %put Notice:
Testing for missing values.; Notice: Testing for missing values.26      if
gender = 'M' and Occupation = ' ' then Occupation='MISSING'; 27   run; NOTE:
There were 5 observations read from the data set WORK.PROFESSIONS.NOTE: The data
set WORK.PROFESSIONS2 has 5 observations and 3 variables.NOTE: DATA statement
used (Total process time): real time           0.01 seconds cpu time
0.01 seconds 28 29   %put Notice: This is the end of the second DATA step.;
Notice: This is the end of the second DATA step.30 31   proc print
data=professions2; 32      title 'Staff Occupations'; 33   run; NOTE: There were
5 observations read from the data set WORK.PROFESSIONS2.NOTE: PROCEDURE PRINT
used (Total process time): real time           0.04 seconds cpu time
0.01 seconds 34   %put Notice: This is the end of the program.; Notice: This is
the end of the program.
```

SAS は次の結果を生成します。

図 23.2　スタッフの職業

Staff Occupations

Obs	Name	Gender	Occupation
1	Shirley Grayson	F	attorney
2	Kristen Hagshould	F	doctor
3	Matthew Rodriguez	M	MISSING
4	Michael Wu	M	mathematician
5	Sophie Majkut	F	physicist

SAS ログで情報を非表示にする

SAS システムオプションを使用したログ出力の抑制

SAS ログに書き込まれないようにしたい情報もあります。SAS ステートメント、システムメッセージおよびエラーメッセージを非表示にするには、SAS システムオプションの NOSOURCE、NONOTES および ERRORS=を使用します。SAS を起動するときに、**OPTIONS** ウィンドウまたは OPTIONS ステートメントで、これらのオプションを指定できます。このセクションでは、オプションは OPTIONS ステートメントに指定します。

注: SAS システムオプションは、途中で変更を加えない限り、SAS セッションが終了するまで有効です。

SAS ステートメントを非表示にする

大規模な SAS プログラムを変更せずに定期的に実行している場合は、NOSOURCE システムオプションを次のように使用して、ログでの SAS ステートメントのリストを非表示にできます。

```
options nosource;
```

NOSOURCE システムオプションは、エラーを含んでいるソース行のみをログに書き込みます。次のように SOURCE システムオプションを指定することによりデフォルトに戻せます。

```
options source;
```

SOURCE システムオプションは後続のソース行をすべてログに書き込みます。

(%INCLUDE ステートメントで含まれるファイルからの)2 次ソースステートメントをログに書き込むかどうかも制御できます。2 次ステートメントを非表示にするには NOSOURCE2 システムオプションを次のように使用します。

```
options nosource2;
```

SOURCE2 システムオプションを使用すると、2 次ソースステートメントがログに含まれます。

```
options source2;
```

システム情報を非表示にする

次の項目を含む、ログによって提供される情報の多くは NOTE メッセージとして現れます。

- 著作権情報
- ライセンスとサイト情報
- データセットにあるオブザベーションと変数の数

SAS は、エラーによってステップの処理が停止したときにも、NOTE メッセージを発行します。

NOTE メッセージをログに表示しない場合は、NONOTES システムオプションを使用して非表示にします。

```
options nonotes;
```

NOTE で始まるすべてのメッセージが非表示になります。NOTES システムオプションを指定することによりデフォルトに戻せます。

```
options notes;
```

エラーメッセージ数の制限

SAS は、ユーザーの SAS プログラムに現れたデータ入力エラーのメッセージを書き込みます。エラーメッセージのデフォルト数は通常 20 ですが、サイトによって異なる場合があります。ERRORS=システムオプションを使用して、エラーメッセージがログに書き込まれるオブザベーションの最大数を指定します。

このオプションは、不正なデータのため生成されるエラーメッセージのみを制限することに注意してください。この種類のエラーは主に、INPUT ステートメントが数値として定義した変数に文字値を読み込もうとすることで起こります。

指定した数より多くのオブザベーションでデータエラーが検出された場合、処理は続行しますが、エラーメッセージにはそれ以上のエラーを書き込みません。たとえば、次の OPTIONS ステートメントは最大 5 つのオブザベーションに対する書き込みを指定しています。

```
options errors=5;
```

しかしながら、“SAS ステートメント、NOTE メッセージ、エラーメッセージを非表示にする” (377 ページ),で説明しているように、エラーメッセージを非表示にすることが利点とはいえない場合があります。

注: 警告メッセージが出ないようにできるオプションはありません。

SAS ログの詳細レベルの調整

MSGLEVEL=システムオプションは、SAS ログに書き込まれるメッセージに詳細なレベルを付けて調整します。MSGLEVEL=の値は、N または I のいずれかです。N では、SAS が NOTE メッセージ、警告メッセージ、およびエラーメッセージのみを書き込みます。これがデフォルトです。I では、インデックス使用状況、マージ処理、並べ替えユーティリティに関係する追加メッセージを SAS が書き込みます。

PRINTMSGLIST システムオプションは、SAS ログへのメッセージの拡張リストの書き込みを調整します。PRINTMSGLIST がデフォルトで、メッセージのリスト全体を SAS ログに書き込みます。NOPRINTMSGLIST では、最上位レベルのメッセージのみ書き込みます。

SAS ステートメント、NOTE メッセージ、エラーメッセージを非表示にする

次の SAS プログラムは、他の例と同じようにテストスコアデータを読み込みますが、この例では、変数 Gender の文字記号が省略されます。さらに、PROC PRINT での BY ステートメントを使用する前にデータが並べ替えられていません。効率を上げるため、ステートメント、NOTE メッセージおよびエラーメッセージが非表示にされています。

```
libname out 'your-data-library';
options nosource nonotes errors=0;

data out.university_test_scores4;
   infile 'your-input-file';
   input test $ gender year TestScore 25-27;
run;
```

```
proc print;
   by test;
run;
```

次の出力は、表示される SAS ログを示しています。SAS システムオプション ERRORS=0 が指定されているので、エラーの限界にすぐに到達してしまい、Gender を数値として読み込もうとすることに起因するエラーはログに書き込まれません。また、NOSOURCE および NONOTES システムオプションを指定すると、確認できる SAS のステートメントおよび SAS の処理を説明する NOTE メッセージはログに含まれません。ログは、OUT.UNIVERSITY_TEST_SCORES4 が昇順で並べ替えられていないことを説明するエラーメッセージを含んでいます。このエラーは無効な入力データによって起きたものではないので、ERRORS=0 オプションはこのエラーには影響を与えません。

ログ 23.9 *SAS ログで情報を非表示にする*

```
1    libname out 'your-data-library'; NOTE: Libref OUT was successfully assigned
as follows: Engine:        V9 Physical Name: your-data-library 2    options
nosource nonotes errors=0; ERROR: Data set OUT.UNIVERSITY_TEST_SCORES4 is not
sorted in ascending sequence.The current BY group has test = Verbal and the next
BY group has test = Math.
```

注: NOSOURCE、NONOTES および ERRORS=システムオプションは領域を節約するために使用されます。これらのオプションは、すでにテスト済みのプログラム(定期的に実行されるものなど)でとても有用です。ただし、このセクションで実証されているように、これらのシステムオプションを使用することが必ずしも適切だとは限りません。新しいプログラムの開発中には、ログのエラーメッセージはデバッグでは欠かせないでしょうし、制限されるべきではありません。同様に、プログラムにある問題を正確に特定する助けになるため、NOTE メッセージは非表示にしないようにします。特に、プログラムに習熟していない誰かにデバッグで助けを求める場合は、それらは特に重要です。つまり、プログラムがエラーなしで実行されるようになるまでは、ログで情報を非表示にしないようにします。

次の部分出力は、前の例が SOURCE、NOTES および ERRORS=オプションで再実行される場合の結果を示します。

```
options source notes errors=4;

data out.university_test_scores5;
   infile 'your-input-file';
   input test $ gender year TestScore;
run;

proc print data=out.university_test_scores5;
   by test;
run;
```

ログ 23.10 出力付きログ: ERRORS=4

```
13   libname out 'your-data-library'; NOTE: Libref OUT was successfully assigned
as follows: Engine:        V9 Physical Name: your-data-library 14 15   options
source notes errors=4; 16 17   data out.university_test_scores5; 18      infile
'your-input-file'; 19      input test $ gender year TestScore; 20   run; NOTE:
The infile 'your-input-file' is: Filename=your-input-file,
RECFM=V,LRECL=32767,File Size (bytes)=544, Last Modified=03May2013:07:18:05,
Create Time=01May2013:10:41:25 NOTE: Invalid data for gender in line 1
8-8.RULE:     ----+----1----+----2----+----3----+----4----+----5----+----6----
+----7----+----8----+-- 1         Verbal m 2005 504  18 test=Verbal gender=.
year=2005 TestScore=504 _ERROR_=1 _N_=1 NOTE: Invalid data for gender in line 2
8-8.2         Verbal f 2005 496 17 test=Verbal gender=. year=2005 TestScore=496
_ERROR_=1 _N_=2 NOTE: Invalid data for gender in line 3 8-8.3         Verbal m
2006 504  18 test=Verbal gender=. year=2006 TestScore=504 _ERROR_=1 _N_=3 NOTE:
Invalid data for gender in line 4 8-8.WARNING: Limit set by ERRORS= option
reached.Further errors of this type will not be printed.4         Verbal f 2006
497 17 test=Verbal gender=. year=2006 TestScore=497 _ERROR_=1 _N_=4 NOTE: 28
records were read from the infile 'your-input-file'.The minimum record length
was 17.The maximum record length was 18.NOTE: The data set
OUT.UNIVERSITY_TEST_SCORES5 has 28 observations and 4 variables.NOTE: DATA
statement used (Total process time): real time           0.04 seconds cpu
time            0.04 seconds 21 22   proc print
data=out.university_test_scores5; 23      by test; 24   run; ERROR: Data set
OUT.UNIVERSITY_TEST_SCORES5 is not sorted in ascending sequence.The current BY
group has test = Verbal and the next BY group has test = Math.NOTE: The SAS
System stopped processing this step because of errors.NOTE: There were 15
observations read from the data set OUT.UNIVERSITY_TEST_SCORES5.NOTE: PROCEDURE
PRINT used (Total process time): real time           0.06 seconds cpu
time            0.04 seconds
```

プログラムの今回の実行では、ログがより効果的な問題解決ツールとなります。ログには、多くの有益な NOTE メッセージだけでなく、プログラムの SAS ステートメントがすべて含まれています。特に、変数 Gender に対する無効データについて十分なメッセージを含んでいるため、問題が特定できます。この情報により、プログラムを修正して正常に実行できるようになります。

ログの表示形式の変更

対話型セッションを除き、PAGE ステートメントと SKIP ステートメントを使用してログをカスタマイズできます。ログで新しいページに移るには PAGE ステートメントを使用します。ログで行をスキップするには SKIP ステートメントを使用します。SKIP ステートメントで、スキップする行数を指定できます。行数を指定しない場合は、1 行がスキップされます。指定した数がそのページで残りの行数を超過する場合、SAS は PAGE ステートメントのように SKIP ステートメントを扱い、次のページのトップへスキップします。PAGE ステートメントと SKIP ステートメントはログには出力されません。

要約

ステートメント

LIST;
: 処理されるオブザベーションの入力バッファのコンテンツを SAS ログにリストします。

SAS ログ
SAS Language Reference: Concepts で、SAS ログについて追加情報が提供されています。

PAGE;
ログの新しいページにスキップします。

PUT '*message*' <*variable-name*>;
PUT <*variable-list*> | <_ALL_>;
SAS ログ、出力ファイル、または FILE ステートメントで指定された任意のファイルに行を書き込みます。DATA ステップの現在の反復で FILE ステートメントが実行されなかった場合には、PUT ステートメントが SAS ログへ書き込みます。

message には、ログに書き込まれる、ユーザー指定のテキストを指定します。

variable-name には、値をログに書き込む変数を指定します。

variable-list には、それらの値がログに書き込まれる変数のリストを指定します。

_ALL_は、_ERROR_と_N_を含むすべての変数の値がログに書き込まれることを示します。

PUT ステートメントは、1 つの DATA ステップ内で有効です。

詳細については、“PUT Statement” (*SAS Statements: Reference*)を参照してください。

%PUT '*message*';
SAS ログにメッセージを書き込みます。

message には、ログに書き込まれる、ユーザー指定のテキストを指定します。

%PUT マクロステートメントは、プログラム内の任意の場所で使用できます。

詳細については、“%PUT Statement” (*SAS Macro Language: Reference*)を参照してください。

SKIP <*n*>;
値 *n* で指定した行数を SAS ログ上でスキップします。値を指定しない場合は、SAS はブランク行をログに書き込みます。指定した数がそのページで残りの行数より多い場合、SAS は PAGE ステートメントのように SKIP ステートメントを扱い、次のページのトップへスキップします。

システムオプション

ERRORS=*n*
データ入力エラーに関するエラーメッセージをログに書き込む、オブザベーションの最大数を指定します。

NOTES | NONOTES
NOTE メッセージをログに書き込むかどうかを制御します。

SOURCE | NOSOURCE
SAS ステートメントをログに書き込むかどうかを制御します。

SOURCE2 | NOSOURCE2
%INCLUDE ステートメントで指定したファイルからの 2 次 SAS ステートメントをログに書き込むかどうかを制御します。

詳細情報

自動変数
: 自動変数_N_および_ERROR_の詳細については、25 章, “エラーの診断と回避” (391 ページ)を参照してください。

デバッグ
: デバッグの詳細については、25 章, “エラーの診断と回避” (391 ページ)を参照してください。

FILE ステートメントおよび PUT ステートメント
: 詳細については、33 章, “SAS 出力とそのカスタマイズについて: 基本” (599 ページ)を参照してください。

ログウィンドウ
: **ログ**ウィンドウの詳細については、“SAS ウィンドウの操作” (739 ページ)および “The SAS Log” (*SAS Language Reference: Concepts*)を参照してください。

動作環境に固有の情報
: 使用する動作環境の SAS ドキュメントには、SAS ログの表示形式や出力先に関する情報と、出力の出力先の指定に関する情報が含まれています。

SAS 環境
: SAS セッションの開始や、プログラムを実行できる別の方法の詳細については、“SAS 環境について” (719 ページ)および“Introduction to the SAS Windowing Environment” (*SAS Language Reference: Concepts*)を参照してください。

SAS ステートメント
: *SAS Statements: Reference* すべての動作環境にわたって機能する SAS ステートメントについての完全な参考情報を提供します。動作環境に固有のオプションについての情報は、使用する動作環境に関する SAS ドキュメントを参照してください。

SAS システムオプション
: *SAS System Options: Reference* すべての動作環境にわたって機能する SAS システムオプションについての完全な参考情報を提供します。動作環境に固有のオプションについての情報は、使用する動作環境に関する SAS ドキュメントを参照してください。

24 章

SAS 出力と SAS ログの出力指定

SAS 出力と SAS ログの出力指定について **383**
目的 383
前提条件 384
例で使用される入力ファイルおよび入力 SAS データセット **384**
PROC PRINTTO を使用した出力と SAS ログの出力先の指定 **385**
出力の代替の出力先の指定 385
SAS ログの代替の出力先の指定 386
デフォルト出力先の復元 387
SAS ウィンドウ環境での出力と SAS ログの格納 **387**
デフォルト出力先について 387
アウトプットウィンドウおよびログウィンドウのコンテンツの格納 388
バッチ(非対話型)環境でのデフォルトの出力先の再定義 **388**
デフォルト出力先の決定 388
デフォルト出力先の変更 388
構成ファイルについて 389
要約 **389**
PROC PRINTTO ステートメントオプション 389
SAS ウィンドウ環境コマンド 389
SAS システムオプション 390
詳細情報 **390**

SAS 出力と SAS ログの出力指定について

目的

SAS は、SAS 出力および SAS ログを異なる出力先へ出力指定するいくつかの方法を提供します。このセクションでは、次の SAS 言語要素の使用方法を学習します。

- PRINTTO プロシジャをプログラムやセッション内から使用して、DATA ステップ出力、SAS ログ、あるいはプロシジャ出力をデフォルト出力先から別の出力先へ送る
- FILE コマンドを SAS ウィンドウ環境で使用して、ログとアウトプットウィンドウのコンテンツをファイルに格納する
- PRINT=および LOG=システムオプションを SAS の起動時に使用して、SAS セッション全体のログと出力の出力先を再定義する

前提条件

このセクションを先に進む前に、次の機能と概念を理解している必要があります。

- DATA ステップまたは PROC ステップの出力の作成
- ログとプロシジャ出力の配置
- 外部ファイルの参照

例で使用される入力ファイルおよび入力 SAS データセット

このセクション中の例は、大学の試験スコアからのデータに基づいています。データには、2005 年から 2011 年までの大学入試クラスの試験スコアが含まれます。入力データの完全なリストについては、“UNIVERSITY_TEST_SCORES データセット” (793 ページ)を参照してください。次に入力ファイルの構造を示します。

```
Verbal f 2009 503
Verbal m 2010 507
Verbal f 2010 503
Verbal m 2011 509
Verbal f 2011 502
Math   m 2005 521
Math   f 2005 484
Math   m 2006 524
Math   f 2006 484
```

入力ファイルには、次に示す値が左から右へと含まれています。

- 試験の種類
- 生徒の性別
- 試験実施年
- 平均スコア

次のプログラムは、このセクションで使用するデータセットを作成します。

```
data test_scores;
   input Test $ Gender $ Year TestScore;
   datalines;
Verbal m 2005 504
Verbal f 2005 496
Verbal m 2006 504
Verbal f 2006 497
Verbal m 2007 501
Verbal f 2007 497
   ... more data lines ...
Math   f 2009 492
Math   m 2010 530
Math   f 2010 494
Math   m 2011 531
Math   f 2011 496
;
```

PROC PRINTTO を使用した出力と SAS ログの出力先の指定

出力の代替の出力先の指定

PRINTTO プロシジャを使用して、SAS プロシジャ出力を HTML 出力先から代替の場所にリダイレクトできます。その場所は次のとおりです。

- 永久ファイル
- SAS カタログエントリ
- 出力を非表示にする役割のダミーファイル

PROC PRINTTO の実行後は、別の PROC PRINTTO ステートメントを実行するまで、あるいはプログラムまたはセッションが終了するまで、プロシジャ出力はすべて代替の場所へ送られます。

プロシジャ出力のデフォルト出力先は、出力の処理方法を SAS でどのように構成するかによって決まります。詳細については、34 章, “SAS 出力とそのカスタマイズについて: The Output Delivery System (ODS)” (625 ページ) と 33 章, “SAS 出力とそのカスタマイズについて: 基本” (599 ページ)にある SAS 出力の説明を参照してください。

注: HTML 出力先を閉じて Output Delivery System (ODS)を使用した場合は、PROC PRINTTO はリダイレクトするべき出力を受け取りません。ただし、プロシジャの結果は常に、ODS で指定した出力先へ行きます。

PROC PRINTTO ステートメントで PRINT=オプションを使用して、プロシジャ出力を格納するファイルまたは SAS カタログの名前を指定します。ファイルを指定する場合は、ファイルの完全な名前を引用符で囲んで使用するか、ファイルにファイル参照名を使用します。(ファイル参照名やファイル名の詳細な情報については、“SAS ジョブでの外部ファイルの使用” (46 ページ)を参照してください。)SAS が出力ファイルの以前のコンテンツを置き換えるように、PROC PRINTTO ステートメントで NEW オプションを指定することもできます。それ以外の場合は、現在そのファイルにある任意の出力に SAS が出力結果を追加します。

代替ファイルを出力先として指定するには、プロシジャ出力を生成する PROC ステップより前のプログラムに PROC PRINTTO ステップを挿入します。次のプログラムは、PROC PRINT 出力の出力先として外部ファイルを指定します。

```
proc printto print='alternate-output-file' new;
run;

proc print data=test_scores;
   title 'Test Scores for Entering University Classes';
run;

proc printto;
run;
```

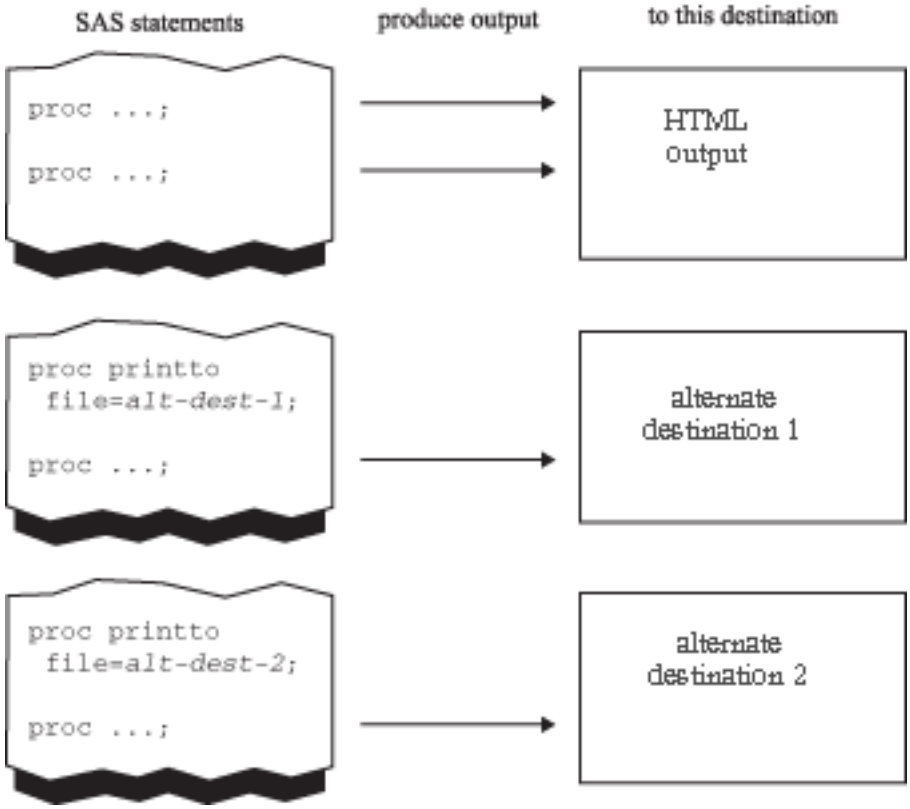
PROC PRINT ステップの実行後、*alternate-output-file* にはプロシジャ出力が含まれます。2 番目の PROC PRINTTO ステップではそのデフォルト場所へ出力をリダイレクトして戻します。

PRINTTO プロシジャは出力を生成しません。そのかわりに、後続プロシジャすべての結果の出力先を、別の PROC PRINTTO ステートメントが実行されるまで SAS に指示

します。したがって、PROC PRINTTO ステートメントは、出力先を指定するプロシジャに先行している必要があります。

次の図は、SAS が PROC PRINTTO を使用してプロシジャ出力の出力先を指定する方法を示しています。また、プログラム内で PROC PRINTTO を複数回使用して、SAS ジョブの異なるステップからの出力を別々のファイルに格納できます。

図 24.1 *PROC PRINTTO による出力先の指定*



SAS ログの代替の出力先の指定

PRINTTO プロシジャを使用して、SAS ログを代替の場所にリダイレクトできます。代替の場所は次のうちの 1 つです。

- 永久ファイル
- SAS カタログエントリ
- ログを非表示にするためのダミーファイル

PROC PRINTTO の実行後、別の PROC PRINTTO ステートメントを実行するまで、あるいはプログラムまたはセッションが終了するまで、ログは、永久外部ファイルあるいは SAS カタログエントリのいずれかへ送られます。

PROC PRINTTO ステートメントで LOG=オプションを使用して、ログを格納するファイルまたは SAS カタログの名前を指定します。ファイルを指定する場合は、ファイルの完全な名前を引用符で囲んで使用するか、ファイルにファイル参照名を使用します。SAS がファイルの前のコンテンツを置き換えるように、PROC PRINTTO ステートメントで NEW オプションを指定することもできます。それ以外の場合は、現在そのファイルにある任意のログに SAS がログを追加します。

次のプログラムは、SAS ログの出力先の代替ファイルを指定します。

```
proc printto log='alternate-log-file';
run;
```

PROC PRINT ステップの実行後、SAS ログは *alternate-log-file* に格納されます。

デフォルト出力先の復元

ログと出力の出力先をデフォルト出力先に戻すには、引数のない PROC PRINTO ステートメントを指定します。

```
proc printto;
run;
```

ログだけあるいはプロシジャ出力だけをデフォルト出力先に戻したい場合もあります。次の PROC PRINTTO ステートメントでは、ログのみデフォルト出力先に戻します。

```
proc printto log=log;
run;
```

次の PROC PRINTTO ステートメントでは、プロシジャ出力のみデフォルト出力先に戻します。

```
proc printto print=print;
run;
```

SAS ウィンドウ環境での出力と SAS ログの格納

デフォルト出力先について

SAS ウィンドウ環境では、ほとんどのプロシジャ出力のデフォルト出力先は**結果ビューア**ウィンドウに現れる HTML 出力です。ただし、Output Delivery System (ODS)を使用して出力の形式を変更できます。

シングルセッションでプロシジャを実行するごとに、SAS は既存の出力に追加します。結果を表示するには、次の方法のいずれかを使用します。

- **結果ビューア**ウィンドウをスクロールすると、生成された順番で出力が含まれ表示されます。
- **結果**ウィンドウを使用して、プロシジャ出力へのリンクであるポインタを選択します。

SAS ウィンドウ環境は ODS の特定の側面と対話して、出力をフォーマット、制御、および管理します。

SAS ウィンドウ環境では、SAS ログメッセージのデフォルト出力先は**ログ**ウィンドウです。プロシジャが実行されると、SAS は**ログ**ウィンドウで既存のログメッセージへログメッセージを追加します。**ログ**ウィンドウをスクロールして結果を見られます。ログメッセージを出力するには、PRINT コマンドを実行してください。**ログ**ウィンドウのコンテンツをクリアするには、CLEAR コマンドを実行してください。セッションが終了するとき、SAS は自動的にウィンドウをクリアします。

SAS ウィンドウ環境内では PRINTTO プロシジャを使用して、他の操作と同じように、ログメッセージやプロシジャ出力をデフォルト場所以外の場所に送れます。詳細については、“PROC PRINTTO を使用した出力と SAS ログの出力先の指定” (385 ページ)を参照してください。また、ODS を使用してもプロシジャ出力の出力形式を変更できます。

ODS の使用、プロシジャ出力の表示、およびプロシジャ出力の出力形式変更についての追加情報は、33 章, “SAS 出力とそのカスタマイズについて: 基本” (599 ページ)を参照してください。

アウトプットウィンドウおよびログウィンドウのコンテンツの格納

結果ビューアウィンドウや**ログ**ウィンドウのコンテンツのコピーをファイルに格納したい場合は、FILE コマンドを使用します。コマンド行で FILE コマンドにファイル名を続けて指定してください。

file '*file-to-store-contents-of-window*'

SAS には、ファイルを誤って上書きしないためビルトイン保護手段があります。不注意に既存のファイルを指定すると、ダイアログボックスが現れます。ダイアログボックスは、行動指針の選択を求め、情報を提供して、ファイルが誤って上書されるのを防ぎます。次のようなことが質問されます。

- ファイルのコンテンツの置き換え
- ファイルのコンテンツの追加
- FILE コマンドのキャンセル

バッチ(非対話型)環境でのデフォルトの出力先の再定義

デフォルト出力先の決定

バッチあるいは非対話型環境では、通常、SAS はプロシジャ出力をリストファイルに送り、SAS ログはログファイルに送ります。これらのファイルは、通常はインストールで定義され、SAS を起動するとき自動的に作成されます。使用しているサイトに関する質問がある場合は、オンサイトの SAS サポート担当者にご連絡ください。

デフォルト出力先の変更

プロシジャ出力のデフォルト出力先を再定義したい場合は、PRINT=システムオプションを使用します。SAS ログのデフォルト出力先を再定義したい場合は、LOG=システムオプションを使用します。これらのオプションの指定は初期化時のみです。

動作環境の情報
SAS システムオプションを使用する場合、出力先の指定方法は、使用する動作環境によって異なります。詳細については、各動作環境向けの SAS ドキュメントを参照してください。

初期化時に指定する必要があるオプションは構成オプションと呼ばれます。構成オプションでは次のようなことに影響があります。

- SAS システムの初期化
- ハードウェアインターフェイス
- オペレーティングシステムインターフェイス

他の SAS システムオプション(出力形式、ファイル操作、システム変数の使用、オブザベーションの処理に影響します)とは異なり、プログラム中の構成オプションを変更することはできません。構成オプションは、SAS を起動するとき、構成ファイル内、あるいは SAS コマンドで指定します。

構成ファイルについて

構成ファイルは、他の SAS システムオプションやそれらの設定だけでなく構成オプションも含んでいる特別なファイルです。SAS を起動するたびに、構成ファイルの設定が検査されます。ユーザーの動作環境に使用される SAS コマンドと同じ形式で、構成ファイルにオプションを指定することができます。たとえば、UNIX では、このファイルのコンテンツに次のことが含まれている場合があります。

```
WORK=WORK
SASUSER=SASUSER
EXPLORER
```

SAS は、構成ファイルに現れるとおりに、自動的にオプションを設定します。構成ファイルと SAS コマンドの両方でオプションを指定した場合、そのオプションは連結されます。SAS コマンドと構成ファイルで同じオプションを指定した場合は、SAS コマンドの設定がファイルの設定を上書きします。たとえば、SAS コマンドに NOEXPLORER オプションを指定することで、構成ファイルの EXPLORER オプションは上書きされ、**エクスプローラ**ウィンドウを表示せずにセッションを始めるよう SAS に指示したことになります。

要約

PROC PRINTTO ステートメントオプション

PROC PRINTTO <PRINT='*alternate-output-file*'> <LOG='*alternate-log-file*'> <NEW>;

PRINT='*alternate-output-file*'
場所を指定します。プロシジャ出力の代替の出力先になります。

LOG='*alternate-log-file*'
場所を指定します。SAS ログの代替の出力先になります。

NEW
現在のログあるいはプロシジャ出力がファイルの以前のコンテンツを上書きすることを指定します。

SAS ウィンドウ環境コマンド

CLEAR
指定されたウィンドウのコンテンツをクリアします。

FILE <*file-to-store-contents-of-window*>
指定したファイルへウィンドウのコンテンツのコピーを送ります。元のコンテンツはそのまま残ります。

PRINT
ウィンドウのコンテンツを書き込みます。

SAS システムオプション

LOG=*system-filename*
system-filename に指定されたファイルへ SAS ログのデフォルト出力先を再定義します。

PRINT=*system-filename*
system-filename に指定されたファイルへプロシジャ出力のデフォルト出力先を再定義します。

詳細情報

Output Delivery System
Output Delivery System の詳細なリファレンスドキュメントについては、*SAS Output Delivery System: User's Guide* を参照してください。

PROC PRINTTO
詳細なリファレンスドキュメントについては、“PRINTTO” (*Base SAS Procedures Guide*)を参照してください。

SAS 環境
SAS 動作方法、ウィンドウ環境での対話型処理の詳細については、40 章, “SAS 環境について” (719 ページ)、41 章, “SAS ウィンドウ環境の使用” (731 ページ) および 42 章, “SAS 環境のカスタマイズ” (771 ページ)を参照してください。

SAS ログ
SAS ログおよびプロシジャ出力の完全な参照情報については、*SAS Language Reference: Concepts* を参照してください。

SAS システムオプション
構成オプションを含む、SAS システムオプションの詳細については、*SAS System Options: Reference* を参照してください。

出力先の指定、PRINT=オプション、LOG=オプション、その他の SAS システムオプションに関する動作環境に固有の情報については、使用する動作環境の SAS ドキュメントを参照してください。

25 章

エラーの診断と回避

エラーの診断と回避について **391**
目的 391
前提条件 392
SAS Supervisor でのジョブのチェック方法について **392**
SAS でのエラーの処理方法について **392**
エラーの種類について **393**
SAS プログラミングエラー 393
構文エラー 393
実行時エラー 393
データエラー 394
セマンティックエラー 394
エラーの診断 **394**
このセクションでの例 394
構文エラーの診断 394
実行時エラーの診断 397
データエラーの診断 398
セマンティックエラーの診断 400
品質管理チェックリストの使用 **401**
詳細情報 **402**

エラーの診断と回避について

目的

このセクションでは、次の概念について学習することにより、プログラム内でのエラーの診断方法を学習します。

- SAS Supervisor がプログラムのエラーをチェックする方法
- エラーの種類を区別する方法
- ログにある NOTE メッセージ、警告メッセージおよびエラーメッセージを解釈する方法
- プログラムの開発時にチェックすること

前提条件

次のセクションで説明した概念を理解している必要があります。

- 3 章, “DATA ステップ処理について” (27 ページ)
- 4 章, “生データから作成する: 基本” (51 ページ)
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)
- 23 章, “SAS ログを使用した SAS セッションの分析” (363 ページ)

SAS Supervisor でのジョブのチェック方法について

発生したエラーをよく理解することで将来のエラーを回避できるようになるため、SAS Supervisor によるジョブのチェック方法を理解していることは重要です。SAS Supervisor は SAS プログラムの実行に責任を負う SAS の部分です。SAS プログラムの構文をチェックするため、SAS Supervisor は次を実施します。

- SAS ステートメントとデータを読み込み
- プログラムのステートメントを実行可能な機械コードあるいは中間コードに翻訳
- データセットを作成
- 要求があった場合は、SAS プロシジャの呼び出し
- エラーメッセージの書き込み
- ジョブを終了

SAS Supervisor は、DATA ステップと PROC ステップについて次の情報を認識しています。

- DATA ステップに存在できるステートメントの形式と種類
- PROC ステップに存在できるステートメントとオプションの種類

プログラムを処理するには、SAS Supervisor がすべての SAS ステートメントをスキャンし、各ステートメントを解釈してワードにします。各ワードは別々に処理されます。ステップにあるワードがすべて処理されたとき、そのステップが実行されます。SAS Supervisor がエラーを検出した場合は、その位置にエラーフラグを設定して、ログに説明を書き込みます。SAS Supervisor は、認識しないものはすべてエラーとします。

SAS でのエラーの処理方法について

SAS がエラーを検出した場合、通常はログでエラーを強調するか、あるいはエラーが検出されたポイントを強調して、エラーは番号で識別します。各番号は各エラーメッセージに一意に関係付けられています。その後、SAS は構文チェックモードに入り、残りのプログラムステートメントを読み込み、その構文をチェックして、必要な場合は追加のエラーを強調します。

バッチや非対話型プログラムでは、DATA ステップステートメントに 1 つエラーがあると、プログラムの残りをチェックするために SAS は構文チェックモードのままになります。外部ファイルあるいは SAS データセットを作成する DATA ステップや PROC ステップをそれ以上は実行しません。プロシジャが SAS データセットから読み込む場合は

0 オブザベーションで実行され、プロシジャが SAS データセットを読み込まない場合は正常に実行されます。PROC ステップの構文エラーは通常そのステップだけに影響します。ステップの終わりに、SAS は、検出した各エラーに関するメッセージをログに書き込みます。

エラーの種類について

SAS プログラミングエラー

SAS がプログラムをコンパイルして実行する際、次の種類のエラーが発生する可能性があります。

- 構文
- 実行時
- データ
- セマンティック

構文エラー

構文エラーは、プログラムの SAS ステートメントに間違いがあります。プログラムステートメントが SAS 言語のルールに合わない場合に発生します。各 DATA ステップおよび PROC ステップをコンパイルするとき、SAS は構文エラーを検出します。構文エラーの種類をいくつか次に示します。

- スペルミスがある SAS キーワード
- 引用符の不一致
- 欠損しているか無効な区切り文字
- 無効なステートメントまたはデータセットオプション

実行時エラー

実行時エラーは、サブミットしてもプログラムが実行に失敗します。ほとんどの実行時エラーは深刻ではなく、SAS ログに NOTE は生成しますが、プログラムは実行を完了できます。ただし、より重大なエラーの場合は、SAS はエラーメッセージを出してすべての処理を停止します。実行時エラーの種類をいくつか次に示します。

- 関数に対して無効な引数
- 0 による除算などの無効な算術演算
- BY グループ処理対象としては間違った順序のオブザベーション
- 存在しない配列メンバの参照
- データ行と一致しない INPUT ステートメント
- INFILE ステートメントでの正しくない参照(たとえば、スペルミス、外部ファイルの不正確な記載など)

データエラー

データエラーは実行時エラーの一種です。データエラーは、SAS プログラムで分析している生データに無効な値が含まれている場合に発生します。たとえば、データが文字データのときに、INPUT ステートメントで数値変数を指定した場合、データエラーが発生します。データエラーによってプログラムが停止することはありません。そのかわり、SAS ログに NOTE が生成されます。データエラーの種類をいくつか次に示します。

- データ値が実際は文字である場合に、変数を数値として定義
- 欠損値の演算を実行した結果、欠損値を生成
- ファイル内で変数が正しい位置にない場合に、INPUT ステートメントで変数を読み取り
- 合計ステートメントで文字変数を使用

セマンティックエラー

セマンティックエラーは別種の実行時エラーです。SAS ステートメントの形式は正しいが、特定の使用法において有効ではない要素がある場合に発生します。セマンティックエラーの種類をいくつか次に示します。

- ある関数に関して間違った数の引数を指定
- 文字変数のみ有効な箇所に数値変数名を使用
- 無効な配列参照を使用
- 未割り当てのライブラリ参照名を使用

エラーの診断

このセクションでの例

このセクションのプログラムの多くは、SAS ログのエラーを説明するために大学の試験スコアを使用しています。このセクションのその他のプログラムは他のデータを使用しています。

構文エラーの診断

SAS で構文エラーが検出される場合

SAS Supervisor は各ステップをコンパイルするときに構文エラーを検出し、SAS は次のことを実施します。

- ログにワード ERROR を書き込む
- エラーの場所を識別
- エラーの説明を書き込む

例: セミコロンの欠損とキーワードのスペルミス

次のプログラムでは、CHART プロシジャがデータの分析に使用されます。DATA ステートメントでセミコロンが省略され、キーワード INFILE にスペルミスがある点に注意してください。

```
libname out 'your-data-library';

data out.error1
   infill 'your-input-file';
   input test $ gender $ year TestScore;
run;

proc chart data=out.error1;
   hbar test / sumvar=TestScore type=mean group=gender discrete;
run;
```

次の出力は、2 つの構文エラーの結果を示します。

ログ 25.1 *構文エラーの診断: セミコロンの欠損とキーワードのスペルミス*

```
11   libname out 'your-data-library'; NOTE: Libref OUT was successfully assigned
as follows: Engine:        V9 Physical Name: your-data-library 12 13   data
out.error1 14      infill 'your-input-file'; 15      input test $ gender $ year
TestScore; 16   run; ERROR: No DATALINES or INFILE statement.ERROR: Extension
for physical file name "your-input-file" does not correspond to a valid member
type.NOTE: The SAS System stopped processing this step because of
errors.WARNING: The data set OUT.ERROR1 may be incomplete.When this step was
stopped there were 0 observations and 4 variables.WARNING: Data set OUT.ERROR1
was not replaced because this step was stopped.WARNING: The data set WORK.INFILL
may be incomplete.When this step was stopped there were 0 observations and 4
variables.WARNING: Data set WORK.INFILL was not replaced because this step was
stopped.NOTE: DATA statement used (Total process time): real time           0.00
seconds cpu time            0.00 seconds 17 18   proc chart data=out.error1;
19      hbar test / sumvar=TestScore type=mean group=gender discrete; 20   run;
NOTE: No observations in data set OUT.ERROR1.NOTE: PROCEDURE CHART used (Total
process time): real time           0.00 seconds cpu time            0.00 seconds
```

ログが示すように、SAS はキーワード DATA を認識して DATA ステップを処理しようとします。DATA ステートメントはセミコロンで終わることが必要なので、SAS は、INFILL がデータセット名で、2 つのデータセット(OUT.ERROR1 と WORK.INFILL)を作成するとみなします。SAS は INFILL をデータセットの名前と考えるため、それを別のステートメントの一部とは認識せず、したがって、スペルミスを検出しません。引用符で囲んだ文字列が DATA ステートメントで無効なので、SAS はここで処理を停止し、どちらのデータセットのオブザベーションも作成しませんでした。

SAS は、このセクションで以前に概略を説明したステップに従い、含まれているステートメントに論理的に基づいてプログラムを実行しようとします。SAS はセミコロンでステートメントが終了するまで DATA ステートメントは有効であると見なすため、第 2 の構文エラー(スペルミスがあるキーワード)は認識されません。重要なポイントは、同じプログラムで複数のエラーがある場合、プログラムを初めて実行するときにそのすべてが検出されるとは限らないことです。また、エラーが単独で発生した場合とグループ内での場合とではフラグの設定され方が異なる可能性もあります。1 つの修正が別のエラーを明らかにしたり、最少でもログの説明が変わったりすることがあります。

このポイントを説明するために、前のプログラムを DATA ステートメントにセミコロンを加えて再び実行します。スペルミスがあるキーワードを修正しようとして、別のスペルエラーが次のように発生します。

```
libname out 'your-data-library';
```

```
data out.error2;
   unfile 'your-input-file';
   input test $ gender $ year TestScore;
run;

proc chart data=out.error2;
   hbar test / sumvar=TestScore type=mean group=gender discrete;
run;
```

ログ 25.2 *構文エラーの診断: キーワードのスペルミス*

```
19   libname out 'your-data-library'; NOTE: Libref OUT was successfully assigned
as follows: Engine:        V9 Physical Name: your-data-library 20 21   data
out.error2; 22      unfile 'your-input-file'; ------ 180 ERROR 180-322:
Statement is not valid or it is used out of proper order.23      input test $
gender $ year TestScore; 24   run; ERROR: No DATALINES or INFILE statement.NOTE:
The SAS System stopped processing this step because of errors.WARNING: The data
set OUT.ERROR2 may be incomplete.When this step was stopped there were 0
observations and 4 variables.NOTE: DATA statement used (Total process time):
real time           0.01 seconds cpu time            0.01 seconds 25 26   proc
chart data=out.error2; 27      hbar test / sumvar=TestScore type=mean
group=gender discrete; 28   run; NOTE: No observations in data set
OUT.ERROR2.NOTE: PROCEDURE CHART used (Total process time): real time
0.01 seconds cpu time            0.01 seconds
```

セミコロンが加えられたので、SAS は 1 つのデータセットだけを作成しようとします。引き続き、SAS は前と同様に SAS ステートメントを読み込み、複数の同じメッセージを発行します。ただし、今回、SAS は UNFILE ステートメントを無効あるいは順序が不適切であるとみなし、データセットのオブザベーションを作成しません。

この例が INFILE という正しいスペルでも、INFILE ステートメントでファイル名のスペルを間違えて再実行した場合は、実行時にエラーが検出され、データは読み込まれません。

例: SET ステートメントでのセミコロンの欠損

次の例では、セミコロンなしで SET ステートメントを使用した場合の SAS ログを示します。最初の DATA ステップでは、データセット INSURANCE が作成されます。2 番目の DATA ステップの SET ステートメントでは、INSURANCE が入力として使用されます。SAS ログに、エラーが検出されたことが示されます。

```
data insurance;
   input PolicyNum $ Name $ Amount 23-27 District $;
   datalines;
4356 Susan Bellingham 45000 North
2678 James Hastings   35000 West
4967 Jan Spiro        49000 North
1367 Robert Hernandez 63000 South
7366 Walter Peters    66000 East
;
run;

data policy;
   set insurance
run;

proc print data=policy;
run;
```

ログ 25.3 SET ステートメントでのセミコロンの欠損についてのログ出力

```
44   data insurance; 45      input PolicyNum $ Name $ Amount 23-27 District $;
46      datalines; NOTE: The data set WORK.INSURANCE has 5 observations and 4
variables. 1 NOTE: DATA statement used (Total process time): real time
0.00 seconds cpu time            0.00 seconds 52   ; 53   run; 54 55   data
policy; 56      set insurance 57   run; ERROR: File WORK.RUN.DATA does not
exist.  2 58 NOTE: The SAS System stopped processing this step because of
errors.  3 WARNING: The data set WORK.POLICY may be incomplete.When this step
was stopped there were 0 observations and 4 variables.  4 WARNING: Data set
WORK.POLICY was not replaced because this step was stopped.NOTE: DATA statement
used (Total process time): real time           0.01 seconds cpu time
0.01 seconds 59   proc print data=policy; 60   run; NOTE: No observations in
data set WORK.POLICY.  5 NOTE: PROCEDURE PRINT used (Total process time): real
time           0.01 seconds cpu time            0.01 seconds
```

次のリストは、前述のログの番号付き項目に対応しています。

1 SAS で INSURANCE データセットが正常に作成されます。

2 2 番目の DATA ステップの SET ステートメントが読み取られると、SAS でエラーが検出されます(ステートメントの終わりを示すセミコロンの欠損)。SAS では、セミコロンを検出するまで、次の行の読み取りが続行されます。

3 NOTE で、プログラムが処理を停止したことが示されます。

4 警告メッセージで、POLICY データセットにオブザベーションがないことが示されます。

5 POLICY データセットが空のため、PRINT プロシジャでは出力が生成されません。

プログラムエラーを修正するには、SET ステートメントの後にセミコロンを追加します。プログラムを再実行してください。

実行時エラーの診断

SAS で実行時エラーが検出される場合

実行時には数種類のエラーが検出されます。言語要素は正しいが、その要素が特定の使用法で有効ではない可能性がある場合に、エラーが発生します。

SAS Supervisor が実行時エラーを検出した場合、次を実施します。

- エラーの重要度に応じて、NOTE、警告またはエラーメッセージをログに書き込み
- 場合によっては、プログラムデータベクトルに格納される値をリスト
- エラーの重要度に応じて、処理を継続、または停止

例: 入力ファイルのスペルミス

次のプログラムが INFILE という正しいスペルでも、INFILE ステートメントでファイル名のスペルを間違えて実行した場合は、実行時にエラーが検出され、データは読み込まれません。

```
   /* misspelled filename in the INFILE statement */
libname out 'your-data-library';

data out.error3;
   infile 'an-incorrect-filename';
   input test $ gender $ year TestScore;
run;
```

```
proc chart data=out.error3;
   hbar test / sumvar=TestScore type=mean group=gender discrete;
run;
```

ログ 25.4 *入力ファイルのスペルミスについてのログ出力*

```
29      /* misspelled filename in the INFILE statement */ 30   libname out 'your-
data-library'; NOTE: Libref OUT was successfully assigned as follows:
Engine:        V9 Physical Name: your-data-library 31 32   data out.error3;
33      infile 'your-input-file'; 34      input test $ gender $ year TestScore
@@; 35   run; ERROR: Physical file does not exist, your-input-file.NOTE: The SAS
System stopped processing this step because of errors.WARNING: The data set
OUT.ERROR3 may be incomplete.When this step was stopped there were 0
observations and 4 variables.NOTE: DATA statement used (Total process time):
real time           0.00 seconds cpu time            0.00 seconds 36 37   proc
chart data=out.error3; 38      hbar test / sumvar=TestScore type=mean
group=gender discrete; 39   run; NOTE: No observations in data set
OUT.ERROR3.NOTE: PROCEDURE CHART used (Total process time): real time
0.00 seconds cpu time            0.00 seconds
```

ログが示すように、SAS はファイルを見つけられません。SAS は、エラーのため処理を停止し、データセットにはオブザベーションを作成しません。

データエラーの診断

SAS でデータエラーが検出される場合

データエラーは、データ値が無効な場合に発生します。SAS が実行中にデータエラーを検出した場合、処理は続行され、次の情報が SAS ログに書き込まれます。

- エラーを記述するメッセージ
- 入力バッファに格納される値のリスト
- プログラムデータベクトルに格納される値のリスト

プログラムデータベクトルにリストされた値には、SAS で自動生成された次の 2 つの変数が含まれることに注意してください。

_N_
: DATA ステップが反復される回数をカウント。

ERROR
: DATA ステップの実行中にエラーが発生したかどうかを示す値。エラーを検出しない場合に変数_ERROR_へ割り当てられる値は 0 で、エラーを検出した場合の値は 1 になります。

これらの自動変数は、一時的に割り当てられるものであり、データセットと一緒には格納されません。

例: 無効な生データ

次のプログラムでは、INPUT ステートメントで記述されたカラムに正しく配置されていない生の入力データが使用されています。6 番目のデータ行がスペース 2 つ分右にシフトし、残りの行は、1 番目以外、スペース 1 つ分右にシフトしています。プログラムでは、変数値の書き込み方法を決定する出力形式が使用されています。

```
   /* data in wrong columns */
proc format;
   value xscore . ='accurate scores unavailable';
run;
```

```
data out.error4;
   input test $ 1-8 gender $ 18 year 20-23 TestScore 25-27;
   format TestScore xscore.;
   datalines;
verbal            m 2008 463
 verbal            f 2008 468
 verbal            m 2011 459
 verbal            f 2011 461
 math              m 2008 514
  math              f 2008 467
 math              m 2011 509
 math              f 2011 509
 ;

proc print data=out.error4;
   title 'Viewing Incorrect Output';
run;
```

次の出力は、SAS プログラムの結果を示しています。

図 25.1 *生データのデータエラーの検出*

Viewing Incorrect Output

Obs	test	gender	year	TestScore
1	verbal	m	2008	463
2	verbal		200	46
3	verbal		201	45
4	verbal		201	46
5	math		200	51
6	math		.	accurate scores unavailable
7	math		201	50
8	math		201	50

このプログラムは出力を生成しますが、期待した出力にはなりません。最初のオブザベーションは正しく現れますが、後続のオブザベーションには次の問題があります。

- 変数 Gender の値が欠損しています。
- 変数 Year では、6 番目のオブザベーションで欠損値であることが示され、それ以外は最初の 3 桁だけが表示されています。
- 変数 TestScore では、6 番目のオブザベーションで欠損値に割り当てられた値が表示され、それ以外は値の 3 桁目が欠損しています。

次の SAS ログでは、プログラム処理について説明しています。

ログ 25.5 *データエラーの診断*

```
61      /* data in wrong columns */ 62   proc format; 63      value
xscore .='accurate scores unavailable'; NOTE: Format XSCORE has been output.64
run; NOTE: PROCEDURE FORMAT used (Total process time): real time           0.01
seconds cpu time            0.00 seconds 65 66   data out.error4; 67      input
test $ 1-8 gender $ 18 year 20-23 TestScore 25-27; 68      format TestScore
xscore.; 69      datalines; NOTE: Invalid data for year in line 75 20-23.NOTE:
Invalid data for TestScore in line 75 25-27.RULE:      ----+----1----+----2----
+----3----+----4----+----5----+----6----+----7----+----8----+- 75
math             f 2008 467 test=math gender=  year=.TestScore=accurate scores
unavailable _ERROR_=1 _N_=6 NOTE: The data set OUT.ERROR4 has 8 observations and
4 variables.NOTE: DATA statement used (Total process time): real time
0.01 seconds cpu time            0.01 seconds 78    ; 79 80   proc print
data=out.error4; 81      title 'Viewing Incorrect Output'; 82   run; NOTE: There
were 8 observations read from the data set OUT.ERROR4.NOTE: PROCEDURE PRINT used
(Total process time): real time           0.03 seconds cpu time            0.03
seconds
```

エラーにはフラグが設定され、75 行目に変数 YEAR の無効データが含まれているという最初のメッセージから始まっています。RULE は、入力データがログに書かれたことを示します。SAS は、プログラムデータベクトルに格納される値をログにリストします。ログにある次の行で、SAS がエラーを検出したことが示されています。

```
NOTE: Invalid data for year in line 75 20-23.
NOTE: Invalid data for TestScore in line 75 25-27.
RULE:      ----+----1----+----2----+----3----+----4----+----5----+----6
----+----7----+----8----+-
16           math             f 2008 467
test=math gender=  year=. TestScore=accurate scores unavailable _ERROR_=1 _N_=6
```

変数 Gender と Year に欠損値が示されます。ログにある NOTE では、入力の 6 行目にエラーが含まれていることを示します。_ERROR_自動変数では、エラーが発生したことが示され、_N_自動変数では、DATA ステップの 6 回目の反復にエラーがあると識別されます。

プログラムをデバッグするために、生データの位置を変えるか、または INPUT ステートメントを書き直すことができます。1 行目を除くすべてのデータ行が少なくとも 1 スペース分右にシフトしていたことを思い出してください。変数 Test は影響を受けませんでしたが、変数 Gender は、最初のオブザベーションを除いて、指定されたフィールドから完全に削除されました。したがって、SAS では、2 番目から 8 番目までのオブザベーションについて変数 Gender が欠損値として読み取られます。6 番目のオブザベーションでは、データが 1 スペース余分に右へシフトしていて、Gender のための文字値は、数値変数 Year のためのフィールドの一部を占有しました。SAS が無効データを検出した場合、その値を欠損値として扱うだけでなく、データが無効であるという NOTE をログに書き込みます。覚えておくべき重要な点は、SAS では、提供するつもりのものではなく、提供された情報だけを使用できるということです。有効な出力のためには、入力データが有効である必要があります。

セマンティックエラーの診断

SAS でセマンティックエラーが検出される場合

言語要素は正しいが、その要素が特定の使用法で有効ではない可能性がある場合に、セマンティックエラーが発生します。

例: 関数での引数の欠損

次の例では、SAS でセマンティックエラーが発生した場合の SAS ログを示します。例では、COUNT 関数を使用して、部分文字列(この場合は"th")が文字列"This is the thistle."内に出現する回数を求めます。COUNT 関数には 2 つの引数があります。最初の引数には検索範囲の文字列が含まれます。2 番目の引数には検索対象の文字列が含まれます。次の例のように、COUNT 関数で引数を 1 つしか使用しなかった場合、エラーが発生します。

```
data _null_;
   x='This is the thistle.';
   y='th';
   occurrences=count(x);
   put occurrences=;
run;
```

ログ 25.6 *関数での引数の欠損*

```
249  data _null_; 250     x='This is the thistle.'; 251     y='th'; 252
occurrences=count(x); ----- 71 ERROR 71-185: The COUNT function call does not
have enough arguments.253     put occurrences=; 254  run; NOTE: The SAS System
stopped processing this step because of errors.NOTE: DATA statement used (Total
process time): real time           0.01 seconds cpu time            0.01 seconds
```

COUNT 関数の引数の 1 つが欠損しているため、エラーが発生しました。エラーを修正するには、COUNT に 2 番目の引数を追加します。(COUNT 関数の構文は **count(*string, substring*);**です。)プログラムを再実行してください。SAS で、文字列から"th"と一致する文字列がすべて検索されます。COUNT 関数では"th"の発生が 2 回識別されるので、Occurrences の値は 2 です。文字列内の最初の語は大文字であるため、"th"と一致しないことに注意してください。

品質管理チェックリストの使用

プログラムを開発するときいくつかの基礎的なガイドラインに従うことで、共通のエラーを回避できます。次のチェックリストを使用して、プログラムをサブミットする前に共通の誤りにフラグを立てて修正してください。

- プログラムの構文をチェック。特に、次の項目が正しいか確認してください。
  - すべての SAS ステートメントはセミコロンで終了する必要があります。セミコロンを省略したり、偶然に間違った文字を入力したりしていないか確認してください。
  - 引用符の開始と終了を正しく対応させる必要があります。一重引用符か二重引用符のいずれかを使用できます。
  - ほとんどの SAS ステートメントは SAS キーワードで始まります。(例外は割り当てステートメントおよび合計ステートメントです。)キーワードをスペルミスや省略していないことを確認してください。
  - すべての DO ステートメントおよび SELECT ステートメントは、その後に END ステートメントが必要です。
- プログラム順序をチェック。

  SAS は、通常、DATA ステップのステートメントを 1 行ずつ、それが現れる順番で実行します。DATA ステップの実行後、SAS は次のステップに移り、同じ方法で続行します。SAS が SAS ステートメントを適切に実行できるように、すべての SAS ス

テートメントが順序立って現れることを確認してください。たとえば、INFILE ステートメントを使用する場合は、INPUT ステートメントより先行する必要があります。

また、必ず RUN ステートメントでステップを終了してください。RUN ステートメントはそれ以前にあるステップを実行するため、プログラムの最後にはこれが特に重要です。

- INPUT ステートメントおよびデータをチェック。

  SAS はすべての変数を文字か数値のいずれかとして分類します。文字あるいは数値としての INPUT ステートメントでの割り当ては、データにある変数の実際の値に対応していることが必要です。また、SAS では、リスト、カラム、フォーマット入力、ネーム入力が許されます。INPUT ステートメントで指定する入力方法は、生データの実際の配置に一致することが必要です。

詳細情報

INFILE ステートメントオプション
"INFILE Statement" (*SAS Statements: Reference*) には、デバッグツールとして、INFILE ステートメントでの MISSOVER オプションと STOPOVER オプションの使用に関する情報が含まれています。

MISSOVER オプションは、すべての INPUT ステートメント変数に対して現在行には値がないことを検出した場合、SAS プログラムがリスト入力で行の終わりを超えて値を読み取らないようにします。SAS は、その後、現在の入力行に値がなかった変数に対して欠損値を割り当てます。

STOPOVER オプションは、リスト入力を使用する INPUT ステートメントがそこにあるすべての変数に対する値を検出していないのに現在のレコードの終わりに到達した場合、DATA ステップの処理を停止します。SAS は、その後、_ERROR_を 1 に設定し、データセットの構築を停止して、不完全なデータ行を書き込みます。

PUT ステートメント
PUT ステートメントの詳細については、"PUT Statement" (*SAS Statements: Reference*)を参照してください。

プログラムデータベクトルと入力バッファ
"DATA ステップ処理について" (27 ページ)および"生データについて" (52 ページ)
には、プログラムデータベクトルと入力バッファについての情報が含まれています。

SAS ログ
23 章, "SAS ログを使用した SAS セッションの分析" (363 ページ)
には、SAS ログとその構造についての情報が含まれています。"The SAS Log" (*SAS Language Reference: Concepts*)には、SAS ログについての詳細情報が含まれています。

SAS 出力
"SAS Output" (*SAS Language Reference: Concepts*) には、SAS 出力についての詳細情報が含まれています。

SAS セッション
SAS セッションに関してより多くの情報を他のセクションで提供しています。"SAS ログを使用した SAS セッションの分析について" (364 ページ)では、ログ構造、ログ内のメッセージの種類、および出力に関するログについて説明しています。

26 章

プログラムの論理エラーの検出

プログラムの論理エラーの検出 **403**
目的 403
前提条件 404

DATA ステップデバッガの使用 **404**

基本的な使用 **404**
デバッガセッションの操作方法 404
ウィンドウの使用 405
コマンドの入力 405
式の処理 405
コマンドをファンクションキーに割り当てる 406

マクロ機能とデバッガの併用 **406**
デバッグツールとしてマクロを使用する 406
マクロを用いてカスタマイズしたデバッグコマンドを作成する 406
マクロで生成された DATA ステップのデバッグ 406

例 **407**
例 1: 結果が出力されないときのシンプルな DATA ステップのデバッグ問題の発見 407
例 2: 出力形式の処理 412
例 3: DO ループのデバッグ 417
例 4: 変数のフォーマット指定された値の検証 418

プログラムの論理エラーの検出

目的

DATA ステップデバッガは、ウィンドウとコマンドグループから成り、SAS プログラムの論理エラー(場合によってはデータエラー)を対話的に特定するのに役立ちます。構文エラーとは違い、論理エラーでプログラムの実行が停止することはありません。そのかわり、論理エラーは、プログラムで予期しない結果が出る原因となります。たとえば、在庫を追跡する DATA ステップを作成したときに、プログラムで、在庫切れでも倉庫が満杯であることが示された場合は、プログラムに論理エラーがあります。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 3 章, “DATA ステップ処理について” (27 ページ)
- 7 章, “DATA ステップ処理の基礎知識” (109 ページ)

DATA ステップデバッガの使用

コマンドを発行することによって、DATA ステップステートメントを 1 つずつ実行したり、一時停止してウィンドウに結果の変数値を表示したりできます。表示される結果を検証して、論理エラーの箇所を特定できます。デバッガが対話型であるため、1 つのデバッグセッションで必要に応じて何度でもコマンド発行および結果観測の処理を繰り返すことができます。デバッガを起動するには、DATA ステートメントに DEBUG オプションを追加し、プログラムを実行します。

DATA ステップデバッガでは、次のタスクを実行できます。

- ステートメントを1つずつまたはグループで実行
- 1 つ以上のステートメントの実行をバイパスする
- DATA ステップの反復中または指定の条件において、特定のステートメントで実行を一時停止し、コマンドで実行の再開
- 選択した変数の値をモニタし、値変更時点で実行を一時停止
- 変数の値を表示し、新しい値を付与
- 変数の属性を表示
- 各デバッガコマンドのヘルプを表示
- ファンクションキーにデバッガコマンドを割り当てる
- マクロ機能を使用して、カスタマイズデバッガコマンドを生成

次のセクションでは、使用法の情報と例が提供されます。

デバッガコマンドとその説明のリストについては、付録 2, “DATA ステップデバッガコマンド” (801 ページ)を参照してください。

基本的な使用

デバッガセッションの操作方法

DEBUG オプション付きの DATA ステップをサブミットすると、SAS はステップをコンパイルし、デバッガウィンドウを表示し、実行開始のデバッガコマンドを入力するまで一時停止します。たとえば、GO コマンドの実行を開始した場合、SAS は DATA ステップの各ステートメントを実行します。DATA ステップの特定の行で実行を一時停止させたい場合、BREAK コマンドを使って選択のステートメントにブレークポイントを設定します。そして GO コマンドを発行します。GO コマンドはブレークポイントに到達するまで、実行を開始または再開します。

DATA ステップを1ステートメントずつまたは数ステートメントずつ実行するには、STEP コマンドを使用します。デフォルトでは STEP コマンドは ENTER キーにマッピングされています。

デバッグセッションでは、DATA ステップのステートメントはデバッグセッション外で実行される回数と同じだけ実行されます。最後の反復終了後、DEBUGGER LOG ウィンドウにメッセージが表示されます。

デバッグセッションでは、DATA ステップの実行終了後は、DATA ステップ実行の再開はできません。SAS セッションで DATA ステップを再度サブミットしてください。ただし、実行終了後に変数の最後の値を検証することはできます。

一回に 1 つの DATA ステップのみデバッグできます。デバッガは DATA ステップでのみ使用でき、PROC ステップでは使用できません。

ウィンドウの使用

DATA ステップデバッガには DEBUGGER LOG ウィンドウと DEBUGGER SOURCE の2つのメインのウィンドウがあります。DEBUG オプションをつけて DATA ステップを実行するとウィンドウが表示されます。

The DEBUGGER LOG ウィンドウは発行するデバッガーコマンドとその結果を記録します。最後の行はデバッガコマンドを発行するデバッガコマンドラインです。デバッガコマンドラインには、より大きい(>)プロンプトが表示されています。

DEBUGGER SOURCE ウィンドウには、デバッグしている DATA ステップを構成する SAS ステートメントが表示されます。ウィンドウはプログラムのデバッグ中に DATA ステップのどの位置かを表示します。ウィンドウでは、SAS ステートメントは SAS ログと同じように行番号を持ちます。

コマンドラインにウィンドウ環境のコマンドを入力できます。ファンクションキーでコマンドを実行することもできます。

コマンドの入力

コマンドとその説明のリストは、付録 2, “DATA ステップデバッガコマンド” (801 ページ)を参照してください。

デバッガコマンドラインに DATA ステップデバッガコマンドを入力します。コマンド入力の際には次のルールに従ってください。

- コマンドは一行に入力します(DO グループは除く)。
- DO グループは複数行にわたることもあります。
- 複数のコマンドを入力する場合は、セミコロンでコマンドを区切ります。

```
examine _all_; set letter='bill'; examine letter
```

式の処理

デバッガ式には“SAS Operators in Expressions” (*SAS Language Reference: Concepts*) に説明されているすべての SAS 演算子が使用できます。デバッガ式に関数を含むことはできません。

デバッガ式は1行内でおさまるようにしてください。式は2行にわたることはできません。

コマンドをファンクションキーに割り当てる

ファンクションキーにデバッガコマンドを割り当てるには、KEYS ウィンドウを開きます。カーソルを割り当てるファンクションキーの定義カラムにおき、DSD のあとにコマンドを入力します。ファンクションキーに複数のコマンドを割り当てるには、コマンドを(セミコロンで区切って)引用符で囲みます。変更を保存するのを忘れないようにしてください。ファンクションキーに割り当てられたコマンドの例です。

- `dsd step3`
- `dsd 'examine cost saleprice; go 120;'`

マクロ機能とデバッガの併用

デバッグツールとしてマクロを使用する

デバッガの DEBUGGER LOG コマンドラインからマクロを起動するには、SAS マクロ機能を使います。マクロを定義して、デバッガコマンドラインから %LET 等のマクロプログラムステートメントを使用できます。

マクロは一連のデバッガコマンドを保存するのに便利です。DEBUGGER LOG コマンドラインでマクロを実行すると、一連のデバッガコマンドが生成されます。マクロと引数を使用して、様々な状況において異なる一連のデバッガコマンドを生成することもできます。

マクロを用いてカスタマイズしたデバッグコマンドを作成する

DEBUGGER LOG のコマンドラインにマクロを定義すればカスタマイズのデバッグコマンドが作成できます。そしてコマンドラインからマクロを起動します。たとえば、変数 COST を検証して、5 つのステートメントを実行して、そして変数 DURATION を検証するには、次のマクロを定義します(このケースでは、マクロは EC と呼ばれています)。この例では、EXAMINE コマンドにはエリアスが使われています。

```
%macro ec; ex cost; step 5; ex duration; %mend ec;
```

コマンドを発行するには、DEBUGGER LOG コマンドラインからマクロ EC を起動します。

```
%ec
```

DEBUGGER LOG は COST の値を表示し、次の 5 ステートメントを実行し、そして DURATION の値を表示します。

注: DEBUGGER LOG コマンドラインでマクロを定義する場合、現在のデバッグセッション中でだけ、そのマクロを使用できます。なぜなら、マクロが永久保存されていないからです。永久保存されたマクロを作成するには、プログラムエディタを使用します。

マクロで生成された DATA ステップのデバッグ

マクロを使って DATA ステップを生成できますが、マクロ生成された DATA ステップのデバッグは困難な場合があります。SAS ログはマクロのコピーは表示しますが、マクロが生成した DATA ステップは表示しません。このときに DEBUG オプションを使用する

と、マクロが生成する文字列はデバッガーに連続するストリームとして認識されます。結果、実行が一時停止できるラインブレークがない状態になります。

マクロで生成される DATA ステップをデバッグするには、

1. プログラムの実行時に MPRINT と MFILE システムオプションを使用する。
2. 既存の外部ファイルにファイル参照名 MPRINT を割り当てる。MFILE でプログラムの出力を外部ファイルに誘導する。プログラムを再実行すると、ファイルに最新の出力が前回の出力に追加されます。
3. SAS セッションでマクロを起動する。
4. 外部ファイルを開くには、プログラムエディタウィンドウで INCLUDE コマンドを発行するか、ファイルメニューを使用します。
5. DATA ステートメントに DEBUG オプションを追加すると、デバッグセッションが開始します。
6. 論理エラーを発見したら、そのステートメントを生成する部分のマクロを修正します。

例

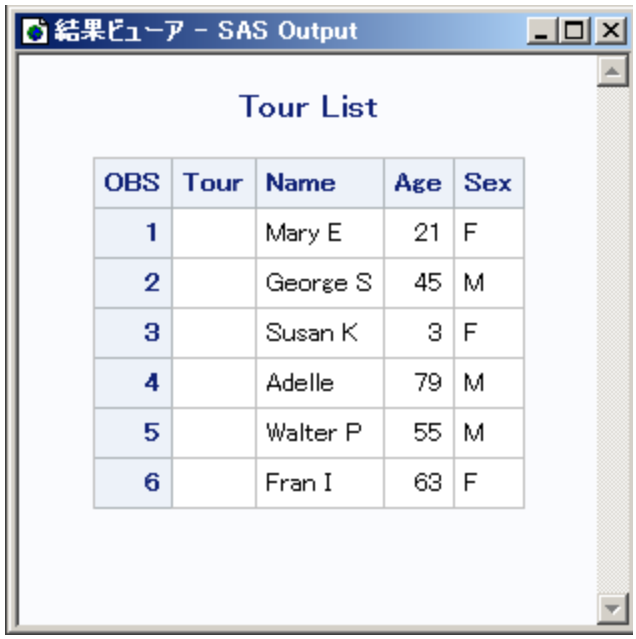
例 1: 結果が出力されないときのシンプルな DATA ステップのデバッグ問題の発見

問題の発見

このプログラムは旅行ツアーのグループの情報を作成します。このデータファイルには2種類のレコードが含まれています。1つはツアーコードで、もう1つは顧客情報です。プログラムは顧客のツアー番号、名前、年齢、性別のリストを作成します。

```
/* first execution */
data tours (drop=type);
input @1 type $ @;
if type='H' then do;
input @3 Tour $20.;
return;
end;
else if type='P' then do;
input @3 Name $10. Age 2. +1 Sex $1.;
output;
end;
datalines;
H Tour 101
P Mary E 21 F
P George S 45 M
P Susan K 3 F
H Tour 102
P Adelle S 79 M
P Walter P 55 M
P Fran I 63 F
;

proc print data=tours;
```

```
title 'Tour List';
run;
```



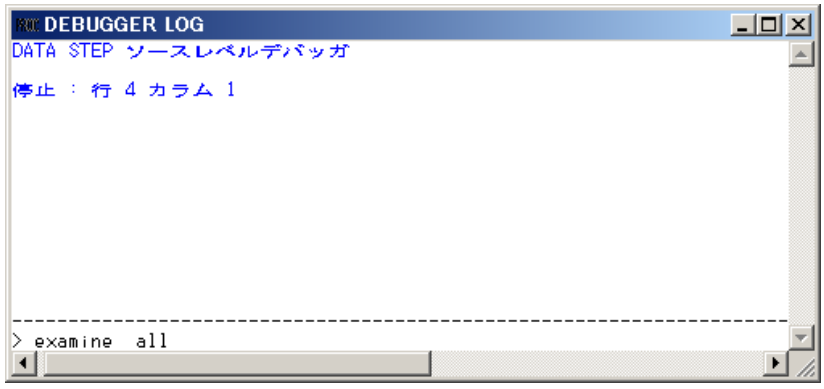
OBS	Tour	Name	Age	Sex
1		Mary E	21	F
2		George S	45	M
3		Susan K	3	F
4		Adelle	79	M
5		Walter P	55	M
6		Fran I	63	F

プログラムエラーなしで実行されましたが、出力結果が予想外でした。出力結果には変数 Tour の値が含まれていません。SAS ログを見てもプログラムのデバッグはできません。なぜなら、データは有効でログにはエラーがないからです。論理エラーを確認するには、DATA ステップデバッガを使って DATA ステップを再度実行します。

最初の反復後にデータ値を検証する

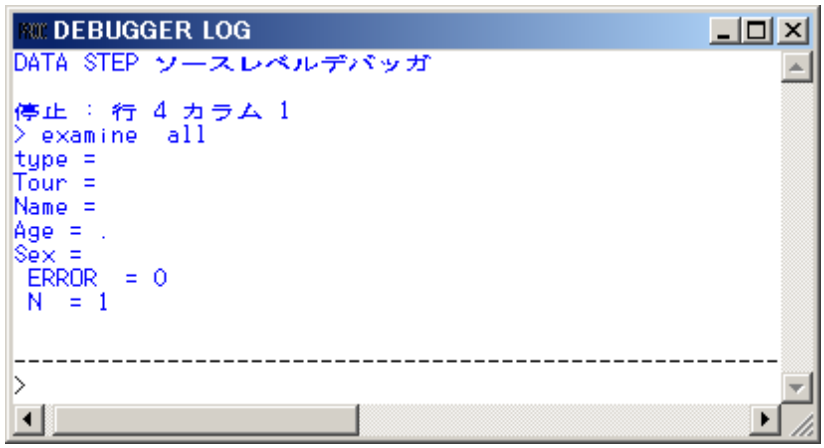
DATA ステップのデバッグは、論理エラーの仮説をたてて、プログラムの様々なポイントで変数値を検証して検定します。たとえば、実行を開始する前にデバッガコマンドラインから EXAMINE コマンドを発行して、プログラムデータ vector のすべての変数の値を表示するようにします。

```
examine _all_
```



注: ほとんどのデバッガコマンドには省略形があり、ファンクションキーにコマンドを割り当てることもできます。このセクションの例では、コマンドを完全形で表示します。すべてのコマンドのリストは、“DATA Step Debugger Commands by Category” (*Base SAS Utilities: Reference*)を参照してください。

ENTER キーを押すと、次が表示されます。



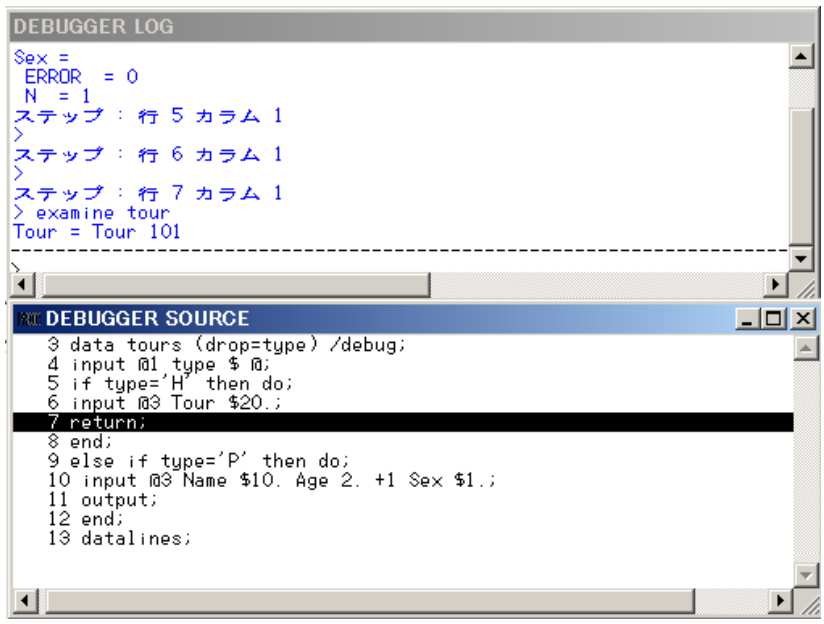
すべての変数の値は、DEBUGGER LOG ウィンドウに表示されます。SAS は INPUT ステートメントをコンパイルしましたが、実行はしていません。

DATA ステップステートメントを1つずつ実行するには、STEP コマンドを使用します。デフォルトで STEP コマンドは ENTER キーに割り当てられています。ENTER キーを繰り返し押して、DATA ステップの最初の反復を実行し、DEBUGGER SOURCE ウィンドウでプログラムの RETURN ステートメントがハイライトしたら、ストップします。

Tour の情報がプログラムの出力から欠損していたため、EXAMINE コマンドを入力して DATA ステップの最初の反復での変数 Tour の値を表示します。

```
examine tour
```

結果は次のように表示されます。



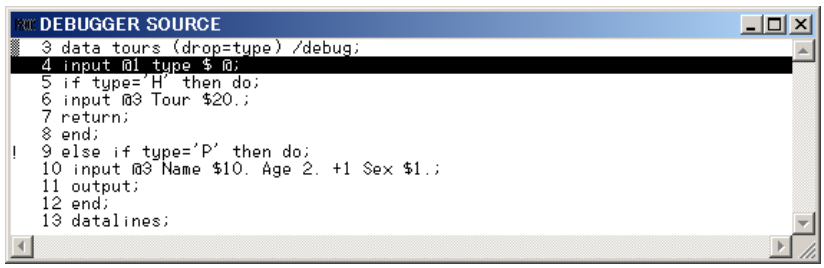
変数 Tour は値 Tour 101 を持っており、Tour が読み取られていることを示します。DATA ステップの最初の反復は、意図されたように動作しています。ENTER を押して DATA ステップの一番上にいきます。

2回目の反復後にデータ値を検証する

指定する特定の行で Data ステップの実行を一時停止するには、BREAK コマンド(ブレークポイントの設定ともいいます)を使います。この例では、ブレークポイントを 9 行目に設定するとこにより、ELSE ステートメントを実行する前に一時停止します。

```
break 9
```

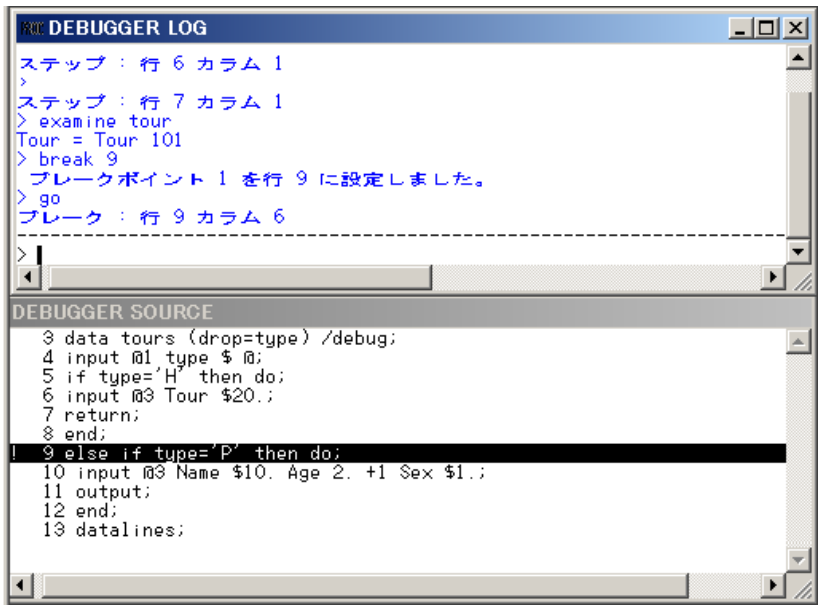
ENTER を押すと、ブレークポイントを示す感嘆符が DEBUGGER SOURCE ウィンドウの 9 行目に表示されます。



GO コマンドを実行すると、ブレークポイント(このケースでは 9 行目)に到達するまで DATA ステップが実行されます。

```
go
```

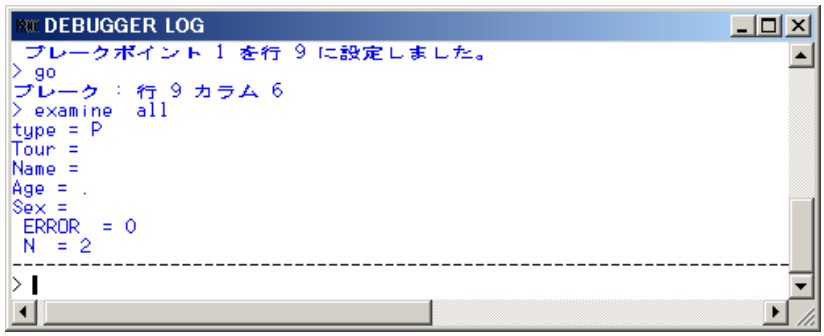
結果は次のように表示されます。



SAS は 7 行目の ELSE ステートメントの直前で実行を一時停止します。この時点で、ステータスを見るためにすべての変数の値を検証します。

```
examine _all_
```

値は次のように表示されます。



Tour の値をみようとしますが、表示されていません。各反復の最初にプログラムデータ vector は欠損値にリセットされるので、Tour の値を保持していません。論理的な問題を解決するためには、SAS プログラムに RETAIN ステートメントを含めなくてはなりません。

デバッガの終了

デバッグセッションを終了するには、デバッガコマンドラインで QUIT コマンドを発行します。

```
quit
```

デバッグウィンドウが消えて、元の SAS セッションが再開されます。

DATA ステップの修正

RETAIN ステートメントを追加して、元のプログラムを修正します。DATA ステップから DEBUG オプションを削除して、プログラムを再度サブミットします。

```
 /* corrected version */
data tours (drop=type);
retain Tour;
input @1 type $ @;
if type='H' then do;
input @3 Tour $20.;
return;
end;
else if type='P' then do;
input @3 Name $10. Age 2. +1 Sex $1.;
output;
end;
datalines;
H Tour 101
P Mary E 21 F
P George S 45 M
P Susan K 3 F
H Tour 102
P Adelle S 79 M
P Walter P 55 M
P Fran I 63 F
;

run;

proc print;
title 'Tour List';
run;
```

今度は Tour の値が出力に表示されています。

OBS	Tour	Name	Age	Sex
1	Tour 101	Mary E	21	F
2	Tour 101	George S	45	M
3	Tour 101	Susan K	3	F
4	Tour 102	Adelle	79	M
5	Tour 102	Walter P	55	M
6	Tour 102	Fran I	63	F

例 2: 出力形式の処理

この例では、Format ステートメントを使用して日付を出力するプログラムのデバッグ方法を示します。次のプログラムは特定の国への旅行日数をリストするレポートを作成します。

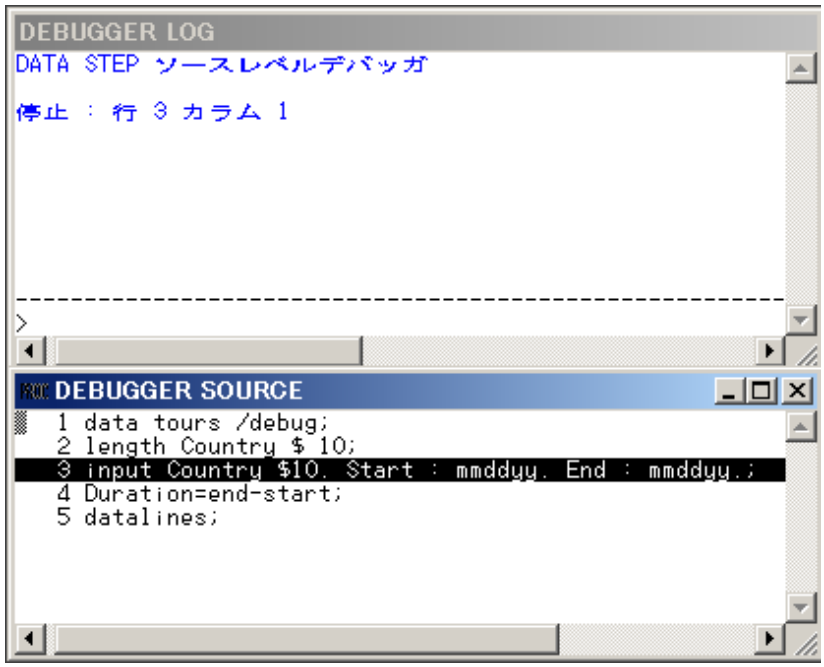
```
data tours;
length Country $ 10;
input Country $10. Start : mmddyy. End : mmddyy.;
Duration=end-start;
datalines;
Italy 033012 041312
Brazil 021912 022812
Japan 052212 061512
Venezuela 110312 11801
Australia 122112 011513
;

proc print data=tours;
format start end date9.;
title 'Tour Duration';
run;
```

結果ビューア - SAS Output

Tour Duration

OBS	Country	Start	End	Duration
1	Italy	30MAR2012	13APR2012	14
2	Brazil	19FEB2012	28FEB2012	9
3	Japan	22MAY2012	15JUN2012	24
4	Venezuela	03NOV2012	18JAN2012	-290
5	Australia	21DEC2012	15JAN2013	25

Venezuela へのツアーの Duration の値は-290 日という負の数字を示しています。エラーを確認するには、DATA ステップデバッガを使って DATA ステップを再度実行します。SAS は次のデバッガウィンドウを表示します。



実行を開始する前に DEBUGGER LOG コマンドラインから EXAMINE コマンドを発行して、プログラムデータ vector のすべての変数の値を表示するようにします。

```
examine _all_
```

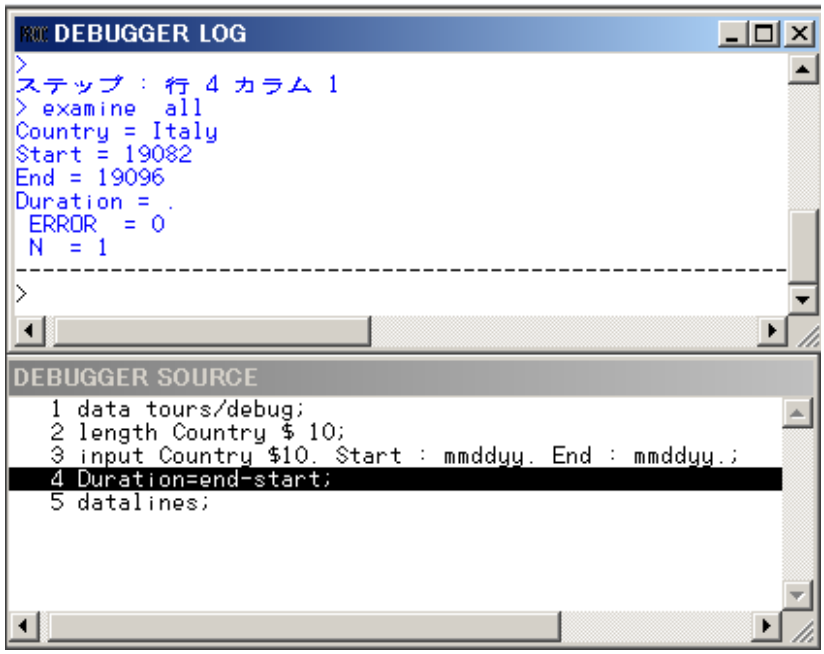
すべての変数の初期値は、DEBUGGER LOG ウィンドウに表示されます。SAS はまだ INPUT ステートメントを実行していません。

ENTER キーを押すと STEP コマンドが発行されます。SAS は INPUT ステートメントを実行し、割り当てステートメントをハイライトします。

EXAMINE コマンドを発行して、すべての変数の現在の値を表示するようにします。

```
examine _all_
```

結果は次のように表示されます。



Venezuela ツアーに問題があるので、Country の値が Venezuela のときに割り当てステートメントの前で実行を一時停止します。次のようにブレークポイントを設定します。

```
break 4 when country='Venezuela'
```

GO コマンドを実行して、プログラムの実行を再開します。

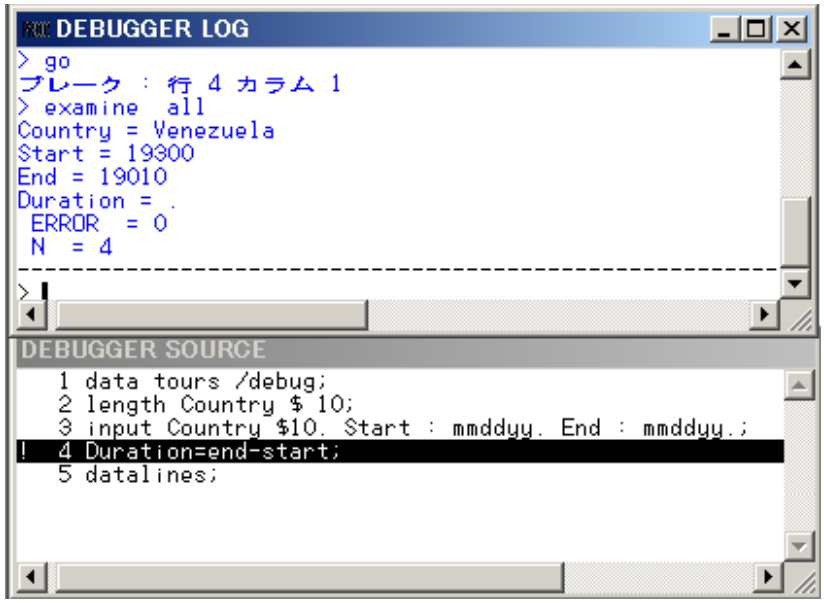
```
go
```

国名が Venezuela のとき、実行は停止します。Venezuela 旅行のツアー Start と End 日程を検証します。割り当てステートメントはハイライトされているため(ステートメントはまだ実行されていないということ)、Duration に値はありません。

EXAMINE コマンドを実行して、実行後の変数の値の変化を表示します。command to view the value of the variables after execution:

```
examine _all_
```

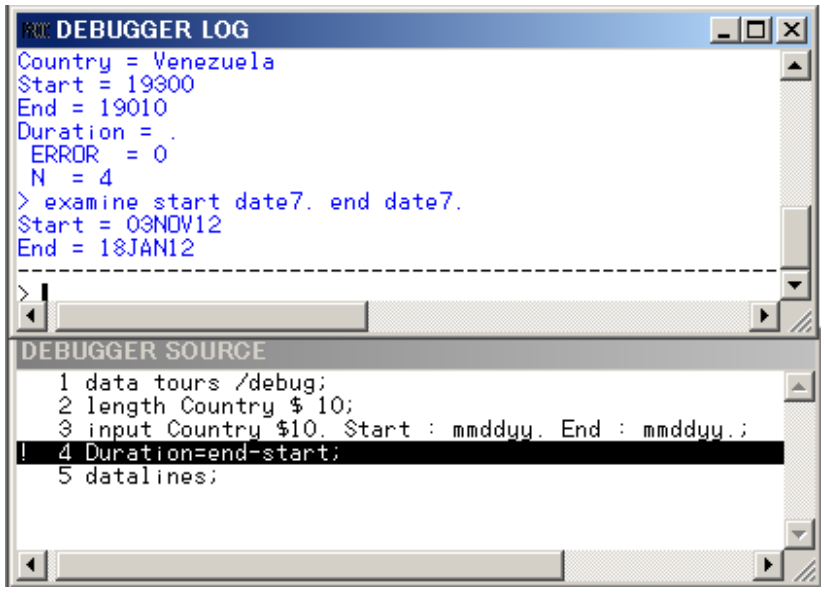
結果は次のように表示されます。



フォーマットされた SAS 日付を表示するには、DATEw.形式を使用する EXAMINE コマンドを発行します。

```
examine start date7. end date7.
```

結果は次のように表示されます。



ツアー終了が January 18, 2012 ではなく November 18, 2012 となっているため、変数 End にエラーがあります。プログラムのソースデータを検証すると、End の値のミスタイプが見つかります。SET コマンドを使用して、暫定的に End の値を November 18 に設定して、期待する結果が得られるか見てみます。DDMMMYY*w*. 形式で SET コマンドを発行します。

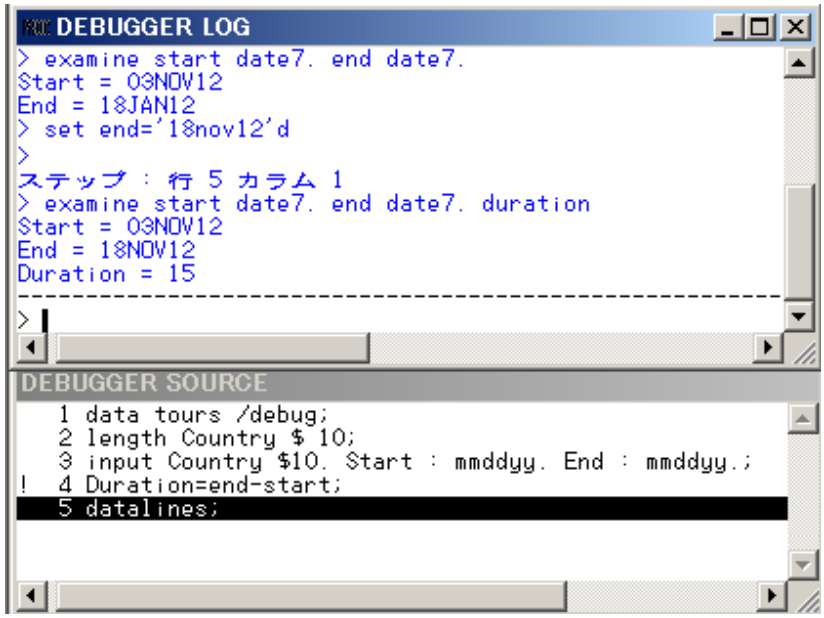
```
set end='18nov12'd
```

ENTER キーを押すと STEP コマンドが発行され、割り当てステートメントが実行されます。

EXAMINE コマンドを発行して、ツアー日付と Duration フィールドを表示します。

```
examine start date7. end date7. duration
```

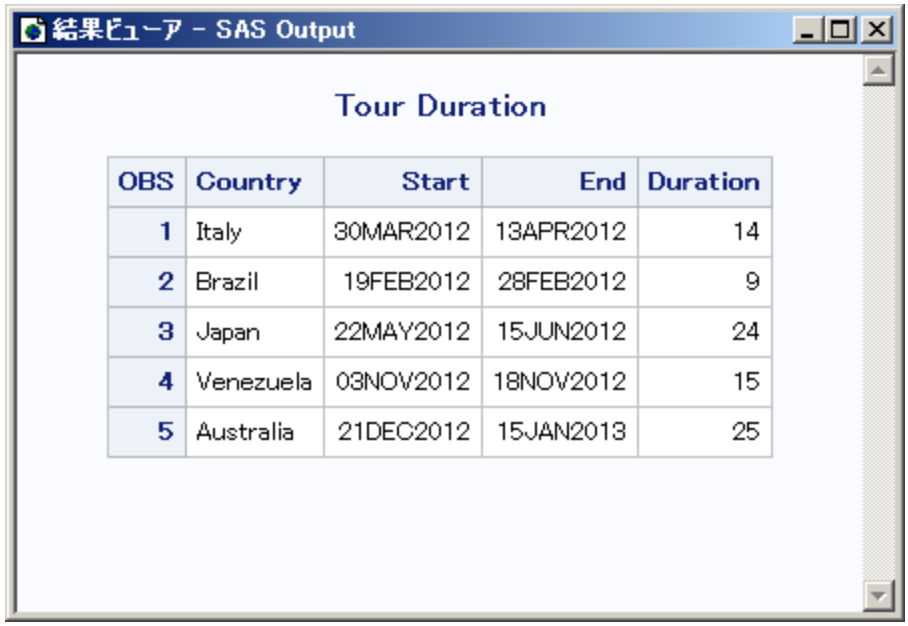
結果は次のように表示されます。



Start と End と Duration フィールドには正しいデータが入っています。

DEBUGGER LOG コマンドラインで QUIT コマンドを発行してデバッグセッションを終了します。SAS プログラムの元のデータを修正して、DBBUG オプションを削除して、プログラムを再度サブミットします。

```
 /* corrected version */

data tours;
length Country $ 10;
input Country $10. Start : mmddyy. End : mmddyy.;
duration=end-start;
datalines;
Italy 033012 041312
Brazil 021912 022812
Japan 052212 061512
Venezuela 110312 111812
Australia 122112 011513
;

proc print data=tours;
format start end date9.;
title 'Tour Duration';
run;
```



OBS	Country	Start	End	Duration
1	Italy	30MAR2012	13APR2012	14
2	Brazil	19FEB2012	28FEB2012	9
3	Japan	22MAY2012	15JUN2012	24
4	Venezuela	03NOV2012	18NOV2012	15
5	Australia	21DEC2012	15JAN2013	25

例 3: DO ループのデバッグ

反復する DO や DO WHILE や DO UNTIL ステートメントは DATA ステップの一回の反復で何回も反復させることができます。DO ループをデバッグする際、BREAK コマンドで AFTER オプションを使うことでループの複数回の反復を検証することができます。AFTER オプションではブレークポイントに到達するまでにループ反復する回数を指定します。BREAK コマンドはプログラムの実行を一時停止します。たとえば、このデータセットで考えてみます。

```
data new / debug;
set old;
do i=1 to 20;
newtest=oldtest+i;
output;
end;
run;
```

割り当てステートメント(この例では 4 行目)に DO ループの 5 回の反復ごとにブレークポイントを設定するには、このコマンドを発行します。

```
break 4 after 5
```

GO コマンドを発行したら、デバッガは DO ループの反復 `i` の値が 5 か 10 か 15 か 20 の場合に、実行を一時停止します。

反復 DO ループでは、AFTER オプションにはループの反復の回数を丁度分割できる値を選択します。たとえば、この DATA ステップでは、は 5 で 20 を丁度分割できます。2 回目の反復では、`i` の値はまた 5、10、15、そして 20 となります。

もしも丁度分割できる値を選択しない場合(この例では 3)、AFTER オプションでの `i` の値が 3、6、9、12、15 および 18 のときにデバッガは一時停止します。DO ループの 2 回目の反復のときは、`i` の値は 1、4、7、10、13 および 16 になります。

例 4: 変数のフォーマット指定された値の検証

EXAMINE コマンドで値を表示するときには、SAS 形式またはユーザー指定の形式を使用できます。たとえば、変数 BEGIN は SAS 日付値を持っているとします。曜日と日付を表示するには、EXAMINE に WEEKDATE*w*. 形式を使用します。

```
examine begin weekdate17.
```

BEGIN の値が 033012 のとき、デバッガの表示は次のようになります。

```
Sun, Mar 30, 2012
```

他の例では、SIZE という名前に形式を作成することもできます。

```
proc format;
value size 1-5='small'
6-10='medium'
11-high='large';
run;
```

変数 STOCKNUM に形式 SIZE を適用する DATA ステップをデバッグするには、形式を EXAMINE と一緒に使います。

```
examine stocknum size.
```

たとえば、STOCKNUM の値が 7 のとき、デバッガの表示は次のようになります。

```
STOCKNUM = medium
```

6 部

レポートの作成

27 章
PRINT プロシジャを使用した詳細レポートの作成 421

28 章
TABULATE プロシジャを使用した要約テーブルの作成 461

29 章
REPORT プロシジャを使用した詳細レポートと要約レポートの作成 489

27 章

PRINT プロシジャを使用した詳細レポートの作成

PRINT プロシジャを使用したレポートの作成について **422**
目的 422
前提条件 422

例で使用される入力ファイルおよび入力 SAS データセット **422**

単純なレポートの作成 **424**
すべての変数の表示 424
オブザベーションの列のラベル付け 425
オブザベーションの列の非表示 426
キー変数の強調 427
選択した変数の値のレポート作成 430
オブザベーションの選択 432

拡張レポートの作成 **434**
レポートの拡張方法 434
変数の出力形式の指定 435
数値変数の合計 436
変数値ごとのオブザベーションのグループ化 437
複数セクションのオブザベーションのグループ化 444

カスタマイズされたレポートの作成 **446**
レポートのカスタマイズ方法 446
タイトルとフットノートについて 446
タイトルとフットノートの追加 447
ラベルの定義 448
複数の行にわたるラベルの分割 449
ブランク行の追加 450
レポートスタイルの変更 451

レポートを変更しやすくする **453**
SAS マクロ機能について 453
自動マクロ変数の使用 453
独自のマクロ変数の使用 454
マクロ変数の定義 455
マクロ変数の参照 455

要約 **457**
PROC PRINT ステートメント 457
PROC SORT ステートメント 459
SAS マクロ言語 459

詳細情報 **460**

PRINT プロシジャを使用したレポートの作成について

目的

PRINT プロシジャで作成するレポートには、レポートに含める対象として選択されたオブザベーションごとに行が 1 つずつ含まれます。レポートは処理されるレコードすべてについての情報を提供します。たとえば、販売会社のレポートには、年の特定の四半期中に発生した売上それぞれに関するすべての情報を含められます。PRINT プロシジャはいくつかあるレポート作成ツールの 1 つで、これを使用してさまざまレポートを作成できます。

このセクションでは、次の操作を行う方法を学習します。

- いくつかの PROC PRINT の基本オプションとステートメントを使用して単純なレポートを生成します。
- 値のフォーマット、列の合計、オブザベーションのグループ化、合計の計算を行うオプションのステートメントを追加して拡張レポートを生成します。
- タイトル、フットノート、列のラベル、レポート出力のスタイルを追加してレポートの外観をカスタマイズします。
- マクロ変数を使用してテキストを置き換えます。

前提条件

このセクションを先に進む前に、次の機能と概念を理解していることを確認してください。

- 割り当てステートメント
- SORT プロシジャ
- BY ステートメント
- プロシジャ出力場所

例で使用される入力ファイルおよび入力 SAS データセット

このセクションの例では、1 つの入力ファイルと 5 つの SAS データセットを使用します。入力データの完全なリストについては、“YEAR_SALES データセット” (794 ページ)を参照してください。

入力ファイルには、コーヒーマシンを供給する会社 TruBlend Coffee Makers の売上記録が含まれています。ファイルの構造は次のとおりです。

```
01      1      Hollingsworth  Deluxe      260      49.50
01      1      Garcia         Standard     41      30.97
01      1      Hollingsworth  Deluxe      330      49.50
01      1      Jensen         Standard   1110      30.97
01      1      Garcia         Standard    715      30.97
01      1      Jensen         Deluxe      675      49.50
```

```
02      1      Jensen          Standard     45      30.97
02      1      Garcia          Deluxe       10      49.50

…more data lines…

12      4      Hollingsworth   Deluxe      125      49.50
12      4      Jensen          Standard   1254      30.97
12      4      Hollingsworth   Deluxe      175      49.50
```

入力ファイルには、次に示す値が左から右へと含まれています。

- 売上が発生した月
- 売上が発生した年の四半期
- 営業担当者の名前
- 売り上げたコーヒーメーカーの種類(Standard または Deluxe)
- 販売ユニット数
- 各ユニットの価格(米ドル)

5 つのうち最初の SAS データセットの名前は YEAR_SALES です。このデータセットには、入力ファイルからのすべての売上データ、および Units を Price で乗算することにより作成される AmountSold という名前の新しい変数が含まれます。その他 4 つのデータセットは YEAR_SALES データセットから作成されます。4 つのデータセットのそれぞれに、各四半期のデータのサブセットが含まれます。このデータセットが QTR01、QTR02、QTR03、QTR04 です。

次のプログラムは、5 つの SAS データセットを作成します。

```
data year_sales;
   infile 'your-input-file';
   input Month $ Quarter $ SalesRep $14. Type $ Units Price;
   AmountSold = Units * Price;
run;

data qtr01;
   set year_sales(where=(quarter='1'));
run;

data qtr02;
   set year_sales(where=(quarter='2'));
run;

data qtr03;
   set year_sales(where=(quarter='3'));
run;

data qtr04;
   set year_sales(where=(quarter='4'));
run;
```

単純なレポートの作成

すべての変数の表示

デフォルトでは、PRINT プロシジャでは、データセット内のすべての変数とオブザベーションの値を表示する単純なレポートが生成されます。たとえば、次の PROC PRINT ステップは第 1 四半期売上のレポートを作成します。

```
proc print data=qtr01;
   title 'TruBlend Coffee Makers First Quarter Sales Report';
run;
```

次の出力は、QTR01 のすべての変数とすべてのオブザベーションの値を示しています。

図 27.1 *すべての変数とすべてのオブザベーションの表示*

TruBlend Coffee Makers First Quarter Sales Report

Obs	Month	Quarter	SalesRep	Type	Units	Price	AmountSold
1	01	1	Hollingsworth	Deluxe	260	49.50	12870.00
2	01	1	Garcia	Standard	41	30.97	1269.77
3	01	1	Hollingsworth	Standard	330	30.97	10220.10
4	01	1	Jensen	Standard	110	30.97	3406.70
5	01	1	Garcia	Deluxe	715	49.50	35392.50
6	01	1	Jensen	Standard	675	30.97	20904.75
7	02	1	Garcia	Standard	2045	30.97	63333.65
8	02	1	Garcia	Deluxe	10	49.50	495.00
9	02	1	Garcia	Standard	40	30.97	1238.80
10	02	1	Hollingsworth	Standard	1030	30.97	31899.10
11	02	1	Jensen	Standard	153	30.97	4738.41
12	02	1	Garcia	Standard	98	30.97	3035.06
13	03	1	Hollingsworth	Standard	125	30.97	3871.25
14	03	1	Jensen	Standard	154	30.97	4769.38
15	03	1	Garcia	Standard	118	30.97	3654.46
16	03	1	Hollingsworth	Standard	25	30.97	774.25
17	03	1	Jensen	Standard	525	30.97	16259.25
18	03	1	Garcia	Standard	310	30.97	9600.70

Obs の列は、各オブザベーションを番号によって特定します。デフォルトでは、SAS により各行の先頭にオブザベーション番号が表示されます。

レポートの上部にタイトルが表示されます。PROC PRINT ステップの TITLE ステートメントによりタイトルが生成されます。TITLE ステートメントの詳細については、“カスタマイズされたレポートの作成” (446 ページ)を参照してください。ここでは、すべての例に、この例に示すタイトルのような、説明的なタイトルを生成する TITLE ステートメントが少なくとも 1 つ含まれることに留意してください。

レポートのコンテンツは、元のデータセット QTR01 のコンテンツと非常によく似ています。ただし、レポートは簡単に作成や拡張ができます。

オブザベーションの列のラベル付け

レポートを変更する簡単な方法は、オブザベーション番号(Obs 列)をラベル付けすることです。次の SAS プログラムには、PROC PRINT ステートメントに Obs 列の列ラベルを変更する OBS=オプションが含まれています。

```
proc print data=qtr01 obs='Observation Number';
   title 'TruBlend Coffee Makers First Quarter Sales Report';
run;
```

レポート出力を次に示します。

図 27.2 オブザベーションの列のラベル付け

TruBlend Coffee Makers First Quarter Sales Report

Observation Number	Month	Quarter	SalesRep	Type	Units	Price	AmountSold
1	01	1	Hollingsworth	Deluxe	260	49.50	12870.00
2	01	1	Garcia	Standard	41	30.97	1269.77
3	01	1	Hollingsworth	Standard	330	30.97	10220.10
4	01	1	Jensen	Standard	110	30.97	3406.70
5	01	1	Garcia	Deluxe	715	49.50	35392.50
6	01	1	Jensen	Standard	675	30.97	20904.75
7	02	1	Garcia	Standard	2045	30.97	63333.65
8	02	1	Garcia	Deluxe	10	49.50	495.00
9	02	1	Garcia	Standard	40	30.97	1238.80
10	02	1	Hollingsworth	Standard	1030	30.97	31899.10
11	02	1	Jensen	Standard	153	30.97	4738.41
12	02	1	Garcia	Standard	98	30.97	3035.06
13	03	1	Hollingsworth	Standard	125	30.97	3871.25
14	03	1	Jensen	Standard	154	30.97	4769.38
15	03	1	Garcia	Standard	118	30.97	3654.46
16	03	1	Hollingsworth	Standard	25	30.97	774.25
17	03	1	Jensen	Standard	525	30.97	16259.25
18	03	1	Garcia	Standard	310	30.97	9600.70

オブザベーションの列の非表示

レポートを単純化する簡単な方法は、オブザベーション番号(Obs 列)を非表示にすることです。通常、各オブザベーションを番号で特定する必要はありません。(場合によっては、オブザベーション番号を表示するほうがよいこともあります。)次の SAS プログラムには、ROC PRINT ステートメントに Obs 列を非表示にする NOOBS オプションが含まれています。

```
proc print data=qtr01 noobs;
   title 'TruBlend Coffee Makers First Quarter Sales Report';
run;
```

レポート出力を次に示します。

図 27.3 オブザベーションの列の非表示

TruBlend Coffee Makers First Quarter Sales Report

Month	Quarter	SalesRep	Type	Units	Price	AmountSold
01	1	Hollingsworth	Deluxe	260	49.50	12870.00
01	1	Garcia	Standard	41	30.97	1269.77
01	1	Hollingsworth	Standard	330	30.97	10220.10
01	1	Jensen	Standard	110	30.97	3406.70
01	1	Garcia	Deluxe	715	49.50	35392.50
01	1	Jensen	Standard	675	30.97	20904.75
02	1	Garcia	Standard	2045	30.97	63333.65
02	1	Garcia	Deluxe	10	49.50	495.00
02	1	Garcia	Standard	40	30.97	1238.80
02	1	Hollingsworth	Standard	1030	30.97	31899.10
02	1	Jensen	Standard	153	30.97	4738.41
02	1	Garcia	Standard	98	30.97	3035.06
03	1	Hollingsworth	Standard	125	30.97	3871.25
03	1	Jensen	Standard	154	30.97	4769.38
03	1	Garcia	Standard	118	30.97	3654.46
03	1	Hollingsworth	Standard	25	30.97	774.25
03	1	Jensen	Standard	525	30.97	16259.25
03	1	Garcia	Standard	310	30.97	9600.70

キー変数の強調

ID ステートメントについて

データセットのキー変数を強調するには、PROC PRINT ステップで ID ステートメントを使用します。ID ステートメントで変数を指定すると、PROC PRINT によりこの変数の値がレポートの各行の最初の列に表示されます。この方法でキー変数を強調すると、データについての質問に答えるのに役立ちます。たとえば、レポートで“各営業担当者の、年の第 1 四半期の売上高はいくらか”という質問に対して回答します。次の 2 つの例で、並べ替えられていないデータと並べ替えられたデータを使用して、この質問に素早く答える方法を示します。

並べ替えられていないキー変数の使用

営業担当者を強調するレポートを作成するには、PROC PRINT ステップに変数 SalesRep を指定する ID ステートメントを含めます。変更後のプログラムは、次のとおりです。

```
proc print data=qtr01;
   id SalesRep;
   title 'TruBlend Coffee Makers First Quarter Sales Report';
run;
```

ID ステートメントによりオブザベーション番号が自動的に非表示されるため、PROC PRINT ステートメントに NOOBS オプションは必要ありません。

レポート出力を次に示します。

図 27.4 並べ替えられていない変数を指定した ID ステートメントの使用

TruBlend Coffee Makers First Quarter Sales Report

SalesRep	Month	Quarter	Type	Units	Price	AmountSold
Hollingsworth	01	1	Deluxe	260	49.50	12870.00
Garcia	01	1	Standard	41	30.97	1269.77
Hollingsworth	01	1	Standard	330	30.97	10220.10
Jensen	01	1	Standard	110	30.97	3406.70
Garcia	01	1	Deluxe	715	49.50	35392.50
Jensen	01	1	Standard	675	30.97	20904.75
Garcia	02	1	Standard	2045	30.97	63333.65
Garcia	02	1	Deluxe	10	49.50	495.00
Garcia	02	1	Standard	40	30.97	1238.80
Hollingsworth	02	1	Standard	1030	30.97	31899.10
Jensen	02	1	Standard	153	30.97	4738.41
Garcia	02	1	Standard	98	30.97	3035.06
Hollingsworth	03	1	Standard	125	30.97	3871.25
Jensen	03	1	Standard	154	30.97	4769.38
Garcia	03	1	Standard	118	30.97	3654.46
Hollingsworth	03	1	Standard	25	30.97	774.25
Jensen	03	1	Standard	525	30.97	16259.25
Garcia	03	1	Standard	310	30.97	9600.70

営業担当者の名前は特定の順序ではありません。レポートは、オブザベーションを営業担当者でグループ化してアルファベット順に並べ替えると読みやすくなります。

並べ替えられているキー変数の使用

データがキー変数で並べ替えられていない場合、PROC SORT を使用してこの変数でオブザベーションを並べ替えます。出力データセットを指定しない場合、PROC SORT により入力データセットのオブザベーションの順序が永久に変更されます。

次のプログラムに、オブザベーションを営業担当者のアルファベット順に並べ替える方法を示します。

```
proc sort data=qtr01; 1
   by SalesRep; 2
run;

proc print data=qtr01;
   id SalesRep; 3
   title 'TruBlend Coffee Makers First Quarter Sales Report';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 PROC SORT ステップは PROC PRINT ステップに優先します。PROC SORT によりデータセットのオブザベーションが BY 変数の値でアルファベット順に並べ替えられ、入力データセットが上書きされます。

2 BY ステートメントにより、オブザベーションが SalesRep のアルファベット順に並べ替えられます。

3 ID ステートメントにより、オブザベーション番号ではなく SalesRep の値でオブザベーションが特定されます。PROC PRINT により SalesRep の並べ替え済みの順序が使用されてレポートが作成されます。

レポート出力を次に示します。

図 27.5 並べ替えられているキー変数を指定した ID ステートメントの使用

TruBlend Coffee Makers First Quarter Sales Report

SalesRep	Month	Quarter	Type	Units	Price	AmountSold
Garcia	01	1	Standard	41	30.97	1269.77
Garcia	01	1	Deluxe	715	49.50	35392.50
Garcia	02	1	Standard	2045	30.97	63333.65
Garcia	02	1	Deluxe	10	49.50	495.00
Garcia	02	1	Standard	40	30.97	1238.80
Garcia	02	1	Standard	98	30.97	3035.06
Garcia	03	1	Standard	118	30.97	3654.46
Garcia	03	1	Standard	310	30.97	9600.70
Hollingsworth	01	1	Deluxe	260	49.50	12870.00
Hollingsworth	01	1	Standard	330	30.97	10220.10
Hollingsworth	02	1	Standard	1030	30.97	31899.10
Hollingsworth	03	1	Standard	125	30.97	3871.25
Hollingsworth	03	1	Standard	25	30.97	774.25
Jensen	01	1	Standard	110	30.97	3406.70
Jensen	01	1	Standard	675	30.97	20904.75
Jensen	02	1	Standard	153	30.97	4738.41
Jensen	03	1	Standard	154	30.97	4769.38
Jensen	03	1	Standard	525	30.97	16259.25

これで、各営業担当者が年の最初の 3 か月間に売り上げた製品がレポートに明らかに表示されるようになりました。

選択した変数の値のレポート作成

デフォルトでは、PRINT プロシジャはデータセットのすべての変数の値をレポートに含めます。ただし、表示する変数および表示順序を制御するには、PROC PRINT ステップに VAR ステートメントを追加します。

たとえば、変数 Quarter、Type、Price の情報は不要です。そのため、レポートには次の順序で指定される変数の値のみを表示する必要があります。

```
SalesRep  Month  Units  AmountSold
```

次のプログラムでは、4 つの変数の値を特定の順序で表示するレポートを作成するための VAR ステートメントを追加します。

```
proc print data=qtr01 noobs;
   var SalesRep Month Units AmountSold;
   title 'TruBlend Coffee Makers First Quarter Sales Report';
run;
```

このプログラムには、ID ステートメントは含まれていません。変数 SalesRep が VAR ステートメントで指定される最初の変数であるため、オブザベーションを特定する必要はありません。PROC PRINT ステートメントの NOOBS オプションによりオブザベーション番号が非表示にされるため、営業担当者がレポートの先頭列に表示されます。

注: ID ステートメントが使用され、そこで VAR ステートメントの変数名のいずれかが指定された場合、情報が重複し、レポートで同じ変数に対して 2 つの列が表示されます。

レポート出力を次に示します。

図 27.6 *選択した変数の表示*

TruBlend Coffee Makers First Quarter Sales Report

SalesRep	Month	Units	AmountSold
Garcia	01	41	1269.77
Garcia	01	715	35392.50
Garcia	02	2045	63333.65
Garcia	02	10	495.00
Garcia	02	40	1238.80
Garcia	02	98	3035.06
Garcia	03	118	3654.46
Garcia	03	310	9600.70
Hollingsworth	01	260	12870.00
Hollingsworth	01	330	10220.10
Hollingsworth	02	1030	31899.10
Hollingsworth	03	125	3871.25
Hollingsworth	03	25	774.25
Jensen	01	110	3406.70
Jensen	01	675	20904.75
Jensen	02	153	4738.41
Jensen	03	154	4769.38
Jensen	03	525	16259.25

レポートは、VAR ステートメントで指定される変数のみを含むため、簡明です。

次の例では、レポートを変更して特定の条件を満たすオブザベーションのみを表示するようにします。

オブザベーションの選択

WHERE ステートメントについて

データセットから特定の条件に一致するオブザベーションを選択するには、WHERE ステートメントを使用します。WHERE ステートメントにより、各オブザベーションが処理に使用できる前に一致する必要がある特定の条件が指定され、入力データがサブセット化されます。

WHERE ステートメントで定義する条件は、算術式または論理式で、通常、一連のオペランドと演算子で構成されます。[1] 文字値を比較するには、それらを一重引用符または二重引用符で囲む必要があり、さらに値は大文字小文字の区別を含め、完全に一致する必要があります。複数の比較を、WHERE ステートメントで論理演算子で結合して指定することもできます。

WHERE ステートメントを使用すると、SAS システムが入力データセットのすべてのオブザベーションを読み込む必要がないため、SAS プログラムの効率向上に効果的です。

1 つの比較の実行

WHERE ステートメントを使用して、1 つの比較に基づいてオブザベーションを選択できます。次のプログラムでは、WHERE ステートメントで 1 つの比較を使用して、Garcia という名前の営業担当者の販売活動を示すレポートを生成します。

```
proc print data=qtr01 noobs;
   var SalesRep Month Units AmountSold;
   where SalesRep='Garcia';
   title 'TruBlend Coffee Makers Quarterly Sales for Garcia';
run;
```

SalesRep は文字変数であるため、WHERE ステートメントで値 **Garcia** は引用符で囲みます。また、値 **Garcia** の文字 G は大文字なので、データセット QTR01 の値と正確に一致します。

1 WHERE ステートメントの構造は、IF および IF-THEN ステートメントの構造と似ています。

レポート出力を次に示します。

図 27.7 1 つの比較の実行

TruBlend Coffee Makers First Quarter Sales Report

SalesRep	Month	Units	AmountSold
Garcia	01	41	1269.77
Garcia	01	715	35392.50
Garcia	02	2045	63333.65
Garcia	02	10	495.00
Garcia	02	40	1238.80
Garcia	02	98	3035.06
Garcia	03	118	3654.46
Garcia	03	310	9600.70

複数の比較の実行

WHERE ステートメントを使用して、複数の比較に基づいてオブザベーションを選択することもできます。ただし、PROC ステップで複数の WHERE ステートメントを使用すると、最後のステートメントのみが使用されます。AND 演算子を使用して、複合比較を作成できます。たとえば、次の WHERE ステートメントは、Garcia によってのみ販売された Deluxe のコーヒーメーカーのオブザベーションを選択します。

```
where SalesRep = 'Garcia' and Type='Deluxe';
```

次のプログラムでは、WHERE ステートメントで 2 つの比較を使用して、年の最初の 1 か月間の営業担当者(Garcia)の販売活動を示すレポートを生成します。

```
proc print data=year_sales noobs;
   var SalesRep Month Units AmountSold;
   where SalesRep='Garcia' and Month='01';
   title 'TruBlend Coffee Makers First Month Sales Report for Garcia';
run;
```

WHERE ステートメントは論理 AND 演算子を使用しています。そのため、PROC PRINT でオブザベーションをレポートに含めるためには、比較が両方とも真である必要があります。

レポート出力を次に示します。

図 27.8 2 つの比較の実行

TruBlend Coffee Makers First Month Sales Report for Garcia

SalesRep	Month	Units	AmountSold
Garcia	01	41	1269.77
Garcia	01	715	35392.50

複数の条件のうち少なくとも 1 つに一致するオブザベーションを選択する場合もあります。次のプログラムでは、WHERE ステートメントで 2 つの比較を使用して、年の第 1 四半期中に行われた 500 ユニットより多いかまたは$20,000 より高額であるすべての売上を示すレポートを作成します。

```
proc print data=qtr01 noobs;
   var SalesRep Month Units AmountSold;
   where Units>500 or AmountSold>20000;
   title 'Sales Rep Q1 Monthly Report for Sales Above 500 Units or $20,000';
run;
```

この WHERE ステートメントは論理 OR 演算子を使用しています。そのため、比較のうち 1 つが真であれば、PROC PRINT によりオブザベーションがレポートに含まれます。

レポート出力を次に示します。

図 27.9 *1つの条件または別の条件との比較の実行*

Sales Rep Q1 Monthly Report for Sales Above 500 Units or $20,000

SalesRep	Month	Units	AmountSold
Garcia	01	715	35392.50
Jensen	01	675	20904.75
Garcia	02	2045	63333.65
Hollingsworth	02	1030	31899.10
Jensen	03	525	16259.25

拡張レポートの作成

レポートの拡張方法

PROC PRINT ステートメントとオプションをいくつか使用するだけでさまざまな詳細レポートを生成できます。レポートを拡張する追加のステートメントとオプションを使用して、PROC PRINT で次の操作を行えます。

- 列のフォーマット
- 数値変数の合計
- 変数の値に基づくオブザベーションのグループ化
- グループの変数の値を合計
- 別々のセクションのオブザベーションをグループ化

このセクションの例では、“例で使用される入力ファイルおよび入力 SAS データセット” (422 ページ)で作成された SAS データセット QTR02 を使用します。

変数の出力形式の指定

変数の出力形式を指定して、レポートを簡単かつ効率的に読みやすくできます。プログラムに FORMAT ステートメントを追加して、変数の出力形式を指定できます。変数の出力形式は、SAS で、変数の値の記述に使用されるパターンです。たとえば、数値にカンマを追加したり、金額にドル記号を追加したり、値をローマ数字で記述したりする出力形式があります。

出力形式を使用すると、変数 Units と AmountSold の値が前のレポートより読みやすくなります。具体的には、Units は、数字を 3 桁ごとに区切るカンマを含み、小数値を省略する、フィールドの合計幅が 7 である COMMA 出力形式を使用できます。AmountSold は、数字を 3 桁ごとに区切るカンマ、小数点、小数点以下 2 桁およびドル記号を含む、フィールドの合計幅が 14 である DOLLAR 出力形式を使用できます。

次のプログラムでは、FORMAT ステートメントを使用してこれらの出力形式を適用する方法を示します。

```
proc print data=qtr02 noobs;
   var SalesRep Month Units AmountSold;
   where Units>500 or AmountSold>20000;
   format Units comma7. AmountSold dollar14.2;
   title 'Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000';
run;
```

PROC PRINT により、変数 Units の値に COMMA7.出力形式が適用され、変数 AmountSold の値に DOLLAR14.2 出力形式が適用されます。

レポート出力を次に示します。

図 27.10　数値変数のフォーマット

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep	Month	Units	AmountSold
Hollingsworth	04	530	$16,414.10
Jensen	04	1,110	$34,376.70
Garcia	04	1,715	$53,113.55
Jensen	04	675	$20,904.75
Hollingsworth	05	1,120	$34,686.40
Hollingsworth	05	1,030	$31,899.10
Garcia	06	512	$15,856.64
Garcia	06	1,000	$30,970.00

AmountSold は DOLLAR14.2 出力形式を使用します。列の最大幅は、14 スペースです。2 スペースは、値の小数部分用に予約されています。残りの 12 スペースに、小数点、整数、ドル記号、カンマ、および値が負の場合にはマイナス符号が含まれます。

Units は COMMA7.出力形式を使用します。列の最大幅は、7 スペースです。列の幅には、数値、カンマ、および値が負の場合にはマイナス符号が含まれます。

出力形式は、SAS データセットに格納されている内部データ値には影響しません。出力形式により変更されるのは、現在の PROC ステップによるレポートでの値の表示方法のみです。

注: 最大値が含まれるように、出力形式で十分な列幅を指定してください。指定する出力形式の幅の広さが、カンマやドル記号などの特殊文字をあわせた最大値を含めるのに十分でない場合、SAS により最も適した出力形式が適用されます。

数値変数の合計

データセットの値のレポート作成に加え、SUM ステートメントを追加して数値変数の小計と合計を計算できます。SUM ステートメントを使用して、1 つ以上の変数の合計を要求できます。

次のプログラムは 2 つの数値変数 Units と AmountSold の合計を示すレポートを生成します。

```
proc print data=qtr02 noobs;
   var SalesRep Month Units AmountSold;
   where Units>500 or AmountSold>20000;
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   title 'Sales Rep Q2 Monthly Report for Sales above 500 Units or $20,000';
run;
```

レポート出力を次に示します。

図 27.11 数値変数の合計

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep	Month	Units	AmountSold
Hollingsworth	04	530	$16,414.10
Jensen	04	1,110	$34,376.70
Garcia	04	1,715	$53,113.55
Jensen	04	675	$20,904.75
Hollingsworth	05	1,120	$34,686.40
Hollingsworth	05	1,030	$31,899.10
Garcia	06	512	$15,856.64
Garcia	06	1,000	$30,970.00
		7,692	**$238,221.24**

Units と AmountSold の合計は、すべての営業担当者による各売上の値を合計することにより計算されます。次の例で示すように、PRINT プロシジャでは営業担当者ごとの小計を個別に計算することもできます。

変数値ごとのオブザベーションのグループ化

変数値ごとのオブザベーションのグループ化の概要

BY ステートメントを使用して、オブザベーションのグループの個別の分析を取得できます。前の例では、SUM ステートメントを使用して、変数 Units と AmountSold の合計を計算しました。ただし、3 人の営業担当者全員を 1 つのグループとしての合計でした。次の例では、PROC PRINT ステップの一部として BY、ID および SUMBY ステートメントを使用して、営業担当者を、個別の 3 つの小計と 1 つの総計を持つ 3 つのグループに分ける方法を示します。

グループの小計の計算

特定の数値変数の個別の小計を取得するには、PROC PRINT ステップに BY ステートメントを追加します。BY ステートメントが使用されると、PRINT プロシジャは、データセットが BY 変数を使用してすでに並べ替え済みであると見なします。そのため、データが正しい順序で並べ替えられていない場合、PROC PRINT ステップの前に PROC SORT ステップを追加する必要があります。

BY ステートメントは、レポートで BY グループごとに個別のテーブルを生成します。各テーブルの上に、BY グループのヘッダーとして BY 行が示されます。

注: VAR ステートメントに BY ステートメントで使用する変数を指定しないでください。指定すると、BY 変数の値がページを横切るヘッダーとして、およびページの下部の列とに、2 回レポートに表示されます。

次のプログラムは、PROC PRINT ステップに BY ステートメントを使用して、営業担当者ごとに変数 Units と AmountSold の個別の小計を取得します。

```
proc sort data=qtr02;
   by SalesRep; 1
run;

proc print data=qtr02 noobs;
   var Month Units AmountSold; 2
   where Units>500 or AmountSold>20000;
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   by SalesRep; 2
   title1 'Sales Rep Q2 Totals for Sales above 500 Units or $20,000';
  run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 PROC SORT ステップの BY ステートメントによりデータが並べ替えられます。

2 変数 SalesRep が、VAR ステートメントではなく BY ステートメントの一部となります。

レポート出力を次に示します。

図 27.12 BY ステートメントによるオブザベーションのグループ化

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep=Garcia

Month	Units	AmountSold
04	1,715	$53,113.55
06	512	$15,856.64
06	1,000	$30,970.00
SalesRep	3,227	$99,940.19

SalesRep=Hollingsworth

Month	Units	AmountSold
04	530	$16,414.10
05	1,120	$34,686.40
05	1,030	$31,899.10
SalesRep	2,680	$82,999.60

SalesRep=Jensen

Month	Units	AmountSold
04	1,110	$34,376.70
04	675	$20,904.75
SalesRep	1,785	$55,281.45
	7,692	$238,221.24

小計と総計のラベル付け

前の例では、小計ラベルと総計ラベルに対してデフォルトラベルが使用されます。小計デフォルトラベルは BY 変数で、総計デフォルトラベルはラベルなしです。デフォルトの小計ラベルを置換するには SUMLABEL=オプション、ブランクの総計ラベルを置換するには GRANDTOTAL_LABEL=オプションを使用します。

次の例では、SUMLABEL=オプションと GRANDTOTAL_LABEL=オプションを PROC PRINT ステートメントに追加します。このプログラムでは、QTR02 データが変数 SalesRep によって事前に並べ替え済みであると見なします。

```
proc print data=qtr02 noobs sumlabel="Total" grandtotal_label="Grand Total";;
   var Month Units AmountSold;
   where Units>500 or AmountSold>20000;
   format Units comma7. AmountSold dollar14.2;
```

```
 sum Units AmountSold;
 by SalesRep;
 title1 'Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000';
run;
```

レポート出力を次に示します。

Sales Rep Q2 Monthy Report for Sales Above 500 Units or $20,000

SalesRep=Garcia

Month	Units	AmountSold
04	1,715	$53,113.55
06	512	$15,856.64
06	1,000	$30,970.00
Total	**3,227**	**$99,940.19**

SalesRep=Hollingsworth

Month	Units	AmountSold
04	530	$16,414.10
05	1,120	$34,686.40
05	1,030	$31,899.10
Total	**2,680**	**$82,999.60**

SalesRep=Jensen

Month	Units	AmountSold
04	1,110	$34,376.70
04	675	$20,904.75
Total	**1,785**	**$55,281.45**
Grand Total	**7,692**	**$238,221.24**

グループの小計の識別

PROC PRINT ステップで BY ステートメントと ID ステートメントの両方を使用して、レポートの外観を変更できます。BY ステートメントと ID ステートメントの両方に同じ変数が指定されると、PRINT プロシジャは ID 変数を使用して BY グループの先頭を識別します。

次の例では、前の例で並べ替えられたデータセットを使用して、PROC PRINT ステップに ID ステートメントを追加します。

```
proc print data=qtr02 sumlabel="Total" grandtotal_label="Grand Total";
   var Month Units AmountSold;
```

```
where Units>500 or AmountSold>20000;
format Units comma7. AmountSold dollar14.2;
sum Units AmountSold;
by SalesRep;
id SalesRep;
title1 'Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000';
run;
```

レポート出力を次に示します。

図 27.13 *BY ステートメントおよび ID ステートメントによるオブザベーションのグループ化*

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep	Month	Units	AmountSold
Garcia	04	1,715	$53,113.55
	06	512	$15,856.64
	06	1,000	$30,970.00
Total		3,227	$99,940.19

SalesRep	Month	Units	AmountSold
Hollingsworth	04	530	$16,414.10
	05	1,120	$34,686.40
	05	1,030	$31,899.10
Total		2,680	$82,999.60

SalesRep	Month	Units	AmountSold
Jensen	04	1,110	$34,376.70
	04	675	$20,904.75
Total		1,785	$55,281.45
Grand Total		7,692	$238,221.24

レポートには 2 つのはっきりとした特徴があります。PROC PRINT によりレポートがグループに分けられ、BY 変数と ID 変数の冗長な値が非表示にされています。PROC PRINT ステップで BY ステートメントと ID ステートメントが併用されているため、グループの上に BY 行は表示されません。

SUM ステートメント、BY ステートメント、ID ステートメントに関する次の一般的な規則に留意してください。

- SUM ステートメントに変数を指定すると、VAR ステートメントでその変数を省略できます。PROC PRINT により、その変数が単に VAR ステートメントの変数のリストに追加されます。
- ID ステートメントまたは BY ステートメントで使用した変数は SUM ステートメントに指定しません。

- BY ステートメントを使用して 1 つの BY 変数のみを指定すると、PROC PRINT により、複数のオブザベーションを含む BY グループごとに SUM 変数の小計が計算されます。
- BY ステートメントを使用して複数の BY 変数を指定すると、PROC PRINT により、BY 変数の値が変わり、その値のオブザベーションが複数存在するときのみ、BY 変数の小計が表示されます。

複数グループの小計の計算

BY ステートメントで複数の変数を使用して、グループやサブグループを定義することもできます。次のプログラムは、最初に営業担当者で、次に月でオブザベーションをグループ化するレポートを生成します。

```
proc sort data=qtr02;
   by SalesRep Month; 1
run;

proc print data=qtr02 noobs n='Sales Transactions:' 2
                            'Total Sales Transactions:'2;
   var Units AmountSold; 3
   where Units>500 or AmountSold>20000;
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   by SalesRep Month 3;
   title1 'Monthly Sales Rep Totals for Sales Above 500 Units or $20,000';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 PROC SORT ステップの BY ステートメントにより、データが SalesRep と Month で並べ替えられます。

2 PROC PRINT ステートメントの N=オプションにより、BY グループのオブザベーションの数がレポートに表示されます。SUM ステートメントがあるため、全体のオブザベーションの総数もレポートの末尾に表示されます。N=により提供される説明テキストの最初の部分が、BY グループごとの数に先行します。N=により提供される説明テキストの 2 番目の部分が、全体の総数に先行します。

3 変数 SalesRep と Month は、BY ステートメントで指定されているため、VAR ステートメントでは省略されています。これにより PROC PRINT でこれらの変数の値が 2 回レポートに表示されるのを防ぎます。

レポート出力を次に示します。

図 27.14 複数の BY 変数によるオブザベーションのグループ化

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep=Garcia Month=04

Units	AmountSold
1,715	$53,113.55
Sales Transactions:1	

SalesRep=Garcia Month=06

Units	AmountSold
512	$15,856.64
1,000	$30,970.00
1,512	**$46,826.64**
3,227	**$99,940.19**
Sales Transactions:2	

SalesRep=Hollingsworth Month=04

Units	AmountSold
530	$16,414.10
Sales Transactions:1	

SalesRep=Hollingsworth Month=05

Units	AmountSold
1,120	$34,686.40
1,030	$31,899.10
2,150	**$66,585.50**
2,680	**$82,999.60**
Sales Transactions:2	

SalesRep=Jensen Month=04

Units	AmountSold
1,110	$34,376.70
675	$20,904.75
1,785	**$55,281.45**
1,785	**$55,281.45**
7,692	**$238,221.24**
Sales Transactions:2 Total Sales Transactions:8	

グループの合計の計算

前の例のように複数の BY 変数を使用する場合、BY 変数の値が変わるごとの、小計を非表示にできます。どの BY 変数により小計の表示が行われるか制御するには、SUMBY ステートメントを使用します。

SUMBY 変数は 1 つのみ指定可能で、この変数は BY ステートメントにも指定する必要があります。PROC PRINT は、次の値が変更された場合に合計を計算します。

- SUMBY 変数の値
- BY ステートメントで SUMBY 変数より前に指定される変数の値

例として、次のステートメントについて考えてみます。

```
by Quarter SalesRep Month;
sumby SalesRep;
```

SalesRep が SUMBY 変数です。BY ステートメントでは、Quarter が SalesRep より前、Month が SalesRep の後です。そのため、これらのステートメントでは、PROC PRINT により Quarter または SalesRep のいずれかの値が変わると合計が計算されますが、Month の値が変わっても計算されません。

次のプログラムは、SalesRep を、それによって合計する変数と指定することで、各営業担当者の月ごとの小計を省略します。

```
proc print data=qtr02 sumlabel="Total" grandtotal_label="Grand Total";
   var Units AmountSold;
   where Units>500 or AmountSold>20000;
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   by SalesRep Month;
   id SalesRep Month;
   sumby SalesRep;
   title1 'Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000';
run;
```

このプログラムでは、QTR02 データが変数 SalesRep と Month によって事前に並べ替え済みであると見なします。

レポート出力を次に示します。

図 27.15 オブザベーションのグループの小計の合計

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep	Month	Units	AmountSold
Garcia	04	1,715	$53,113.55

SalesRep	Month	Units	AmountSold
Garcia	06	512	$15,856.64
		1,000	$30,970.00
Total		3,227	$99,940.19

SalesRep	Month	Units	AmountSold
Hollingsworth	04	530	$16,414.10

SalesRep	Month	Units	AmountSold
Hollingsworth	05	1,120	$34,686.40
		1,030	$31,899.10
Total		2,680	$82,999.60

SalesRep	Month	Units	AmountSold
Jensen	04	1,110	$34,376.70
		675	$20,904.75
Total		1,785	$55,281.45
Grand Total		7,692	$238,221.24

複数セクションのオブザベーションのグループ化

PAGEBY ステートメントと BY ステートメントを使用して、複数のセクションがあるレポートを作成することもできます。PAGEBY ステートメントで BY ステートメントの変数を指定し、次の値に変化があった場合に PRINT プロシジャによりレポートが新しいセクションで開始されるようにします。

- BY 変数の値
- BY ステートメントで指定した BY 変数に先行する BY 変数の値

次のプログラムは、PAGEBY ステートメントと BY ステートメントを使用して複数のセクションを持つレポートを作成します。このプログラムでは、QTR02 データが変数 SalesRep と Month によって事前に並べ替え済みであると見なします。

```
proc print data=qtr02 sumlabel="Total" grandtotal_label="Grand Total";
   var Units AmountSold;
   where Units>500 or AmountSold>20000;
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   by SalesRep Month;
   id SalesRep Month;
   sumby SalesRep;
   pageby SalesRep;
   title1 'Sales Rep Quarterly Totals for Sales above 500 Units or $20,000';
run;
```

レポート出力を次に示します。

図 27.16 オブザベーションをグループ化して個別のページに表示

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep	Month	Units	AmountSold
Garcia	04	1,715	$53,113.55

SalesRep	Month	Units	AmountSold
Garcia	06	512	$15,856.64
		1,000	$30,970.00
Total		3,227	$99,940.19

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep	Month	Units	AmountSold
Hollingsworth	04	530	$16,414.10

SalesRep	Month	Units	AmountSold
Hollingsworth	05	1,120	$34,686.40
		1,030	$31,899.10
Total		2,680	$82,999.60

Sales Rep Q2 Monthly Report for Sales Above 500 Units or $20,000

SalesRep	Month	Units	AmountSold
Jensen	04	1,110	$34,376.70
		675	$20,904.75
Total		1,785	$55,281.45
Grand Total		7,692	$238,221.24

変数 SalesRep の値が **Garcia** から **Hollingsworth** へ変わるとき、また **Hollingsworth** から **Jensen** へと変わるときに、レポートで新しいセクションが発生します。

カスタマイズされたレポートの作成

レポートのカスタマイズ方法

前述の例で説明したとおり、PRINT プロシジャを使用して単純なレポートを素早く簡単に作成できます。追加のステートメントとオプションを使用して、レポートを読みやすくできます。たとえば、次の操作を行えます。

- 説明的なタイトルとフットノートの追加。
- 複数の行にわたるラベルの分割と定義。
- ダブルスペースの追加。
- レポートの各ページで列幅が均等になるように指定。
- 出力スタイルの変更。

タイトルとフットノートについて

説明的なタイトルとフットノートを追加することは、レポートの外観を改善する最も効率的で簡単な方法のひとつです。TITLE ステートメントを使用して、1-10 行のテキストをレポートの最上部に含めることができます。FOOTNOTE ステートメントを使用して、1-10 行のテキストをレポートの最下部に含めることができます。

TITLE ステートメントで、キーワード TITLE の直後に *n* を指定して、TITLE ステートメントのレベルを示すことができます。*n* は、1-10 の数値で、TITLE の行番号を示します。各タイトルのテキストを一重引用符または二重引用符で囲む必要があります。

いくつかの *n* の値をスキップした場合、それらの行がブランクであることを示します。たとえば、TITLE1 ステートメントと TITLE3 ステートメントを指定したが TITLE2 をスキップした場合、1 行目と 3 行目の間にブランク行が生成されます。

タイトルが指定されると、SAS では、キャンセルまたはその行が別のタイトルを定義されないかぎり、すべての後続の出力にそのタイトルが使用されます。特定の行に TITLE ステートメントが指定されると、その行、およびその行以降(つまり、*n* の値がより大きい)のすべての行の、以前の TITLE ステートメントがキャンセルされます。

既存のタイトルをすべてキャンセルするには、*n* の値を使用せずに TITLE ステートメントを指定します。

```
title;
```

n とそれ以降のタイトルを非表示にするには、次のステートメントを使用します。

```
titlen;
```

フットノートはタイトルと同様に機能します。FOOTNOTE ステートメントで、キーワード FOOTNOTE の直後に *n* を指定して、FOOTNOTE ステートメントのレベルを示すことができます。*n* は、1-10 の数値で、FOOTNOTE の行番号を示します。各フットノートのテキストを一重引用符または二重引用符で囲む必要があります。TITLE ステートメントと同様に、*n* の値をスキップした場合、スキップした行がブランクであることを示します。

フットノートはレポートの最下部を起点として配置されることに注意してください。つまり、最も行番号が大きい FOOTNOTE ステートメントが最下行に表示されます。

フットノートが指定されると、SAS では、キャンセルまたはその行に別のフットノートが定義されないかぎり、すべての後続の出力にそのフットノートが使用されます。フットノートは、タイトルをキャンセルおよび非表示するのと同様の方法でキャンセル、非表示にできます。

注: タイトルとフットノートの許容最大長は、動作環境と LINESIZE=システムオプションの値によって異なります。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

タイトルとフットノートの追加

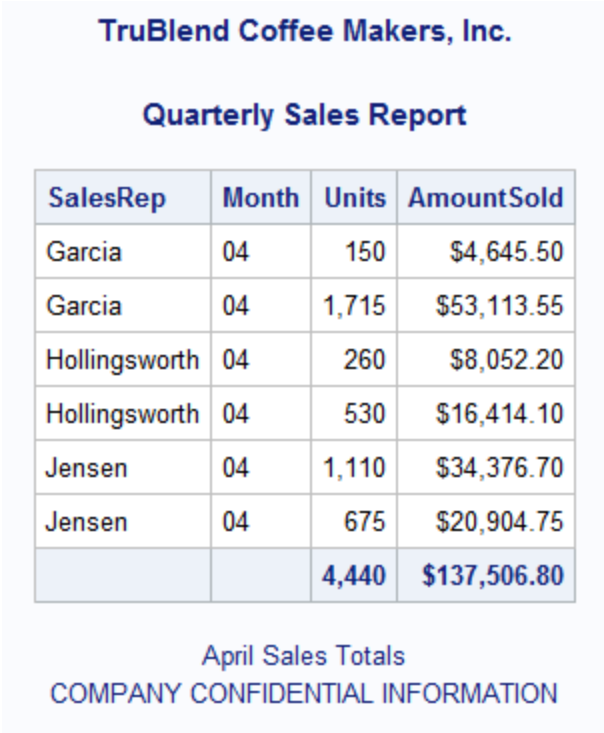
次のプログラムは、4 月の第 2 四半期売上レポートにタイトルとフットノートを含めます。

```
proc sort data=qtr02;
   by SalesRep;
run;

proc print data=qtr02 noobs;
   var SalesRep Month Units AmountSold;
   where Month='04';
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   title1 'TruBlend Coffee Makers, Inc.';
   title3 'Quarterly Sales Report';
   footnote1 'April Sales Totals';
   footnote2 'COMPANY CONFIDENTIAL INFORMATION';
run;
```

レポートには、タイトル行が 3 行とフットノート行が 2 行含まれます。プログラムでは TITLE2 ステートメントが省略されているため、タイトルの 2 行目はブランクです。

レポート出力を次に示します。

図 27.17 タイトルとフットノートの追加



TruBlend Coffee Makers, Inc.

Quarterly Sales Report

SalesRep	Month	Units	AmountSold
Garcia	04	150	$4,645.50
Garcia	04	1,715	$53,113.55
Hollingsworth	04	260	$8,052.20
Hollingsworth	04	530	$16,414.10
Jensen	04	1,110	$34,376.70
Jensen	04	675	$20,904.75
		4,440	$137,506.80

April Sales Totals
COMPANY CONFIDENTIAL INFORMATION

ラベルの定義

デフォルトでは、SAS では変数名が列ヘッダーに使用されます。ただし、レポートの外観を向上させるために、独自の列ヘッダーを指定できます。

デフォルトのヘッダーを上書きするには、次の操作を行う必要があります。

- PROC PRINT ステートメントに LABEL オプションを追加します。
- LABEL ステートメントでラベルを定義します。

LABEL オプションを指定すると、レポートの列ヘッダーに、変数名のかわりにラベルが表示されます。LABEL ステートメントを使用して特定の変数にラベルを割り当てます。ラベルは、ブランクを含めて最長 256 文字で、一重引用符または二重引用符で囲む必要があります。SAS データセット作成時にラベルを割り当てている場合、PROC PRINT ステップから LABEL ステートメントを省略できます。

次のプログラムは、前述のプログラムを変更して、変数 SalesRep、Units、AmountSold のラベルを定義します。

```
proc sort data=qtr02;
   by SalesRep;
run;

proc print data=qtr02 noobs label;
   var SalesRep Month Units AmountSold;
   where Month='04';
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
```

```
   label SalesRep   = 'Sales Rep.'
         Units      = 'Units Sold'
         AmountSold = 'Amount Sold';
   title 'TruBlend Coffee Maker Sales Report for April';
   footnote;
run;
```

TITLE ステートメントにより最初のタイトルが再定義され、以前定義されていた可能性があるその他のタイトルがすべてキャンセルされます。FOOTNOTE ステートメントにより、以前定義されていた可能性があるフットノートがすべてキャンセルされます。

レポート出力を次に示します。

図 27.18 ラベルの定義

TruBlend Coffee Maker Sales Report for April

Sales Rep.	Month	Units Sold	Amount Sold
Garcia	04	150	$4,645.50
Garcia	04	1,715	$53,113.55
Hollingsworth	04	260	$8,052.20
Hollingsworth	04	530	$16,414.10
Jensen	04	1,110	$34,376.70
Jensen	04	675	$20,904.75
		4,440	$137,506.80

複数の行にわたるラベルの分割

ラベルが 1 行に収まるには長すぎて、複数の行にラベルを分割した方がよい場合があります。SPLIT=オプションを使用して、ラベルの複数行への分割場所を制御できます。

SPLIT=オプションが、PROC PRINT ステートメントの LABEL オプションに置き換わります。(SPLIT=により PROC PRINT でラベルが使用されることを意味するため、SPLIT=と LABEL 両方を使用する必要はありません。)SPLIT=オプションで、ラベルの分割場所を示す英数字を指定します。SPLIT=オプションを使用するには、次の操作を行う必要があります。

- PROC PRINT ステートメントの一部として分割文字を定義します。
- LABEL ステートメントでラベルと分割文字を定義します。

次の PROC PRINT ステップは、スラッシュ(/)を分割文字として定義し、LABEL ステートメントにスラッシュを含めて、ラベル Sales Representative、Units Sold と Amount Sold をそれぞれ 2 行に分割します。

```
proc sort data=qtr02;
   by SalesRep;
run;

proc print data=qtr02 noobs split='/';
```

```
   var SalesRep Month Units AmountSold;
   where Month='04';
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   title 'TruBlend Coffee Maker Sales Report for April';
   label SalesRep   = 'Sales/Representative'
         Units      = 'Units/Sold'
         AmountSold = 'Amount/Sold';
   run;
```

レポート出力を次に示します。

図 27.19 *レポート作成: 2 行へのラベルの分割*

TruBlend Coffee Maker Sales Report for April

Sales Representative	Month	Units Sold	Amount Sold
Garcia	04	150	$4,645.50
Garcia	04	1,715	$53,113.55
Hollingsworth	04	260	$8,052.20
Hollingsworth	04	530	$16,414.10
Jensen	04	1,110	$34,376.70
Jensen	04	675	$20,904.75
		4,440	$137,506.80

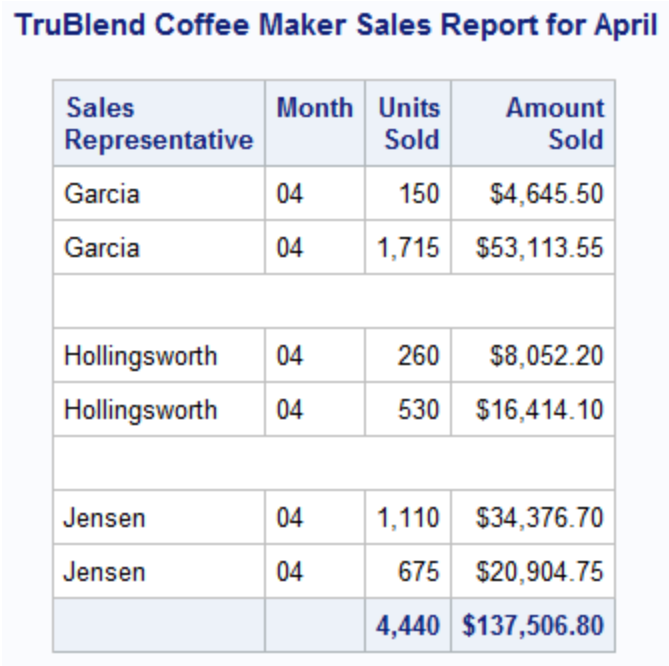
ブランク行の追加

1 つ以上のブランク行をレポートの行間に追加して、レポートの外観を向上できます。次のプログラムでは、PROC PRINT ステートメントで BLANKLINE オプションを使用して、レポートの営業担当者のオブザベーション間に行を追加します。

```
proc sort data=qtr02;
   by SalesRep;
run;

proc print data=qtr02 noobs split='/' blankline=2;
   var SalesRep Month Units AmountSold;
   where Month='04';
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   title 'TruBlend Coffee Maker Sales Report for April';
   label SalesRep   = 'Sales/Representative'
         Units      = 'Units/Sold'
         AmountSold = 'Amount/Sold';
   run;
```

レポート出力を次に示します。

図 27.20 ブランク行の追加



TruBlend Coffee Maker Sales Report for April

Sales Representative	Month	Units Sold	Amount Sold
Garcia	04	150	$4,645.50
Garcia	04	1,715	$53,113.55
Hollingsworth	04	260	$8,052.20
Hollingsworth	04	530	$16,414.10
Jensen	04	1,110	$34,376.70
Jensen	04	675	$20,904.75
		4,440	$137,506.80

レポートスタイルの変更

デフォルトでは、Output Delivery System (ODS)で、HTMLBlue のデフォルトスタイルを使用して HTML 出力先に対してレポートが作成されます。LISTING 出力先以外のすべての ODS 出力先、および出力が作成されない出力先について、レポートスタイルを変更できます。ODS を明示的に使用してレポートスタイルを変更するかわりに、PROC PRINT ステートメントで STYLE=オプションを使用することもできます。STYLE=オプションは ODS と連動してレポートスタイルを変更します。フォント、色、テキスト位置調整、テーブルセルサイズなどのレポート属性を変更できます。STYLE=オプションの構文は、次のとおりです。

STYLE <(*location(s)*)>=<*style-element-name*> <[*style-attribute-name=style-attribute-value*]>

*location(s)*では、レポートで STYLE=オプションの影響が及ぶ部分が識別されます。レポート場所には、テーブル構造、列ヘッダー、セルのデータ、BY ラベル、BY グループ合計とレポート総計を含む SUM 行、N=オプション値、OBS 列のヘッダーとデータが含まれます。

style-element-name は、ODS で登録されたスタイル要素の名前です。スタイル要素は、SAS プログラムの出力の特定部分に適用するスタイル属性の集合です。たとえば、スタイル要素には、列ヘッダーの表示またはテーブルセル内データの表示についての命令などが含まれます。各場所にデフォルトスタイル要素があります。

style-attribute-name=style-attribute-value では、変更するスタイル属性とその値が記述されます。

スタイル要素とスタイル属性のリストについては、*SAS Output Delivery System: User's Guide* を参照してください。

次のプログラムでは、前述のプログラムが変更されて、テーブルのセルスペース、ヘッダーの色とフォントスタイル、および総計が変更されます。ヘッダーのスタイルは、headerstrong スタイル要素に基づきます。

```
proc sort data=qtr02;
  by SalesRep;
run;

proc print data=qtr02 noobs split='/'
           style(table)={cellpadding=10}
           style(header)=headerstrong{backgroundcolor=very light green
                                      color=green fontstyle=italic}
           style(grandtotal)={backgroundcolor=very light green color=green};
   var SalesRep Month Units AmountSold;
   where Month='04';
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   label SalesRep= 'Sales/Representative'
        Units = 'Units/Sold'
            AmountSold= 'Amount/Sold';
   title 'TruBlend Coffee Maker Sales Report for April';
run;
```

レポート出力を次に示します。

図 27.21 *レポートスタイルの変更*

TruBlend Coffee Maker Sales Report for April

Sales Representative	Month	Units Sold	Amount Sold
Garcia	04	150	$4,645.50
Garcia	04	1,715	$53,113.55
Hollingsworth	04	260	$8,052.20
Hollingsworth	04	530	$16,414.10
Jensen	04	1,110	$34,376.70
Jensen	04	675	$20,904.75
		4,440	$137,506.80

レポートを変更しやすくする

SAS マクロ機能について

Base SAS には、SAS をカスタマイズして、共通タスクの実行に必要な入力テキストを減らすためのツールとしてマクロ機能が含まれています。マクロ機能を使用して、文字列または SAS プログラミングステートメントのグループに名前を割り当てることができます。

これ以降は、テキスト自体ではなく、名前を使用して作業できます。SAS プログラムでマクロ機能名を使用すると、必要に応じて、マクロ機能により SAS ステートメントや SAS コマンドが生成されます。その後、SAS はそれらのステートメントを受け取り、通常の方法で入力したステートメントと同様に使用します。

マクロ機能を使用して、SAS プログラムでテキストを置き換えるためのマクロ変数を作成できます。マクロ変数を使用する大きな利点の 1 つは、これを使用してプログラムの 1 か所で変数の値を変更すると、プログラム全体を通じ複数の参照に変更を反映できることです。自動マクロ変数、または(ユーザーが定義して値を割り当てる)独自のマクロ変数を使用して、テキストを置き換えることができます。

自動マクロ変数の使用

SAS マクロ機能には、多数の自動マクロ変数が含まれています。自動マクロ変数に関連付けられている値の一部は、ユーザーの動作環境によって異なります。自動マクロ変数を使用し、コンピュータの内部時計やその他の処理情報に基づいて、時刻、曜日、日付を提供できます。

テキスト文字列“Produced on”とそれに続いて今日の日付を表示する 2 行目のタイトルをレポートに含めるには、次の TITLE ステートメントをプログラムに追加します。

```
title2 "Produced on &SYSDATE9";
```

このステートメントの構文に注意してください。最初に、SYSDATE9 に先行するアンパサンドにより、SAS マクロ機能にその割り当てられている値で参照を置き換えるように指示されます。この場合、割り当てられている値は SAS セッションが開始された日付で、*ddmmmyyyy* で表されます。

- *dd* は、2 桁の日付です。
- *mmm* は、月名の最初の 3 文字になります。
- *yyyy* は、4 桁の年です。

次に、TITLE ステートメントのテキストが二重引用符で囲まれています。TITLE ステートメントと FOOTNOTE ステートメントのマクロ変数の参照が二重引用符で囲まれている場合のみ、SAS マクロ機能によりそれらが解決されるためです。

次の PROC SORT ステップと TITLE ステートメントを含むプログラムでは、SYSDATE9 自動マクロ変数の使用方法を示します。

```
proc sort data=qtr04;
   by SalesRep;
run;

proc print data=qtr04 noobs split='/';
   var SalesRep Month Units AmountSold;
   format Units comma7. AmountSold dollar14.2;
```

```
   sum Units AmountSold;
   title1 'TruBlend Coffee Maker Quarterly Sales Report';
   title2 "Produced on &SYSDATE9";
   label SalesRep    = 'Sales/Rep.'
         Units       = 'Units/Sold'
         AmountSold  = 'Amount/Sold';
run;
```

レポート出力を次に示します。

図 27.22 自動マクロ変数の使用

TruBlend Coffee Maker Quarterly Sales Report
Produced on 09APR2013

Sales Rep.	Month	Units Sold	Amount Sold
Garcia	10	250	$7,742.50
Garcia	10	365	$11,304.05
Garcia	11	198	$6,132.06
Garcia	11	120	$3,716.40
Garcia	12	1,000	$30,970.00
Hollingsworth	10	530	$16,414.10
Hollingsworth	10	265	$8,207.05
Hollingsworth	11	1,230	$38,093.10
Hollingsworth	11	150	$7,425.00
Hollingsworth	12	125	$6,187.50
Hollingsworth	12	175	$5,419.75
Jensen	10	975	$30,195.75
Jensen	10	55	$1,703.35
Jensen	11	453	$14,029.41
Jensen	11	70	$2,167.90
Jensen	12	876	$27,129.72
Jensen	12	1,254	$38,836.38
		8,091	$255,674.02

独自のマクロ変数の使用

自動マクロ変数の使用に加え、%LET ステートメントを使用して独自のマクロ変数を定義し、それらをアンパサンド接頭辞で参照できます。プログラムの開始部分でマクロ変

数を定義すると、プログラムの他の部分を簡単に変更できます。このセクションの例では、2 つのマクロ変数 Quarter と Year を定義する方法、および TITLE ステートメントでそれらを参照する方法を示します。

マクロ変数の定義

柔軟なレポートタイトルを生成する 2 つのマクロ変数を使用するには、最初に、マクロ変数を定義します。次の%LET ステートメントは 2 つのマクロ変数を定義します。

```
%let Quarter=Fourth;
%let Year=2011;
```

最初のマクロ変数の名前は Quarter で、値 Fourth が割り当てられます。2 番目のマクロ変数の名前は Year で、値 2011 が割り当てられます。

これらのマクロ変数名は、次の SAS 名の規則に従います。

- マクロ変数名の長さは、1-32 文字です。
- マクロ変数名は、文字またはアンダースコアで開始します。
- 文字、数値やアンダースコアが最初の文字に続きます。

このような単純な状況では、不一致の引用符またはセミコロンを含む値を、マクロ変数に割り当てないでください。値に先行ブランクまたは後置ブランクが含まれる場合、SAS によりブランクが削除されます。

マクロ変数の参照

マクロ変数の値を参照するには、変数名の前にアンパサンド接頭辞を配置します。次の TITLE ステートメントには、前に%LET ステートメントで定義したマクロ変数 Quarter と Year の値への参照が含まれます。

```
title3 "&Quarter Quarter &Year Sales Totals";
```

2 つの%LET ステートメントと TITLE3 ステートメントを含む完全なプログラムを次に示します。

```
%let Quarter=Fourth;  1
%let Year=2013;  2

proc sort data=qtr04;
   by SalesRep;
run;

proc print data=qtr04 noobs split='/' width=uniform;
   var SalesRep Month Units AmountSold;
   format Units comma7. AmountSold dollar14.2;
   sum Units AmountSold;
   title1 'TruBlend Coffee Maker Quarterly Sales Report';
   title2 "Produced on &SYSDATE9";
   title3 "&Quarter Quarter &Year Sales Totals";  3
   label SalesRep    = 'Sales/Rep.'
         Units       = 'Units/Sold'
         AmountSold  = 'Amount/Sold';
run;
```

1 %LET ステートメントにより、売上四半期を使用してマクロ変数が作成されます。アンパサンドが Quarter に先行すると、SAS マクロ機能により&Quarter への参照がすべて、割り当てられた値である Fourth で置き換えられます。

2 %LET ステートメントにより、年を使用してマクロ変数が作成されます。アンパサンドが Year に先行すると、SAS マクロ機能により&Year への参照がすべて、割り当てられた値である 2011 で置き換えられます。

3 TITLE2 ステートメントと TITLE3 ステートメントのテキストは二重引用符で囲まれているため、SAS マクロ機能によって解決されます。

次のリストは、前述のプログラムの番号付き項目に対応しています。

レポート出力を次に示します。

図 27.23 *独自のマクロ変数の使用*

TruBlend Coffee Maker Quarterly Sales Report
Produced on 09APR2013
Fourth Quarter 2012 Sales Totals

Sales Rep.	Month	Units Sold	Amount Sold
Garcia	10	250	$7,742.50
Garcia	10	365	$11,304.05
Garcia	11	198	$6,132.06
Garcia	11	120	$3,716.40
Garcia	12	1,000	$30,970.00
Hollingsworth	10	530	$16,414.10
Hollingsworth	10	265	$8,207.05
Hollingsworth	11	1,230	$38,093.10
Hollingsworth	11	150	$7,425.00
Hollingsworth	12	125	$6,187.50
Hollingsworth	12	175	$5,419.75
Jensen	10	975	$30,195.75
Jensen	10	55	$1,703.35
Jensen	11	453	$14,029.41
Jensen	11	70	$2,167.90
Jensen	12	876	$27,129.72
Jensen	12	1,254	$38,836.38
		8,091	$255,674.02

マクロ変数を使用すると、プログラムを簡単に変更できます。たとえば、前述のプログラムに Quarter と Year への参照が多数含まれている場合、次の 3 か所を変更するだけで、まったく異なるレポートを生成できます。

- %LET ステートメントの 2 つの値

- PROC PRINT ステートメントのデータセット名

要約

PROC PRINT ステートメント

PROC PRINT <DATA=*SAS-data-set*> <*option(s)*>;
BY *variable(s)*;
FOOTNOTE<*n*> <*'footnote'*>;
FORMAT *variable(s) format-name*;
ID *variable(s)*;
LABEL *variable='label'*;
PAGEBY *variable*;
SUM *variable(s)*;
SUMBY *variable*;
TITLE<*n*> <*'title'*>;
VAR *variable(s)*;
WHERE *where-expression*;

PROC PRINT <DATA=*SAS-data-set*> <*option(s)*>;
プロシジャを開始し、単独で使用された場合に、レポートに SAS-data-set のすべてのオブザベーションの全変数を表示します。次にリストにする他のステートメントを使用して、レポートの対象を制御できます。

PROC PRINT ステートメントでは、次のオプションを使用できます。

BLANKLINE=*n* | BLANKLINE=(COUNT=*n* <STYLE=[*style-attribute-specification(s)*]>)
n オブザベーションごとにブランク行を挿入するように指定します。すべての ODS 出力先についてすべての BY グループの最初にオブザベーション数が 0 にリセットされます。

DATA=*SAS-data-set*
PROC PRINT で使用される SAS データセットの名前を指定します。DATA=が省略されると、PROC PRINT は最後に作成されたデータセットを使用します。

GRANDTOTAL_LABEL="*label*"
総計行にラベルが表示されます。

LABEL
ラベルが定義されているすべての変数について、変数名のかわりに変数ラベルが列ヘッダーとして使用されます。変数ラベルは、LABEL オプションまたは SPLIT=オプションが使用される場合のみ表示されます。PROC PRINT ステップや、データセットを作成する DATA ステップの LABEL ステートメントでラベルを指定します。LABEL オプションが指定されない場合、または変数のラベルが存在しない場合、PROC PRINT は変数名を使用します。

N<="*string-1*" <"*string-2*">>
データセットのオブザベーション数または BY グループのオブザベーション数、あるいはその両方が表示されます。数値と一緒に含める説明テキストを指定することもできます。

NOOBS
出力でオブザベーション番号を非表示にします。このオプションは、ID ステートメントを省略し、オブザベーション番号を表示する予定がない場合に便利です。

OBS="*column-header*"
各オブザベーションを番号によって識別する列の列ヘッダーを指定します。

SPLIT='*split-character*'
列ヘッダーの改行を制御する分割文字を指定します。PROC PRINT は、分割文字に達すると列ヘッダーを改行し、次の行にヘッダーを続けます。分割文字は列ヘッダーの一部ではありません。

PROC PRINT は、LABEL オプションまたは SPLIT=オプションが使用された場合にのみ、変数ラベルを使用します。SPLIT=はラベルの使用を意味するため、LABEL オプションと SPLIT=オプションの両方を使用する必要はありません。

STYLE=<*location(s)*>=<*style-element-name*> <[*style-attribute-specification(s)*]>
レポートのさまざまな部分に使用する Output Delivery System のスタイル要素を 1 つ以上指定します。

SUMLABEL="*label*"
要約行で BY 変数名のかわりにラベルが表示されます。

WIDTH=UNIFORM
各変数のフォーマットされた幅をその列幅としてすべてのページで使用します。変数にフィールド幅を明示的に指定する出力形式が適用されていない場合、PROC PRINT は最も幅が広いデータ値を列幅として使用します。このオプションがない場合、PROC PRINT は可能な限り多くの変数とオブザベーションを 1 ページに収めようとします。そのため、レポートのページごとに含まれる列の数が異なる場合があります。

BY *variable(s)*;
BY グループごとにレポートの個別のセクションを生成します。BY グループは、指定する *variables* で構成されます。BY ステートメントを使用すると、プロシジャは入力データセットが *variables* で並べ替えられているものと見なします。

FOOTNOTE<*n*> <'*footnote*'>;
フットノートを指定します。引数 *n* は、ワード FOOTNOTE の直後にブランクを空けず続く 1-10 の数値で、FOOTNOTE の行番号を指定します。各フットノートのテキストは一重引用符または二重引用符で囲む必要があります。フットノートの許容最大長は、動作環境と LINESIZE=システムオプションの値によって異なります。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

FORMAT *variable(s) format-name*;
format-name として指定する特殊なパターンを使用して、*variable* の値をレポートに記述します。

ID *variable(s)*;
PROC PRINT でオブザベーション番号のかわりに使用される、レポートでオブザベーションを識別するための 1 つ以上の変数を指定します。

LABEL *variable*='*label*';
列ヘッダーにラベルを使用するように指定します。*Variable* にラベル付けする変数名を指定し、ブランクを含めて最大 256 文字の文字列を *label* に指定します。*label* は、一重引用符または二重引用符で囲む必要があります。

PAGEBY *variable*;
PROC PRINT で、指定の *variable* の値が変わるとき、または BY ステートメントでその変数より先に記述されている変数の値が変わるとき、新しいページを開始しま

す。PAGEBY ステートメントは BY ステートメントとあわせて使用する必要があります。

SUM *variable(s)*;
レポートで総計する数値変数を指定します。SUM ステートメントで変数を指定すると、PROC PRINT により VAR リストに変数が追加されるため、VAR ステートメントでその変数を省略できます。PROC PRINT により BY 変数と ID 変数を総計する要求が無視されます。一般に、BY ステートメントも一緒に使用する場合は、SUM ステートメントによって BY 変数の値が変わるたびに小計が生成されます。

SUMBY *variable*;
レポートに表示される合計の数を制限します。PROC PRINT は、*variable* の値が変わるときか、または BY ステートメントでその変数より先に記述されている変数の値が変わるときのみ、合計をレポートに記述します。SUMBY ステートメントは BY ステートメントとあわせて使用する必要があります。

TITLE<*n*> <'*title*'>;
タイトルを指定します。引数 *n* は、ワード TITLE の直後にブランクを空けずに続ける 1-10 の数値で、TITLE のレベルを指定します。各 *title* のテキストは一重引用符または二重引用符で囲む必要があります。指定可能なタイトルの最大長は、ご使用の動作環境や LINESIZE=システムオプションの値によって異なります。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

VAR *variable(s)*;
レポートに表示する 1 つ以上の変数を指定します。変数は VAR ステートメントに記述される順番で表示されます。VAR ステートメントを省略すると、すべての変数がレポートに表示されます。

WHERE *where-expression*;
各オブザベーションが処理に使用できる前に一致する必要がある特定の条件を指定し、入力データセットをサブセット化します。*Where-expression* は条件を定義します。条件は、有効な算術式または論理式で、通常、一連のオペランドと演算子で構成されます。

PROC SORT ステートメント

PROC SORT <DATA=*SAS-data-set*>;

BY *variable(s)*;

PROC SORT DATA=*SAS-data-set*;
SAS データセットを BY ステートメントに記述した変数の値で並べ替えます。

BY *variable(s)*;
PROC SORT でオブザベーションの並べ替えに使用される 1 つ以上の変数を指定します。デフォルトでは、PROC SORT によりデータセットが値の昇順(最小値から最大値へ)で配置されます。

SAS マクロ言語

%LET *macro-variable=value*;
マクロステートメントで、macro-variable を定義してこれに value を割り当てます。%LET ステートメントで定義される value が、出力で macro-variable と置き換えられます。プログラムで macro-variable を使用するには、その前にアンパサンド(&)接頭辞を含めます。

SYSDATE9
SAS ジョブまたは SAS セッションの実行開始日付を含む自動マクロ変数です。SYSDATE9 には DATE9 出力形式(ddmmmyyyy)で SAS 日付値が含まれます。日付には、2 桁の日付、月名の最初の 3 文字、4 桁の西暦年が表示されます。プログラムで使用するには、SYSDATE9 の前にアンパサンド(&)接頭辞を含めます。

詳細情報

データセットのインデックス
データセットのインデックス付けの詳細については、*SAS データセットオプション: リファレンス*を参照してください。データセットに BY ステートメントで指定されている変数のインデックスがある場合、PRINT プロシジャでの BY ステートメント使用前にデータセットを並べ替える必要はありません。

スタイル要素とスタイル属性
スタイル要素とスタイル属性の詳細については、*SAS Output Delivery System: User's Guide* を参照してください。

PROC PRINT
詳細なドキュメントについては、*Base SAS Procedures Guide* を参照してください。

PROC SORT
詳細については、12 章, “グループ化されたオブザベーションや並べ替えられたオブザベーションの操作” (181 ページ)を参照してください。SORT プロシジャの詳細なリファレンスドキュメントについては、*Base SAS Procedures Guide* を参照してください。

SAS 出力形式
詳細なドキュメントについては、*SAS Formats and Informats: Reference* を参照してください。SAS で使用可能な出力形式には、小数値、16 進値、ローマ数字、社会保障番号、日付と時刻、および文字として記述される数値があります。

SAS マクロ機能
詳細なリファレンスドキュメントについては、*SAS Macro Language: Reference* を参照してください。

WHERE ステートメント
詳細なリファレンスドキュメントについては、*SAS Statements: Reference* を参照してください。WHERE 処理の詳細については、*SAS Language Reference: Concepts* を参照してください。

28 章

TABULATE プロシジャを使用した要約テーブルの作成

TABULATE プロシジャを使用した要約テーブルの作成について **462**
目的 462
前提条件 462

要約テーブルの設計について **462**

TABULATE プロシジャの基礎知識 **464**
TABULATE プロシジャの必須ステートメント 464
PROC TABULATE ステートメントから開始 465
CLASS ステートメントでの分類変数の指定 465
VAR ステートメントでの分析変数の指定 465
TABLE ステートメントでのテーブル構造の定義 465
分類変数の欠損値の特定 466

例で使用される入力ファイルおよび入力 SAS データセット **467**

単純な要約テーブルの作成 **468**
基本の 1 次元要約テーブルの作成 468
基本の 2 次元要約テーブルの作成 469
基本の 3 次元要約テーブルの作成 470
1 つの PROC TABULATE ステップによる複数のテーブルの生成 471

複雑な要約テーブルの作成 **473**
サブグループのレポート用の階層テーブルの作成 473
出力のフォーマッティング 475
記述統計量の計算 475
複数統計量のレポートの作成 476
コードの削減と複数の要素への単一ラベルの適用 477
すべての変数の要約の取得 478
ラベルの定義 479
スタイルと Output Delivery System の使用 480
分類変数の並べ替え 483

要約 **484**
グローバルステートメント 484
TABULATE プロシジャステートメント 484

詳細情報 **487**

TABULATE プロシジャを使用した要約テーブルの作成について

目的

要約テーブルは、1 つのデータセットの変数間に存在する関係を表示します。データセットの変数により、要約テーブルの列、行およびページが形成されます。列と行の各交点(各セル)にあるデータは、変数と変数との間の関係を示します。TABULATE プロシジャを使用して、さまざまな要約テーブルを作成できます。

このセクションでは、次の操作を行う方法を学習します。

- いくつかの PROC TABULATE の基本オプションとステートメントを使用して単純な要約テーブルを生成します。
- 変数間のより複雑な関係を要約し、変数への出力形式適用、変数の統計量の計算を行って、拡張された要約テーブルを作成します。
- ラベルの使用、Output Delivery System を使用したフォントと色の指定、および分類変数の並べ替えを行ってテーブルの外観を整えます。

前提条件

このセクションの例を理解するために、次の機能と概念を理解していることを確認してください。

- “プロシジャ出力の出力先” (603 ページ)
- “タイトルとフットノートの追加” (447 ページ)

要約テーブルの設計について

前もって要約テーブルを設計すると、時間を節約することが可能で、より簡単な SAS コードの記述で要約テーブルを作成できます。要約テーブルの設計と構築の基本ステップを次に示します。詳細な設計プロセスの手順を追った例については、*PROC TABULATE by Example* を参照してください。

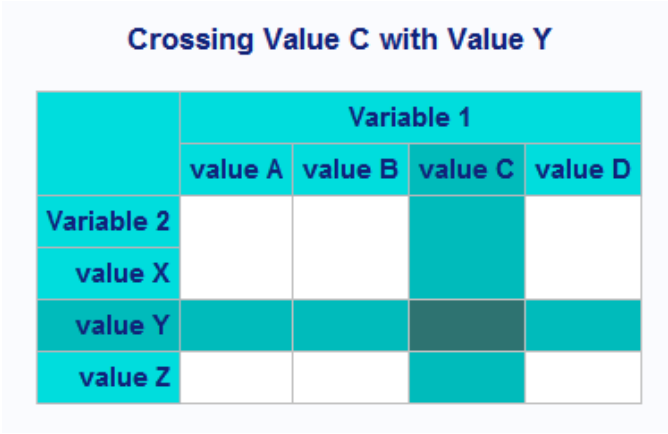
要約テーブルの設計の前に、2 つ以上の変数の値が交差する場合は常に、要約テーブルにより要約データが生成されることを理解しておくことが重要です。交点は、セルです。2 つ以上の変数の値が交差する場合、その変数は交差と言います。交差する変数が交点を形成するプロセスは、クロス集計と呼ばれます。列、行およびページの変数は交差して要約データを生成できます。次のプログラムでは、2 つの変数がどのように交差するかを示す要約テーブルを作成します。入力ファイルおよび SAS データセットについては、“例で使用される入力ファイルおよび入力 SAS データセット” (467 ページ)を参照してください。

```
proc format;
   value $slrpclr 'Hollingsworth'='cx00BBBB'
                  other ='cx00DDDD';
   value $slrpmsk 'Hollingsworth'='value Y'
                  'Garcia'       ='value X'
```

```
                'Jensen'       ='value Z';
   value $quarclr '3'='cx00BBBB'
                  other ='cx00DDDD';
   value $quarmsk '1'='value A'
                  '2'='value B'
                  '3'='value C'
                  '4'='value D';
   value sumclr   222290-222291 ='cx00BBBB'
                  225326-225327 ='cx00BBBB'
                  108900-110000 ='cx2E7371'
                   81000- 82000 ='cx00BBBB'
                   96000- 97000 ='cx00BBBB'
                   59000- 59650 ='cx00BBBB'
                  other ='w';
run;

ods html body='/dept/pub/doc/701misc/authoring/TW6025/sgml/delete.htm';
proc tabulate data=year_sales format=comma10.;
   title 'Crossing Value C with Value Y';
   class SalesRep Quarter/ style=[background=cx00DDDD];
   classlev salesrep / style=[background=$slrpclr. just=right];
   classlev quarter / style=[background=$quarclr.];
   var AmountSold / style=[background=cx00DDDD];
   table SalesRep='Variable 2',
         (AmountSold='Variable 1'*sum=' '*Quarter=' ')*
         [s=[background=sumclr. foreground=sumclr.]] /
         box={style={background=cx00DDDD}};
   format salesrep $slrpmsk. Quarter $quarmsk.;
run;
ods html close;
```

次の要約テーブルに、2 つの変数が交差する様子をそれぞれの変数の単一の値を強調して表示します。

図 28.1　交差変数



次に、要約テーブルを設計して構築する基本ステップを示します。

1. 要約テーブルを使用して答えを求めたい質問より開始します。
2. 質問に答えるために必要な変数を特定します。

- 特定した変数が、使用中のデータセットのいずれかですでに使用されているかどうか確認します。使用されていない場合、これらのデータセットの変数の値を、FORMAT プロシジャを使用して再分類できる可能性があります。その場合はこれにより、必要なデータを生成できます。

  たとえば、変数 MONTH の値に新しい出力形式を適用して、変数 QUARTER の値にすることができます。これを行うには、最初の 3 か月を表している値を第 1 四半期の値に割り当て、2 番目の 3 か月のセットを表している値を第 2 四半期に、といった方法で続けます。

- 可能であれば、カテゴリやヘッダーには、連続変数でなく、離散変数を使用します。連続変数を使用する必要がある場合、カテゴリを作成すると役立ちます。たとえば、年齢を、年齢 15-19、年齢 20-35、年齢 36-55、年齢 56 以上などのカテゴリにグループ化できます。これにより、およそ 56 以上ものカテゴリではなく、4 つのカテゴリが作成されます。PROC FORMAT を使用して、データを分類できます。

- 要約テーブルに表示する変数とデータの出力形式を選択します。データセットのデータが使用可能な出力形式であるか確認してください。PROC FORMAT を使用して新しい出力形式を作成するか、別のデータセットから変数の出力形式をコピーしてデータが同様にフォーマットされるようにする必要がある場合があります。

3. レポートのディスクレパンシの原因となりうるものがないか、データを確認します。
   - ニーズに関連のないデータを削除します。
   - 欠損データを特定します。
   - データ全体に論理的整合性があるか確認します。
4. 質問の答えを得るのに有用な統計量を選択します。統計量の詳細なリストについては、*Base SAS Procedures Guide* の“Statistics Available in PROC TABULATE”を参照してください。
5. テーブルの基本構造を決定します。特定した変数を使用して、列、行、ページのヘッダーを決定します。変数の値はサブヘッダーとなります。統計量は、通常、サブヘッダーとして表されますが、ヘッダーとして表される場合もあります。交差変数は、非常に基本的なテーブルのテンプレートの例です。
6. テーブルのスタイルを決定します。ODS を使用して、出力の見栄えをカスタマイズできます。ODS (Output Delivery System)の詳細については、*SAS Output Delivery System: User's Guide* を参照してください。

TABULATE プロシジャの基礎知識

TABULATE プロシジャの必須ステートメント

TABULATE プロシジャでは、3 つのステートメントが、通常、次の順序で必要です。

1. PROC TABULATE ステートメント
2. CLASS ステートメントまたは VAR ステートメント、あるいはその両方
3. TABLE ステートメント

CLASS ステートメント、VAR ステートメント、TABLE ステートメントは複数使用できます。

PROC TABULATE ステートメントから開始

TABULATE プロシジャは PROC TABULATE ステートメントから開始します。PROC TABULATE ステートメントでは多数のオプションが使用可能です。しかしながら、このセクションのほとんどの例では、2 つのオプション、DATA=オプションと FORMAT=オプションのみを使用します。次の PROC TABULATE ステートメントは、このセクションの例のすべてで使用されます。

```
proc tabulate data=year_sales format=comma10.;
```

DATA=オプションを使用して、PROC TABULATE で特定の SAS データセットを使用するように指定できます。現在のジョブまたはセッションで DATA=オプションを省略すると、TABULATE プロシジャは最後に作成された SAS データセットを使用します。

FORMAT=オプションを使用して、PROC TABULATE にデフォルト出力形式を指定して、テーブル内の各セルの値に適用できます。有効な SAS 数値の出力形式またはユーザー定義の出力形式を指定できます。

CLASS ステートメントでの分類変数の指定

CLASS ステートメントを使用して、分類変数とする変数を指定します。分類変数には、カテゴリを形成するのに使用される値が含まれています。要約テーブルでは、カテゴリが列、行およびページのヘッダーとして使用されます。カテゴリは交差して、記述統計量を取得します。交差するカテゴリ(変数の値)の例については、図 28.1 (463 ページ)を参照してください。

分類変数は、文字または数値のいずれかです。分類変数のデフォルトの統計量は N で、変数の欠損値がないデータセットのオブザベーションの数または度数です。

次の CLASS ステートメントは、変数 SalesRep と Type を分類変数として指定します。

```
class SalesRep Type;
```

欠損値を持つ分類変数が CLASS ステートメントに記述されているが TABLE ステートメントで使用されていない場合の、PROC TABULATE の動作の仕様についての重要な情報は、“分類変数の欠損値の特定” (466 ページ)を参照してください。

VAR ステートメントでの分析変数の指定

VAR ステートメントを使用して、分析変数とする変数を指定します。分析変数には、統計量を計算する数値が含まれます。分析変数のデフォルトの統計量は、SUM です。

次の VAR ステートメントは、分析変数として、変数 AmountSold を指定しています。

```
var AmountSold;
```

TABLE ステートメントでのテーブル構造の定義

TABLE ステートメントの構文

TABLE ステートメントを使用して、PROC TABULATE で生成するテーブルの構造を定義します。TABLE ステートメントは、カンマで区切られた 1 次元式から 3 次元式で構成されます。次元式により要約テーブルの列、行、ページが定義されます。オプションがある場合は次元式の後に続きます。PROC TABULATE ステップにはデフォルトのテーブルがないため、少なくとも 1 つの TABLE ステートメントを指定する必要があります。次に基本の TABLE ステートメントの 3 つの応用を示します。

```
TABLE column-expression;
TABLE row-expression, column-expression;
TABLE page-expression, row-expression, column-expression;
```

この構文における規則:

- 列の式は必須です。
- 行の式はオプションです。
- ページの式はオプションです。
- 式は、ページの式、行の式、列の式の順番である必要があります。

次に、3 次元式での基本的な TABLE ステートメントの例を示します。

```
table SalesRep, Type, AmountSold;
```

この TABLE ステートメントは 3 次元の要約テーブルを定義し、次のものを配置します。

- 変数 AmountSold の値を列の次元へ
- 変数 Type の値を行の次元へ
- 変数 SalesRep の値をページの次元へ

TABLE ステートメントの制限

次に、TABLE ステートメントの制限を示します。

- TABLE ステートメントでは、列の次元は必須です。
- TABLE ステートメントの次元式で使用される変数はすべて、CLASS ステートメントまたは VAR ステートメントのいずれかに含められる必要がありますが、両方に含める必要はありません。
- すべての分析変数は同じ次元にある必要があり、交差できません。そのため、TABLE ステートメントの 1 つの次元のみに分析変数を含めることができます。

分類変数の欠損値の特定

MISSING オプションで分類変数の欠損値を特定できます。デフォルトでは、オブザベーションにいずれかの分類変数で欠損値が含まれていると、そのオブザベーションはすべてのテーブルから除外されます。その変数が、1 つ以上のテーブルに対する TABLE ステートメントに使用されていない場合でも、その値は除外されます。そのため、少なくとも 1 度 MISSING オプションを使用してプログラムを実行して欠損値を特定することは有用です。

MISSING オプションは、要約テーブルに欠損値用のカテゴリを別に作成します。PROC TABULATE ステートメントまたは CLASS ステートメントで使用できます。次のように PROC TABULATE ステートメントで MISSING オプションを指定すると、そのプロシジャでは欠損値をすべての分類変数の有効な水準として見なします。

```
proc tabulate data=year_sales format=comma10. missing;
   class SalesRep;
   class Month Quarter;
   var AmountSold;
```

この例では、PROC TABULATE ステートメントに MISSING オプションがあるため、SalesRep、Month または Quarter の欠損値を持つオブザベーションが要約テーブルに表示されます。

次のように CLASS ステートメントで MISSING オプションを指定すると、PROC TABULATE では、欠損値をその CLASS ステートメントで指定される分類変数の有効な水準として見なします。

```
proc tabulate data=year_sales format=comma10.;
   class SalesRep;
   class Month Quarter / missing;
   var AmountSold;
```

この例では、MISSING オプションが 2 番目の CLASS ステートメントで使用されているため、Month または Quarter の欠損値を持つオブザベーションは要約テーブルに表示されますが、SalesRep の欠損値を持つオブザベーションは表示されません。

データセットに欠損値を持つ分類変数がある場合、欠損値を持つオブザベーションをすべてのテーブルから省略するかどうかを決定する必要があります。欠損値を持つオブザベーションを省略しない場合は、適切な場所に欠損値を書き込むか、または MISSING オプションを使用して PROC TABULATE ステップの実行を継続できます。欠損値の処理に関するその他のオプションについては、*PROC TABULATE by Example* の“Handling Missing Data”を参照してください。欠損値に関する全般的な情報については、*SAS Language Reference: Concepts* の“Missing Values”を参照してください。

例で使用される入力ファイルおよび入力 SAS データセット

このセクションの例では、1 つの入力ファイルと 1 つの SAS データセットを使用します。詳細については、“YEAR_SALES データセット” (794 ページ)を参照してください。

入力ファイルには、コーヒーマシンを供給する会社 TruBlend Coffee Makers の売上記録が含まれています。ファイルの構造は次のとおりです。

```
01      1      Hollingsworth  Deluxe      260      49.50
01      1      Garcia         Standard     41      30.97
01      1      Hollingsworth  Deluxe      330      49.50
01      1      Jensen         Standard   1110      30.97
01      1      Garcia         Standard    715      30.97
01      1      Jensen         Deluxe      675      49.50
02      1      Jensen         Standard     45      30.97
02      1      Garcia         Deluxe       10      49.50

…more data lines…

12      4      Hollingsworth  Deluxe      125      49.50
12      4      Jensen         Standard   1254      30.97
12      4      Hollingsworth  Deluxe      175      49.50
```

入力ファイルには、次に示すデータが左から右へと含まれています。

- 売上が発生した月
- 売上が発生した年の四半期
- 営業担当者の名前
- 売り上げたコーヒーメーカーの種類(Standard または Deluxe)
- 販売ユニット数
- 各ユニットの価格(米ドル)

SAS データセットの名前は YEAR_SALES です。このデータセットには、入力ファイルからのすべての売上データ、および Units を Price で乗算することにより作成される AmountSold という名前の新しい変数が含まれます。

次のプログラムは、このセクションで使用する SAS データセットを作成します。

```
data year_sales;
   infile 'your-input-file';
   input Month $ Quarter $ SalesRep $14. Type $ Units Price;
   AmountSold = Units * Price;
run;
```

単純な要約テーブルの作成

基本の 1 次元要約テーブルの作成

最も単純な要約テーブルには、複数の列と行が 1 つのみ含まれます。これは、列の次元のみを持つため、1 次元要約テーブルと呼ばれます。次の PROC TABULATE ステップは、1 次元要約テーブルを作成し、各営業担当者が何回売り上げたか、を調べることができます。

```
   options linesize=84 pageno=1 nodate;

   proc tabulate data=year_sales format=comma10.;
      title1 'TruBlend Coffee Makers, Inc.';
      title2 'Number of Sales by Each Sales Representative';
      class SalesRep; 1
      table SalesRep; 2
   run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 変数 SalesRep は、分類変数として CLASS ステートメントで指定されています。TABLE ステートメントで SalesRep が使用されるたびに、SalesRep の各値に対してカテゴリが作成されます。

2 変数 SalesRep は、TABLE ステートメントの列の次元に指定されています。SalesRep のそれぞれのカテゴリに対して列が作成されます。各列には、そのカテゴリに属する値がそのデータセットに出現する回数(N)が表示されます。

次の要約テーブルにこのプログラムの結果を示します。

図 28.2 *基本の 1 次元要約テーブル*

TruBlend Coffee Makers, Inc.
Number of Sales by Each Sales Representative

SalesRep		
Garcia	Hollingsworth	Jensen
N	N	N
40	32	38

値 40、32 と 38 は、それぞれの営業担当者の名前(Garcia、Hollingsworth と Jensen)がデータセットに出現した度数です。このデータセットの場合、データセットでの営業担当者の名前のそれぞれの出現は、1 つの売り上げを表します。

基本の 2 次元要約テーブルの作成

最もよく使用される要約テーブルの形式は、少なくとも 1 つの列と複数の行を持ち、2 次元要約テーブルと呼ばれます。次の PROC TABULATE ステップは、2 次元要約テーブルを作成し、各営業担当者の売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Amount Sold by Each Sales Representative';
   class SalesRep; 1
   var AmountSold; 2
   table SalesRep, 3
         AmountSold; 4 ;
run;
```

次のリストは、前述のプログラムで影付きの部分に対応しています。

1 変数 SalesRep は、分類変数として CLASS ステートメントで指定されています。TABLE ステートメントで SalesRep が使用されるたびに、SalesRep の各値に対してカテゴリが作成されます。

2 変数 AmountSold は、分析変数として VAR ステートメントで指定されています。AmountSold が TABLE ステートメントに使用されている場合、AmountSold の値が統計量の計算に使用されます。

3 TABLE ステートメントの行の次元には、変数 SalesRep が指定されています。SalesRep のそれぞれの値(カテゴリ)に対して 1 行が作成されます。

4 TABLE ステートメントの列の次元には、変数 AmountSold が指定されています。分析変数のデフォルトの統計量、SUM が AmountSold の値の要約に使用されます。

次の要約テーブルにこのプログラムの結果を示します。変数 AmountSold は変数 SalesRep と交差されて、要約テーブルの各セルを生成しています。列ヘッダー

AmountSold にはサブヘッダー SUM が含まれます。列の次元に表示される値は、各営業担当者の売上金額の合計です。

図 28.3 基本の 2 次元要約テーブル

TruBlend Coffee Makers, Inc.
Amount Sold by Each Sales Representative

	AmountSold
	Sum
SalesRep	
Garcia	512,071
Hollingsworth	347,246
Jensen	461,163

基本の 3 次元要約テーブルの作成

3 次元要約テーブルは、各ページに行と列を配置し、出力を個別のページに生成します。次の PROC TABULATE ステップは、3 次元要約テーブルを作成し、各営業担当者の年の四半期ごとの売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Quarterly Sales by Each Sales Representative';
   class SalesRep Quarter; 1
   var AmountSold; 2
   table SalesRep, 3
         Quarter, 4
         AmountSold; 5
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 変数 SalesRep と Quarter は、分類変数として CLASS ステートメントで指定されています。TABLE ステートメントで SalesRep が使用されるたびに、SalesRep の各値に対してカテゴリが作成されます。同様に、TABLE ステートメントで Quarter が使用されるたびに、Quarter の各値に対してカテゴリが作成されます。

2 変数 AmountSold は、分析変数として VAR ステートメントで指定されています。AmountSold が TABLE ステートメントに使用されている場合、AmountSold の値が統計量の計算に使用されます。

3 TABLE ステートメントのページの次元には、変数 SalesRep が使用されます。SalesRep のそれぞれの値(カテゴリ)のページが作成されます。

4 TABLE ステートメントの行の次元には、変数 Quarter が使用されます。Quarter のそれぞれの値またはカテゴリに対して 1 行が作成されます。

5 TABLE ステートメントの列の次元には、変数 AmountSold が使用されます。分析変数のデフォルトの統計量、SUM が AmountSold の値の要約に使用されます。

次の要約テーブルにこのプログラムの結果を示します。この要約テーブルでは営業担当者ごとにページが分かれています。各営業担当者について、四半期ごとに売上金額がレポートに記述されます。列ヘッダー AmountSold にはサブヘッダー SUM が含まれます。この列に表示される値は、各営業担当者による売上金額合計を米ドルで示します。

図 28.4 *基本の 3 次元要約テーブル*

TruBlend Coffee Makers, Inc.
Quarterly Sales by Each Sales Representative

SalesRep Garcia

	AmountSold
	Sum
Quarter	
1	118,020
2	108,860
3	225,326
4	59,865

TruBlend Coffee Makers, Inc.
Quarterly Sales by Each Sales Representative

SalesRep Hollingsworth

	AmountSold
	Sum
Quarter	
1	59,635
2	96,161
3	109,704
4	81,747

TruBlend Coffee Makers, Inc.
Quarterly Sales by Each Sales Representative

SalesRep Jensen

	AmountSold
	Sum
Quarter	
1	50,078
2	74,731
3	222,291
4	114,063

1 つの PROC TABULATE ステップによる複数のテーブルの生成

1 つの PROC TABULATE ステップで複数のテーブルを生成できます。ただし、変数の使用方法または定義をステップの途中で変更することはできません。つまり、CLASS ステートメントや VAR ステートメントの変数は、PROC TABULATE ステップのすべて

の TABLE ステートメントに対して 1 回のみ定義されます。1 つの変数を異なる TABLE ステートメントに使用または定義する方法を変更する必要がある場合は、複数の PROC TABULATE ステップ内で、それらの TABLE ステートメントを使用し、変数を定義する必要があります。次のプログラムは、TABULATE プロシジャの 1 回の実行時に 3 つの要約テーブルを生成します。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Sales of Deluxe Model Versus Standard Model';
   class SalesRep Type;
   var AmountSold Units;
   table Type; 1
   table Type, Units; 2
   table SalesRep, Type, AmountSold; 3
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 最初の TABLE ステートメントは、変数 Type の値を列の次元に使用して、1 次元要約テーブルを生成します。

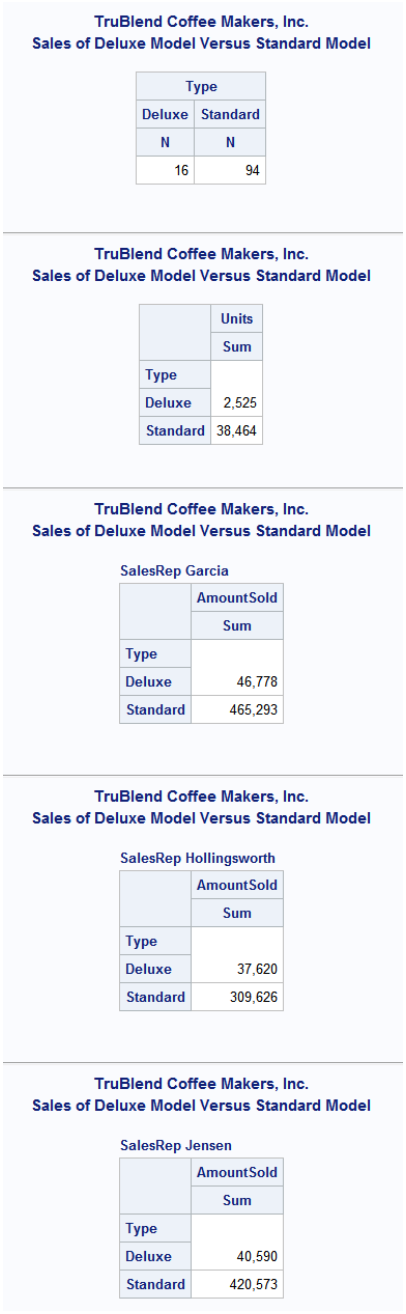
2 2 番目の TABLE ステートメントは、変数 Type の値を行の次元に使用し、変数 Units の値を列の次元に使用して、2 次元要約テーブルを生成します。

3 3 番目の TABLE ステートメントは、次の値を使用して 3 次元の要約テーブルを生成します。

- 変数 SalesRep の値をページの次元に
- 変数 Type の値を行の次元に
- 変数 AmountSold を列の次元に

次の要約テーブルにこのプログラムの結果を示します。

図 28.5 1 つの PROC TABULATE ステップによって生成された複数のテーブル



TruBlend Coffee Makers, Inc.
Sales of Deluxe Model Versus Standard Model

Type	
Deluxe	Standard
N	N
16	94

TruBlend Coffee Makers, Inc.
Sales of Deluxe Model Versus Standard Model

	Units
	Sum
Type	
Deluxe	2,525
Standard	38,464

TruBlend Coffee Makers, Inc.
Sales of Deluxe Model Versus Standard Model

SalesRep Garcia

	AmountSold
	Sum
Type	
Deluxe	46,778
Standard	465,293

TruBlend Coffee Makers, Inc.
Sales of Deluxe Model Versus Standard Model

SalesRep Hollingsworth

	AmountSold
	Sum
Type	
Deluxe	37,620
Standard	309,626

TruBlend Coffee Makers, Inc.
Sales of Deluxe Model Versus Standard Model

SalesRep Jensen

	AmountSold
	Sum
Type	
Deluxe	40,590
Standard	420,573

複雑な要約テーブルの作成

サブグループのレポート用の階層テーブルの作成

次元内で要素を交差させてデータのサブグループのレポート用の階層テーブルを作成できます。要素の交差とは、分類変数、分析変数、フォーマット修飾子、統計量、ス

タイルなど 2 つ以上の要素を組み合わせる操作です。次元と次元は自動的に交差されます。単一の次元式で変数を交差させると、1 つの変数の値が同一次元のその他の変数の値内に配置されます。これにより変数の階層が形成され、階層テーブルが形成されます。変数が交差するときに変数が記述されている順序により、テーブルのヘッダーの順序が決定されます。変数が、列の次元では上から下へ、行の次元では左から右へ、ページの次元では前から後ろへ積み重ねられていきます。次元式の要素は、要素間にアスタリスクを挿入することで交差します。2 つの分析変数は交差できないことに注意してください。また、次元と次元は自動的に交差されるため、すべての分析変数は 1 つの次元に出現する必要があります。

次の PROC TABULATE ステップは、2 つの変数を交差して 2 次元要約テーブルを作成し、各営業担当者のコーヒーメーカーの種類ごとの売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Amount Sold Per Item by Each Sales Representative';
   class SalesRep Type;
   var AmountSold;
   table SalesRep*Type,
         AmountSold;
run;
```

行の次元の式 **SalesRep*Type** は、アスタリスク演算子を使用して、変数 SalesRep の値と変数 Type の値を交差させています。交差時に SalesRep が Type より前に記述されているため、そして要素は行の次元で交差しているため、Type の値は SalesRep の値の右にリストされます。Type の各値が SalesRep の各値に対して繰り返されます。

次の要約テーブルに結果を示します。

図 28.6 *交差変数*

TruBlend Coffee Makers, Inc.
Amount Sold Per Item by Each Sales Representative

		AmountSold
		Sum
SalesRep	Type	
Garcia	Deluxe	46,778
	Standard	465,293
Hollingsworth	Deluxe	37,620
	Standard	309,626
Jensen	Deluxe	40,590
	Standard	420,573

Type の値が SalesRep の各値の右に繰り返されて値の階層が作成されることに注意してください。

出力のフォーマッティング

要約テーブルの結果の出力形式を、変数とフォーマット修飾子を交差させることで上書きできます。変数とフォーマット修飾子は、間にアスタリスクを挿入して交差させます。

次の PROC TABULATE ステップは、変数とフォーマット修飾子を交差して 2 次元要約テーブルを作成し、各営業担当者のコーヒーメーカーの種類ごとの売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Amount Sold Per Item by Each Sales Representative';
   class SalesRep Type;
   var AmountSold;
   table SalesRep*Type,
         AmountSold*f=dollar16.2;
run;
```

列の次元の式 **AmountSold*f=dollar16.2** は、アスタリスク演算子を使用して変数 **AmountSold** の値と SAS フォーマット修飾子 **f=dollar16.2** とを交差させます。AmountSold の値が DOLLAR16.2 出力形式を使用して表示されるようになります。DOLLAR16.2 出力形式の方が、PROC TABULATE ステートメントでデフォルトとして指定されている COMMA10.出力形式より、ドルの金額に適しています。

次の要約テーブルに結果を示します。

図 28.7 *変数とフォーマット修飾子の交差*

TruBlend Coffee Makers, Inc.
Amount Sold Per Item by Each Sales Representative

		AmountSold
		Sum
SalesRep	Type	
Garcia	Deluxe	$46,777.50
	Standard	$465,293.28
Hollingsworth	Deluxe	$37,620.00
	Standard	$309,626.10
Jensen	Deluxe	$40,590.00
	Standard	$420,572.60

記述統計量の計算

変数の記述統計量を、その変数と適切な統計量キーワードを交差させて要求できます。統計量と分類変数または分析変数のいずれかを交差させて、PROC TABULATE で実行する計算の種類を指定できます。2 つの統計量は交差できないことに注意して

ください。また、次元と次元は自動的に交差されるため、統計量はすべて 1 つの次元に出現する必要があります。

分類変数と交差されるデフォルトの統計量は、N 統計量(度数)です。分類変数は、度数統計量およびパーセント度数統計量とのみ交差できます。分析変数と交差されるデフォルトの統計量は、SUM 統計量です。分析変数は、よく使用される統計量 MIN、MAX、MEAN、STD および MEDIAN などを含め、PROC TABULATE で使用可能なさまざまな記述統計量のいずれとも交差できます。分析変数に使用可能な統計量の詳細なリストについては、*Base SAS Procedures Guide* の“Statistics Available in PROC TABULATE”を参照してください。

次の PROC TABULATE ステップは、要素と統計量を交差して 2 次元要約テーブルを作成し、各営業担当者のコーヒーメーカーの種類別売上ごとの平均金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Average Amount Sold Per Item by Each Sales Representative';
   class SalesRep Type;
   var AmountSold;
   table SalesRep*Type,
         AmountSold*mean*f=dollar16.2;
run;
```

このプログラムでは、列の次元で変数 **AmountSold** と統計量 **mean** およびフォーマット修飾子 **f=dollar16.2** とが交差されます。MEAN 統計量により AmountSold の算術平均が算出されます。

次の要約テーブルに結果を示します。

図 28.8 *変数と統計量の交差*

TruBlend Coffee Makers, Inc.
Average Amount Sold Per Item by Each Sales Representative

		AmountSold
		Mean
SalesRep	Type	
Garcia	Deluxe	$11,694.38
	Standard	$12,924.81
Hollingsworth	Deluxe	$4,702.50
	Standard	$12,901.09
Jensen	Deluxe	$10,147.50
	Standard	$12,369.78

複数統計量のレポートの作成

変数を連結することにより、2 つ以上の統計量についてレポートを記述する要約テーブルを作成できます。*連結*は、分類変数、分析変数や統計量など 2 つ以上の要素の情報を、2 番目とその後続の要素の出力を最初の要素の出力の直後に続けて配置することにより、連結する操作です。次元式の要素は、要素間にブランクを挿入して連結します。

次の PROC TABULATE ステップは、連結を使用して 2 次元要約テーブルを作成し、各営業担当者のコーヒーメーカーの種類ごとの販売数と合計売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Sales Summary by Representative and Product';
   class SalesRep Type;
   var AmountSold;
   table SalesRep*Type,
         AmountSold*n AmountSold*f=dollar16.2;
run;
```

このプログラムでは、列の次元の式 **AmountSold*n** と **AmountSold*f=dollar16.2** がブランクによって区切られているため、これらの出力が連結されます。

次の要約テーブルに結果を示します。この要約テーブルでは、AmountSold の度数(N)が AmountSold の SUM と同じテーブルに示されています。

図 28.9 *変数の連結*

TruBlend Coffee Makers, Inc.
Sales Summary by Representative and Product

		AmountSold	AmountSold
		N	Sum
SalesRep	Type		
Garcia	Deluxe	4	$46,777.50
	Standard	36	$465,293.28
Hollingsworth	Deluxe	8	$37,620.00
	Standard	24	$309,626.10
Jensen	Deluxe	4	$40,590.00
	Standard	34	$420,572.60

コードの削減と複数の要素への単一ラベルの適用

共通の要素と連結または交差されている連結要素(変数、出力形式、統計量など)を、かっこを使用してグループ化できます。これにより使用するコードを減らし、ラベルの表示方法を変更できます。次の PROC TABULATE ステップは、AmountSold と交差する要素をかっこを使用してグループ化し、各営業担当者のコーヒーメーカーの種類ごとの販売数と合計売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Sales Summary by Representative and Product';
   class SalesRep Type;
```

```
   var AmountSold;
   table SalesRep*Type,
         AmountSold*(n sum*f=dollar16.2);
run;
```

このプログラムでは、**AmountSold*(n sum*f=dollar16.2)** が、**AmountSold*n AmountSold*f=dollar16.2** のかわりに使用されます。**AmountSold*f=dollar16.2** のデフォルトの統計量 SUM を、ここでは式に含める必要があることに注意してください。これは、フォーマット修飾子は変数または統計量と交差している必要があるためです。フォーマット修飾子だけで式に含めることはできません。

次の要約テーブルに結果を示します。

図 28.10 *かっこを使用した要素のグループ化*

TruBlend Coffee Makers, Inc.
Sales Summary by Representative and Product

		AmountSold	
		N	Sum
SalesRep	Type		
Garcia	Deluxe	4	$46,777.50
	Standard	36	$465,293.28
Hollingsworth	Deluxe	8	$37,620.00
	Standard	24	$309,626.10
Jensen	Deluxe	4	$40,590.00
	Standard	34	$420,572.60

ラベル AmountSold は、図 28.9 (477 ページ)のように、要約テーブルに 2 つ表示されるのではなく、複数の列にまたがって表示されることに注意してください。

すべての変数の要約の取得

ある次元内の分類変数すべてを、ユニバーサル分類変数 ALL を使用して要約できます。ALL は、TABLE ステートメントの 3 つの次元それぞれと、かっこによって区切られるグループ内の要素と連結できます。次の PROC TABULATE ステップは、ユニバーサル分類変数 ALL を使用して 2 次元要約テーブルを作成し、各営業担当者、およびすべての営業担当者を 1 つのグループとした、販売数、売上ごとの平均金額、売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Sales Report';
   class SalesRep Type;
   var AmountSold;
   table SalesRep*Type all,
         AmountSold*(n (mean sum)*f=dollar16.2);
```

```
run;
```

このプログラムでは、TABLE ステートメントの行の次元にユニバーサル分類変数 ALL が指定されています。SalesRep と Type が要約されます。

次の要約テーブルに結果を示します。この要約テーブルは、SalesRep と Type のそれぞれのカテゴリの、AmountSold の度数(N)、MEAN および SUM をレポートします。このデータは、SalesRep と Type のすべてのカテゴリについて、All とラベル付けされた行に要約されています。

図 28.11 ユニバーサル分類変数 ALL との交差

TruBlend Coffee Makers, Inc.
Sales Report

		AmountSold		
		N	Mean	Sum
SalesRep	Type			
Garcia	Deluxe	4	$11,694.38	$46,777.50
	Standard	36	$12,924.81	$465,293.28
Hollingsworth	Deluxe	8	$4,702.50	$37,620.00
	Standard	24	$12,901.09	$309,626.10
Jensen	Deluxe	4	$10,147.50	$40,590.00
	Standard	34	$12,369.78	$420,572.60
All		110	$12,004.36	$1,320,479.48

ラベルの定義

TABLE ステートメントで変数にラベルを割り当てることで、要約テーブルに独自のラベルを追加したり、要約テーブルからヘッダーを削除したりできます。単純に、変数の後に等号(=)を付けて、目的のラベルまたはブランクを引用符で囲んで続けます。引用符で囲んだブランクを指定すると、要約テーブルからヘッダーが削除されます。次の PROC TABULATE ステップは、TABLE ステートメントでラベルを使用して 2 次元要約テーブルを作成し、各営業担当者による、コーヒーメーカーの種類ごととコーヒーメーカー全種類の合計売上金額のパーセント、および平均売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Sales Performance';
   class SalesRep Type;
   var AmountSold;
   table SalesRep='Sales Representative'1 *
         (Type='Type of Coffee Maker'1 all) all,
         AmountSold=' '4 *
         (N='Sales'2
         SUM='Amount'2 *f=dollar16.2
         colpctsum='% Sales'3
         mean='Average Sale'2 *f=dollar16.2);
```

```
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 変数 SalesRep と変数 Type にラベルが割り当てられます。

2 度数統計量 N、統計量 SUM、統計量 MEAN にラベルが割り当てられます。

3 統計量 COLPCTSUM が、列の値の合計に対する単一のテーブルセルの値のパーセンテージの計算に使用され、ラベル'% Sales'が割り当てられます。

4 変数 AmountSold にブランクラベルが割り当てられます。その結果、AmountSold のヘッダーは要約テーブルに表示されません。

次の要約テーブルに結果を示します。このテーブルでは、変数 AmountSold のヘッダーは表示されません。ラベル'Sales'、'Amount'、'% Sales'と'Average Sale'により、度数(N)、SUM、COLPCTSUM と MEAN がそれぞれ置き換えられます。ラベルにより変数 SalesRep と変数 Type が置き換えられます。

図 28.12 *ラベルを使用した要約テーブルのカスタマイズ*

TruBlend Coffee Makers, Inc.
Sales Performance

		Sales	Amount	% Sales	Average Sale
Sales Representative	Type of Coffee Maker				
Garcia	Deluxe	4	$46,777.50	4	$11,694.38
	Standard	36	$465,293.28	35	$12,924.81
	All	40	$512,070.78	39	$12,801.77
Hollingsworth	Type of Coffee Maker				
	Deluxe	8	$37,620.00	3	$4,702.50
	Standard	24	$309,626.10	23	$12,901.09
	All	32	$347,246.10	26	$10,851.44
Jensen	Type of Coffee Maker				
	Deluxe	4	$40,590.00	3	$10,147.50
	Standard	34	$420,572.60	32	$12,369.78
	All	38	$461,162.60	35	$12,135.86
All		110	$1,320,479.48	100	$12,004.36

スタイルと Output Delivery System の使用

Output Delivery System を使用して、LISTING や OUTPUT 以外の任意の出力先へ PROC TABULATE からの出力を作成する場合、次の操作を行えます。

- プロシジャによってテーブルのさまざまな要素に対して使用される、特定のスタイル要素(フォントスタイル、フォントの太さ、色など)を設定します。
- CLASS ステートメントにオプションを追加して、変数のラベルのスタイル要素を指定します。
- STYLE=オプションと次元式の要素とを交差させることで、要約テーブルのセルのスタイル要素を指定します。

次元式で使用する場合、STYLE=オプションは、大かっこ([および])または中かっこ({および})で囲む必要があります。次の PROC TABULATE ステップは、CLASS ステートメントと TABLE ステートメントで STYLE=オプションを使用して 2 次元要約テーブルを作成し、各営業担当者による、コーヒーメーカーの種類ごとと コーヒーメーカー全種類の合計売上金額のパーセント、および平均売上金額はいくらか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

ods html file='summary-table.htm';1
ods pdf file='summary-table.pdf';2

proc tabulate data=year_sales format=comma10.;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Sales Performance';
   class SalesRep;
   class Type / style=[font_style=italic]3;
   var AmountSold;
   table SalesRep='Sales Representative'*(Type='Type of Coffee Maker'
         all*[style=[background=yellow font_weight=bold]]4)
         all*[style=[font_weight=bold]]5,
         AmountSold=' '*(colpctsum='% Sales' mean='Average Sale'*
         f=dollar16.2);
run;

ods html close;6
ods pdf close;7
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 HTML 出力先はデフォルトで開いています。ODS HTML を使用して html ファイルの名前を指定できます。FILE=オプションで、HTML 出力を含むファイルを識別します。ブラウザによっては、ファイル名に HTM または HTML の拡張子が必要な場合があります。

2 ODS PDF ステートメントは、PDF 出力先を開き、PDF 出力を作成します。FILE=オプションで、PDF 出力を含むファイルを識別します。

3 2 番目の CLASS ステートメントで STYLE=オプションが指定され、Type のラベルのフォントスタイルが斜体に設定されます。SalesRep のラベルは別の CLASS ステートメントにあるため、この STYLE=オプションによる影響を受けません。

4 ユニバーサル分類変数 ALL と STYLE=オプションとが交差され、テーブルセルの背景は黄色に、およびこれらのセルのフォントの太さは太字に設定されます。

5 ユニバーサル分類変数 ALL と STYLE=オプションとが交差され、テーブルセルのフォントの太さは太字に設定されます。

6 ODS HTML CLOSE ステートメントによって、HTML 出力先とそれに関連付けられているファイルがすべて閉じられます。

7 ODS PDF CLOSE ステートメントによって、PDF 出力先が閉じられます。PDF 出力を表示する前に、PDF 出力先を閉じる必要があります。

次の要約テーブルに HTML 結果を示します。

図 28.13 スタイル属性と ODS HTML ステートメントの使用

TruBlend Coffee Makers, Inc.
Sales Performance

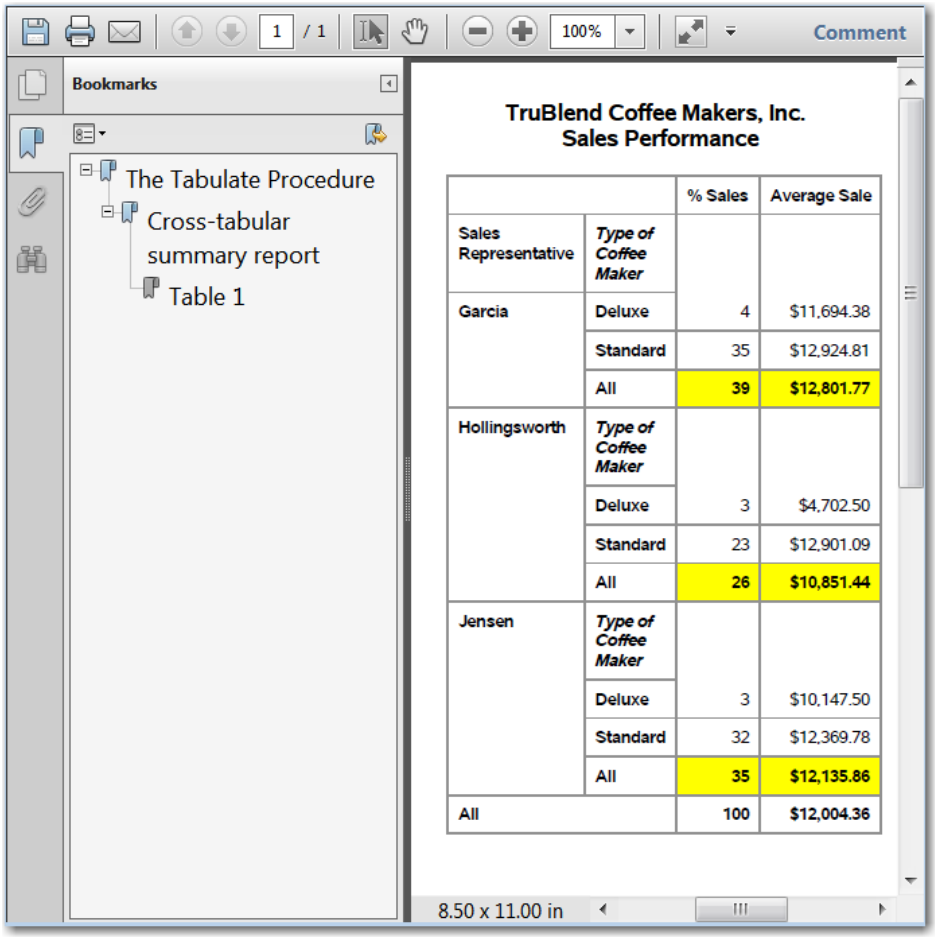
		% Sales	Average Sale
Sales Representative	*Type of Coffee Maker*		
Garcia	Deluxe	4	$11,694.38
	Standard	35	$12,924.81
	All	**39**	**$12,801.77**
Hollingsworth	*Type of Coffee Maker*		
	Deluxe	3	$4,702.50
	Standard	23	$12,901.09
	All	**26**	**$10,851.44**
Jensen	*Type of Coffee Maker*		
	Deluxe	3	$10,147.50
	Standard	32	$12,369.78
	All	**35**	**$12,135.86**
All		**100**	**$12,004.36**

この要約テーブルは、前述の SAS プログラムの 3 つの STYLE=オプションと ODS HTML ステートメントの使用による、次の効果を表しています。

- 繰り返しラベル、Type of Coffee Maker は斜体です。
- 営業担当者のすべての値の小計はより淡い色(黄色)で強調表示され、太字になります。
- すべての営業担当者の合計は太字です。

次の要約テーブルに PDF 結果を示します。

図 28.14 スタイル属性とODS PDF ステートメントの使用



TruBlend Coffee Makers, Inc.
Sales Performance

Sales Representative	Type of Coffee Maker	% Sales	Average Sale
Garcia	Deluxe	4	$11,694.38
	Standard	35	$12,924.81
	All	**39**	**$12,801.77**
Hollingsworth	Type of Coffee Maker		
	Deluxe	3	$4,702.50
	Standard	23	$12,901.09
	All	**26**	**$10,851.44**
Jensen	Type of Coffee Maker		
	Deluxe	3	$10,147.50
	Standard	32	$12,369.78
	All	**35**	**$12,135.86**
All		**100**	**$12,004.36**

この要約テーブルは、前述の SAS プログラムの 3 つの STYLE=オプションと ODS PDF ステートメントの使用による、次の効果を表しています。

- 繰り返しラベル、Type of Coffee Maker は斜体です。
- 営業担当者のすべての値の小計は強調表示され、太字になります。
- すべての営業担当者の合計は太字です。

分類変数の並べ替え

要約テーブルに分類変数の値とヘッダーが表示される順序を、ORDER=オプションを使用して制御できます。ORDER=オプションは、PROC TABULATE ステートメントおよび個別の CLASS ステートメントで使用できます。構文は、**ORDER=sort-order** です。4 つの使用可能な並べ替え順序(DATA、FORMATTED、FREQ、UNFORMATTED)は、“要約” (484 ページ)に定義されています。次の PROC TABULATE ステップは、PROC TABULATE ステートメントで ORDER=オプションを使用してすべての分類変数を度数によって並べ替えて 2 次元要約テーブルを作成し、どの四半期が最大販売数を上げたか、どの営業担当者が全体での最大販売数を達成したか、を調べることができます。

```
options linesize=84 pageno=1 nodate;

proc tabulate data=year_sales format=comma10. order=freq;
   title1 'TruBlend Coffee Makers, Inc.';
```

```
   title2 'Quarterly Sales and Representative Sales by Frequency';
   class SalesRep Quarter;
   table SalesRep all,
         Quarter all;
run;
```

次の要約テーブルにこのプログラムの結果を示します。分類変数 Quarter の値の順序により、最大多数の売上は第 3 四半期に、続いて第 1 四半期、第 2 四半期そして第 4 四半期に発生したことが示されます。分類変数 SalesRep の値の順序により、Garcia が全体を通じて最大数の売り上げを達成し、そして Jensen、Hollingsworth の順に続くことが示されます。ユニバーサル分類変数 ALL は、この例では両方の次元に含まれていて、要約テーブルの作成時にデータの並べ替えに使用された度数データを示しています。

図 28.15 *分類変数の並べ替え*

TruBlend Coffee Makers, Inc.
Quarterly Sales and Representative Sales by Frequency

	Quarter				All
	3	1	2	4	
	N	N	N	N	N
SalesRep					
Garcia	21	8	6	5	40
Jensen	21	5	6	6	38
Hollingsworth	15	5	6	6	32
All	57	18	18	17	110

要約

グローバルステートメント

TITLE<n> <'title'>;

タイトルを指定します。引数 *n* は、ワード TITLE の直後にブランクを空けずに続ける 1-10 の数値で、TITLE のレベルを指定します。各 *title* のテキストは最大 132 文字長(一部の動作環境では、256 文字長)で、一重引用符または二重引用符で囲む必要があります。

TABULATE プロシジャステートメント

PROC TABULATE <*option(s)*>;

CLASS *variable(s)*</*option(s)*>;

VAR *analysis-variable(s)*;

TABLE <<*page-expression*,> *row-expression*,> *column-expression*;

PROC TABULATE <option(s)>;
プロシジャを開始します。

PROC TABULATE ステートメントでは、次の *options* を指定できます。

DATA=SAS-data-set
PROC TABULATE で使用される *SAS-data-set* を指定します。DATA=オプションを省略すると、TABULATE プロシジャは現在のジョブまたはセッションで最後に作成された SAS データセットを使用します。

FORMAT=format-name
テーブルの各セルの値のフォーマットに使用されるデフォルトの出力形式を指定します。有効な SAS 数値の出力形式またはユーザー定義の出力形式を指定できます。

MISSING
欠損値を有効な値とみなして、分類変数の組み合わせを作成します。各欠損値のヘッダーがテーブルに表示されます。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
テーブルのヘッダーを形成する、分類変数の値の一意の組み合わせの作成に使用される並べ替え順序を指定します。次に、それぞれの並べ替え順序の簡単な説明を示します。

DATA
入力データセットでの順序に従って値を並べ替えます。

FORMATTED
フォーマットされた値の昇順で値を並べ替えます。この順序は、使用している動作環境によって異なります。

FREQ
度数カウントの降順で値を並べ替えます。

ORDER=
CLASS ステートメントで使用される ORDER=は、PROC TABULATE ステートメントで使用される ORDER=に優先します。

UNFORMATTED
フォーマットされていない値で値を並べ替えます。PROC SORT を使用するのと同じ順序になります。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を経時的に表示するのに特に便利です。

CLASS variable(s)/option(s);
テーブルの分類変数を識別します。分類変数により、PROC TABULATE で統計量の計算に使用されるカテゴリが決定されます。

MISSING
欠損値を有効な値と見なして、分類変数の組み合わせを作成します。各欠損値のヘッダーがテーブルに表示されます。MISSING を分類変数のサブセットのみに適用する場合、MISSING を個別の CLASS ステートメントで分類変数のサブセットと一緒に指定します。

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
テーブルのヘッダーを形成する、分類変数の値の一意の組み合わせの作成に使用される並べ替え順序を指定します。ORDER=を分類変数のサブセットのみに適用する場合、ORDER=を個別の CLASS ステートメントで分類変数のサブセットと一緒に指定します。このようにすると、それぞれの分類変数に対して異なる並べ替え順序を指定できます。次に、それぞれの並べ替え順序の簡単な説明を示します。

DATA
入力データセットでの順序に従って値を並べ替えます。

FORMATTED
フォーマットされた値の昇順で値を並べ替えます。この順序は、使用している動作環境によって異なります。

FREQ
度数カウントの降順で値を並べ替えます。

ORDER=
CLASS ステートメントで使用される ORDER=は、PROC TABULATE ステートメントで使用される ORDER=に優先します。

UNFORMATTED
フォーマットされていない値で値を並べ替えます。PROC SORT を使用するのと同じ順序になります。この順序は、使用している動作環境によって異なります。この並べ替え順序は、日付を経時的に表示するのに特に便利です。

VAR analysis-variable(s);
テーブルの分析変数を指定します。分析変数には、統計量を計算する値が含まれます。

TABLE <<page-expression, >row-expression,> column-expression;
PROC TABULATE で生成するテーブルを定義します。少なくとも 1 つの TABLE ステートメントを指定する必要があります。TABLE ステートメントで page-expressions、row-expressions、column-expressions を指定します。これらはすべて同様の方法で構成され、集合的に次元式と呼ばれます。次元式は、カンマを使用して区切ります。次元内の変数、統計量およびその他の要素間の関係を、1 つ以上の演算子を使用して組み合わせて定義します。演算子は、PROC TABULATE で変数、統計量およびその他の要素に対して実行する操作を指示する記号です。次の表に、一般的な演算子とそれらが表す操作をリストで表示します。

表 28.1 *TABLE ステートメント演算子*

演算子	アクション
, カンマ	テーブルの次元を区切ります。
* アスタリスク	次元内の要素を交差します。
ブランク	次元内の要素を連結します。
= 等号	デフォルトのセルの出力形式を上書きするか、または要素にラベルを割り当てます。
( ) 丸かっこ	要素をグループ化し、グループのそれぞれの連結要素と演算子とを関連付けます。
[ ] 大かっこ	交差用に STYLE=オプションをグループ化し、STYLE=オプション内のスタイル属性の指定をグループ化します。
{ } 中かっこ	交差用に STYLE=オプションをグループ化し、STYLE=オプション内のスタイル属性の指定をグループ化します。

詳細情報

プロシジャ出力の出力先
: 33 章, “SAS 出力とそのカスタマイズについて: 基本” (599 ページ)を参照してください。

欠損値
: 欠損値の説明については、*SAS Language Reference: Concepts* を参照してください。欠損値の処理の詳細については、*PROC TABULATE by Example* にも説明が記載されています。

ODS
: ODS (Output Delivery System)の使用方法の詳細なドキュメントについては、*SAS Output Delivery System: User's Guide* を参照してください。

PROC TABULATE
: *Base SAS Procedures Guide* の TABULATE プロシジャを参照してください。

  TABULATE プロシジャの詳細およびさまざまな例については、*PROC TABULATE by Example* を参照してください。

SAS 出力形式
: *SAS Formats and Informats: Reference* を参照してください。SAS では、小数値、16 進値、ローマ数字、社会保障番号、日付と時刻、および文字として記述される数値など、多くの出力形式が使用可能です。

統計量
: TABULATE プロシジャで使用可能な統計量のリストについては、*Base SAS Procedures Guide* の TABULATE プロシジャの概念の説明を参照してください。リストの統計量の詳細については、*Base SAS Procedures Guide* の Appendix の基本的な統計量の説明を参照してください。

スタイル属性
: Output Delivery System を使用してスタイル要素に設定できるスタイル属性の詳細については、*Base SAS Procedures Guide* または *SAS Output Delivery System: User's Guide* を参照してください。

要約テーブル
: さまざまな要約テーブルの生成方法のその他の例については、*SAS Guide to Report Writing Examples* を参照してください。

  要約テーブルを作成するための REPORT プロシジャの使用方法の詳細については、*Base SAS Procedures Guide* の REPORT プロシジャを参照してください。

表形式レポート
: 対話型オンラインの例と詳細については、SAS Online Tutor for Version 8: SAS Programming の表形式レポートの作成に関するレッスンを参照してください。

TITLE ステートメント
: 27 章, “PRINT プロシジャを使用した詳細レポートの作成” (421 ページ)を参照してください。

29 章

REPORT プロシジャを使用した詳細レポートと要約レポートの作成

REPORT プロシジャを使用した詳細レポートと要約レポートの作成について **489**
目的 489
前提条件 490
レポートの作成方法について **490**
レポート作成ツールの使用 490
レポートの種類 490
レポートのレイアウト 491
例で使用される入力ファイルおよび入力 SAS データセット **492**
単純なレポートの作成 **493**
すべての変数の表示 493
列の指定と並べ替え 495
行の並べ替え 496
単一行に複数のオブザベーションをまとめる 498
行のデフォルトの順序の変更 499
高度なレポートの作成 **502**
列のレイアウトの調整 502
列ヘッダーのカスタマイズ 503
出力形式の指定 504
列ヘッダーとして変数の値を使用 505
オブザベーションのグループの要約 508
要約 **510**
PROC REPORT ステートメント 510
詳細情報 **514**

REPORT プロシジャを使用した詳細レポートと要約レポートの作成について

目的

SAS には、詳細レポートや要約レポートの作成に使用できるさまざまなレポート作成ツールが用意されています。レポートを使用して、データに関する情報を、系統立てて簡潔に伝達できます。REPORT プロシジャを使用すると、単一のレポート作成ツールで詳細レポートと要約レポートを作成できます。

このセクションでは、PROC REPORT を使用して次の操作を行う方法を学習します。

- 単純な詳細レポートの生成
- 単純な要約レポートの生成
- オブザベーションの並べ替えとグループ化、列の合計、総合計の計算を行うオプションのステートメントを追加して拡張レポートを生成
- 列の間隔、列のラベル、行の区切りおよび出力形式を追加してレポートの外観をカスタマイズ

前提条件

このセクションの例を理解するために、次の機能と概念を理解していることを確認してください。

- データセットオプション
- TITLE ステートメント
- LABEL ステートメント
- WHERE 処理
- SAS 出力形式の作成と割り当て

レポートの作成方法について

レポート作成ツールの使用

REPORT プロシジャは、PROC MEANS、PROC PRINT、および PROC TABULATE の機能と DATA ステップのレポート作成機能とをあわせ持つ、強力なレポート作成ツールです。PROC REPORT を使用して、次の操作を行えます。

- カスタマイズされた、プレゼンテーション品質のレポートの作成
- 構造とレイアウトを制御するレポート定義の開発と保存
- 定義済みのレポートの表示
- 1 つのレポート定義からの複数レポートの生成

PROC REPORT を使用してレポートを作成するには、次の 3 つの異なる方法があります。

- 非ウィンドウ環境で PROC REPORT を使用して一連のステートメントをサブミットこれがデフォルト環境です。
- ウィンドウ環境でプロンプト機能を使用
- ウィンドウ環境でプロンプト機能を使用しない

ウィンドウ環境では、レポート開発の際、必要な SAS プログラミング技術は最小限で、即時の視覚的なフィードバックが可能です。このセクションでは、非ウィンドウ環境を使用して要約レポートと詳細レポートを作成する方法について説明します。

レポートの種類

REPORT プロシジャを使用して、次の 2 種類のレポートを作成できます。

詳細レポート
　各行が、レポートに選択した各オブザベーションを示します。詳細については、アウトプット 29.1 (494 ページ)を参照してください。これらの行はすべて詳細行です。

要約レポート
　データがまとめられ、各行は複数のオブザベーションを示します。詳細については、アウトプット 29.5 (499 ページ)を参照してください。これらの行もすべて詳細行と呼ばれます。

詳細レポートと要約レポートの両方ともに要約行、詳細行を含めることができます。要約行は、詳細行のセットまたはすべての詳細行の数値データを要約します。PROC REPORT を使用して、デフォルトの要約とカスタマイズされた要約の両方を生成できます。

レポートのレイアウト

レイアウトの決定

最初にレポートのレイアウトを決定しておくと、レポートの作成が容易になります。次の項目を決定する必要があります。

- レポートに表示する列
- 列と行の順序
- 列と行のラベル付け方法
- 表示する統計量
- 特定の変数の各値の列を表示するかどうか
- すべてのオブザベーションについて行を表示するか、または複数のオブザベーションを単一の行にまとめるかどうか

レポートのレイアウトが決定したら、PROC REPORT ステップに COLUMN ステートメントと DEFINE ステートメントを使用してレイアウトを作成します。

レイアウトの作成

COLUMN ステートメントでは、レポートの列として含めるレポート項目、列の配置を記述し、複数の列にまたがるヘッダーを定義します。レポート項目は、データセットの変数、計算された統計量、またはレポートの他の項目に基づいてユーザーが計算する変数です。

DEFINE ステートメントは、レポートの項目の特性を定義します。これらの特性には、PROC REPORT におけるレポートの項目の使用方法、列ヘッダーのテキスト、値を表示する出力形式が含まれます。

DEFINE ステートメントで指定する変数の使用方法により、レポートのレイアウトの多くの部分を制御することができます。変数の使用方法の種類を次に示します。

ACROSS
　ACROSS 変数の各値の列を作成します。

ANALYSIS
　レポートのセルで表されるすべてのオブザベーションについて数値変数から統計量を計算します。変数の値は、レポート内の位置によって変わります。デフォルトでは、PROC REPORT はすべての数値変数を ANALYSIS 変数として処理し、合計を計算します。

COMPUTED
レポート用に定義する変数からレポート項目を計算します。これらは入力データセットに含まれておらず、また、PROC REPORT によって入力データセットに追加されません。

%DISPLAY
入力データセットのすべてのオブザベーションについて、各オブザベーションを 1 行として表示します。デフォルトでは、PROC REPORT はすべての文字変数を DISPLAY 変数として処理します。

GROUP
データセットから、各 GROUP 変数のフォーマットされた値の組み合わせが同じすべてのオブザベーションを 1 行にまとめます。

ORDER
入力データセットの各オブザベーションを示す行を、ORDER 変数のフォーマットされた値の昇順に並べ替えます。

各変数のレポート内での位置と使用法によりレポートの構造とコンテンツが決定されます。たとえば、PROC REPORT はレポートの詳細行を、ORDER 変数と GROUP 変数の値に従って左から右へ並べ替えます。同様に、PROC REPORT は ACROSS 変数の列を変数の値に従って上から下へ並べ替えます。PROC REPORT によるレポートレイアウトの決定方法の詳細については、*Base SAS Procedures Guide* を参照してください。

例で使用される入力ファイルおよび入力 SAS データセット

このセクションの例では、1 つの入力ファイルと 1 つの SAS データセットを使用します。詳細については、“YEAR_SALES データセット” (794 ページ)を参照してください。

入力ファイルには、コーヒーマシンを供給する会社 TruBlend Coffee Makers の売上記録が含まれています。ファイルの構造は次のとおりです。

```
01      1      Hollingsworth  Deluxe      260      49.50
01      1      Garcia         Standard     41      30.97
01      1      Hollingsworth  Deluxe      330      49.50
01      1      Jensen         Standard   1110      30.97
01      1      Garcia         Standard    715      30.97
01      1      Jensen         Deluxe      675      49.50
02      1      Jensen         Standard     45      30.97
02      1      Garcia         Deluxe       10      49.50

…more data lines…

12      4      Hollingsworth  Deluxe      125      49.50
12      4      Jensen         Standard   1254      30.97
12      4      Hollingsworth  Deluxe      175      49.50
```

入力ファイルには、次に示す値が左から右へと含まれています。

- 売上が発生した月
- 売上が発生した年の四半期
- 営業担当者の名前
- 売り上げたコーヒーメーカーの種類(Standard または Deluxe)

- 販売ユニット数
- 各ユニットの価格(米ドル)

SAS データセットの名前は YEAR_SALES です。このデータセットには、入力ファイルからのすべての売上データ、および Units を Price で乗算することにより作成される AmountSold という名前の新しい変数が含まれます。

次のプログラムは、このセクションで使用する SAS データセットを作成します。

```
data year_sales;
infile 'your-input-file';
input Month $ Quarter $ SalesRep $14. Type $ Units Price;
AmountSold = Units * Price;
run;
```

単純なレポートの作成

すべての変数の表示

デフォルトでは、PROC REPORT はデータセットのすべての変数を使用します。レポートのレイアウトは、データセットの変数の種類によって異なります。データセットに文字変数が含まれている場合、PROC REPORT は、データセットのすべての変数とオブザベーションの値をリスト表示する単純な詳細レポートを生成します。データセットに数値変数のみが含まれている場合、PROC REPORT はデータセットのすべてのオブザベーションのそれぞれの変数の値を合計し、1 行の合計の要約を生成します。数値のみを含むデータセットの詳細レポートを生成するには、レポートの列を定義する必要があります。

デフォルトでは、PROC REPORT が結果を SAS プロシジャ出力に送ります。NOWINDOWS (NOWD)オプションを指定する必要はありません。PROC REPORT で **REPORT** ウィンドウを開くようにするには、WINDOWS オプションを指定します。**REPORT** ウィンドウを使用すると、レポートを繰り返し変更して、その変更をすぐに確認できます。

次の PROC REPORT ステップは、第 1 四半期売上の、デフォルトの詳細レポートを作成します。

```
proc report data=year_sales;
   where quarter='1';
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'First Quarter Sales Report';
run;
```

WHERE ステートメントには、YEAR_SALES データセットからのオブザベーションの選択に使用される条件が指定されます。PROC REPORT でレポートが作成される前に、SAS でオブザベーションが選択的に処理され、レポートに第 1 四半期のオブザベーションのデータのみが含まれるようになります。WHERE 処理の詳細については、“オブザベーションの選択” (432 ページ)を参照してください。

次の詳細レポートは、第 1 四半期売上データを含む、YEAR_SALES にあるオブザベーションのすべての変数値を示しています。

アウトプット 29.1 *データセットに文字値が含まれる場合のデフォルトのレポート*

TruBlend Coffee Makers, Inc. 2
First Quarter Sales Report

Month 1	Quarter	SalesRep	Type	Units	Price	AmountSold
01	1	Hollingsworth	Deluxe	260	49.5	12870
01	1	Garcia	Standard	41	30.97	1269.77
01	1	Hollingsworth	Standard	330	30.97	10220.1
01	1	Jensen	Standard	110	30.97	3406.7
01	1	Garcia	Deluxe	715	49.5	35392.5
01	1	Jensen	Standard	675	30.97	20904.75
02	1	Garcia	Standard	2045	30.97	63333.65
02	1	Garcia	Deluxe	10	49.5	495
02	1	Garcia	Standard	40	30.97	1238.8
02	1	Hollingsworth	Standard	1030	30.97	31899.1
02	1	Jensen	Standard	153	30.97	4738.41
02	1	Garcia	Standard	98	30.97	3035.06
03	1	Hollingsworth	Standard	125	30.97	3871.25
03	1	Jensen	Standard	154	30.97	4769.38
03	1	Garcia	Standard	118	30.97	3654.46
03	1	Hollingsworth	Standard	25	30.97	774.25
03	1	Jensen	Standard	525	30.97	16259.25
03	1	Garcia	Standard	310	30.97	9600.7

次のリストは、前述のレポートの番号付き項目に対応しています。

1 列の順序はデータセット内での変数の位置に対応しています。

2 レポートの最上部には、TITLE ステートメントによって生成されるタイトルが配置されます。

次の PROC REPORT ステップは、YEAR_SALES データセットに数値のみが含まれている場合に、デフォルトの要約レポートを生成します。

```
proc report data=year_sales (keep=Units AmountSold);
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'Total Yearly Sales';
run;
```

KEEP=データセットオプションにより、数値変数 Units と数値変数 Amountsold のみが処理されるように指定されています。PROC REPORT によりこれらの変数を使用してレポートが作成されます。

次のレポートは、2 つの数値変数の 1 行の要約を示します。

アウトプット 29.2 *データセットに数値のみが含まれる場合のデフォルトのレポート*

TruBlend Coffee Makers, Inc.
Total Yearly Sales

Units	AmountSold
40989	1320479.5

PROC REPORT により、データセットのすべてのオブザベーションの各変数の値が合計されて、Units と AmountSold の 1 行の要約が計算されます。

列の指定と並べ替え

レポートの作成における最初のステップは、レポートに表示する列を選択することです。デフォルトでは、レポートには各変数の列が含まれており、列の順序はデータセットの変数の順序に対応しています。

COLUMN ステートメントを使用して、レポートで使用する変数と列の配置を指定します。COLUMN ステートメントでは、データセット変数、PROC REPORT によって計算される統計量、またはレポートの他の項目から計算される変数をリストできます。

次のプログラムは、4 列の第 1 四半期売上レポートを作成します。

```
proc report data=year_sales;
   where Quarter='1';
   column SalesRep Month Type Units;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'First Quarter Sales Report';
run;
```

COLUMN ステートメントにより、レポートの項目の順序が決定されます。最初の列には SalesRep の値が、2 番目の列は Month の値がリストされます(以下同様に続く)。

次の出力はレポートを示しています。

アウトプット 29.3 *選択した列の表示*

TruBlend Coffee Makers, Inc.
First Quarter Sales Report

SalesRep	Month	Type	Units
Hollingsworth	01	Deluxe	260
Garcia	01	Standard	41
Hollingsworth	01	Standard	330
Jensen	01	Standard	110
Garcia	01	Deluxe	715
Jensen	01	Standard	675
Garcia	02	Standard	2045
Garcia	02	Deluxe	10
Garcia	02	Standard	40
Hollingsworth	02	Standard	1030
Jensen	02	Standard	153
Garcia	02	Standard	98
Hollingsworth	03	Standard	125
Jensen	03	Standard	154
Garcia	03	Standard	118
Hollingsworth	03	Standard	25
Jensen	03	Standard	525
Garcia	03	Standard	310

行の並べ替え

変数の使用方法を決定することで、レポートのレイアウトの大部分を制御できます。DEFINE ステートメントで変数の使用オプションを指定して、PROC REPORT での変数の使用方法を指示します。

レポートの行の順序を指定するために、1 つ以上の DEFINE ステートメントで ORDER オプションを使用できます。PROC REPORT では ORDER 変数の値に従って、レポートの行が並べ替えられます。レポートに複数の ORDER 変数が含まれる場合、PROC REPORT ではまず、COLUMN ステートメントの最初の ORDER 変数の値に従って行が並べ替えられます。[1] 最初の ORDER 変数の各値の範囲内では、COLUMN ステートメントの 2 番目の ORDER 変数の値に従ってプロシジャにより行が並べ替えられます(以下同様に続く)。

1 COLUMN ステートメントを省略した場合は、PROC REPORT は入力データセットでの位置に従って ORDER 変数を処理します。

次のプログラムは、営業担当者と月で並べ替えられた、第 1 四半期売上の詳細レポートを作成します。

```
proc report data=year_sales nowindows;
   where Quarter='1';
   column SalesRep Month Type Units;
   define SalesRep / order;
   define Month / order;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'First Quarter Sales Report';
run;
```

DEFINE ステートメントにより SalesRep と Month が ORDER 変数であることが示されています。COLUMN ステートメントにより列の順序が指定されます。デフォルトでは、行が SalesRep のフォーマットされた値の昇順で並べ替えられます。各営業担当者の行が Month の値で並べ替えられます。

次の出力はレポートを示しています。

アウトプット 29.4 *行の並べ替え*

TruBlend Coffee Makers, Inc.
First Quarter Sales Report

SalesRep	Month	Type	Units
Garcia	01	Standard	41
		Deluxe	715
	02	Standard	2045
		Deluxe	10
		Standard	40
		Standard	98
	03	Standard	118
		Standard	310
Hollingsworth	01	Deluxe	260
		Standard	330
	02	Standard	1030
	03	Standard	125
		Standard	25
Jensen	01	Standard	110
		Standard	675
	02	Standard	153
	03	Standard	154
		Standard	525

PROC REPORT では、ある行と次の行の値が同じ場合、ORDER 変数の値は繰り返されません。

単一行に複数のオブザベーションをまとめる

PROC REPORT で 1 つ以上の GROUP 変数を定義して要約レポートを作成できます。グループは、すべての GROUP 変数に対して一意の値の組み合わせを持つオブザベーションのセットです。PROC REPORT では、各グループをレポートの 1 行にまとめること(要約)が試みられます。

1 行にすべての列をまとめるには、レポートのすべての変数を GROUP、ANALYSIS、COMPUTED、または ACROSS のいずれかとして定義する必要があります。1 つ以上の DEFINE ステートメントで GROUP オプションを使用して、PROC REPORT でグループの形成に使用される変数を指定します。GROUP 変数として複数の変数を定義できますが、GROUP 変数は、他の種類の使用方法の変数に先行する必要があります。PROC REPORT では COLUMN ステートメントの変数の順序によって、ネストが決定されます。変数の使用方法の定義に関する詳細については、“レイアウトの作成” (491 ページ)を参照してください。

グループの ANALYSIS 変数の値は、統計量の値で、PROC REPORT によってグループ内のすべてのオブザベーションに対して計算されます。各 ANALYSIS 変数について、DEFINE ステートメントで統計量を指定します。デフォルトでは、PROC REPORT はすべての数値変数を ANALYSIS 変数として使用し、SUM 統計量を計算します。DEFINE ステートメントで要求できる統計量を次に示します。

表 29.1 *記述統計*

記述統計量のキーワード	
CSS	PCTSUM
CV	RANGE
MAX	STD
MEAN	STDERR
MIN	SUM
N	SUMWGT
NMISS	USS
PCTN	VAR
分位点統計量のキーワード	
MEDIAN | P50	Q3 | P75
P1	P90
P5	P95
P10	P99
Q1 | P25	QRANGE

仮説検定のキーワード	
PRT | PROBT	T

これらの統計量の定義と説明については、“SAS Elementary Statistics Procedures” (*Base SAS Procedures Guide*)を参照してください。

次のプログラムは、各営業担当者の年間合計売上を示す要約レポートを作成します。

```
proc report data=year_sales;
   column SalesRep Units AmountSold;
   define SalesRep /group;1
   define Units / analysis sum;2
    define AmountSold/ analysis sum;3
   title1 'TruBlend Coffee Makers Sales Report';
   title2 'Total Yearly Sales';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

- DEFINE ステートメントにより SalesRep が GROUP 変数であることが示されています。
- DEFINE ステートメントで Units は ANALYSIS 変数であることが示され、PROC REPORT で SUM 統計量を計算するよう指定します。
- DEFINE ステートメントで AmountSold は ANALYSIS 変数であることが示され、PROC REPORT で SUM 統計量を計算するよう指定します。

次の出力はレポートを示しています。

アウトプット 29.5 *要約レポートでの複数のオブザベーションのグループ化*

TruBlend Coffee Makers Sales Report
Total Yearly Sales

SalesRep	Units	AmountSold
Garcia	15969	512070.78
Hollingsworth	10620	347246.1
Jensen	14400	461162.6

レポートの各行は 1 つのグループを表し、一意の SalesRep の値を持つすべてのオブザベーションを要約します。PROC REPORT はこれらの行を GROUP 変数の昇順で並べ替えます。この例では、営業担当者のアルファベット順です。ANALYSIS 変数の値は、グループのすべてのオブザベーションの Units と AmountSold の合計です。この例では、各営業担当者の合計販売ユニット数と合計売上金額です。

行のデフォルトの順序の変更

DEFINE ステートメントで ORDER=または DESCENDING オプションを使用して、レポートの行のデフォルトの並べ替え順序を変更できます。ORDER=オプションにより変数の並べ替え順序が指定されます。次に示す順序で行を並べ替えできます。

DATA
入力データセットのデータの順序。

FORMATTED
フォーマットされた値の昇順。

FREQ
度数カウントの昇順。

INTERNAL
フォーマットされていない値(内部に格納されている値)の昇順。

デフォルトでは、PROC REPORT では変数のフォーマットされた値を使用して行が並べ替えられます。DESCENDING オプションは並べ替え順序を逆にし、PROC REPORT により値の降順で行が並べ替えられます。

次のプログラムは、販売数で並べ替えられた、第 1 四半期売上の詳細レポートを作成します。

```
proc report data=year_sales;
   where Quarter='1';
   column SalesRep Type Units Month;
   define SalesRep / order order=freq;
   define Units / order descending;
   define Type / order;
   title1 'TruBlend Coffee Makers, Inc.';
   title2 'First Quarter Sales Report';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

- DEFINE ステートメントにより、SalesRep、Units および Type が ORDER 変数であることが示され、各営業担当者が売り上げた販売数に対応しています。
- ORDER=FREQ オプションによりレポートの行が SalesRep の度数で並べ替えられます。
- DESCENDING オプションにより UNITS の行が最大値から最小値へ並べ替えられます。

次の出力はレポートを示しています。

アウトプット 29.6 *行の並べ替え順序の変更*

TruBlend Coffee Makers, Inc.
First Quarter Sales Report

SalesRep (2)	Type (3)	Units	Month (1)
Hollingsworth	Deluxe	260	01
	Standard	1030	02
		330	01
		125	03
		25	03
Jensen	Standard (4)	675	01
		525	03
		154	03
		153	02
		110	01
Garcia	Deluxe	715	01
		10	02
	Standard	2045	02
		310	03
		118	03
		98	02
		41	01
		40	02

次のリストは、前述のレポートの番号付き項目に対応しています。

1 列の順序は COLUMN ステートメントでの変数の指定順序に対応しています。DEFINE ステートメントの順序は、列の順序に影響しません。

2 行の順序は SalesRep の度数の昇順であるため、最小の販売数(オブザベーション数)の営業担当者が最初に表示され、最大の販売数を持つ営業担当者が最後に表示されます。

3 SalesRep 内の行の順序は、Type のフォーマットされた値の昇順であるため、Deluxe のコーヒーメーカーに関する売上情報が、Standard のコーヒーメーカーのものより先に表示されます。

4 Type 内の行の順序は Units のフォーマットされた値の降順であるため、最大の販売ユニット数のオブザベーションが最初に表示されます。

高度なレポートの作成

列のレイアウトの調整

ODS LISTING 出力の列幅とスペースについて

ODS LISTING 出力先に出力する場合、PROC REPORT ステートメントまたは DEFINE ステートメントのいずれかに指定するオプションによって、列スペース (SPACING=)および列幅(COLWIDTH=)を変更できます。

列と列の間のスペースを制御するには、次のステートメントに SPACING=オプションを使用します。

- すべての列間のデフォルトのブランク文字数を指定するには、PROC REPORT ステートメント
- デフォルト値を上書きして、特定の列の左側のブランク文字数を指定するには、DEFINE ステートメント

LISTING 出力で、PROC REPORT は列と列の間に 2 つのブランクスペースを挿入します。列と列の間のスペースを削除するには、SPACING=0 を指定します。PROC REPORT で許可される列と列の間の最大スペースは、レポートの列数によって異なります。すべての列幅の合計に各列の左側のブランク文字を加えたものが、行サイズを超えることはできません。

列幅を指定するには、次のオプションを使用します。

- 計算変数または数値のデータセット変数を含む列のデフォルトの文字数を指定するには、PROC REPORT ステートメントで COLWIDTH=オプションを使用します。
- PROC REPORT でレポート項目の表示に使用される列の幅を指定するには、DEFINE ステートメントで WIDTH=オプションを使用します。

デフォルトでは、数値の列幅は 9 文字です。列幅は最小 1 文字から最大行サイズまで指定できます。PROC REPORT では、まず DEFINE ステートメントの WIDTH=オプションに従って列の幅が設定されます。WIDTH=が省略されると、PROC REPORT では、レポート項目の出力形式を格納するのに十分な広さの列幅が使用されます。出力形式が割り当てられていない場合、列幅は、文字変数の長さまたは COLWIDTH=オプションの値のいずれかになります。

レポート項目のフォーマットされた値の整列方法、および列ヘッダーと列幅を指定して、列のレイアウトを調整できます。DEFINE ステートメントに次のオプションを使用して列を整列します。

CENTER
　列の値と列ヘッダーを中央揃えにします。

LEFT
　列の値と列ヘッダーを左揃えにします。

RIGHT
　列の値と列ヘッダーを右揃えにします。

列幅とスペースの変更

次のプログラムは、各営業担当者の年間合計売上を示す要約レポートの列のスペースを変更します。

```
options linesize=80 pageno=1 nodate;
```

```
proc report data=year_sales spacing=3;
   column SalesRep Units AmountSold;
   define SalesRep /group right;
   define Units / analysis sum width=5;
   define AmountSold/ analysis sum width=10;
   title1 'TruBlend Coffee Makers Sales Report';
   title2 'Total Yearly Sales';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

- PROC REPORT ステートメントの SPACING=オプションによりすべての列間に 3 つのブランク文字が挿入されます。
- DEFINE ステートメントの RIGHT オプションにより営業担当者の名前と列ヘッダーがその列で右揃えにされます。
- DEFINE ステートメントの WIDTH=オプションにより列ヘッダーを 1 行に格納するのに十分なスペースが指定されます。

次の出力はレポートを示しています。

アウトプット 29.7 *LISTING 出力の列幅とスペースの調整*

```
TruBlend Coffee Makers Sales Report                      1 Total Yearly Sales
SalesRep   Units   AmountSold Garcia   15969    512070.78 Hollingsworth
10620     347246.1 Jensen   14400     461162.6
```

SalesRep の列幅は変数の長さである 14 文字幅です。

列ヘッダーのカスタマイズ

列ヘッダーの構造について

ODS LISTING 出力では、PROC REPORT によってヘッダーと詳細行を視覚的に区別するために列ヘッダーの下に縦のスペースは挿入されません。ODS LISTING 出力先を使用して生成されるレポートの外観を見やすくするために、列ヘッダーにアンダーラインを引いたり、列ヘッダーの下にブランク行を挿入したり、独自の列ヘッダーを指定したりできます。PROC REPORT ステートメントでは、HEADLINE オプションを使用して列ヘッダーにアンダーラインを引いたり、HEADSKIP オプションを使用して列ヘッダーの後にブランク行を挿入したりできます。

デフォルトでは、データセット変数にラベルが事前に割り当てられている場合、SAS は変数名または変数ラベルを列ヘッダーに使用します。異なる列ヘッダーを指定するには、そのレポート項目の DEFINE ステートメントにテキストを一重引用符または二重引用符で囲んで設定します。

デフォルトでは、列の幅に基づいて PROC REPORT が列ヘッダーに改行を生成します。ラベルに複数の引用符のセットを使用した場合、各セットにより個別のヘッダー行が定義されます。ラベルに分割文字を含めた場合、PROC REPORT では分割文字に達するとヘッダーが改行され、次の行にヘッダーが続けられます。デフォルトでは、分割文字はスラッシュ(/)です。かわりの分割文字を指定するには、PROC REPORT ステートメントに SPLIT=オプションを指定します。

LISTING 出力の列ヘッダーの変更

次のプログラムは、列ヘッダーが複数行になる、変数 SalesRep、Units および AmountSold の要約レポートを作成します。

```
ods listing;
ods html close;
options linesize=80 pageno=1 nodate;

proc report data=year_sales nowindows spacing=3 headskip;
   column SalesRep Units AmountSold;
   define SalesRep /group 'Sales/Representative';
   define Units / analysis sum 'Units Sold' width=5;
   define AmountSold/ analysis sum 'Amount' 'Sold';
   title1 'TruBlend Coffee Makers Sales Report';
   title2 'Total Yearly Sales';
run;
ods listing close;
ods html;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

- HEADSKIP オプションにより列ヘッダーの後にブランク行が挿入されます。
- 引用符で囲んだテキストで列ヘッダーを指定します。

次の出力はレポートを示しています。

アウトプット 29.8 *LISTING 出力の列ヘッダーの変更*

```
TruBlend Coffee Makers Sales Report                      1 Total Yearly Sales
Sales            Units      Amount Representative    Sold        Sold
Garcia           15969   512070.78 Hollingsworth    10620    347246.1
Jensen           14400    461162.6
```

ラベル Units Sold は、このレポート項目の列幅が 5 文字幅のため、2 行に分割されます。PROC REPORT ステートメントの SPLIT=オプションは、SalesRep のラベル、Sales Representative(営業担当者)がどこで分割されるかを識別します。逆に、AmountSold のラベルでは、一重引用符のセットを複数使用してラベルの分割位置を識別します。

出力形式の指定

SAS 出力形式の使用

レポート項目の出力形式を指定して、レポートを簡単かつ効率的に読みやすくできます。列に出力形式を割り当てるには、FORMAT ステートメントを使用するか、または DEFINE ステートメントに FORMAT=オプションを使用します。FORMAT ステートメントは、データセットの変数にのみ作用します。FORMAT=オプションでは、任意のレポート項目に SAS 出力形式またはユーザー定義の出力形式を割り当てられます。

PROC REPORT はレポート項目のフォーマット方法を決定するために、使用する出力形式を次の箇所から次の順序で検索します。

1. DEFINE ステートメントの FORMAT=オプション
2. FORMAT ステートメント
3. データセット

PROC REPORT では、最初に検出した出力形式が使用されます。出力形式が割り当てられていない場合、PROC REPORT では、BEST9.出力形式が数値変数に、$w.出力形式が文字変数に使用されます。

レポート項目への出力形式の適用

次のプログラムは、各営業担当者の年間合計売上の要約レポートの列に出力形式を適用する方法を示しています。

```
proc report data=year_sales;
   column SalesRep Units AmountSold;
   define SalesRep / group 'Sales/Representative';
   define Units / analysis sum 'Units Sold' format=comma7.;
   define AmountSold / analysis sum 'Amount' 'Sold' format=dollar14.2;
   title1 'TruBlend Coffee Makers Sales Report';
   title2 'Total Yearly Sales';
run;
```

PROC REPORT により、変数 Units の値に COMMA7.出力形式が適用され、変数 AmountSold の値に DOLLAR14.2 出力形式が適用されます。

次の出力はレポートを示しています。

アウトプット 29.9 *数値の列のフォーマット*

TruBlend Coffee Makers Sales Report
Total Yearly Sales

Sales Representative	Units Sold	Amount Sold
Garcia	15,969	$512,070.78 1
Hollingsworth	2 10,620	$347,246.10
Jensen	14,400	$461,162.60

次のリストは、前述のレポートの番号付き項目に対応しています。

1 変数 AmountSold に最大列幅が 14 スペースの DOLLAR14.2 出力形式が使用されます。2 スペースは、値の小数部分用に予約されています。残りの 12 スペースに、小数点、整数、ドル記号、カンマ、および値が負の場合にはマイナス符号が含まれます。

2 変数 Units に最大列幅が 7 スペースの COMMA7.出力形式が使用されます。列の幅には、数値、カンマ、および値が負の場合にはマイナス符号が含まれます。

これらの出力形式は、SAS データセットに格納されている実際のデータ値には影響しません。つまり、出力形式は、値のレポートでの表示方法にのみ影響します。

列ヘッダーとして変数の値を使用

列ヘッダーの作成

データセット変数の値から列ヘッダーを作成してクロス集計表を生成するには、DEFINE ステートメントで ACROSS オプションを使用します。ACROSS 変数が定義されると、PROC REPORT では、ACROSS 変数の各値に対して列が作成されます。

ACROSS 変数によって作成される列には、統計量または計算値が含まれます。ACROSS 変数の前後に変数がない場合、PROC REPORT により、レポートのセルに

属する、入力データセットのオブザベーションの数が表示されます(N 統計量)。セルは、レポートの単一ユニットで、行と列の交差によって形成されます。

このセクションの例では、ANALYSIS 変数の計算された度数カウント(N 統計量)と統計量の表示方法を示します。レポートのセルへの計算変数の配置に関する詳細については、“REPORT Procedure” (*Base SAS Procedures Guide*)を参照してください。

度数カウントの作成

次のプログラムは、各営業担当者の販売数を表形式で表すレポートを作成します。

```
proc report data=year_sales colwidth=5;
   column SalesRep Type n;
   define SalesRep /group 'Sales Representative';
   define Type / across 'Type of Coffee Maker';
   define n / 'Total';
   title1 'TruBlend Coffee Makers Yearly Sales Report';
   title2 'Number of Sales';
 run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

- COLUMN ステートメントにより、レポートに 2 つのデータセット変数と計算された統計量 N を含むように指定されています。この N 統計量が指定されたため、PROC REPORT により 3 番目の列が追加され、各営業担当者のオブザベーションの数が表示されます。
- DEFINE ステートメントにより Type が ACROSS 変数であることが示されています。

次の出力はレポートを示しています。

アウトプット 29.10 *度数カウントの表示*

TruBlend Coffee Makers Yearly Sales Report
Number of Sales

	Coffee Maker 1		
Sales Representative	**Deluxe**	**Standard**	**Total** 2
Garcia	4	36	40
Hollingsworth	8	24	32
Jensen	4	34	38

次のリストは、前述のレポートの番号付き項目に対応しています。

1 Type は ACROSS 変数であり、その前後には変数がありません。そのため、入力データセットに含まれる、営業担当者とコーヒーメーカーの種類ごとのオブザベーションの数がレポートに示されます。

2 N 統計量の列には、Total とラベル付けされ、各営業担当者のオブザベーションの合計数が含まれます。

デフォルトでは、PROC REPORT で ACROSS 変数の列がそのフォーマットされた値に従って並べ替えられました。DEFINE ステートメントに ORDER=オプションを使用して、

ACROSS 変数の並べ替え順序を変更できます。詳細については、次を参照してください。

複数の ANALYSIS 変数による列の共有

ACROSS 変数が作成する列に ANALYSIS 変数の値を表示させることで、高度なクロス集計表を作成できます。ACROSS 変数が 1 つ以上の ANALYSIS 変数と列を共有する場合、PROC REPORT により列がスタックされます。たとえば、ACROSS 変数 Type の列を ANALYSIS 変数 Units と共有し、各列にコーヒーメーカーの種類ごとの販売ユニット数が含まれるようにできます。

ACROSS 変数によって作成される列に ANALYSIS 変数の値をスタックするには、次のように、COLUMN ステートメントでその変数を ACROSS 変数の次に指定します。

```
column SalesRep Type, Unit;
```

ANALYSIS 変数と ACROSS 変数の間はカンマで区切ります。複数の ANALYSIS 変数を指定するには、次のように、変数の名前をかっこに入れてリストし、COLUMN ステートメントの ACROSS 変数の次に指定します。

```
column SalesRep Type,(Unit AmountSold);
```

ANALYSIS 変数の前に ACROSS 変数を置くと、レポートでは ACROSS 変数の名前と値が ANALYSIS 変数の名前の上に表示されます。ANALYSIS 変数の後に ACROSS 変数を置くと、レポートでは ACROSS 変数の名前と値が ANALYSIS 変数の名前の下に表示されます。

デフォルトでは、PROC REPORT で ANALYSIS 変数の SUM 統計量が計算されます。列に別の統計量を表示するには、DEFINE ステートメントを使用して、計算する ANALYSIS 変数の統計量を指定します。使用可能な統計量のリストについては、表 29.1 (498 ページ)を参照してください。

次のプログラムは、各営業担当者の販売コーヒーメーカー数と平均売上金額(米ドル)を表形式で表すレポートを作成します。

```
proc report data=year_sales;
   column SalesRep Type,(Units Amountsold);
   define SalesRep /group 'Sales Representative';
   define Type / across '';
   define units / analysis sum 'Units Sold' format=comma7.;
   define AmountSold / analysis mean 'Average/Sale' format=dollar12.2;
   title1 'TruBlend Coffee Makers Yearly Sales Report';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

- COLUMN ステートメントにより SalesRep と Type の列が作成されます。ACROSS 変数 Type はその列を ANALYSIS 変数の Units および AmountSold と共有します。
- DEFINE ステートメントは列ヘッダーの Type のラベルとしてブランクを使用します。
- DEFINE ステートメントは ANALYSIS 変数 Units を使用して SUM 統計量を計算します。
- DEFINE ステートメントは ANALYSIS 変数 AmountSold を使用して MEAN 統計量を計算します。

次の出力はレポートを示しています。

アウトプット 29.11 *複数の ANALYSIS 変数による列の共有*

TruBlend Coffee Makers Yearly Sales Report

	Deluxe		Standard	
Sales Representative	Units Sold	Average Sale	Units Sold	Average Sale
Garcia	945	$11,694.38	15,024	$12,924.81
Hollingsworth	760	$4,702.50	9,860	$12,901.09
Jensen	820	$10,147.50	13,580	$12,369.78

特定の種類のコーヒーメーカーの列の値は、各営業担当者の合計販売ユニット数と平均売上金額(米ドル)です。

オブザベーションのグループの要約

グループ要約の使用

レポートによっては、オブザベーションのグループの情報を要約して、視覚的に各グループを分離する場合があります。これを行うには、レポートの各グループの前または後に区切りを作成します。

各グループを視覚的に区切るには、区切りに区切り行と呼ばれるテキストの行を挿入します。区切り行は、レポートの先頭や末尾、各ページの最上部や最下部、およびグループや ORDER 変数の値が変わるごとに配置できます。区切り行には、次の項目を含めることができます。

- テキスト(ブランクを含む)
- 統計量の要約
- レポート変数
- 計算変数

グループ要約を作成するには、BREAK ステートメントを使用します。BREAK ステートメントには次の項目を示す順序で含める必要があります。

- キーワード BREAK
- 区切りの場所(BEFORE または AFTER)
- BREAK 値と呼ばれる GROUP 変数の名前

PROC REPORT により、BREAK 変数の値が変わるたびに区切りが作成されます。要約を各グループの最初の行の前に表示するには、BEFORE 引数を使用します。要約を各グループの最後の行の後に表示するには、AFTER 引数を使用します。

レポート全体の要約情報を作成するには、RBREAK ステートメントを使用します。RBREAK ステートメントには次の項目を示す順序で含める必要があります。

- キーワード RBREAK
- 区切りの場所(BEFORE または AFTER)

RBREAK ステートメントを使用すると、PROC REPORT により、テキスト、レポート全体の要約統計量、または計算変数が、レポートの詳細行の先頭または末尾に挿入されます。要約をレポートの最初の行の前に表示するには、BEFORE 引数を使用します。要約を各グループの最後の行の後に表示するには、AFTER 引数を使用します。

BREAK ステートメントと RBREAK ステートメントのどちらも、グループ要約とレポート要約の外観を制御するオプションをサポートしています。ステートメント内では、任意の順序で任意の組み合わせのオプションを使用できます。使用可能なオプションのリストについては、“REPORT Procedure” (*Base SAS Procedures Guide*)を参照してください。

グループ要約の作成

次のプログラムは、区切り行を使用した要約レポートを作成して、営業担当者ごとの小計と年間売上、および営業担当者すべての年間総計を表示します。

```
ods listing;
options linesize=80 pageno=1 nodate linesize=84;

proc report data=year_sales headskip;
   column Salesrep Quarter Units AmountSold;
   define SalesRep / group 'Sales Representative';
   define Quarter / group center;
   define Units / analysis sum 'Units Sold' format=comma7.;
   define AmountSold / analysis sum 'Amount/Sold' format=dollar14.2;
   break after SalesRep / summarize skip ol suppress;
   rbreak after / summarize skip dol;
   title1 'TruBlend Coffee Makers Sales Report';
   title2 'Total Yearly Sales';
run;
ods listing close;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

- DEFINE ステートメントの CENTER オプション(LISTING 出力のみ)により、変数 Quarter の値と列ヘッダーのラベルが中央揃えにされます。
- GROUP 変数 SalesRep の値が変わった後に、BREAK ステートメントにより区切り行が追加されます。SUMMARIZE オプションにより、区切り行の各グループの統計量を要約するための要約行が書き込まれます。SKIP オプション(LISTING 出力のみ)により、区切り行の各グループの後にブランク行が挿入されます。OL オプション(LISTING 出力のみ)により、要約行の各値の上にハイフン(-)の行が書かれます。SUPPRESS オプションにより、BREAK 変数の値の出力と BREAK 変数の列の上線が非表示にされます。
- RBREAK ステートメントは、レポートの末尾に区切り行を追加します。SUMMARIZE オプションにより、ANALYSIS 変数の Units および AmountSold の SUM 統計量を要約する要約行が記述されます。SKIP オプション(LISTING 出力のみ)により、区切り行の前にブランク行が挿入されます。DOL オプション (LISTING 出力のみ)により、要約行の各値の上に等号(=)の行が書かれます。

次に、LISTING 出力先のレポートを表示します。

アウトプット 29.12 *グループ要約の作成*

```
TruBlend Coffee Makers Sales Report 1 Total Yearly Sales
Sales                       Units          Amount Representative  Quarter
Sold            Sold Garcia             1        3,377     $118,019.94 2
3,515     $108,859.55 3        7,144     $225,326.28 4        1,933
$59,865.01 -------  -------------- 15,969 1 $512,070.78 1 Hollingsworth
1        1,770      $59,634.70 2        3,090      $96,160.55 3
3,285     $109,704.35 4        2,475      $81,746.50 -------  --------------
10,620     $347,246.10 Jensen             1        1,617      $50,078.49
2        2,413      $74,730.61 3        6,687     $222,290.99 4
3,683     $114,062.51 -------  -------------- 14,400     $461,162.60 =======
============== 40,989 2 $1,320,479.48 2
```

次のリストは、前述の LISTING 出力の番号付き項目に対応しています。

1 グループ要約行の ANALYSIS 変数の Units と AmountSold の値は、グループのすべての行の合計(小計)です。

2 レポート要約行の ANALYSIS 変数の Units と AmountSold の値は、レポートのすべての行の合計(総計)です。

このレポートでは、Units と AmountSold は、SUM 統計量の計算に使用される ANALYSIS 変数です。これらの変数で異なる統計量を計算するように定義されている場合、要約行の値は、グループのすべての行とレポートのすべての行の統計量の値になります。

要約

PROC REPORT ステートメント

PROC REPORT <DATA=*SAS-data-set*> <*option(s)*>;

BREAK *location break-variable* <*/option(s)*>;

COLUMN *column-specification(s)*;

DEFINE *report-item* /<*usage*> <*option(s)*>;

RBREAK *location*<*/option(s)*>;

TITLE<*n*> <*'title'*>;

WHERE *where-expression*;

PROC REPORT <DATA=SAS-data-set> <option(s)>;

プロシジャを開始します。他のステートメントが使用されていない場合、SAS-data-set のすべての変数が **REPORT** ウィンドウの詳細レポートに表示されます。データセットに数値データのみが含まれている場合、PROC REPORT は要約レポートにすべての変数を表示します。次に示すその他のステートメントを使用して、レポートの構造を制御できます。

PROC REPORT ステートメントでは、次のオプションを指定できます。

COLWIDTH=column-width

計算変数または数値のデータセット変数を含む、列のデフォルトの文字数を指定します。このオプションは LISTING 出力にのみ影響します。

DATA=SAS-data-set
PROC REPORT で使用される SAS データセットの名前を指定します。DATA= が省略されると、PROC REPORT は最後に作成されたデータセットを使用します。

HEADLINE
レポートの各ページの上部にある列ヘッダーの下にハイフン(-)の行を挿入します。このオプションは LISTING 出力にのみ影響します。

HEADSKIP
レポートの各ページの上部にあるすべての列ヘッダーの下(または HEADLINE オプションによって挿入される行の下)にブランク行を挿入します。このオプションは LISTING 出力にのみ影響します。

SPACING=space-between-columns
列と列の間のブランク文字数を指定します。各列の列幅とその左側にある列との間のブランク文字の合計が、行サイズを超えることはできません。このオプションは LISTING 出力にのみ影響します。

SPLIT='character'
分割文字を指定します。PROC REPORT は、分割文字に達すると列ヘッダーを改行し、次の行にヘッダーを続けます。出現するすべての分割文字は、ラベルの 256 文字の制限にカウントされますが、分割文字自体は列ヘッダーの一部ではありません。LISTING 出力では、分割文字はヘッダー行とデータ値で有効です。それ以外のすべての出力先では、分割文字はヘッダー行でのみ有効です。

WINDOWS | NOWINDOWS
ウィンドウ環境または非ウィンドウ環境を選択します。

NOWINDOWS を使用すると、PROC REPORT は **REPORT** ウィンドウを表示せずに実行され、SAS プロシジャ出力に結果が送られます。NOWINDOWS(エイリアスは NOWD)がデフォルトです。WINDOWS を使用する場合、**REPORT** ウィンドウが SAS により開かれるため、レポートを繰り返し変更し、その変更をすぐに確認することができます。

BREAK location break-variable </option(s)>;
デフォルトの要約を区切り(GROUP 変数または ORDER 変数の値で変わる場所)に生成します。要約の情報はオブザベーションのセットに対応しています。オブザベーションは、BREAK 変数と、レポートで BREAK 変数の左側にあるその他すべての GROUP 変数または ORDER 変数との一意の値の組み合わせを共有します。

BREAK ステートメントで次の引数を指定する必要があります。

location
区切り行の配置を次の位置に制御します。

AFTER
区切り行を、同じ BREAK 変数の値を持つ行のセットごとに最後の行の直後に配置します。

BEFORE
区切り行を、同じ BREAK 変数の値を持つ行のセットごとに最初の行の直前に配置します。

break-variable
GROUP 変数または ORDER 変数です。PROC REPORT では、この変数の値が変わるたびに区切り行が書き込まれます。BREAK ステートメントでは、次のオプションを指定できます。

OL
要約行に表示される各値の上にハイフン(-)の行を挿入します。このオプションは LISTING 出力にのみ影響します。

SKIP
最後の区切り行にブランク行を書き込みます。このオプションは LISTING 出力にのみ影響します。

SUMMARIZE
区切り行の各グループの要約行を書き込みます。

SUPPRESS
要約行の BREAK 変数の値、および区切り行のアンダーラインやオーバーラインの出力を非表示にします。

COLUMN <column-specification(s)>;
レポートの列を形成する項目を指定して、すべての列の配置を記述します。COLUMN ステートメントに次の column-specification(s)を指定できます。

- *report-item(s)*
- *report-item-1*, *report-item-2 <..., report-item-n>*

report-item はレポートの列を形成する項目を示します。report-item はデータセット変数の名前、計算変数、または統計量のいずれかです。

report-item-1, report-item-2 <..., report-item-n>
列のコンテンツを一括設定するレポート項目を示します。各項目によりヘッダーが生成され、ヘッダーの上に他のヘッダーがスタックされるため、これらの項目は、レポートでスタックされると言います。最左の項目のヘッダーが最上部に配置されます。項目の 1 つが ANALYSIS 変数の場合、計算変数、または統計量の値がレポートのその部分のセルに格納されます。そうでない場合、PROC REPORT によりセルに度数カウントが格納されます。

DEFINE report-item / <usage> <option(s)>;
レポート項目の使用方法と表示方法を記述します。レポート項目は、データセット変数の名前かエイリアス(COLUMN ステートメントで作成)、計算変数、または統計量のいずれかです。レポート項目の使用法を次に示します。

- ACROSS
- ANALYSIS
- COMPUTED
- %DISPLAY
- GROUP
- ORDER

DEFINE ステートメントでは、次のオプションを指定できます。

CENTER
レポート項目のフォーマットされた値を列幅内で中央揃えにし、列ヘッダーを値の上に中央揃えにして配置します。

column-header
レポート項目の列ヘッダーを定義します。それぞれのヘッダーを一重引用符または二重引用符で囲みます。複数の列ヘッダーを指定すると、PROC REPORT はそれぞれに個別の行を使用します。分割文字を使用して列ヘッダーを複数の行に分割することもできます。

DESCENDING
PROC REPORT で行(GROUP 変数、ORDER 変数、ACROSS 変数の値)を表示する順序を逆にします。

FORMAT=format
SAS 出力形式またはユーザー定義の出力形式をレポート項目に割り当てます。この出力形式は、PROC REPORT での表示時に *report-item* に適用されます。データセットの変数に関連付けられている出力形式はこの出力形式によって変更されません。

ORDER=DATA | FORMATTED | FREQ | INTERNAL
GROUP 変数、ORDER 変数、ACROSS 変数の値を次に示す指定の順序に従って並べ替えます。

DATA
入力データセットでの順序に従って値を並べ替えます。

FORMATTED
フォーマットされた(外部)値で値を並べ替えます。デフォルトでは、順序は昇順です。

FREQ
度数カウントの昇順で値を並べ替えます。

INTERNAL
フォーマットされていない値で値を並べ替えます。PROC SORT を使用した場合と同じ順序になります。この順序は使用している動作環境によって異なります。この並べ替え順序は、日付を経時的に表示するのに特に便利です。

RIGHT
指定のレポート項目のフォーマットされた値を列幅内で右揃えにし、列ヘッダーを値の上に右揃えにして配置します。出力形式の幅と列幅が同じ場合、RIGHT は値の配置に影響しません。このオプションは LISTING 出力にのみ影響します。

SPACING=horizontal-positions
定義しようとしている列とそのすぐ左隣りの列との間に残すブランク文字数を指定します。各列の列幅とその左側にある列との間のブランク文字の合計が、行サイズを超えることはできません。このオプションは LISTING 出力にのみ影響します。

statistic
統計量と ANALYSIS 変数を関連付けます。PROC REPORT によりこの統計量を使用して、レポートの各セルで表されるオブザベーションの ANALYSIS 変数の値が計算されます。統計量と変数を関連付けない場合、PROC REPORT では SUM 統計量が計算されます。これ以外の種類の変数定義では、*statistic* は使用できません。

WIDTH=column-width
PROC REPORT が *report-item* を表示する列の幅を定義します。このオプションは LISTING 出力にのみ影響します。

RBREAK location </option(s)>;
レポートの先頭または末尾にデフォルトの要約を生成します。

RBREAK ステートメントで次の引数を指定する必要があります。

location
区切り行の配置を制御し、値は次のいずれかです。

AFTER
区切り行をレポートの末尾に配置します。

BEFORE
区切り行をレポートの先頭に配置します。

RBREAK ステートメントでは、次のオプションを指定できます。

DOL
要約行に表示されるそれぞれの値の上に二重線を引くよう指定します。このオプションは LISTING 出力にのみ影響します。

SKIP
レポートの先頭に配置される区切りの最後の区切り行の後にブランク行を書き込みます。このオプションは LISTING 出力にのみ影響します。

SUMMARIZE
要約行を区切り行の一部として含みます。レポートの先頭または末尾の要約行には、統計量、ANALYSIS 変数、または計算変数の値が含まれます。

TITLE<n> <'title'>;
タイトルを指定します。引数 *n* は、ワード TITLE の直後にブランクを空けずに続ける 1-10 の数値で、TITLE のレベルを指定します。各 *title* のテキストは一重引用符または二重引用符で囲む必要があります。タイトルの最大長は、動作環境と LINESIZE=システムオプションの値によって異なります。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

WHERE where-expression;
各オブザベーションが処理に使用できる前に一致する必要がある特定の条件を指定し、入力データセットをサブセット化します。*Where-expression* は条件を定義します。条件は、有効な算術式または論理式で、通常、一連のオペランドと演算子で構成されます。

詳細情報

KEEP=データセットオプション
その他の例については、“選択した変数の読み込み” (96 ページ)を参照してください。KEEP=データセットオプションの詳細なドキュメントについては、“KEEP= Data Set Option” (*SAS Data Set Options: Reference*) を参照してください。

PROC PRINT
さまざまな種類の詳細レポートの作成方法の説明については、27 章, “PRINT プロシジャを使用した詳細レポートの作成” (421 ページ)を参照してください。

PROC REPORT
詳細なドキュメントについては、“REPORT Procedure” (*Base SAS Procedures Guide*) を参照してください。

PROC TABULATE
さまざまな種類の要約レポートの作成方法の説明については、28 章, “TABULATE プロシジャを使用した要約テーブルの作成” (461 ページ)を参照してください。

SAS 出力形式
詳細なドキュメントについては、*SAS Formats and Informats: Reference* を参照してください。SAS ソフトウェアでは、小数値、16 進値、ローマ数字、社会保障番号、日付と時刻、および文字として記述される数値など、多くの出力形式が使用可能です。

WHERE ステートメント
詳細な説明については、“WHERE ステートメントについて” (432 ページ)を参照してください。WHERE ステートメントの詳細な参照ドキュメントについては、*SAS Statements: Reference* を参照してください。WHERE 処理の詳細については、*SAS Language Reference: Concepts* を参照してください。

7 部

プロットとチャートの作成

30 章
変数間のリレーションシップのプロット 519

31 章
変数を要約するチャートを作成する 539

30 章

変数間のリレーションシップのプロット

変数間のリレーションシップのプロットについて **519**
概要 519
前提条件 520

例で使用される入力ファイルおよび入力 SAS データセット **520**

1 つの組み合わせの変数のプロット **524**
PLOT ステートメントについて 524
例 525

プロットの拡張 **525**
軸ラベルの指定 525
目盛値の指定 526
プロット記号の指定 528
凡例の削除 529

複数の組み合わせの変数のプロット **530**
複数のプロットのそれぞれ異なるページへの作成 530
複数のプロットの同一ページへの作成 532
複数の組み合わせの変数の同じ軸へのプロット 534

要約 **536**
PROC PLOT ステートメント 536

詳細情報 **537**

変数間のリレーションシップのプロットについて

概要

変数の値のプロットは、変数間のリレーションシップを検証する効率的な方法です。PLOT プロシジャを使用して、データのリレーションシップとパターンを示すことができます。

このセクションでは、次のトピックを取り上げます。

- 1 つの組み合わせの変数をプロットする
- プロットの外観を向上させる
- 複数のプロットをそれぞれ異なるページに作成する
- 複数のプロットを同一ページに作成する

- 複数の組み合わせの変数を同じ軸のペアにプロットする

前提条件

このセクションの例を理解するために、次の機能と概念を理解していることを確認してください。

- LOG 関数
- FORMAT ステートメント
- LABEL ステートメント
- TITLE ステートメント

例で使用される入力ファイルおよび入力 SAS データセット

このセクションの例では、1 つの入力ファイルと 1 つの SAS データセットを使用します。入力ファイルには、1968 年から 2008 年までの、ダウジョーンズ工業株価平均(Dow Jones Industrial Average)の高値と安値の情報が含まれています。次に入力ファイルの構造を示します。

```
1968  03DEC1968  985.21  21MAR1968  825.13
1969  14MAY1969  968.85  17DEC1969  769.93
1970  29DEC1970  842.00  06MAY1970  631.16
1971  28APR1971  950.82  23NOV1971  797.97
1972  11DEC1972 1036.27  26JAN1972  889.15
...more data lines...
2005  04MAR2005 10940.55 20APR2005 10012.36
2006  27DEC2006 12510.57 20JAN2006 10667.39
2007  09OCT2007 14164.53 05MAR2007 12050.41
2008  02MAY2008 13058.20 10OCT2008 8451.19
```

入力ファイルには、次に示す値が左から右へと含まれています。

- オブザベーションに記述される年
- ダウジョーンズ工業株価平均(Dow Jones Industrial Average)の各年の最高値の日付
- ダウジョーンズ工業株価平均(Dow Jones Industrial Average)の各年の最高値の値
- ダウジョーンズ工業株価平均(Dow Jones Industrial Average)の各年の最安値の日付
- ダウジョーンズ工業株価平均(Dow Jones Industrial Average)の各年の最安値の値

次のプログラムは、SAS データセット HIGHLOW を作成します。

```
data highlow;
   infile 'your-input-file';
   input Year @7 DateOfHigh:date9. DowJonesHigh @26 DateOfLow:date9. DowJonesLow;
   format LogDowHigh LogDowLow 5.2 DateOfHigh DateOfLow date9.;
   LogDowHigh=log(DowJonesHigh);
   LogDowLow=log(DowJonesLow);
```

```
run;
```

計算変数 LogDowHigh と LogDowLow には、ダウジョーンズ工業株価平均(Dow Jones Industrial Average)の各年の最高値と最安値のログ変換が含まれます。

```
proc print data=highlow;
   title 'Dow Jones Industrial Average Yearly High and Low Values';
run;


data highlow;
   input Year @7 DateOfHigh:date9. DowJonesHigh @26 DateOfLow:date9. DowJonesLow;
   format LogDowHigh LogDowLow 5.2 DateOfHigh DateOfLow date9.;
   LogDowHigh=log(DowJonesHigh);
   LogDowLow=log(DowJonesLow);
datalines;
1968  03DEC1968  985.21  21MAR1968  825.13
1969  14MAY1969  968.85  17DEC1969  769.93
1970  29DEC1970  842.00  06MAY1970  631.16
1971  28APR1971  950.82  23NOV1971  797.97
1972  11DEC1972 1036.27  26JAN1972  889.15
1973  11JAN1973 1051.70  05DEC1973  788.31
1974  13MAR1974  891.66  06DEC1974  577.60
1975  15JUL1975  881.81  02JAN1975  632.04
1976  21SEP1976 1014.79  02JAN1976  858.71
1977  03JAN1977  999.75  02NOV1977  800.85
1978  08SEP1978  907.74  28FEB1978  742.12
1979  05OCT1979  897.61  07NOV1979  796.67
1980  20NOV1980 1000.17  21APR1980  759.13
1981  27APR1981 1024.05  25SEP1981  824.01
1982  27DEC1982 1070.55  12AUG1982  776.92
1983  29NOV1983 1287.20  03JAN1983 1027.04
1984  06JAN1984 1286.64  24JUL1984 1086.57
1985  16DEC1985 1553.10  04JAN1985 1184.96
1986  02DEC1986 1955.57  22JAN1986 1502.29
1987  25AUG1987 2722.42  19OCT1987 1738.74
1988  21OCT1988 2183.50  20JAN1988 1879.14
1989  09OCT1989 2791.41  03JAN1989 2144.64
1990  16JUL1990 2999.75  11OCT1990 2365.10
1991  31DEC1991 3168.83  09JAN1991 2470.30
1992  01JUN1992 3413.21  09OCT1992 3136.58
1993  29DEC1993 3794.33  20JAN1993 3241.95
1994  31JAN1994 3978.36  04APR1994 3593.35
1995  13DEC1995 5216.47  30JAN1995 3832.08
1996  27DEC1996 6560.91  10JAN1996 5032.94
1997  06AUG1997 8259.31  11APR1997 6391.69
1998  23NOV1998 9374.27  31AUG1998 7539.07
1999  31DEC1999 11497.12 22JAN1999 9120.67
2000  14JAN2000 11722.98 07MAR2000 9796.04
2001  21MAY2001 11337.92 21SEP2001 8235.81
2002  19MAR2002 10635.25 09OCT2002 7286.27
2003  31DEC2003 10453.92 11MAR2003 7524.06
2004  28DEC2004 10854.54 25OCT2004 9749.99
2005  04MAR2005 10940.55 20APR2005 10012.36
2006  27DEC2006 12510.57 20JAN2006 10667.39
2007  09OCT2007 14164.53 05MAR2007 12050.41
2008  02MAY2008 13058.20 10OCT2008 8451.19
```

```
;
run;
```

図 30.1 HIGHLOW データセットの SAS 出力

Dow Jones Industrial Average Yearly High and Low Values

Obs	Year	DateOfHigh	DowJonesHigh	DateOfLow	DowJonesLow	LogDowHigh	LogDowLow
1	1968	03DEC1968	985.21	21MAR1968	825.13	6.89	6.72
2	1969	14MAY1969	968.85	17DEC1969	769.93	6.88	6.65
3	1970	29DEC1970	842.00	06MAY1970	631.16	6.74	6.45
4	1971	28APR1971	950.82	23NOV1971	797.97	6.86	6.68
5	1972	11DEC1972	1036.27	26JAN1972	889.15	6.94	6.79
6	1973	11JAN1973	1051.70	05DEC1973	788.31	6.96	6.67
7	1974	13MAR1974	891.66	06DEC1974	577.60	6.79	6.36
8	1975	15JUL1975	881.81	02JAN1975	632.04	6.78	6.45
9	1976	21SEP1976	1014.79	02JAN1976	858.71	6.92	6.76
10	1977	03JAN1977	999.75	02NOV1977	800.85	6.91	6.69
11	1978	08SEP1978	907.74	28FEB1978	742.12	6.81	6.61
12	1979	05OCT1979	897.61	07NOV1979	796.67	6.80	6.68
13	1980	20NOV1980	1000.17	21APR1980	759.13	6.91	6.63
14	1981	27APR1981	1024.05	25SEP1981	824.01	6.93	6.71
15	1982	27DEC1982	1070.55	12AUG1982	776.92	6.98	6.66
16	1983	29NOV1983	1287.20	03JAN1983	1027.04	7.16	6.93
17	1984	06JAN1984	1286.64	24JUL1984	1086.57	7.16	6.99
18	1985	16DEC1985	1553.10	04JAN1985	1184.96	7.35	7.08
19	1986	02DEC1986	1955.57	22JAN1986	1502.29	7.58	7.31
20	1987	25AUG1987	2722.42	19OCT1987	1738.74	7.91	7.46
21	1988	21OCT1988	2183.50	20JAN1988	1879.14	7.69	7.54
22	1989	09OCT1989	2791.41	03JAN1989	2144.64	7.93	7.67
23	1990	16JUL1990	2999.75	11OCT1990	2365.10	8.01	7.77
24	1991	31DEC1991	3168.83	09JAN1991	2470.30	8.06	7.81
25	1992	01JUN1992	3413.21	09OCT1992	3136.58	8.14	8.05
26	1993	29DEC1993	3794.33	20JAN1993	3241.95	8.24	8.08
27	1994	31JAN1994	3978.36	04APR1994	3593.35	8.29	8.19
28	1995	13DEC1995	5216.47	30JAN1995	3832.08	8.56	8.25
29	1996	27DEC1996	6560.91	10JAN1996	5032.94	8.79	8.52
30	1997	06AUG1997	8259.31	11APR1997	6391.69	9.02	8.76
31	1998	23NOV1998	9374.27	31AUG1998	7539.07	9.15	8.93
32	1999	31DEC1999	11497.12	22JAN1999	9120.67	9.35	9.12
33	2000	14JAN2000	11722.98	07MAR2000	9796.04	9.37	9.19
34	2001	21MAY2001	11337.92	21SEP2001	8235.81	9.34	9.02
35	2002	19MAR2002	10635.25	09OCT2002	7286.27	9.27	8.89
36	2003	31DEC2003	10453.92	11MAR2003	7524.06	9.25	8.93
37	2004	28DEC2004	10854.54	25OCT2004	9749.99	9.29	9.19
38	2005	04MAR2005	10940.55	20APR2005	10012.36	9.30	9.21
39	2006	27DEC2006	12510.57	20JAN2006	10667.39	9.43	9.27
40	2007	09OCT2007	14164.53	05MAR2007	12050.41	9.56	9.40
41	2008	02MAY2008	13058.20	10OCT2008	8451.19	9.48	9.04

1 つの組み合わせの変数のプロット

PLOT ステートメントについて

PLOT プロシジャは、1 つの座標軸セット内で、1 つの変数を別の変数に対してプロットする 2 次元のグラフを生成します。プロットの各ポイントの座標は、2 つの変数の値に対応しています。グラフは自動的に使用データの値に拡大縮小しますが、座標軸を指定してスケールを制御できます。

次の PLOT ステートメントを使用して、1 つの組み合わせのメジャーの単純な 2 次元プロットを作成します。

PROC PLOT <DATA=*SAS-data-set*>;

PLOT *vertical***horizontal*;

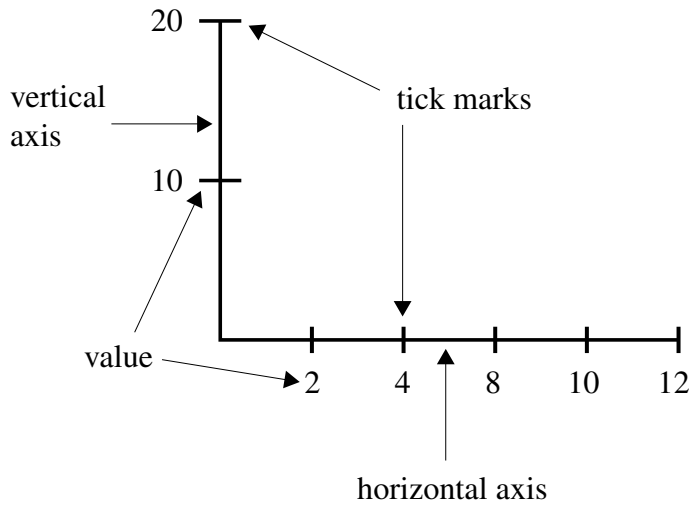
vertical は縦軸にプロットする変数の名前で、*horizontal* は横軸にプロットする変数の名前です。

デフォルトでは、PROC PLOT によりプロット記号が選択されます。データにより軸のラベル、軸の値、目盛の値が決定されます。プロットに次の項目が表示されます。

- vertical 変数の名前(縦軸の横)と horizontal 変数の名前(横軸の下)
- 均等にスペースが挿入された間隔(等間隔)に基づく軸と目盛
- 1 つのオブザベーションを示すためのプロット記号としての文字 A、2 つのオブザベーションが一致する場合のプロット記号としての文字 B、3 つが一致する場合の文字 C(以下同様に続く)
- プロットの変数の名前と凡例、およびプロット記号の意味

次の図では、プロット上の軸、値、目盛を示します。

図 30.2 *軸、値、目盛の図*



注: PROC PLOT は対話型プロシジャです。PROC PLOT ステートメントの発行後、プロシジャで有効な任意のステートメントを、PROC ステートメントを再サブミットすることなくサブミットできます。そのため、簡単かつ素早くラベルや目盛の値などを変更してみることができます。

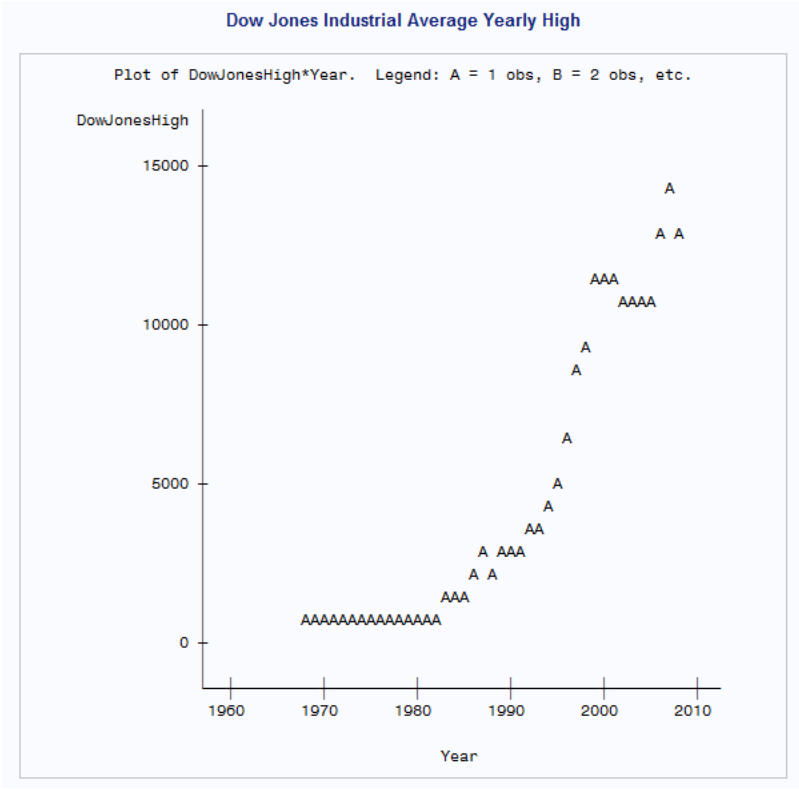
例

次のプログラムは、PLOT ステートメントを使用して、1968 年から 2008 年のダウジョーンズの高値のトレンドを示す単純なプロットを作成します。

```
proc plot data=highlow;
   plot DowJonesHigh*Year;
   title 'Dow Jones Industrial Average Yearly High';
run;
```

プロット出力を次に示します。

図 30.3 *単純なプロットを使用したデータのトレンドの表示*



プロットは、グラフィカルに過去 50 年間のダウジョーンズ工業株価平均の高値の指数トレンドを示します。最大の成長は、過去 10 年間に起こっており、およそ 6,000 ポイント増大しました。

プロットの拡張

軸ラベルの指定

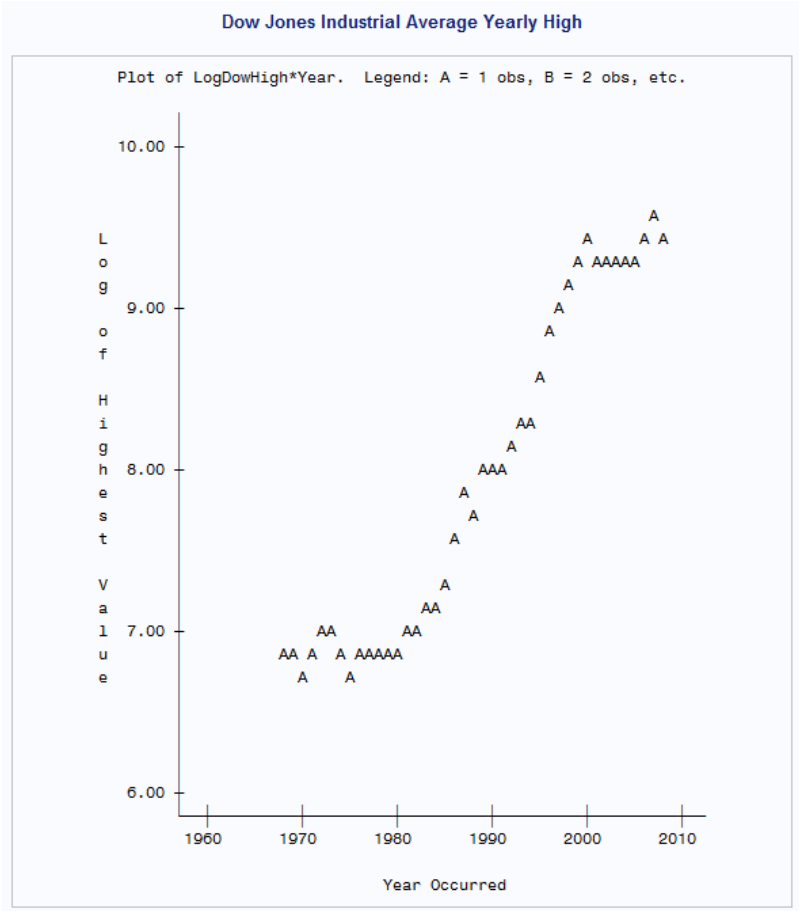
軸の追加情報を提供すると、有用な場合があります。縦軸と横軸のラベルを指定してプロットを拡張できます。

次のプログラムは、各年の DowJonesHigh のログ変換をプロットし、LABEL ステートメントを使用して軸のラベルを変更します。

```
proc plot data=highlow;
   plot LogDowHigh*Year;
   label LogDowHigh='Log of Highest Value'
         Year='Year Occurred';
   title 'Dow Jones Industrial Average Yearly High';
run;
```

プロット出力を次に示します。

図 30.4 *軸ラベルの指定*



DowJonesHigh のログ変換をプロットすると、指数トレンドから線形トレンドに変わります。各変数のラベルは、軸と平行に中央揃えにされます。

目盛値の指定

前のプロットでは、横軸の範囲は 1960 年から 2010 年までです。年を表す目盛とラベルは 10 の間隔が空けられています。PLOT ステートメントの HAXIS=オプションを使用して、横軸の範囲選択と間隔選択を制御できます。対応する PLOT ステートメントのオプション VAXIS=により縦軸の目盛の値を制御します。

HAXIS=オプションと VAXIS=オプションの形式を次に示します。PLOT ステートメントの最初のオプションにスラッシュが先行する必要があります。

PLOT *vertical*horizontal* / HAXIS=*tick-value-list*;

PLOT *vertical*horizontal* / VAXIS=*tick-value-list*;

tick-value-list は、目盛に割り当てるすべての値のリストです。

たとえば、目盛を 1960 年から 2010 年までの間 5 年ごとに指定するには、次のオプションを使用します。

```
haxis=1960 1965 1970 1975 1980 1985 1990 1995 2000 2005 2010
```

または、この目盛リストを次のように省略形にできます。

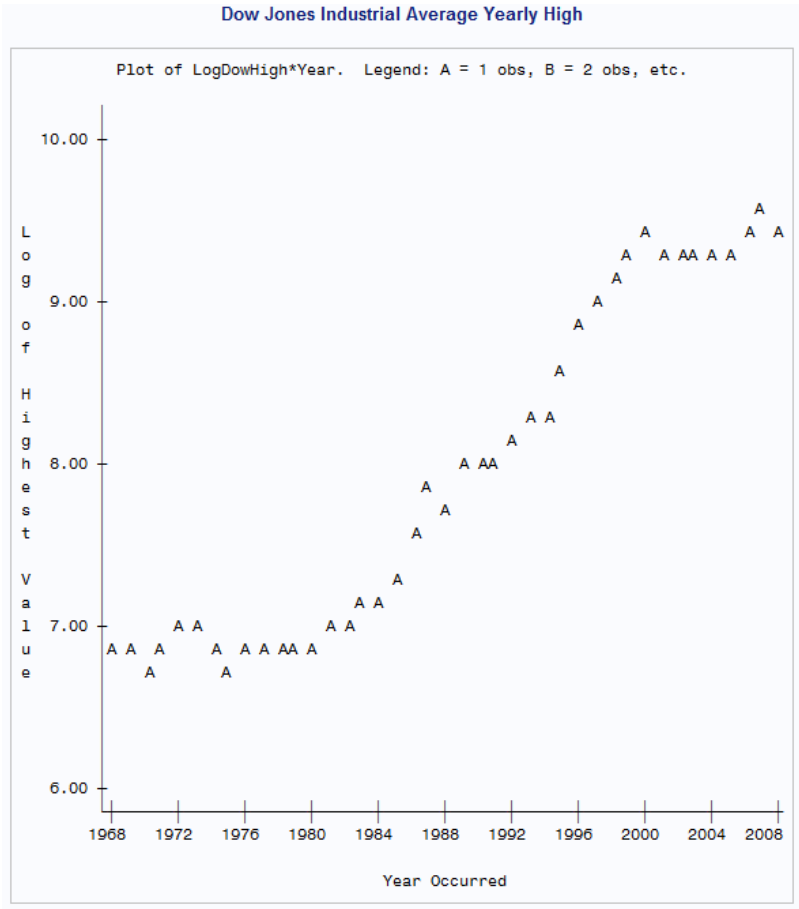
```
haxis=1960 to 2010 by 5
```

次のプログラムは、HAXIS=オプションを使用して横軸の目盛値を指定します。

```
proc plot data=highlow;
   plot LogDowHigh*Year / haxis=1968 to 2008 by 4;
   label LogDowHigh='Log of Highest Value'
         Year='Year Occurred';
   title 'Dow Jones Industrial Average Yearly High';
run;
```

プロット出力を次に示します。

図 30.5 *横軸の範囲と間隔の指定*



横軸の範囲は 1968 年から 2008 年までで、目盛は 4 年間隔に配置されました。

プロット記号の指定

デフォルトでは、PROC PLOT で、1 つのオブザベーションを示すためのプロット記号として文字 A、2 つのオブザベーションが一致する場合はプロット記号として文字 B、3 つが一致する場合には文字 C が使用されます(以下同様に続く)。文字 Z は 26 以上の一致するオブザベーションを表します。

同じ軸のペア上に 2 セットのデータをプロットする場合は、次の形式の PLOT ステートメントを使用して独自のプロット記号を指定できます。

PLOT *vertical*horizontal='character'*;

character は、プロット上の各ポイントを示すプロット記号です。PROC PLOT はこの文字を使用して 1 つ以上のオブザベーションからの値を表します。

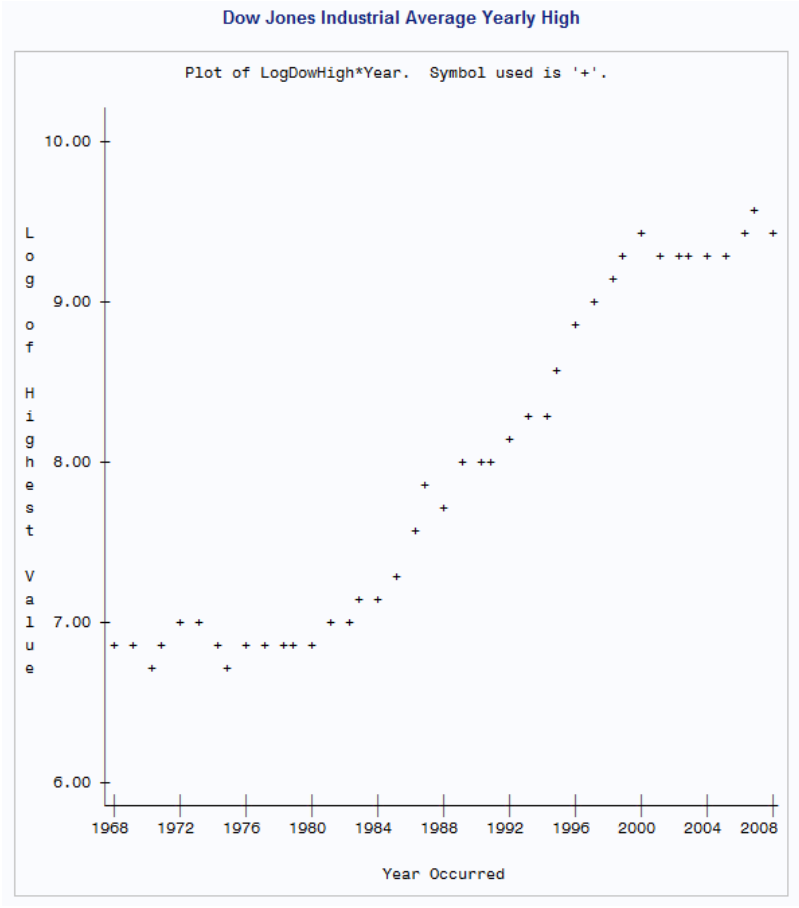
次のプログラムは、プロットのプロット記号としてプラス符号(+)を使用します。

```
proc plot data=highlow;
   plot LogDowHigh*Year='+' / haxis=1968 to 2008 by 4;
   label LogDowHigh='Log of Highest Value'
         Year='Year Occurred';
   title 'Dow Jones Industrial Average Yearly High';
run;
```

プロット記号は、一重引用符または二重引用符で囲む必要があります。

プロット出力を次に示します。

図 30.6 *プロット記号の指定*



注: プロット記号が指定される場合、PROC PLOT では、一致するオブザベーションの数にかかわらず、プロット上のすべてのポイントにその記号を使用します。オブザベーションが一致する場合、いくつのオブザベーションが非表示になっているかを示すメッセージが、プロットの下部に表示されます。

凡例の削除

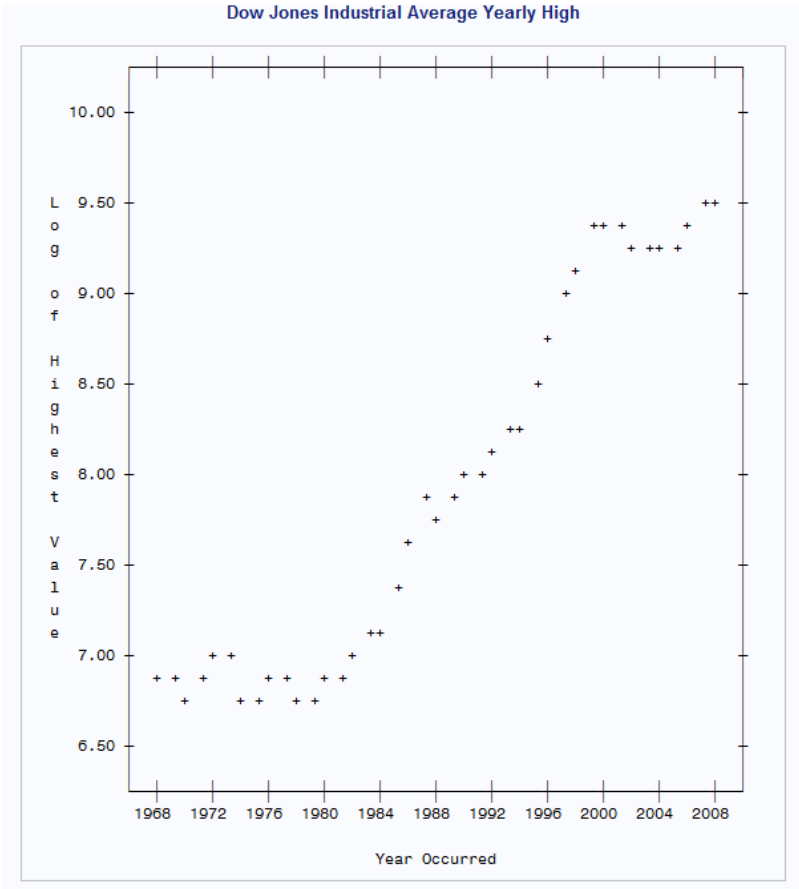
プロットにいくつか単純な変更を行うことで、外観を向上できます。左側と下だけでなく、プロット全体の周りにフレームを引くことができます。これにより、プロット記号が表すプロットの左側の値を特定しやすくなります。また、ラベルがプロットの変数を明確に表す場合や、プロット記号と変数との間の関連付けが明らかな場合、凡例を非表示にできます。

次のプログラムは、NOLEGEND オプションを使用して凡例を非表示にし、BOX オプションを使用して全プロットを囲みます。

```
proc plot data=highlow nolegend;
   plot LogDowHigh*Year='+' / haxis=1968 to 2008 by 4
                              box;
   label LogDowHigh='Log of Highest Value'
         Year='Year Occurred';
   title 'Dow Jones Industrial Average Yearly High';
run;
```

プロット出力を次に示します。

図 30.7 *凡例の削除*



複数の組み合わせの変数のプロット

複数のプロットのそれぞれ異なるページへの作成

複数のプロットを作成して異なるセットのメジャーのトレンドを比較できます。同じ SAS データセットから複数のプロットを要求するには、単純に PLOT ステートメントに追加の変数のセットを指定します。次にステーメントの形式を示します。

PLOT *vertical*-1**horizontal*-1 *vertical*-2**horizontal*-2;

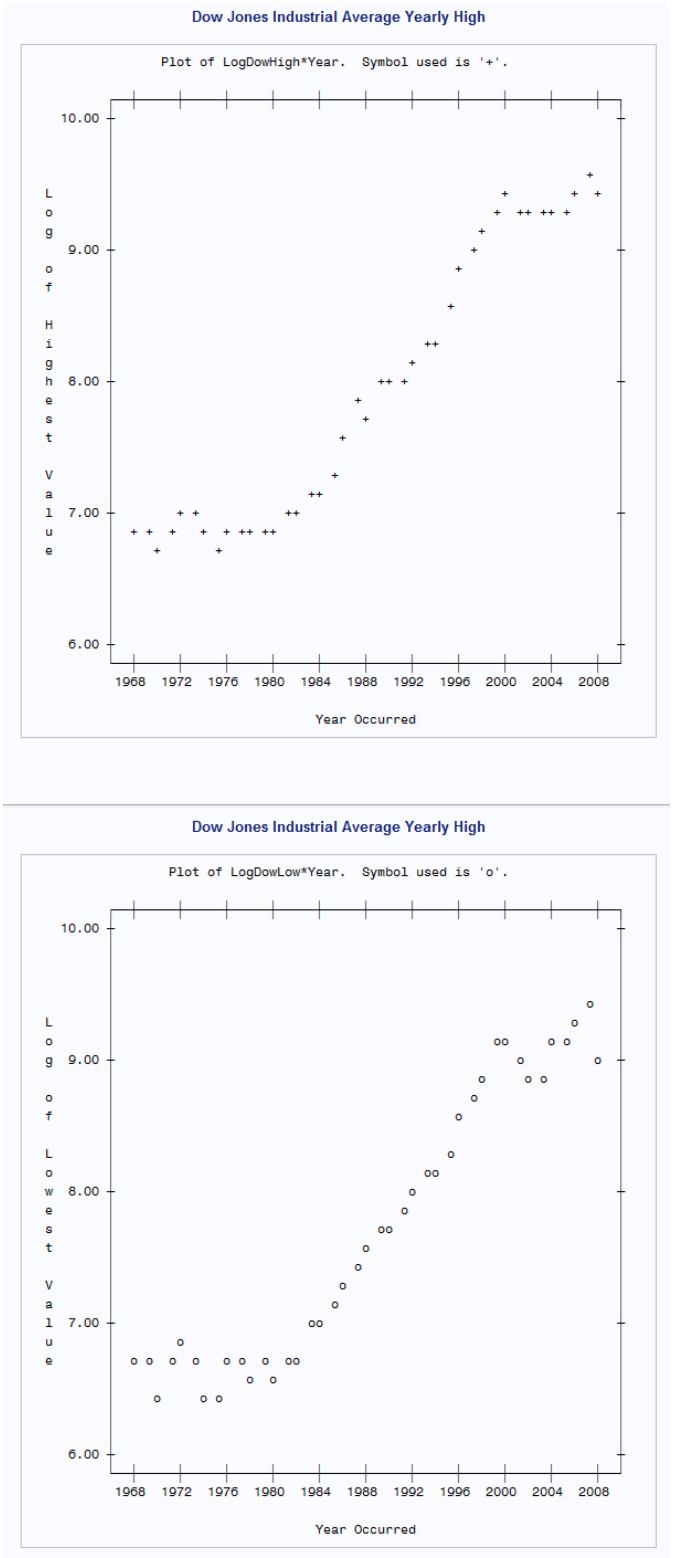
PLOT ステートメントに記述するすべてのオプションは、ステートメントによって生成されるすべてのプロットに対して適用されます。

次のプログラムは、PLOT ステートメントを使用して、1968 年から 2008 年までのダウジョーンズ工業株価平均の最高値と最安値の個別のプロットを生成します。

```
proc plot data=highlow;
   plot LogDowHigh*Year='+' LogDowLow*Year='o'
                           / haxis=1968 to 2008 by 4 box;
   label LogDowHigh='Log of Highest Value'
         LogDowLow='Log of Lowest Value'
         Year='Year Occurred';
   title 'Dow Jones Industrial Average Yearly High';
run;
```

プロット出力を次に示します。

図 30.8 複数のプロットのそれぞれ異なるページへの作成



プロットが異なる縦軸を使用して、個別のページに表示されます。ダウジョーンズ工業株価平均の高値と安値が、それぞれ異なるプロット記号によって表されます。

複数のプロットの同一ページへの作成

プロットを同じページに表示すると、異なるセットのメジャーのトレンドをより容易に比較できます。PROC PLOT には、複数のプロットを同一ページに表示する次の 2 つのオプションがあります。

- VPERCENT=オプション
- HPERCENT=オプション

これらのオプションを、PROC PLOT ステートメントに次の形式のいずれかを使用して指定します。

PROC PLOT <DATA=*SAS-data-set*> VPERCENT=*number*;

PROC PLOT <DATA=*SAS-data-set*> HPERCENT=*number*;

number は、各プロットに指定される縦または横のスペースのパーセントです。これらのオプションはエイリアス VPCT=と HPCT=で置き換えることができます。

2 つのプロットを 図 30.9 (532 ページ)のように 1 ページの上下に収めるには、VPERCENT=50 を使用します。3 つのプロットを収めるには、VPERCENT=33 を使用します(以下同様に続く)。2 つのプロットを 1 ページの左右に収めるには、HPERCENT=50 を使用します。 図 30.10 (533 ページ)のように 3 つのプロットを収めるには、HPERCENT=33 を使用します(以下同様に続く)。図 30.11 (533 ページ)では、これらのオプションの両方を同じ PLOT ステートメントで組み合わせ、プロットのマトリックスを作成しています。VPERCENT=オプションと HPERCENT=オプションは PROC PLOT ステートメントに含まれるため、PROC PLOT ステップで作成されるすべてのプロットに影響します。

次の例は、プロットの位置を示しています。

図 30.9 *VPERCENT=50 で作成されたプロット*

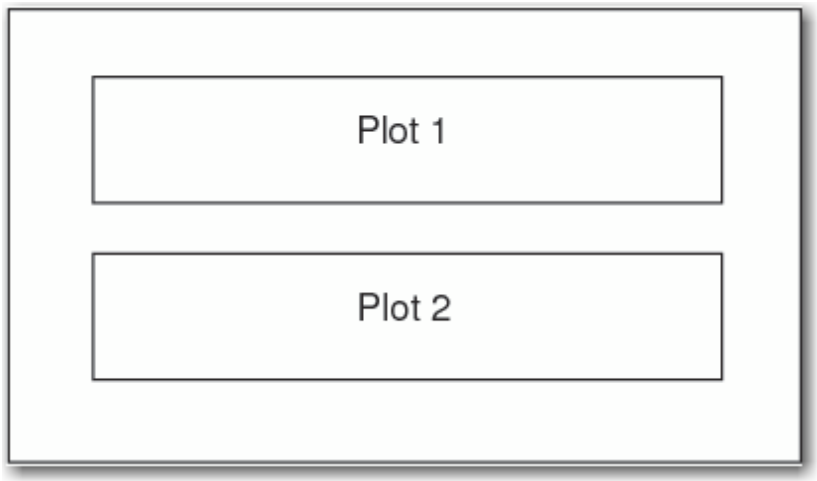


図 30.10 HPERCENT=33 で作成されたプロット

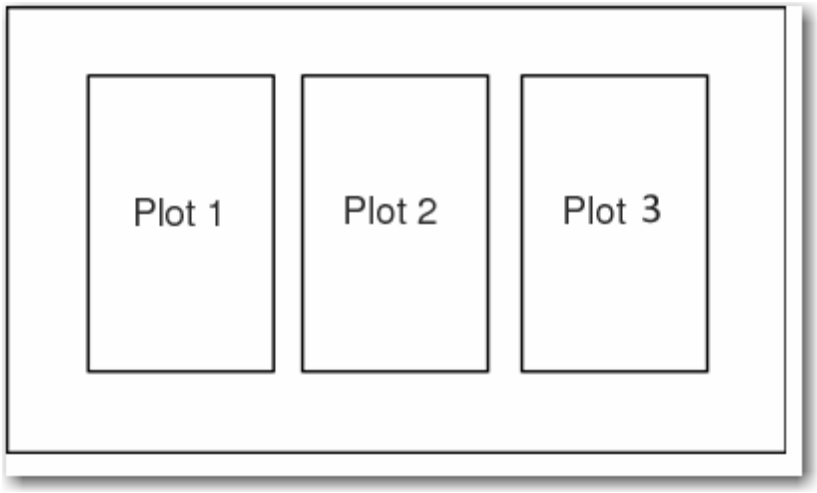
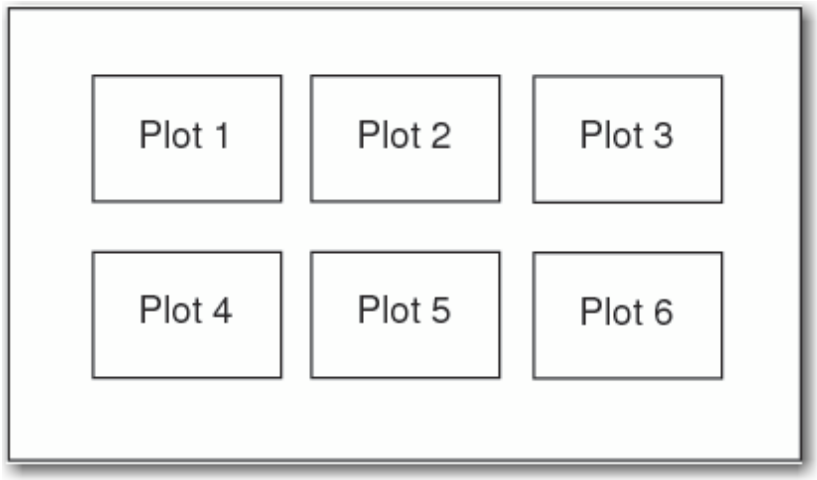


図 30.11 VPERCENT=50 と HPERCENT=33 で作成されたプロット



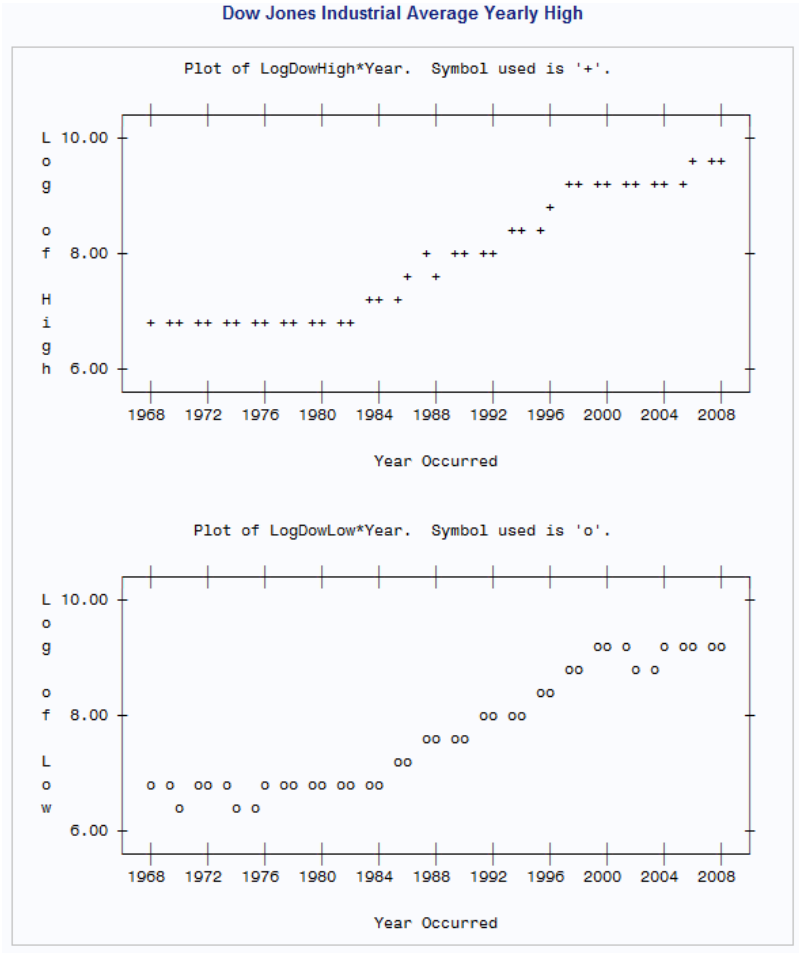
次のプログラムは、VPERCENT=オプションを使用して、ダウジョーンズの高値と安値をより容易に比較できるように 2 つのプロットを同一ページに表示します。

```
proc plot data=highlow vpercent=50;
   plot LogDowHigh*Year='+' LogDowLow*Year='o'
                          / haxis=1968 to 2008 by 4 box;
   label LogDowHigh='Log of High'
         LogDowLow='Log of Low'
         Year='Year Occurred';
   title 'Dow Jones Industrial Average Yearly High';
run;
```

出力では、PROC PLOT によってページの縦のスペースの 50%がそれぞれのプロットの表示に使用されます。

プロット出力を次に示します。

図 30.12 複数のプロットの同一ページへの作成



2 つのプロットが同一ページの上下に表示されます。

複数の組み合わせの変数の同じ軸へのプロット

複数のセットのメジャーのトレンドを比較する最も簡単な方法は、PLOT ステートメントの OVERLAY オプションを使用して軸の 1 セット上にプロットを重ね合わせることです。1 番目のプロットの変数名、または存在する場合は変数ラベルが、軸ラベルになります。HAXIS=オプションまたは VAXIS=オプションを使用しない場合、PROC PLOT によって自動的にすべての変数が最適に収まるように軸が拡大縮小されます。

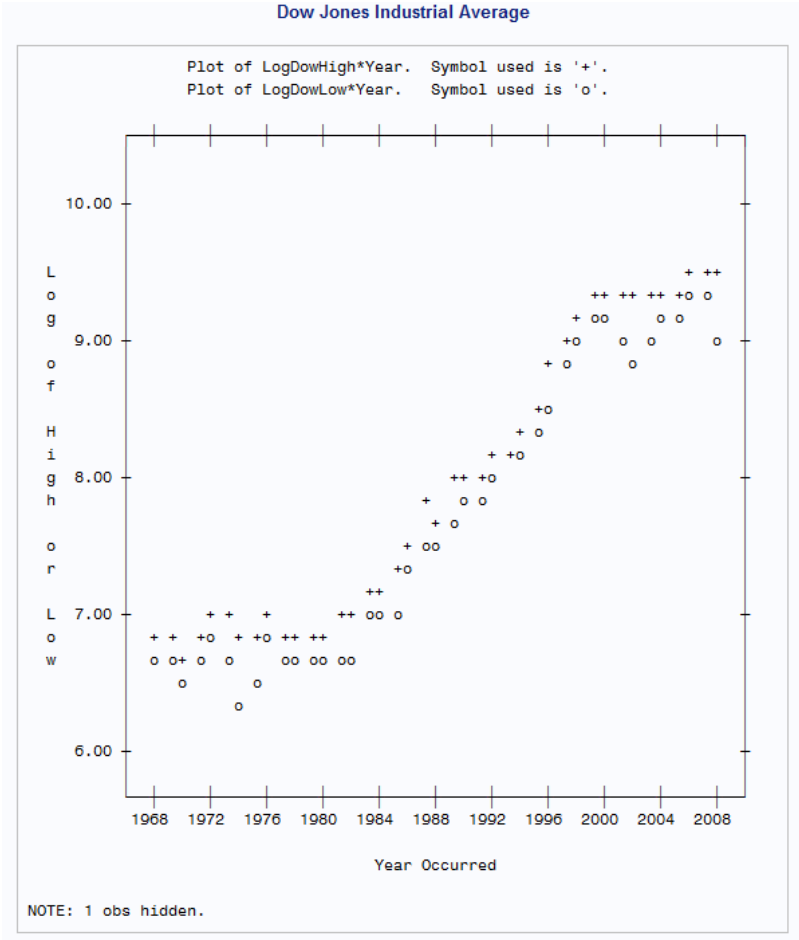
次のプログラムは、OVERLAY オプションを使用して、ダウジョーンズ工業株価平均の高値と安値を同じ軸のペア上にプロットします。

```
proc plot data=highlow;
   plot LogDowHigh*Year='+' LogDowLow*Year='o'
                          / haxis=1968 to 2008 by 4
                            overlay box;
   label LogDowHigh='Log of High or Low'
         Year='Year Occurred';
   title 'Dow Jones Industrial Average';
run;
```

PROC PLOT では、変数 LogDowHigh のみを使用して縦軸がラベル付けされるため、この変数に新しいラベルが指定されます。

プロット出力を次に示します。

図 30.13 *2 つのプロットの重ね合わせ*



1968 年から 2008 年にかけてのダウジョーンズの高値と安値の線形トレンドを簡単に見てとることができます。

注: SAS システムオプション OVP が有効でオーバープリントができる場合、プロットは重ね合わせられます。そうでない場合、NOOVP が有効であれば、PROC PLOT は 1 番目のプロットからのプロット記号を使用して複数のプロットに表示されるポイントを表します。そのような場合、非表示のオブザベーションの数を示すメッセージが出力に含まれます。

要約

PROC PLOT ステートメント

PROC PLOT <DATA=*SAS-data-set*> <*options*>;

LABEL *variable='label'*;

PLOT *request-list* </*option(s)*>;

TITLE<*n*> <*'title'*>;

PROC PLOT <DATA=*SAS-data-set*> <*option(s)*> ;

PLOT プロシジャを開始します。PROC PLOT ステートメントでは、次の *option(s)*を指定できます。

DATA=*SAS-data-set*

PROC PLOT で使用される SAS データセットの名前を指定します。DATA=が省略されると、PROC PLOT は最後に作成されたデータセットを使用します。

HPERCENT=*percent(s)*

各プロットに使用する、使用可能な横のスペースのパーセントを 1 以上指定します。HPERCENT=を使用すると複数のプロットを 1 ページに表示できます。PROC PLOT は、できるだけ多数のプロットを収めようと試みます。それぞれの *percent(s)*の使用後、PROC PLOT はリストの先頭に戻ります。リスト内にゼロ(0)がある場合、PROC PLOT は、次のプロットが同じページに収まる場合でも強制的に新しいページに移動されます。

NOLEGEND

デフォルトの凡例を非表示にします。凡例にはプロットされている変数の名前とプロットで使用されているプロット記号がリストにされます。

VPERCENT=*percent(s)*

各プロットに使用する、使用可能な縦のスペースのパーセントを 1 以上指定します。100 より大きいパーセントを使用すると、PROC PLOT はプロットのセクションの一部を連続したページに出力します。

LABEL *variable='label'*;

軸のラベルを使用するように指定します。*Variable* にラベル付けする変数名を指定し、ブランクを含めて最大 256 文字の文字列を *label* に指定します。*label* は、一重引用符または二重引用符で囲む必要があります。

PLOT *request-list* </*option(s)*>;

PLOT ステートメントの *request-list* の個別のプロットを要求できます。リストの各要素の形式を次に示します。

*vertical*horizontal*<='symbol'>*

vertical と *horizontal* は軸上に表示される変数の名前で、*symbol* はプロットのすべてのポイントに使用される文字です。

1 つの PROC PLOT ステップに任意の数の PLOT ステートメントを要求できます。単一の PLOT ステートメントに関するオプションのリストを次に示します。

BOX

左側と下だけでなく、プロット全体の周りにボックスを描きます。

HAXIS=<*tick-value-list*>

横軸の目盛値を指定します。*tick-value-list* は、目盛の値のリストで構成されます。

OVERLAY
PLOT ステートメントで要求されたプロットをすべて 1 セットの軸上に重ね合わせます。1 番目のプロットの変数名、または存在する場合は変数ラベルが、軸のラベルに使用されます。HAXIS=オプションまたは VAXIS=オプションを使用しない場合、PROC PLOT によって自動的にすべての変数が最適に収まるように軸が拡大縮小されます。

VAXIS=<*tick-value-list*>
縦軸の目盛値を指定します。*tick-value-list* は、目盛の値のリストで構成されます。

TITLE<*n*> <'*title*'>;
タイトルを指定します。引数 *n* は、ワード TITLE の直後にブランクを空けずに続ける 1-10 の数値で、TITLE のレベルを指定します。各 *title* のテキストは一重引用符または二重引用符で囲む必要があります。指定可能なタイトルの最大長は、ご使用の動作環境や LINESIZE=システムオプションの値によって異なります。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

詳細情報

PROC CHART と PROC UNIVARIATE
グラフィック表現を用意するにあたり、一部のデータはチャートに適しており、一部のデータはプロットにより適していることに注意してください。さまざまなチャートの作成方法の詳細については、31 章, “変数を要約するチャートを作成する” (539 ページ)を参照してください。

PROC PLOT
PROC PLOT を使用して、等高線プロットを作成したり、プロットの特定の値のところで参照線を引いたり、プロットの罫線を変更したりすることもできます。詳細なドキュメントについては、*Base SAS Procedures Guide* を参照してください。

SAS 関数
SAS には、算術関数、三角関数、双曲線関数、確率分布、基本統計量、乱数の生成を含むさまざまな数値関数が用意されています。詳細なドキュメントについては、*SAS Functions and CALL Routines: Reference* を参照してください。

31 章

変数を要約するチャートを作成する

変数を要約するチャートの作成について **540**
目的 540
前提条件 540

チャート作成ツールについて **540**

例で使用される入力ファイルおよび入力 SAS データセット **541**

CHART プロシジャを使用した度数グラフの作成 **543**
度数グラフの種類 543
縦棒グラフの作成 544
横棒グラフの作成 546
ブロックチャートの作成 548
円グラフの作成 549

度数グラフのカスタマイズ **551**
範囲の数の変更 551
数値変数の中間点の指定 552
グラフの中間点の数の指定 554
すべての値のチャートの作成 555
文字変数の度数グラフの作成 557
平均値のチャート作成 560
3 次元グラフの作成 561

高解像度ヒストグラムの作成 **562**
HISTOGRAM ステートメントの使用方法について 562
SAS/GRAPH を使用したヒストグラムの作成方法について 563
単純なヒストグラムの作成 563
ヒストグラムの軸の変更 565
ヒストグラムでの要約統計量の表示 570
比較ヒストグラムの作成 572

要約 **574**
PROC CHART ステートメント 574
PROC UNIVARIATE ステートメント 576
GOPTIONS ステートメント 578
FORMAT ステートメント 578

詳細情報 **578**

変数を要約するチャートの作成について

目的

プロットなどのチャートは、データをグラフィカルに要約する手法を提供します。チャートを使用して、単一の変数または複数の変数の値を表示できます。棒グラフを使用して、変数の値の分布をグラフィカルに検証することもできます。

このセクションでは、次に示す項目を作成する方法を学習します。

- 縦棒グラフ
- 横棒グラフ
- 円グラフ
- ブロックチャート
- 高解像度ヒストグラムと比較ヒストグラム

例の範囲は、単純な度数棒グラフから、変数をグループ化して要約統計量を含む複雑な棒グラフまで複雑です。

前提条件

このセクションの例を理解するために、次の機能と概念を理解していることを確認してください。

- LABEL ステートメント
- TITLE ステートメント
- SAS システムオプション
- SAS 出力形式の作成と割り当て

チャート作成ツールについて

Base SAS は、チャートを生成する 2 つのプロシジャを提供します。

- PROC CHART
- PROC UNIVARIATE

PROC CHART は、文字変数または数値変数のさまざまなチャートを生成します。チャートには、縦棒グラフと横棒グラフ、ブロックチャート、円グラフ、およびスターチャートがあります。これらの種類のチャートは、これらの値と関連付けられている変数の値または統計量をグラフィカルに表示します。PROC UNIVARIATE は、連続尺度の数値変数のヒストグラムを生成し、データの分布を視覚化できます。

PROC CHART は、データを迅速に視覚化する、便利なツールです。ただし、サイトで SAS/GRAPH のライセンスを所有している場合、PROC GCHART[1] を使用して、高解像度で出版品質である、色と各種フォントを含む棒グラフを生成できます。PROC UNIVARIATE を使用して、要約統計量とテーブルをグラフィカル表示に直接追加する

1 PROC GCHART と PROC CHART は、同一のチャートを生成します。

ことにより、ヒストグラムをカスタマイズできます。PROC UNIVARIATE を使用して、当てはめ密度曲線またはカーネル密度推定値とヒストグラムとを重ね合わせて、データの分布を評価できます。

例で使用される入力ファイルおよび入力 SAS データセット

このセクションの例では、1 つの入力ファイルと 1 つの SAS データセットを使用します。入力データの完全なリストについては、“YEAR_SALES データセット” (794 ページ)を参照してください。入力ファイルには、化学の入門コースの登録者および試験の成績が含まれています。コースに登録した 50 人の生徒は、いくつかの講義および週に一日ディスカッションセクションに参加します。入力ファイルの構造は次のとおりです。

```
Abdallah       F Mon  46 Anderson       M Wed  75
Aziz           F Wed  67 Bayer          M Wed  77
Bhatt          M Fri  79 Blair          F Fri  70
Bledsoe        F Mon  63 Boone          M Wed  58
Burke          F Mon  63 Chung          M Wed  85
Cohen          F Fri  89 Drew           F Mon  49
Dubos          M Mon  41 Elliott        F Wed  85
…more data lines…
Simonson       M Wed  62 Smith N        M Wed  71
Smith R        M Mon  79 Sullivan       M Fri  77
Swift          M Wed  63 Wolfson        F Fri  79
Wong           F Fri  89 Zabriski       M Fri  89
```

入力ファイルには、次に示す値が左から右へと含まれています。

- 生徒の姓(および、必要に応じて名のイニシャル)
- 生徒の性別(F または M)
- 生徒のディスカッションセクションの曜日(Mon、Wed、Fri)
- 生徒の最初の試験の成績

次のプログラムは、このセクションで使用する GRADES データセットを作成します。この例では、最初の 15 オブザベーションを示します。

```
options pagesize=60 linesize=80 pageno=1 nodate;

data grades;
   infile 'your-input-file';
   input Name & $14. Gender : $2. Section : $3. ExamGrade1 @@;
run;

proc print data=grades;
   title 'Introductory Chemistry Exam Scores';
run;

options obs=15;
data grades;
   input Name &$14. Gender :$2. Section :$3. ExamGrade1 @@;
   datalines;
Abdallah       F Mon  46 Anderson       M Wed  75
Aziz           F Wed  67 Bayer          M Wed  77
Bhatt          M Fri  79 Blair          F Fri  70
```

```
Bledsoe        F Mon  63 Boone          M Wed  58
Burke          F Mon  63 Chung          M Wed  85
Cohen          F Fri  89 Drew           F Mon  49
Dubos L        M Mon  41 Elliott        F Wed  85
Farmer         F Wed  58 Franklin       F Wed  59
Freeman        F Mon  79 Friedman       M Mon  58
Gabriel        M Fri  75 Garcia         M Mon  79
Harding        M Mon  49 Hazelton       M Mon  55
Hinton         M Fri  85 Hung           F Fri  98
Jacob          F Wed  64 Janeway        F Wed  51
Jones          F Mon  39 Jorgensen      M Mon  63
Judson         F Fri  89 Kuhn           F Mon  89
LeBlanc        F Fri  70 Lee            M Fri  48
Litowski       M Fri  85 Malloy         M Wed  79
Meyer          F Fri  85 Nichols        M Mon  58
Oliver         F Mon  41 Park           F Mon  77
Patel          M Wed  73 Randleman      F Wed  46
Robinson       M Fri  64 Shien          M Wed  55
Simonson       M Wed  62 Smith N        M Wed  71
Smith R        M Mon  79 Sullivan       M Fri  77
Swift          M Wed  63 Wolfson        F Fri  79
Wong           F Fri  89 Zabriski       M Fri  89
;
```

注: このセクションの大部分の出力は、OPTIONS ステートメントを使用し、PAGESIZE=40 と LINESIZE=80 を指定します。その他の例では、チャートを読みやすくするため、異なる行サイズやページサイズを OPTIONS ステートメントで使用します。PAGESIZE=オプションと LINESIZE=オプションが設定されると、オプションを別の OPTIONS ステートメントで再設定するか、SAS セッションを終了しない限り、有効のままです。

次の出力では、最初の 15 オブザベーションが表示されています。

図 31.1 化学入門試験スコア

Introductory Chemistry Exam Scores

Obs	Name	Gender	Section	ExamGrade1
1	Abdallah	F	Mon	46
2	Anderson	M	Wed	75
3	Aziz	F	Wed	67
4	Bayer	M	Wed	77
5	Bhatt	M	Fri	79
6	Blair	F	Fri	70
7	Bledsoe	F	Mon	63
8	Boone	M	Wed	58
9	Burke	F	Mon	63
10	Chung	M	Wed	85
11	Cohen	F	Fri	89
12	Drew	F	Mon	49
13	Dubos	M	Mon	41
14	Elliott	F	Wed	85
15	Farmer	F	Wed	58

このデータセットを使用して棒グラフを作成し、次を行えます。

- 成績の分布を評価します。
- それぞれの生徒の成績(A-F)を決定します。
- 各セクションの生徒数を比較します。
- 各セクションの男子生徒数と女子生徒数を比較します。
- 他のセクションの生徒とパフォーマンスを比較します。

CHART プロシジャを使用した度数グラフの作成

度数グラフの種類

デフォルトでは、PROC CHART は、グラフのそれぞれの棒、セクションまたはブロックが値の範囲を表す度数グラフを作成します。デフォルトでは、PROC CHART はチャート変数の値に基づいて範囲を選択します。それぞれの範囲の中央は、中間点です。中

間点は、必ずしもチャート変数の実際の値に対応しません。それぞれの棒、ブロック、セクションのサイズは、その範囲に収まるオブザベーションの数を表します。

PROC CHART は、次の種類のグラフを作成します。

縦棒グラフおよび横棒グラフ
: 棒の長さまたは高さでデータの大きさを表示します。

ブロックチャート
: さまざまなサイズのブロックでデータの相対的な大きさを表示します。

円グラフ
: データを、各セクションの全円に対する寄与率を表す、円のくさび形のセクションとして表示します。

スターチャート
: データを、(車輪のスポークのように)中心点から放射状に伸びる棒として表示します。

各種チャートの形状により、データの特定の側面が強調されます。データの性質および強調する側面によって、選択するチャートは異なります。

縦棒グラフの作成

縦棒グラフについて

縦棒グラフは、個別の範囲を強調します。横軸(中間点の軸)は、変数の値を範囲に分けて表示します。デフォルトでは、縦軸は、指定した範囲の値の度数を示します。それぞれの棒の高さによって、含まれているオブザベーションが多い範囲、および含まれているオブザベーションが少ない範囲を素早く判別できます。

PROC CHART ステップの VBAR ステートメントは、縦棒グラフを生成します。オプションなしで VBAR ステートメントを使用すると、PROC CHART は自動的に次を実行します。

- 縦軸の拡大縮小
- 棒の幅の決定
- 棒と棒の間のスペース調整
- 軸のラベル付け

連続尺度の数値データの場合、PROC CHART は、チャート変数の最小値と最大値から棒の数と各棒の中間点を決定します。文字変数、または離散数値変数の場合、PROC CHART は、チャート変数のそれぞれの値の棒を作成します。ただし、オプションを使用して PROC CHART での軸の決定方法を変更できます。

注: 行ごとの文字数(LINESIZE=)が縦棒を表示するのに十分でない場合、PROC CHART は自動的に横棒グラフを生成します。

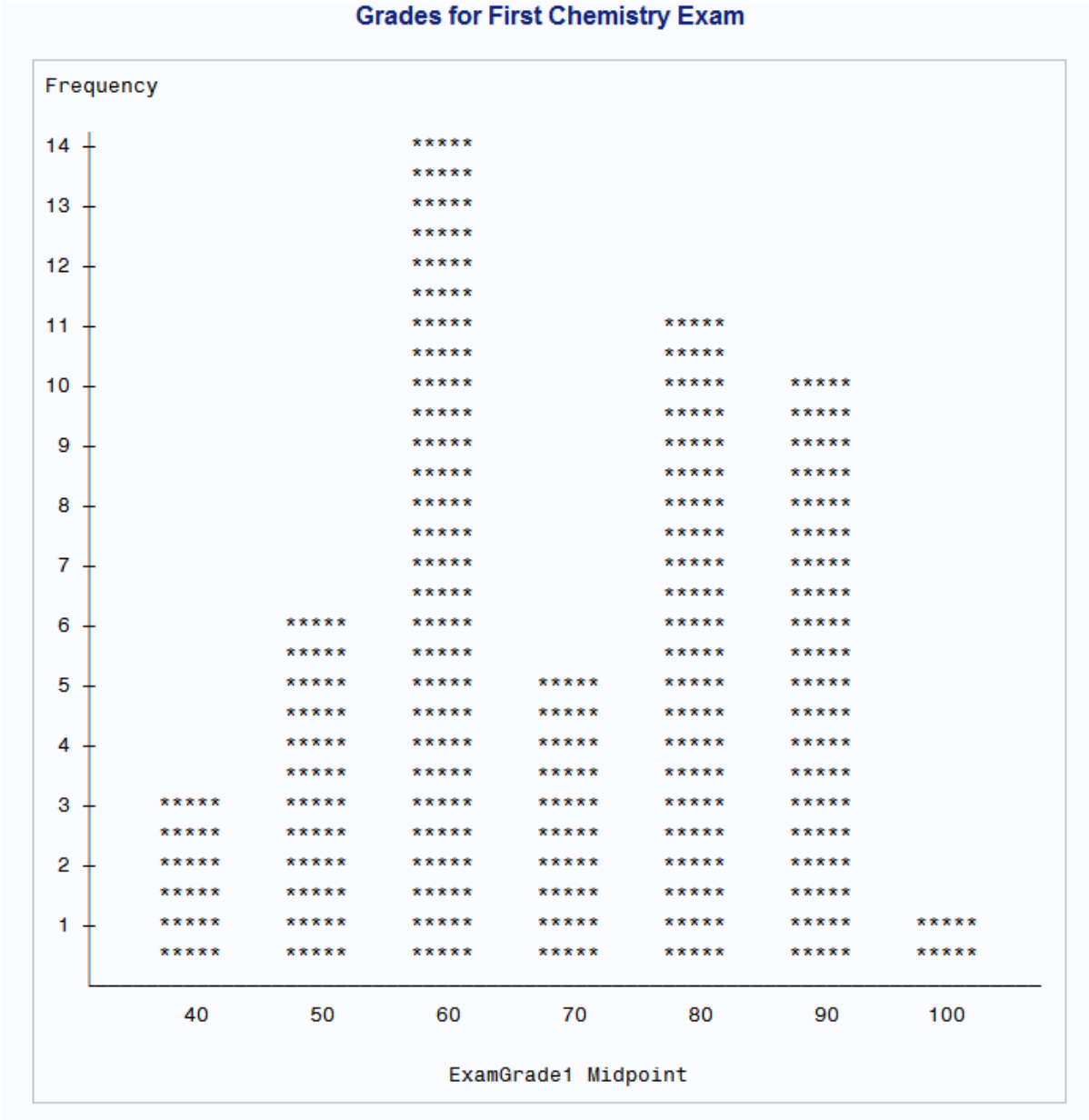
プログラム

次のプログラムは、VBAR ステートメントを使用して数値変数 ExamGrade1 の度数の縦棒グラフを作成します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   vbar ExamGrade1;
   title 'Grades for First Chemistry Exam';
run;
```

棒グラフの出力を次に示します。

図 31.2 縦棒グラフを使用した度数の表示



前述のグラフの中間点の軸の範囲は、40-100 で、10 の間隔で増分します。次の表に、それぞれの棒の値と度数を示します。

表 31.1 値と度数

範囲	中間点	度数
35-44	40	3
45-54	50	6

範囲	中間点	度数
55-64	60	14
65-74	70	5
75-84	80	11
85-94	90	10
95-104	10	1

注: PROC CHART は、範囲のサイズを選択し、それらの中間点の位置を数値変数のすべての値に基づいて選択するため、最も高い範囲と最も低い範囲はデータの値の範囲を外れて伸びる場合があります。この例では、最も低い成績は 39 で、最低範囲は 35-44 です。同様に、最も高い成績は 98 で、最高範囲は 95-104 です。

横棒グラフの作成

横棒グラフについて

横棒グラフは、基本的に縦棒グラフと同様の性質を持っています。いずれのグラフも、個別の範囲を強調します。ただし、横棒グラフでは棒が回転されて、横軸が度数を、縦軸がチャート変数の値を示します。横棒の右に、PROC CHART によりデータを要約する統計量のテーブルが表示されます。

PROC CHART ステップの HBAR ステートメントは、横棒グラフを生成します。デフォルトでは、統計量のテーブルには度数、累積度数、パーセント、累積パーセントが含まれます。特定の統計量を要求し、それらの統計量と度数のみがテーブルに含まれるようにできます。

HBAR 統計量について

デフォルトの横棒グラフは、その他の形状のグラフより使用するスペースが少なくすみます。PROC CHART は、横棒グラフの小さいサイズの利点を活かし、グラフの右に統計量を表示します。次の統計量が含まれます。

度数
: 指定範囲内のオブザベーションの数。

累積度数
: 指定範囲と指定範囲までのすべての範囲内のオブザベーションの数。最後の範囲の累積度数はデータセット内のオブザベーションの数に等しいものです。

パーセント
: 指定範囲内のオブザベーションの割合。

累積パーセント
: 指定範囲と指定範囲までのすべての範囲内のオブザベーションの割合。最後の範囲の累積パーセントは常に 100 です。

さまざまなオプションを使用して、テーブルに表示する統計量を制御できます。FREQ、CFREQ、PERCENT、CPERCENT の各オプションを使用して統計量を選択できます。統計量のテーブルを非表示にするには、NOSTAT オプションを使用します。

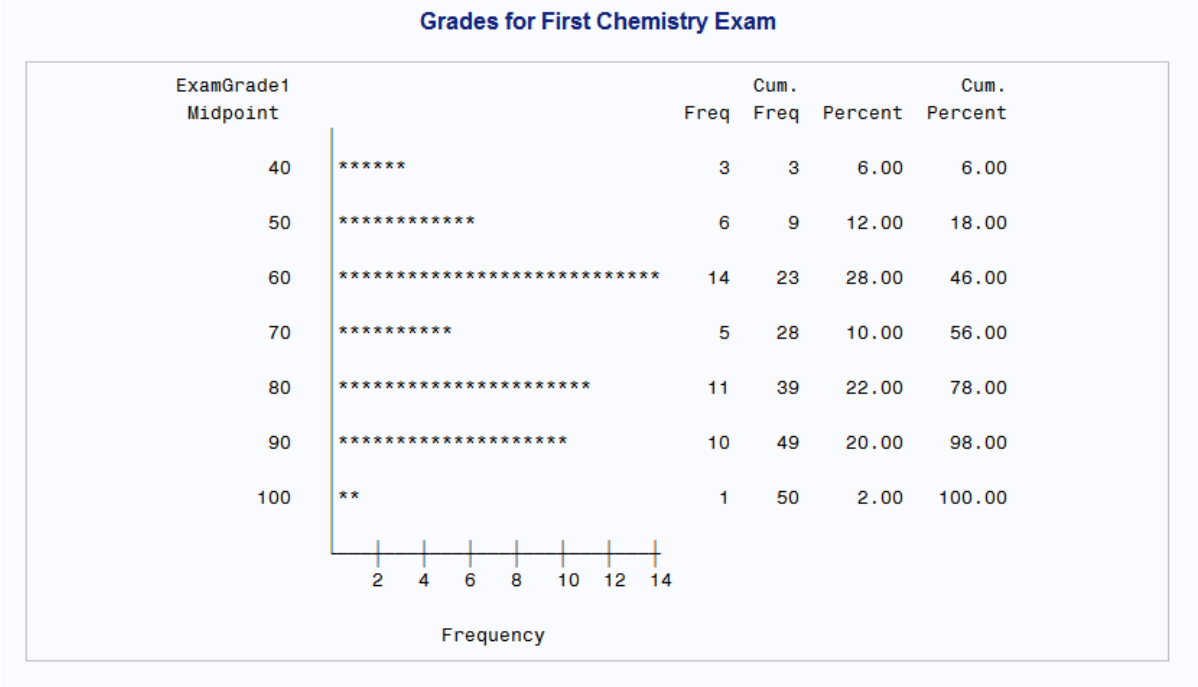
プログラム

次のプログラムは、HBAR ステートメントを使用して変数 ExamGrade1 の度数の横棒グラフを作成します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   hbar Examgrade1;
   title 'Grades for First Chemistry Exam';
run;
```

棒グラフの出力を次に示します。

図 31.3 横棒グラフを使用した度数の表示



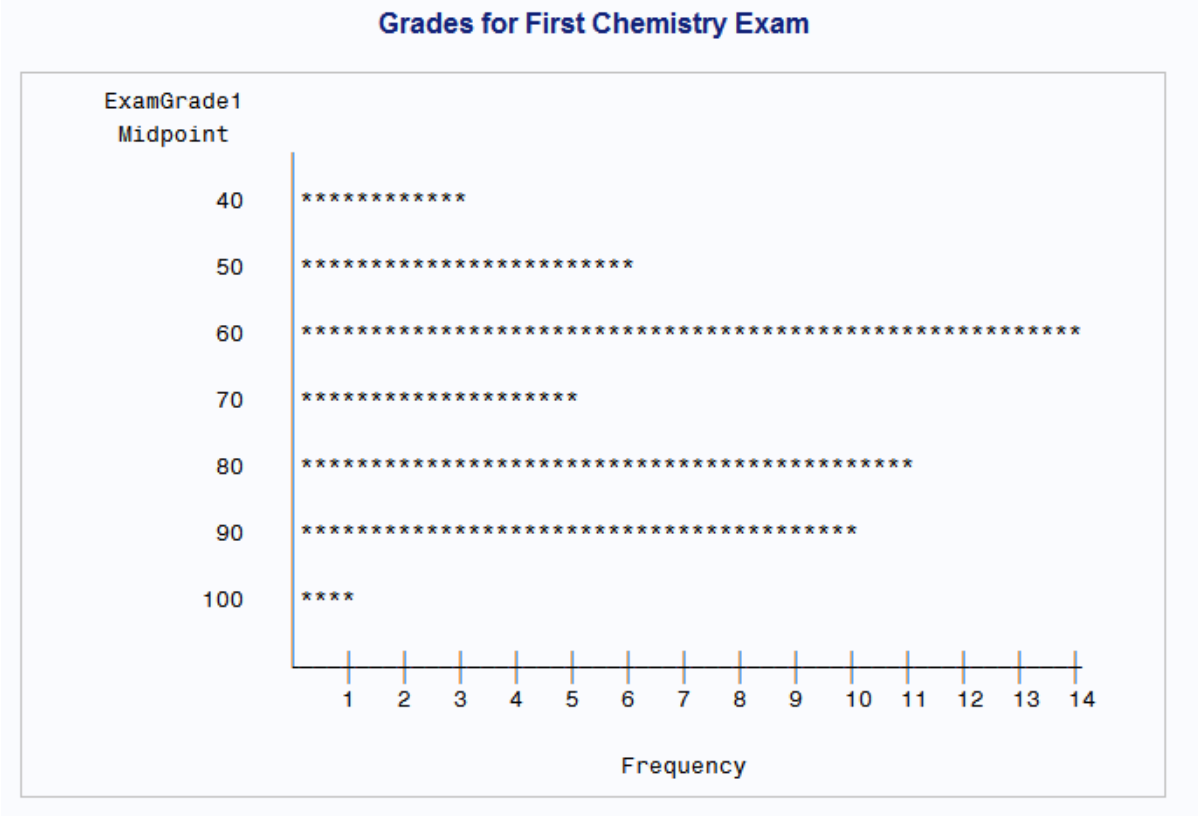
累積パーセントは試験の中央値の成績(オブザベーションの 50%ずつがその上下に存在する成績)が中間点の 70 の内に存在することを示します。

次の例は前述の横棒グラフと同じものを生成しますが、プログラムで NOSTAT オプションを使用し、統計量のテーブルを除外します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   hbar Examgrade1 / nostat;
   title 'Grades for First Chemistry Exam';
run;
```

棒グラフの出力を次に示します。

図 31.4 横棒グラフからの統計量の削除



ブロックチャートの作成

ブロックチャートについて

ブロックチャートは、さまざまな高さのブロックを使用してデータの相対的な大きさを示します。四角形の各ブロックはデータのカテゴリを表します。ブロックチャートは縦棒グラフと似ています。個別の範囲を強調するため、より複雑なデータ表現が使用されます。ただし、ブロックの最大の高さは 10 行であるため、ブロックチャートは棒グラフより精度が低くなります。

PROC CHART ステップの BLOCK ステートメントは、ブロックチャートを生成します。または、BLOCK ステートメントを使用して 3 次元の度数グラフを作成します。例については、“3 次元グラフの作成” (561 ページ)を参照してください。多数のチャート値を使用してブロックチャートを作成すると、ブロックチャートが 1 ページに収まるように SAS システムオプション LINESIZE=と PAGESIZE=の調整が必要な場合があります。

注: 行サイズまたはページサイズがすべての棒を表示するのに十分でない場合、PROC CHART は自動的に横棒グラフを生成します。

プログラム

次のプログラムは、BLOCK ステートメントを使用して数値変数 ExamGrade1 のブロック度数チャートを作成します。

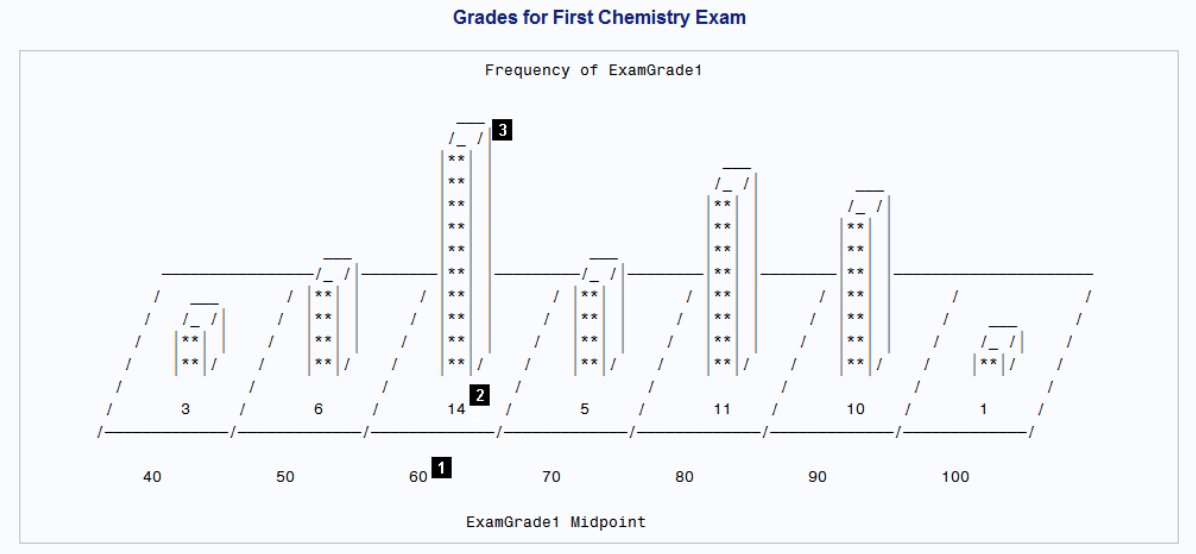
```
options linesize=120 pagesize=40 pageno=1 nodate;
```

```
proc chart data=grades;
   block Examgrade1;
   title 'Grades for First Chemistry Exam';
run;
```

OPTIONS ステートメントで行サイズを 120 に増やします。

ブロックチャートの出力を次に示します。

図 31.5 ブロックチャートを使用した度数の表示



チャートは、BLOCK ステートメントの使用による効果を示します。

1. PROC CHART は、棒グラフとブロックチャートの両方に同じ中間点を使用します。中間点はチャートの下に表示されます。
2. 各ブロックによって表されるオブザベーションの数はブロックの下に表示されます。
3. ブロックの高さは、ブロックのオブザベーションの数に比例します。

円グラフの作成

円グラフについて

円グラフは部分(値の範囲)の全体に対する寄与率を強調します。成績の分布を円グラフとしてグラフ化すると、縦棒グラフとまったく同様に、他に対する各範囲のサイズを示します。ただし、円グラフを使用すると、範囲内の成績の数を成績の総数に対して視覚的に比較することもできます。

PROC CHART ステップの PIE ステートメントは、円グラフを生成します。PROC CHART は、次の 1 つの例外を除き、縦棒グラフの棒の数を決定するのと同じ方法で円グラフのセクションの数を決定します。円のいずれかのスライスの占める印刷位置が 3 より少ない場合、PROC CHART はそれらのスライスを“Other”と呼ばれるカテゴリにグループ化します。

PROC CHART は、円グラフの境界線の周りに中間点の値を表示します。グラフの各セクションの内側には、PROC CHART により、範囲内のオブザベーションの数とその数値が表すオブザベーションの割合が表示されます。

SAS システムオプション LINESIZE=および PAGESIZE=により、円のサイズが決定されます。ご使用のプリンタが 1 インチあたり 6 行(6lpi)および 1 インチあたり 10 列の印刷をサポートしない場合、円は楕円形状の外観になります。正円の円グラフを作成するには、PROC CHART ステートメントで LPI=オプションを使用する必要があります。詳細については、*Base SAS Procedures Guide* の CHART プロシジャに関する説明を参照してください。

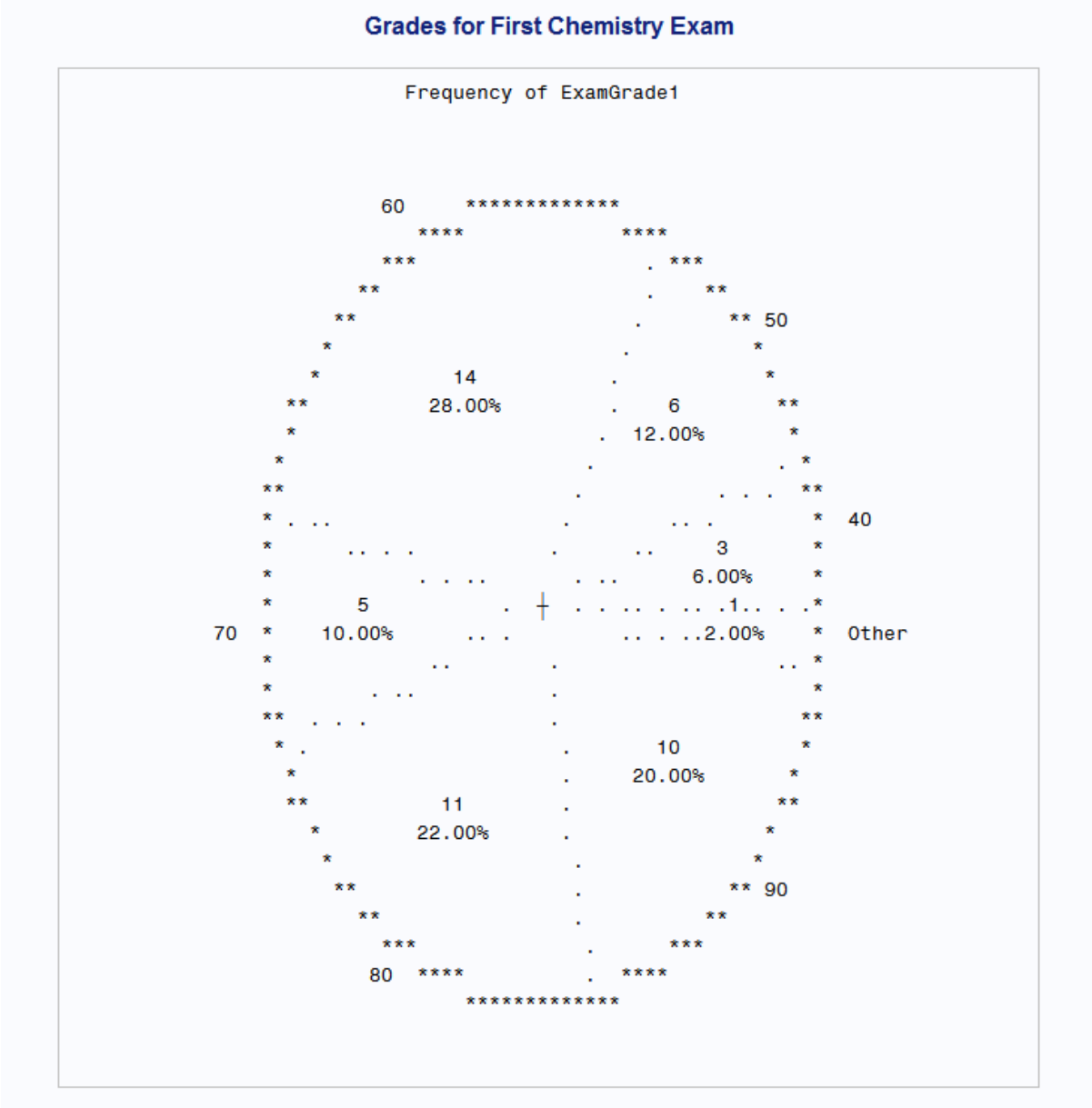
プログラム

次のプログラムは、PIE ステートメントを使用して数値変数 ExamGrade1 の度数の円グラフを作成します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   pie ExamGrade1;
   title 'Grades for First Chemistry Exam';
run;
```

円グラフの出力を次に示します。

図 31.6 円グラフを使用した度数の表示



この円グラフでは、Other セクションは、中間点が 100 である範囲内の 1 つの成績を表しています。セクションのサイズは、その範囲に該当するオブザベーションの数に対応しています。

度数グラフのカスタマイズ

範囲の数の変更

次に示す方法でグラフの外観を変更できます。

表 31.2 グラフの外観

アクション	オプション
各棒、ブロックまたはセクションが表す、値の範囲を定義する中間点を指定します。	MIDPOINTS=オプション
グラフの棒の数を指定し、PROC CHART で中間点を計算します。	LEVELS=オプション
離散数値を含む変数を指定します。PROC CHART で、重複しない値を棒で表す棒グラフを生成します。	DISCRETE オプション

注: このセクションのほとんどの例は縦棒グラフを使用します。ただし、特に記載のない場合において、PIE、BLOCK、HBAR の各ステートメントで任意のオプションを使用できます。

数値変数の中間点の指定

VBAR ステートメントの MIDPOINTS=オプションを使用して、連続尺度の数値変数の中間点を指定できます。次に、このオプションの形式を示します。

VBAR *variable* / MIDPOINTS=*midpoints-list*;

ここで midpoints-list は中間点として使用する数値のリストです。

たとえば、中間点が 55-95 である、従来の成績付け範囲を指定するには、次のオプションを使用します。

```
midpoints=55 65 75 85 95
```

または、中間点のリストを次のように省略形にできます。

```
midpoints=55 to 95 by 10
```

対応する範囲を次に示します。

```
  50 to 59
  60 to 69
  70 to 79
  80 to 89
  90 to 99
```

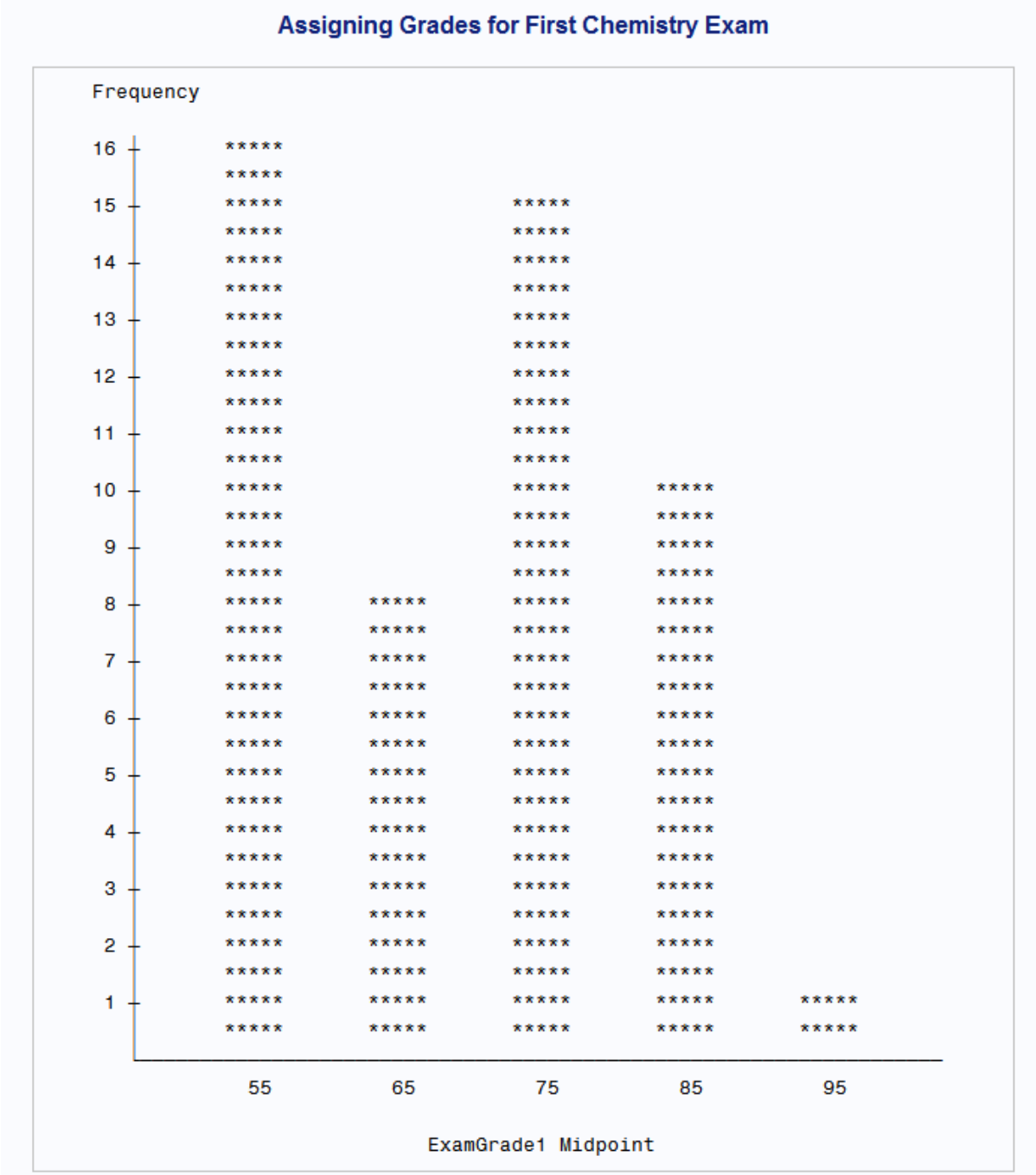
次のプログラムは、MIDPOINTS=オプションを使用して ExamGrade1 の棒グラフを作成します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   vbar Examgrade1 / midpoints=55 to 95 by 10;
   title 'Assigning Grades for First Chemistry Exam';
run;
```

MIDPOINTS=オプションは、PROC CHART で 5 つの棒を試験成績の従来の中間点を中心に中央揃えにするよう強制します。

棒グラフの出力を次に示します。

図 31.7 *縦棒グラフの中間点の指定*



成績を割り当てる従来の方法は、データが正規分布であるとみなします。ただし、棒は正規分布(釣鐘型)曲線として表示されません。成績が、これらの中間点、および従来の 60 の合格か不合格かの境界に基づいて割り当てられると、他のどの棒よりも中間点が 55 の棒により多数のオブザベーションが該当するためクラスの相当な部分が試験に不合格になります。

グラフの中間点の数の指定

LEVELS=オプションを使用して、グラフの中間点の値ではなく中間点の数を指定します。プロシジャにより中間点が選択されます。

オプションの形式を次に示します。

VBAR *variable* / LEVELS=*number-of-midpoints*;

ここで number-of-midpoints により中間点の数が指定されます。

次のプログラムは LEVELS=オプションを使用して棒が 5 つある棒グラフを作成します。[1]

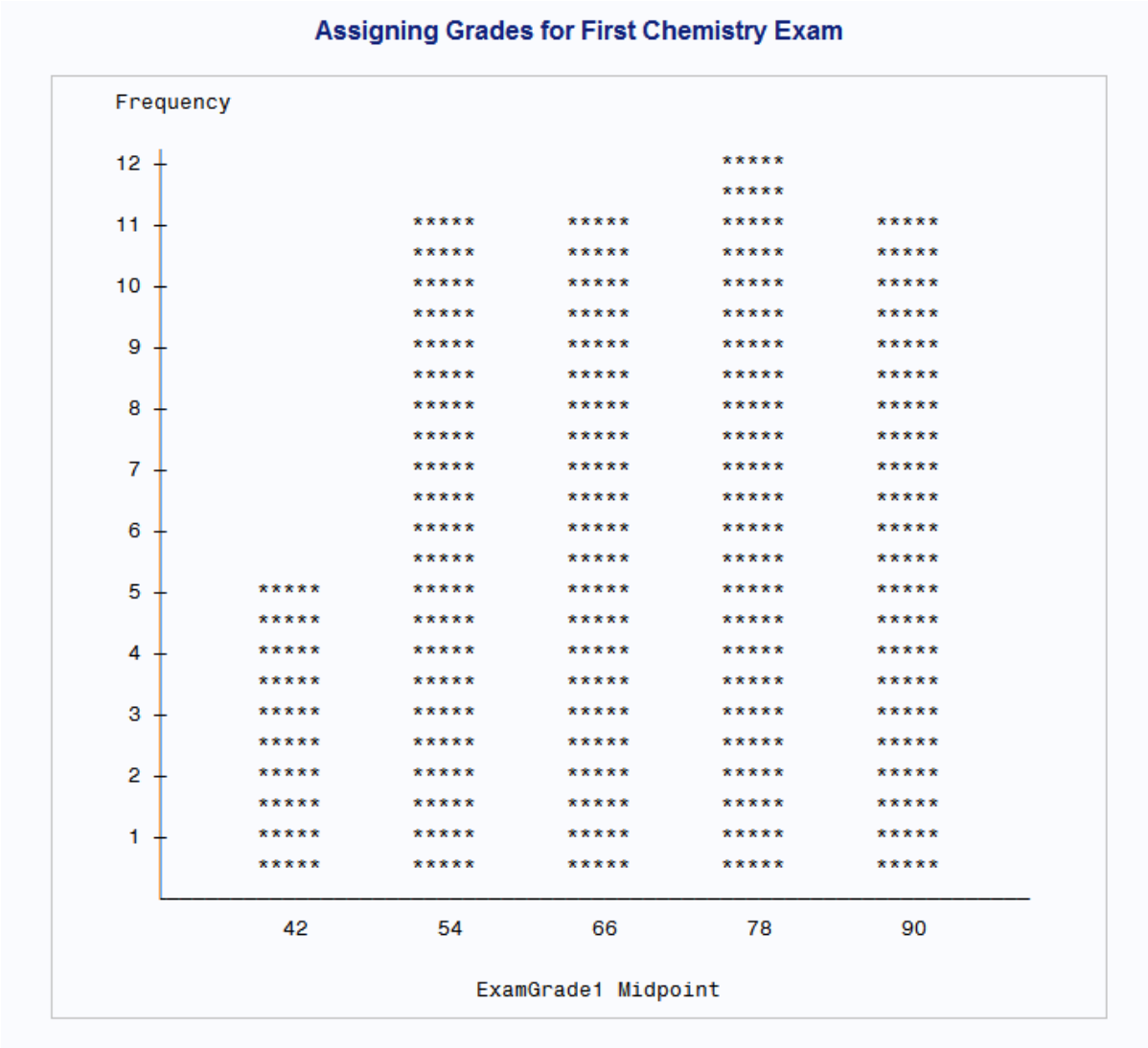
```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   vbar Examgrade1 / levels=5;
   title 'Assigning Grades for First Chemistry Exam';
run;
```

LEVELS=オプションにより、PROC CHART で中間点を 5 つのみ計算するように強制します。

1 SAS を使用して、チャート作成前にデータを正規化できます。

棒グラフの出力を次に示します。

図 31.8 縦棒の 5 つの中間点の指定



これらの中間点に成績を割り当てると、3 人の生徒の試験成績が最も低い範囲のものになります。

すべての値のチャートの作成

デフォルトでは、PROC CHART は、すべての数値変数が連続尺度であるとみなし、MIDPOINTS=または LEVELS=を使用しない限り、自動的にその間隔を選択します。数値変数が連続尺度の数値変数ではなく離散数値変数であることを DISCRETE オプションを使用して指定できます。PROC CHART は、離散数値変数の重複しない値それぞれを表す棒がある度数グラフを作成します。

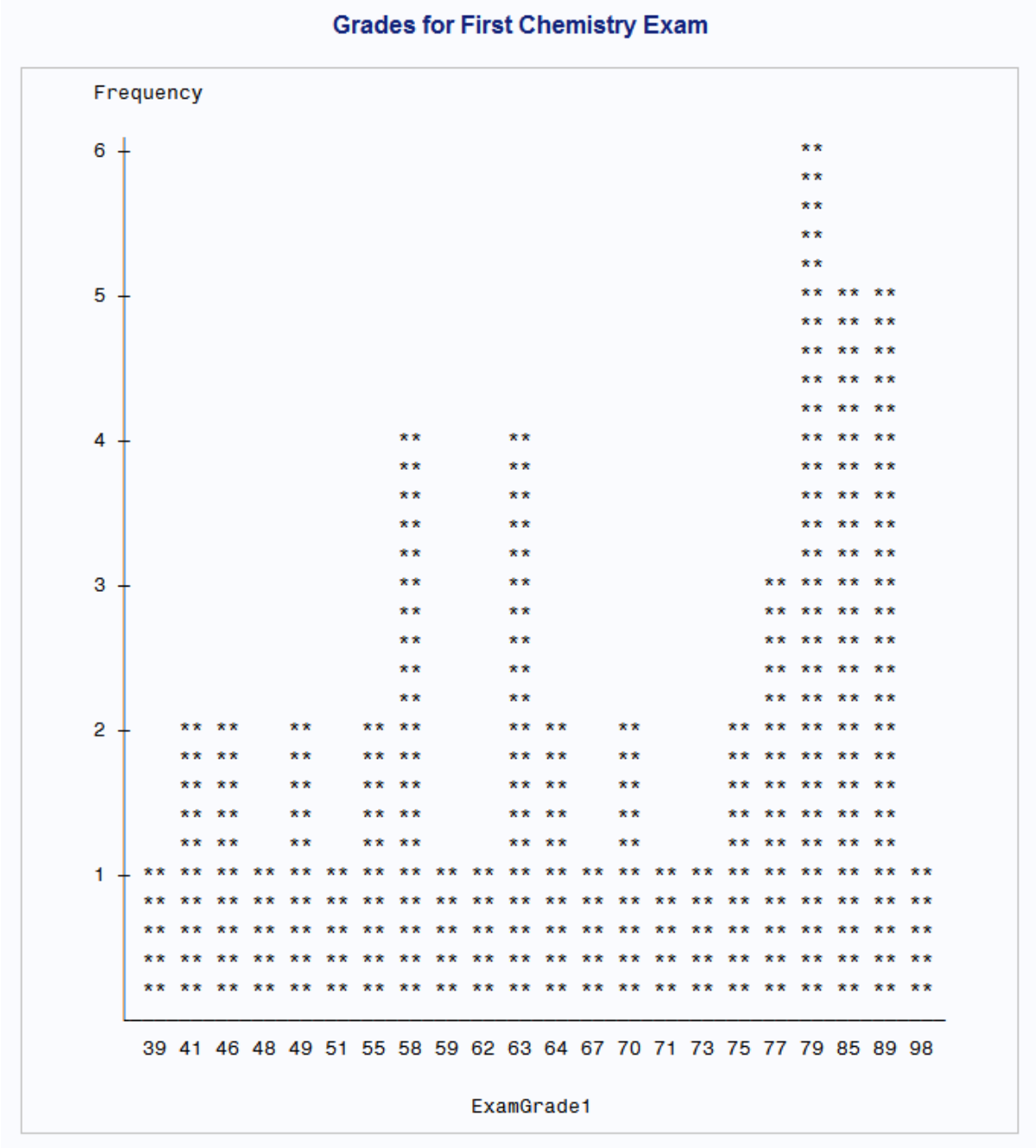
次のプログラムは、DISCRETE オプションを使用して ExamGrade1 のそれぞれの値を表す棒がある棒グラフを作成します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
```

```
   vbar Examgrade1 / discrete;
   title 'Grades for First Chemistry Exam';
run;
```

棒グラフの出力を次に示します。

図 31.9 試験の成績ごとの棒の指定



グラフは、ほとんどの場合、特定の成績を 1 人または 2 人の生徒のみが取ったことを示しています。ただし、3 人以上の生徒の集団が 58、63、77、79、85、89 の成績をとっています。この試験の最頻値は(最も頻繁に取得された試験成績)は 79 です。

注: PROC CHART は離散数値変数の値を横軸上に比例配置しません。

文字変数の度数グラフの作成

概要

数値変数だけでなく、文字変数のチャートを作成できます。たとえば、PROC CHART で、登録者数をセクション間で比較するため、セクションごとの生徒数を示すチャートを作成します。

文字変数の度数グラフの作成方法は、数値変数の度数グラフの作成方法と同様です。ただし、数値変数のチャート作成と文字変数のチャート作成の主な違いは、PROC CHART が中間点を選択する方法です。デフォルトでは、PROC CHART は、DISCRETE オプションが有効であるかのように文字変数の各値を中間点とします。中間点の選択肢を変数の値のサブセットに限定できます。チャート変数の出力形式を定義しない場合、単一の棒、ブロックまたはセクションが変数の単一の値を表します。

文字変数の中間点の指定

デフォルトでは、PROC CHART が文字変数に使用する中間点はアルファベット順です。ただし、MIDPOINTS=オプションを使用して、中間点の順序は簡単に並べ替えることができます。文字変数に MIDPOINTS=オプションを使用する場合、それぞれの中間点の値を一重引用符または二重引用符で囲む必要があります。また、その値はデータセットの値に対応している必要があります。たとえば、

```
midpoints='Mon' 'Wed' 'Fri'
```

では、クラスのセクションの中間点として 3 つの曜日が使用されます。

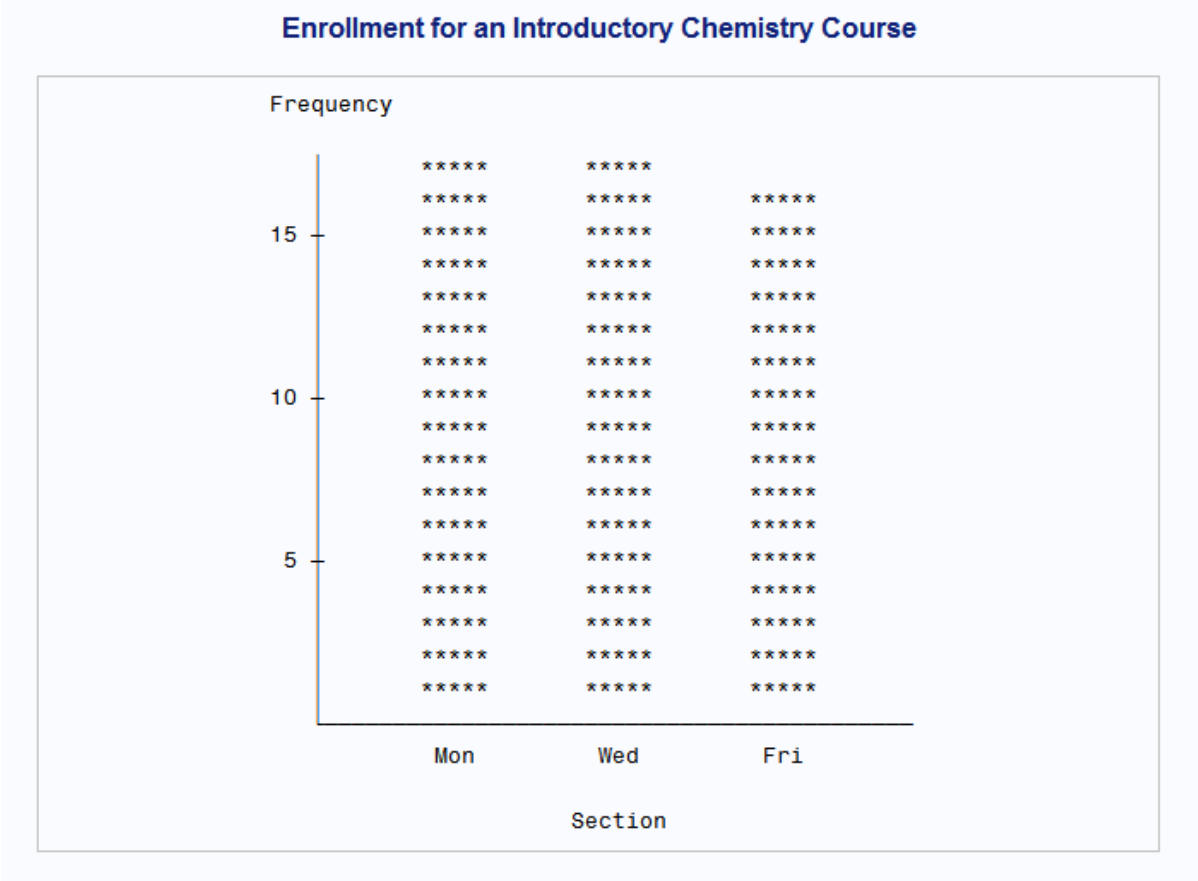
次のプログラムは、MIDPOINTS=オプションを使用して各セクションに登録した生徒の数を示す棒グラフを作成します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   vbar Section / midpoints='Mon' 'Wed' 'Fri';
   title 'Enrollment for an Introductory Chemistry Course';
run;
```

MIDPOINTS=オプションによりグラフが変更され、曜日がアルファベット順ではなく時系列で表示されます。

棒グラフの出力を次に示します。

図 31.10 文字の中間点の時系列での並べ替え



グラフは月曜日および水曜日のセクションに同数の生徒がおり、金曜日のセクションは生徒の数が 1 人少ないことを示しています。

範囲内のサブグループの作成

BLOCK ステートメント、HBAR ステートメントまたは VBAR ステートメントで SUBGROUP=オプションを使用して、各棒またはブロックをサブグループがどのように構成しているかを示すことができます。たとえば、SUBGROUP=オプションを使用して母集団内の性差のパターンを調査できます。

SUBGROUP=オプションはサブグループ変数と呼ばれる変数を定義します。PROC CHART は、複数の値が同一の最初の文字で開始する場合を除いて、各値の最初の文字を使用してその値に対応する棒やブロックの部分を埋めます。該当する場合、PROC CHART は文字 A、B、C 以降の文字を使用して棒またはブロックを埋めます。

変数に出力形式を割り当てた場合、PROC CHART はフォーマットされた値の最初の文字を使用します。PROC CHART がグラフで使用する文字とそれらが表す値は、グラフの下部の凡例に示されます。

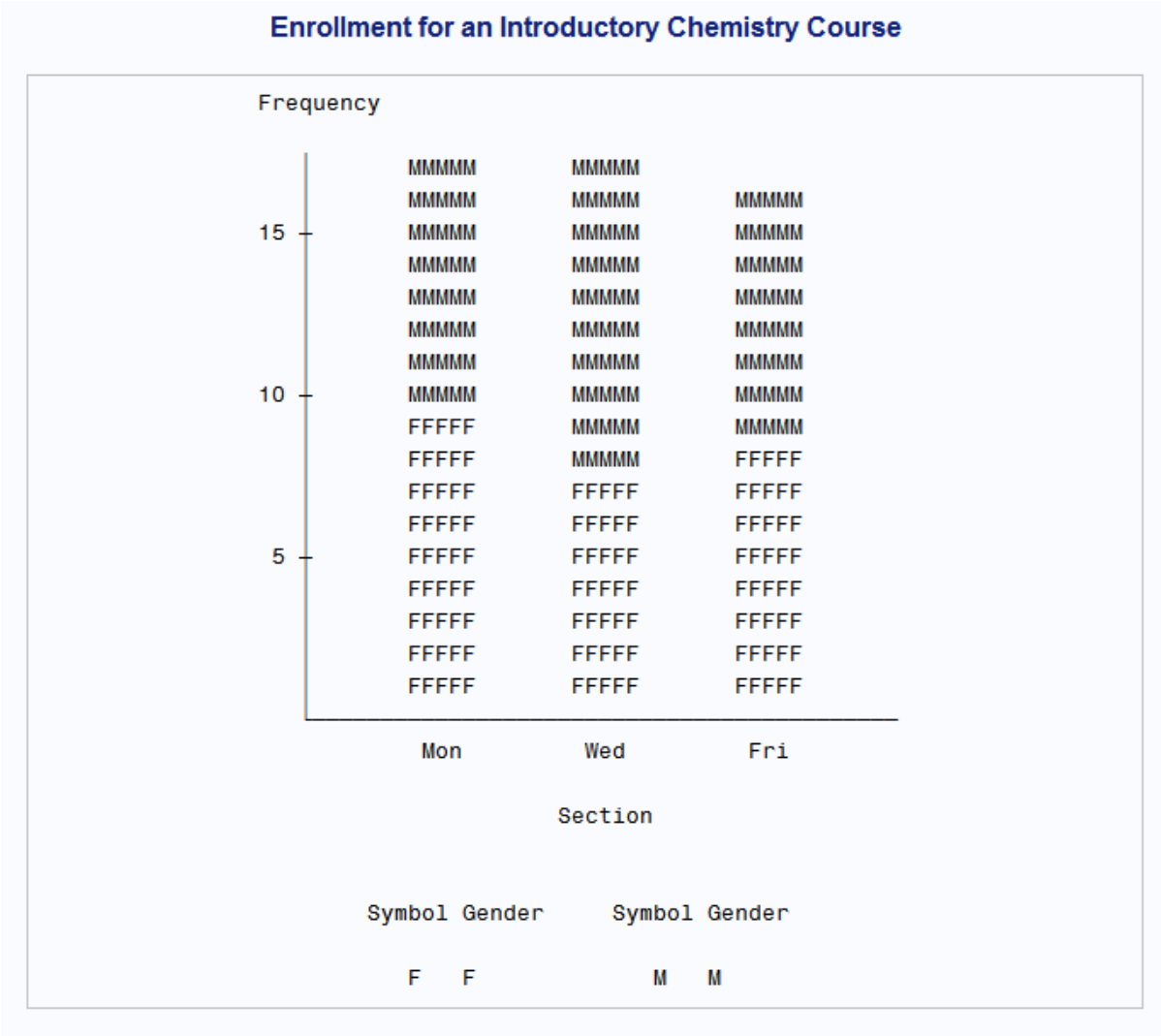
PROC CHART はサブグループの記号を A-Z、および 0-9 といったように文字を昇順で並べ替えます。PROC CHART は各サブグループの棒の高さまたはブロックの高さを個別に計算し、棒の合計に対する割合を切り上げまたは切り捨てます。そのため、棒の合計の高さは、SUBGROUP=オプションを使用しない場合の同一の棒の高さより大きいまたはより小さい場合があります。

次のプログラムは GENDER をサブグループ変数として使用し、各セクションの男性メンバ数、女性メンバ数を示します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   vbar Section / midpoints='Mon' 'Wed' 'Fri'
                  subgroup=Gender;
   title 'Enrollment for an Introductory Chemistry Course';
run;
```

棒グラフの出力を次に示します。

図 31.11 *性別を使用したサブグループの形成*



PROC CHART は、グラフの各棒を変数 GENDER の値を表す文字で埋めます。F で埋められた棒の部分は女性に対応するオブザベーションの数を表し、M で埋められた部分は男性に対応するオブザベーションの数を表します。Gender の値には単一の文字(F または M)が含まれているため、PROC CHART が埋め込み文字として使用する記号は変数の値と一致します。

平均値のチャート作成

PROC CHART では、チャートの棒またはセクションが表す項目を指定できます。デフォルトでは、それぞれの棒、ブロックやセクションは、チャート変数の度数を表します。また、その値によってチャートの棒、ブロックまたはセクションのサイズを決定する変数を指定できます。

SUMVAR=オプションを使用して sumvar 変数と呼ばれる変数を定義します。SUMVAR=オプションとともに、TYPE=オプションを使用して Sumvar 変数の合計または Sumvar 変数の平均によって棒のサイズまたはセクションのサイズを決定するかどうか指定できます。使用可能な種類を次に示します。

SUM
: 各範囲の Sumvar 変数の値を合計します。PROC CHART は合計を使用してそれぞれの棒、ブロックまたはセクションのサイズを決定します。SUM はデフォルトの種類です。

MEAN
: 各範囲の Sumvar 変数の平均値を決定します。その後、PROC CHART は平均値を使用してそれぞれの棒、ブロックまたはセクションのサイズを決定します。

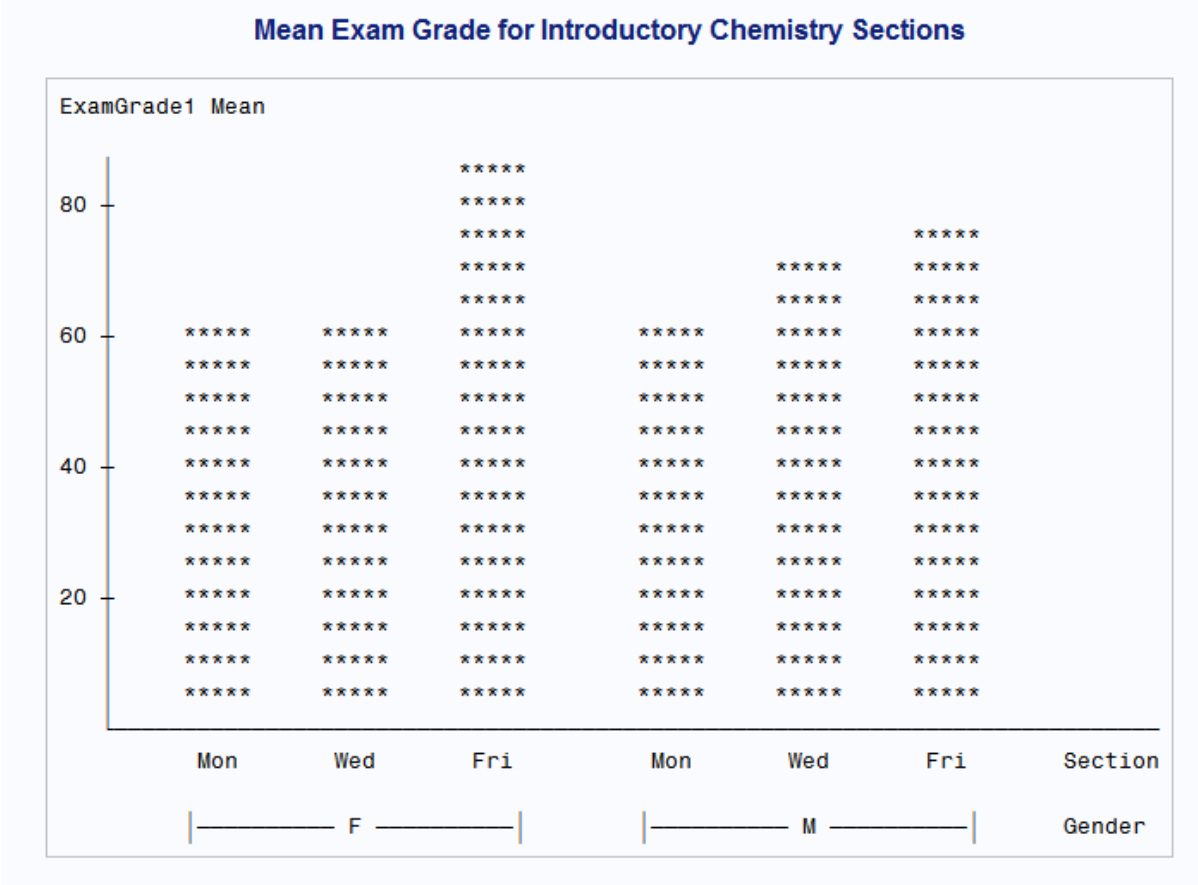
次のプログラムは、性別でグループ化された棒グラフを作成し、各セクションのすべての成績の平均値を比較します。

```
options pagesize=40 linesize=80 pageno=1 nodate;

proc chart data=grades;
   vbar Section / midpoints='Mon' 'Wed' 'Fri' group=Gender
                  sumvar=Examgrade1 type=mean;
   title 'Mean Exam Grade for Introductory Chemistry Sections';
run;
```

SUMVAR=オプションにより、ExamGrade1 の値で棒のサイズを決定するように指定されます。TYPE=MEAN オプションにより、各グループの平均の成績を比較するように指定されます。

棒グラフの出力を次に示します。

図 31.12 SUMVAR=オプションを使用した平均値の比較



チャートは、金曜日セクションの女性達が最も高い成績を残し、同じセクションの男性達が次点であることを示します。

3 次元グラフの作成

GROUP=オプションを使用してチャート作成されたものなど複雑なリレーションシップは、3 次元のブロックチャートとして表示するとわかりやすいことがあります。次のプログラムは、BLOCK ステートメントを使用して数値変数 ExamGrade1 のブロックチャートを作成します。

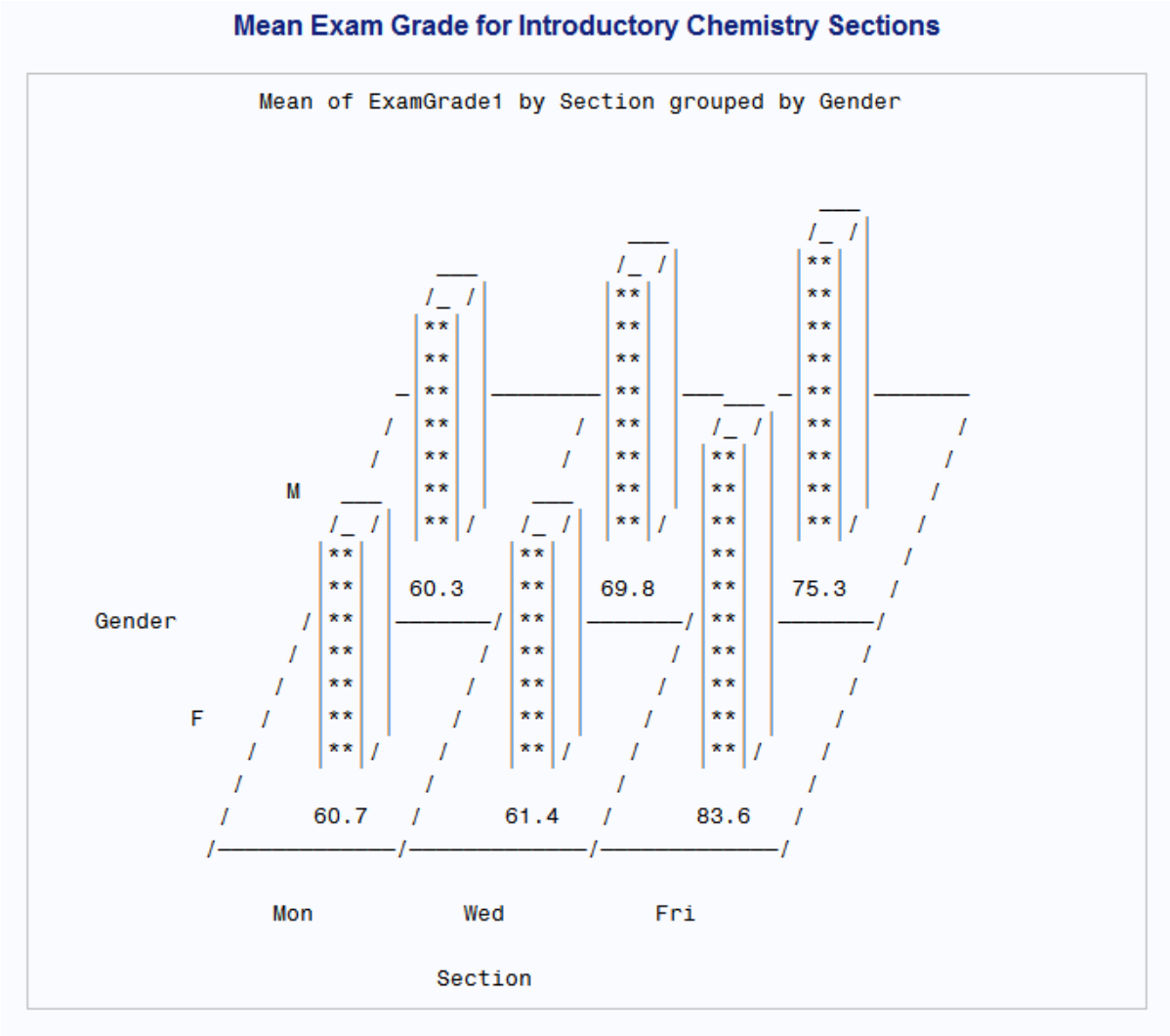
```
options linesize=120 pagesize=40 pageno=1 nodate;
proc chart data=grades;
   block Section / midpoints='Mon' 'Wed' 'Fri'
                   sumvar=Examgrade1 type=mean
                   group=Gender;
   format Examgrade1 4.1;
   title 'Mean Exam Grade for Introductory Chemistry Sections';
run;
```

FORMAT ステートメントは、PROC CHART で ExamGrade1 の平均値の、各ブロックの下での報告に使用する小数点以下桁数を指定します。

注: 行サイズまたはページサイズがすべての棒を表示するのに十分でない場合、PROC CHART は横棒グラフを生成します。

ブロックチャートの出力を次に示します。

図 31.13 ブロックチャートを使用したグループ平均の比較



各ブロックの下に表示される値は、その Section と Gender の組み合わせの ExamGrade1 の平均です。金曜日セクションの女性と男性の両方が他のセクションの対抗者より高い成績を残したことが容易にわかります。

高解像度ヒストグラムの作成

HISTOGRAM ステートメントの使用方法について

ヒストグラムは縦棒グラフと類似しています。この種類の棒グラフは連続尺度の数値変数の個別の範囲を強調し、データの分布を評価できるようにします。

PROC UNIVARIATE ステップの HISTOGRAM ステートメントは、ヒストグラムと比較ヒストグラムを生成します。PROC UNIVARIATE は、データを等しい長さの区間に分割し、各区間のオブザベーションの数をカウントして、カウントを(各区間の中間点を中心として配置される)縦棒としてプロットすることでヒストグラムを作成します。

オプションを使用せずに HISTOGRAM ステートメントを使用すると、PROC UNIVARIATE は自動的に次を実行します。

- 区間内のオブザベーションのパーセントを示すための縦軸のスケール調整
- Terrell and Scott (1985)の手法に基づいた棒幅の決定
- 軸のラベル付け

HISTOGRAM ステートメントには、ヒストグラムのレイアウトを制御し、グラフの拡張を可能にするさまざまなオプションが用意されています。ヒストグラムに密度曲線のファミリを当てはめてカーネル密度推定値を重ね合わせ、データ分布の評価に活用することもできます。SAS が計算する密度曲線の詳細情報については、*Base SAS Procedures Guide* の UNIVARIATE プロシジャを参照してください。

SAS/GRAPH を使用したヒストグラムの作成方法について

サイトで SAS/GRAPH ソフトウェアのライセンスを所有している場合、HISTOGRAM ステートメントを使用して高解像度グラフを作成できます。グラフィックデバイスを使用してグラフを作成する場合、AXIS、LEGEND、PATTERN、SYMBOL の各ステートメントもプロットの拡張に使用できます。

高解像度グラフの外観を制御するには、グラフを作成する PROC ステップより前に、GOPTIONS ステートメントを指定します。GOPTIONS ステートメントは、グラフィック出力が作成されるときに SAS によって使用されるグラフィックオプションの値を変更します。グラフィックオプションは、サイズ、色、フォントの種類、塗りつぶしパターン、および線の太さなどのグラフの特徴に影響します。さらに、表示の外観、作成される出力の種類、出力先などのデバイスパラメータの設定にも影響します。

このセクションのほとんどの例では、次の GOPTIONS ステートメントを使用します。

```
goptions reset=global
         gunit=pct
         hsize= 5.625 in
         vsize= 3.5 in
         htitle=4
         htext=3
         vorigin=0 in
         horigin= 0 in
         cback=white border
         ctext=black
         colors=(black blue green red yellow)
         ftext=swiss
         lfactor=3;
```

グラフィック出力の外観の変更方法の詳細については、*SAS/GRAPH: Reference* を参照してください。

単純なヒストグラムの作成

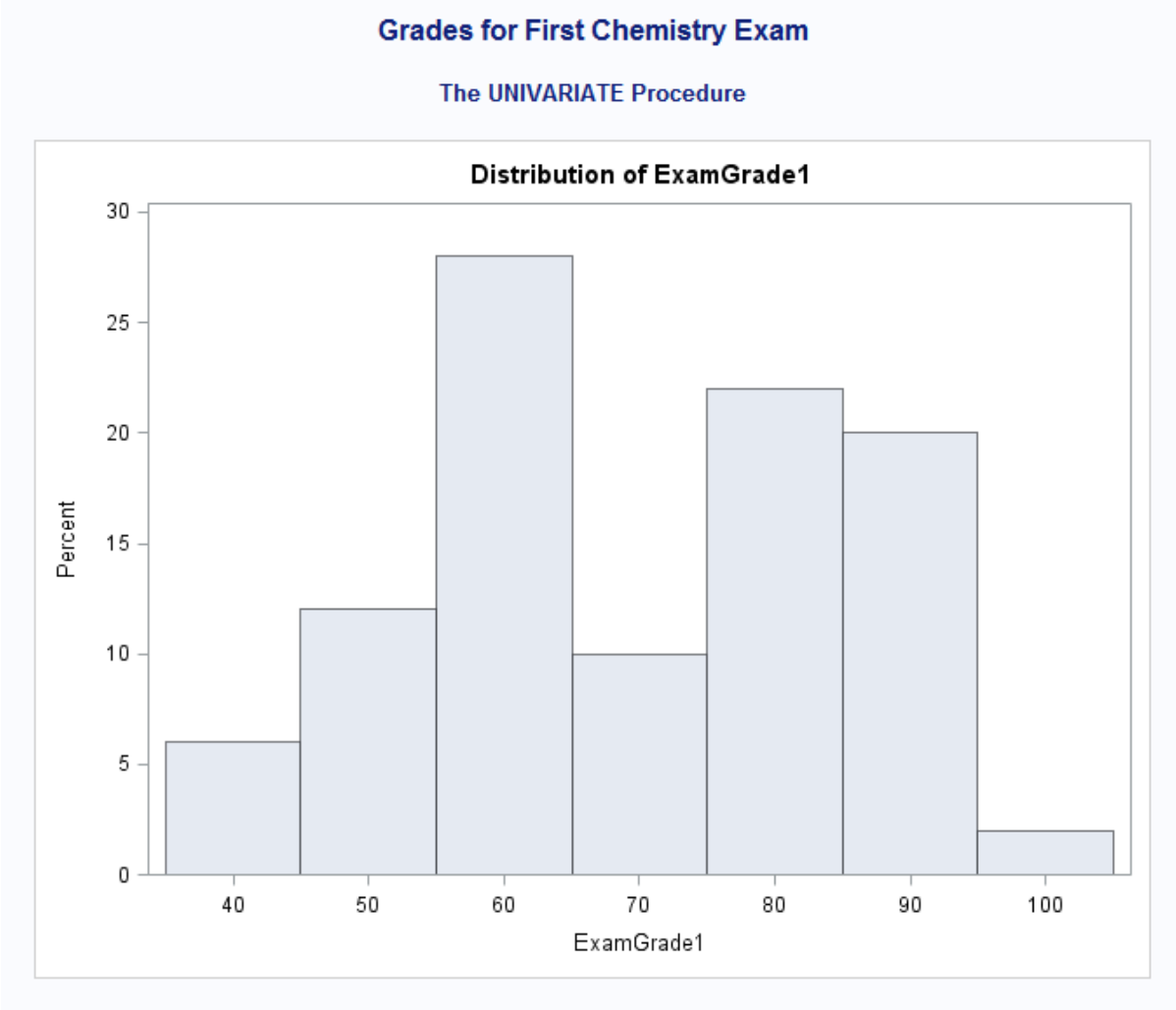
次のプログラムは、HISTOGRAM ステートメントを使用して数値変数 ExamGrade1 のヒストグラムを作成します。

```
proc univariate data=grades noprint;
   histogram ExamGrade1;
   title 'Grades for First Chemistry Exam';
run;
```

NOPRINT オプションは、PROC UNIVARIATE ステートメントによって作成される統計量のテーブルを非表示にします。

次の図にヒストグラムを示します。

図 31.14 ヒストグラムを使用したパーセントの表示



前述のヒストグラムの中間点の軸は 40-100 で、10 の間隔で増分します。次の表に値を示します。

表 31.3 ヒストグラム値

区間	中間点
35-44	40
45-54	50
55-64	60
65-74	70
75-84	80

区間	中間点
85-94	90
95-104	10

注: PROC UNIVARIATE は、区間のサイズとその中間点の位置を、数値変数のすべての値に基づいて選択するため、最高区間と最低区間はデータの値を超えて延長される場合があります。この例では、最も低い成績は 39 で、最低区間は 35-44 です。同様に、最も高い成績は 98 で、最高区間は 95-104 です。

ヒストグラムの軸の変更

縦軸の拡張

ヒストグラムの棒の正確な値は、判別するのが難しい場合があります。デフォルトでは、PROC UNIVARIATE は、縦軸の値(主目盛)の間に補助目盛を提供しません。VMINOR=オプションを使用して主目盛間の補助目盛の数を指定できます。

主目盛の位置をわかりやくするには、GRID オプションを使用してヒストグラムにグリッド線を追加します。グリッド線は横線で、縦軸の主目盛に配置されます。PROC UNIVARIATE には、グリッド線の外観を変更する 2 つのオプションがあります。

CGRID= グリッド線の色を設定します。

LGRID= グリッド線の線の種類を指定します。

デフォルトでは、PROC UNIVARIATE はデバイスのカラーリストの最初の色を使用して実線を描きます。使用可能な線の種類のリストについては、*SAS/GRAPH: Reference* を参照してください。

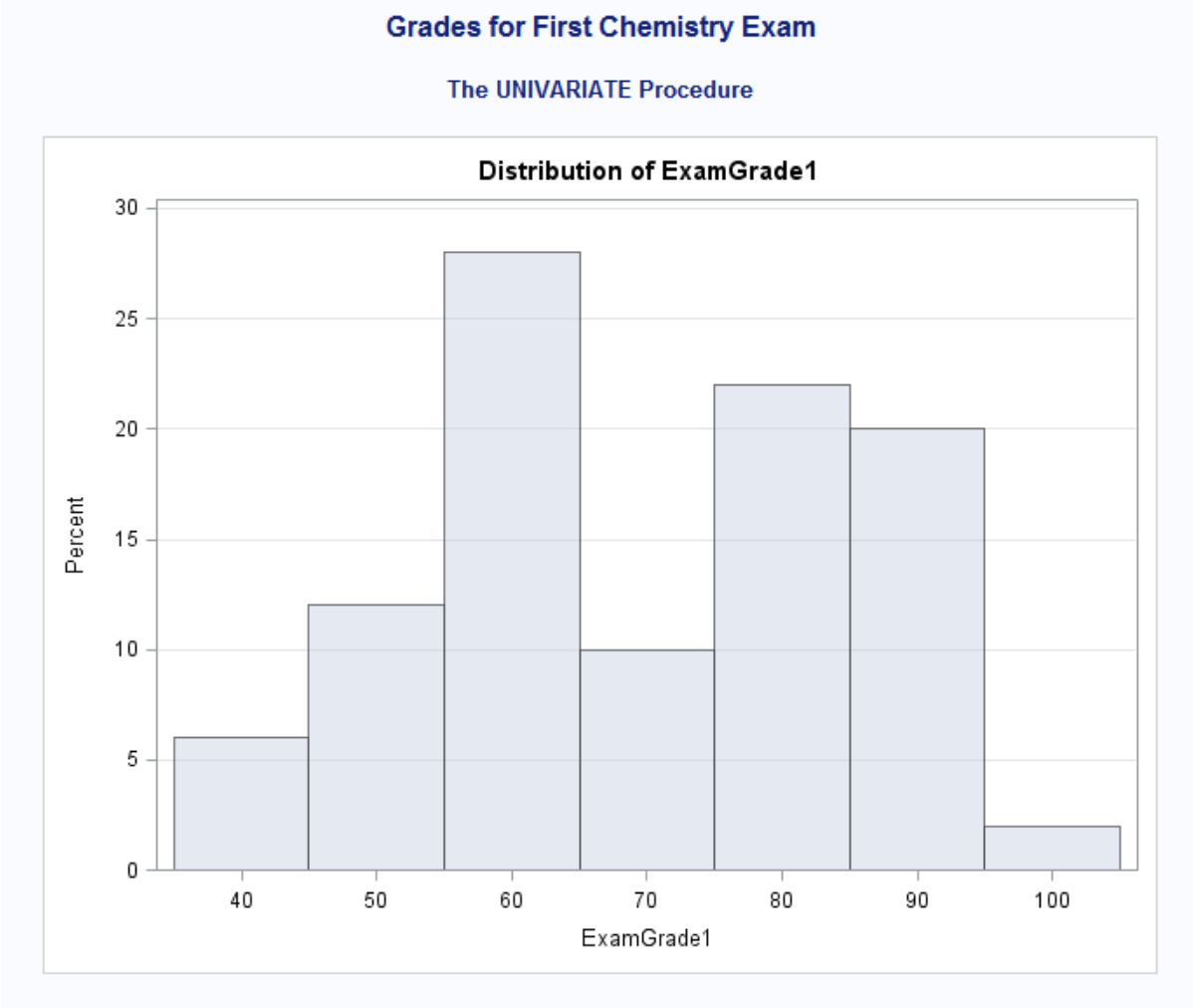
次のプログラムは、補助目盛およびグリッド線を表示する、数値変数 ExamGrade1 のヒストグラムを作成します。

```
proc univariate data=grades noprint;
   histogram Examgrade1 / vminor=4 grid lgrid=34;
   title 'Grades for First Chemistry Exam';
run;
```

各主目盛間に 4 つの補助目盛が挿入されます。間隔を狭くとった点を使用してグリッド線を描きます。

次の図にヒストグラムを示します。

図 31.15 ヒストグラムのグリッド線の指定



それぞれのヒストグラムの棒の高さが簡単にグラフから判別できるようになりました。次の表に各区間が表すパーセントを示します。

表 31.4 各区間のパーセント

区間	パーセント
35-44	6
45-54	12
55-64	28
65-74	10
75-84	22
85-94	20

区間	パーセント
95-104	2

縦軸の値の指定

PROC UNIVARIATE では、ヒストグラムの棒が表す項目、および縦軸の値を指定できます。デフォルトでは、それぞれの棒は、指定区間に該当するオブザベーションのパーセントを表します。

VSCALE=オプションを使用して、次のスケールを縦軸に指定できます。

- COUNT
- PERCENT
- PROPORTION

VAXIS=オプションを使用して、均等に配置された目盛値を縦軸に指定できます。次に、このオプションの形式を示します。

HISTOGRAM *variable* / VAXIS=*value-list*;

ここで value-list は主目盛の値として使用する数値のリストです。最初の値は常にゼロに等しく、最後の値は常に最大の棒の高さに等しいかより大きいです。

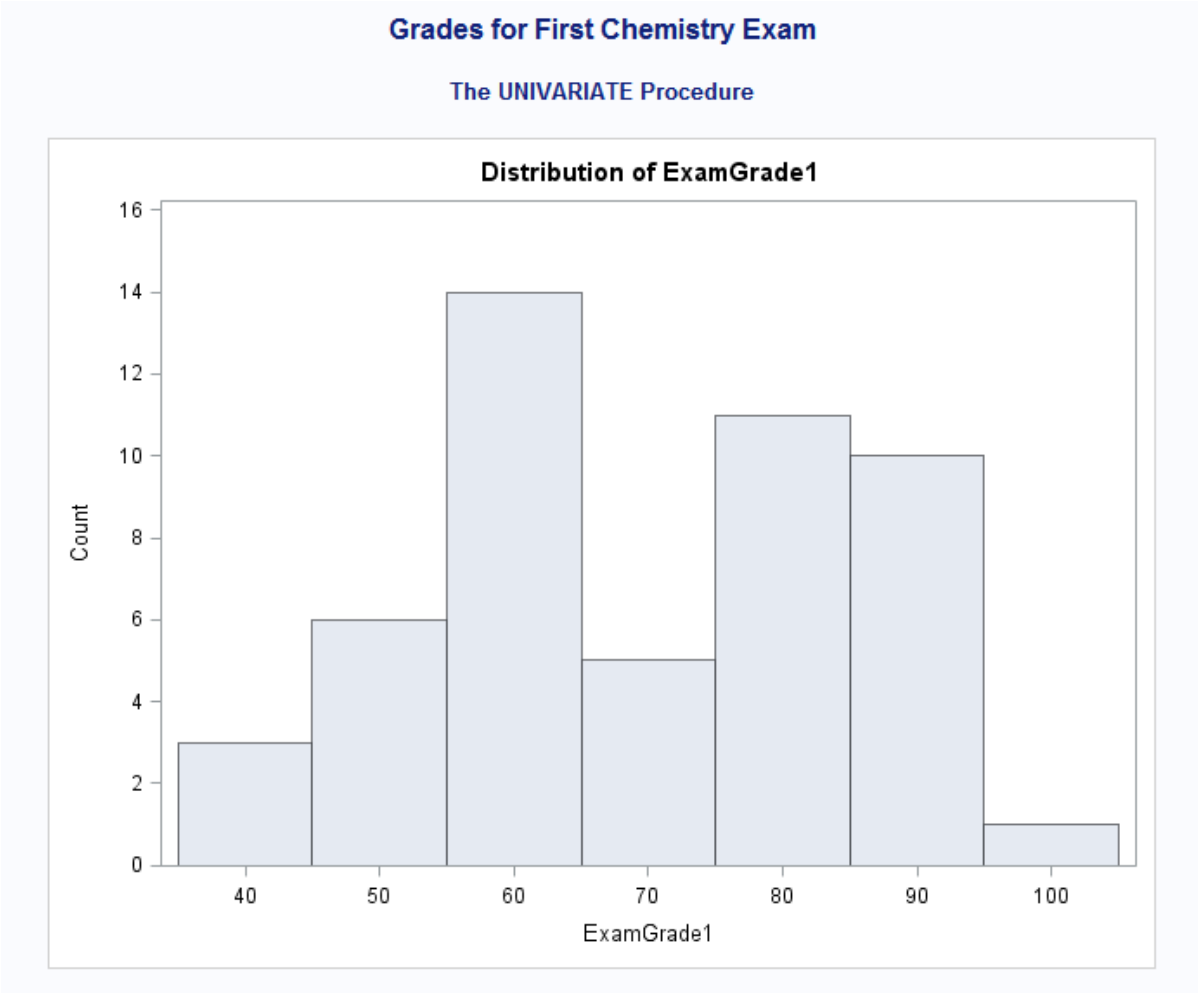
次のプログラムは数値変数 ExamGrade1 のカウントを縦軸に表示するヒストグラムを作成します。

```
proc univariate data=grades noprint;
   histogram Examgrade1 / vscale=count vaxis=0 to 16 by 2 vminor=1;
   title 'Grades for First Chemistry Exam';
run;
```

縦軸の値は、0-16 の範囲で 2 ずつ増分します。各主目盛間に 1 つの補助目盛が挿入されます。

次の図にヒストグラムを示します。

図 31.16 ヒストグラムを使用したカウントの表示



ヒストグラムの中間点の指定

MIDPOINTS=オプションを使用してヒストグラムの棒の幅を制御できます。PROC UNIVARIATE は中間点の値を使用してヒストグラムの棒の幅を決定します。連続する中間点と中間点の間の差が棒の幅です。

中間点を指定するには、HISTOGRAM ステートメントで MIDPOINTS=オプションを使用します。MIDPOINTS=オプションの形式を次に示します。

HISTOGRAM *variable* / MIDPOINTS=*midpoint-list*;

ここで midpoint-list は中間点として使用する数値のリストです。等間隔の中間点を昇順で記述して使用する必要があります。

たとえば、中間点が 55-95 である、従来の成績付け範囲を指定するには、次のオプションを使用します。

```
midpoints=55 65 75 85 95
```

または、この中間点リストを次のように省略形にできます。

```
midpoints=55 to 95 by 10
```

次のプログラムは、MIDPOINTS=オプションを使用して数値変数 ExamGrade1 のヒストグラムを作成します。

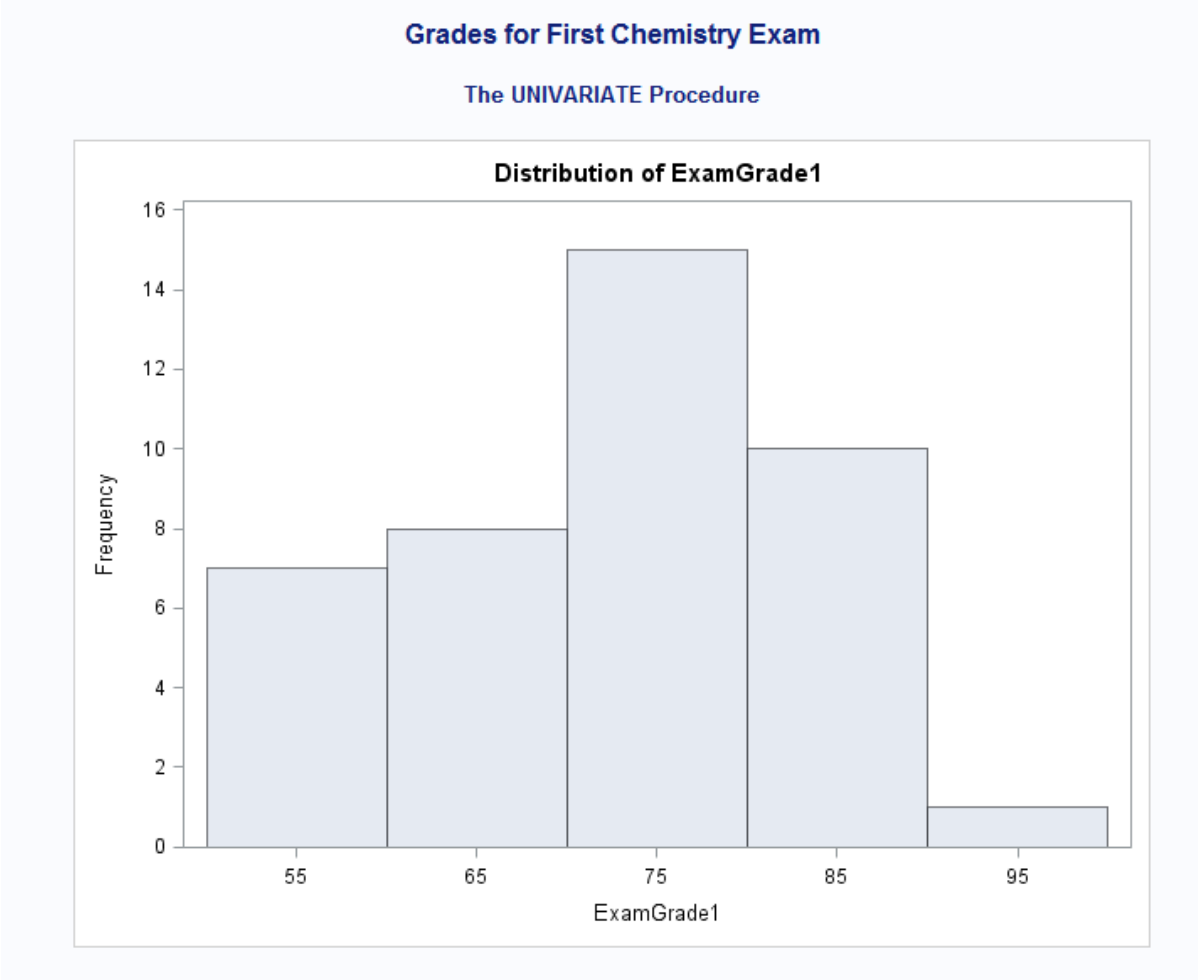
```
proc univariate data=grades
   (where=(examgrade1 ge 55 and examgrade1 le 100 )) noprint;
   histogram Examgrade1 / vscale=count vaxis=0 to 16 by 2 vminor=1
                          midpoints=55 65 75 85 95 hoffset=10
                          vaxislabel='Frequency';
   title 'Grades for First Chemistry Exam';
run;
```

次のリストは、前述のプログラムの項目に対応しています。

- MIDPOINTS=オプションは、PROC UNIVARIATE で 5 つの棒を試験成績の従来の中間点を中心に配置するように強制します。
- HOFFSET=オプションは、横軸の両端に 10%のオフセットを使用します。
- VAXISLABEL=オプションは Frequency を縦軸のラベルとして使用します。デフォルトのラベルは Count です。

次の図にヒストグラムを示します。

図 31.17 *ヒストグラムの 5 つの中間点の指定*



前述のヒストグラムの中間点の軸は 55-95 で、10 の間隔で増分します。50 より低い試験成績はヒストグラムからすべて除外されます。

ヒストグラムでの要約統計量の表示

INSET ステートメントの使用方法について

PROC UNIVARIATE を使用して、インセットと呼ばれる要約統計量のボックスまたはテーブルをヒストグラムに直接追加できます。通常、インセットは PROC UNIVARIATE によって計算された統計量を示しますが、SAS データセットで指定した値を表示することもできます。

要約統計量のテーブルを追加するには、INSET ステートメントを使用します。UNIVARIATE プロシジャで複数の INSET ステートメントを使用して、ヒストグラムに複数のテーブルを追加できます。INSET ステートメントは、拡張する対象プロットを作成する HISTOGRAM ステートメントの後である必要があります。インセットは先行の HISTOGRAM ステートメントが生成するすべてのグラフに表示されます。

次に INSET ステートメントの形式を示します。

INSET *<keyword(s)> </ option(s)>*

インセットの統計量のキーワード(N、MIN、MAX、MEAN、STD など)を、ワード INSET の直後に指定します。キーワード DATA=の後に続けて SAS データセットの名前を指定して、SAS データセットに格納されているカスタマイズされた統計量を表示することもできます。統計量はキーワードを指定した順に表示されます。

デフォルトでは、PROC UNIVARIATE は適切なラベルおよび適切な出力形式を使用してインセットの統計量を表示します。ラベルをカスタマイズするには、キーワードに続けて等記号(=)と使用するラベルを引用符で囲んで指定します。出力形式をカスタマイズするには、キーワードの後に数値の出力形式をかっこに囲んで指定します。最長 24 文字までのラベルを割り当てることができます。キーワードにラベルと出力形式の両方を指定する場合、ラベルを出力形式より前に記述する必要があります。たとえば、

```
inset n='Sample Size' std='Std Dev' (5.2);
```

は 2 つの統計量(サンプルサイズと標準偏差)にカスタマイズされたラベルを要求します。標準偏差には、小数点以下桁数 2 桁を含む 5 のフィールド幅の出力形式も割り当てられます。

さまざまなオプションを使用してインセットの外観をカスタマイズできます。たとえば、次の操作を行えます。

- インセットの位置を指定します。
- インセットテーブルのヘッダーを指定します。
- 背景色、テキストの色、テキストの高さ、テキストフォント、ドロップシャドウなどのグラフィカルな拡張を指定します。

INSET ステートメントで使用可能なキーワードとオプションの完全なリストについては、*Base SAS Procedures Guide* を参照してください。

プログラム

次のプログラムは、INSET ステートメントを使用して数値変数 ExamGrade1 の要約統計量をヒストグラムに追加します。

```
proc univariate data=grades noprint;
   histogram Examgrade1 /vscale=count vaxis=0 to 16 by 2 vminor=1 hoffset=10
                         midpoints=55 65 75 85 95 vaxislabel='Frequency';
   inset n='No. Students' mean='Mean Grade' min='Lowest Grade' 1
         max='Highest Grade' / header='Summary Statistics' position=ne 2 3
                               format=3. 4 ;
```

```
   title 'Grade Distribution for the First Chemistry Exam';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

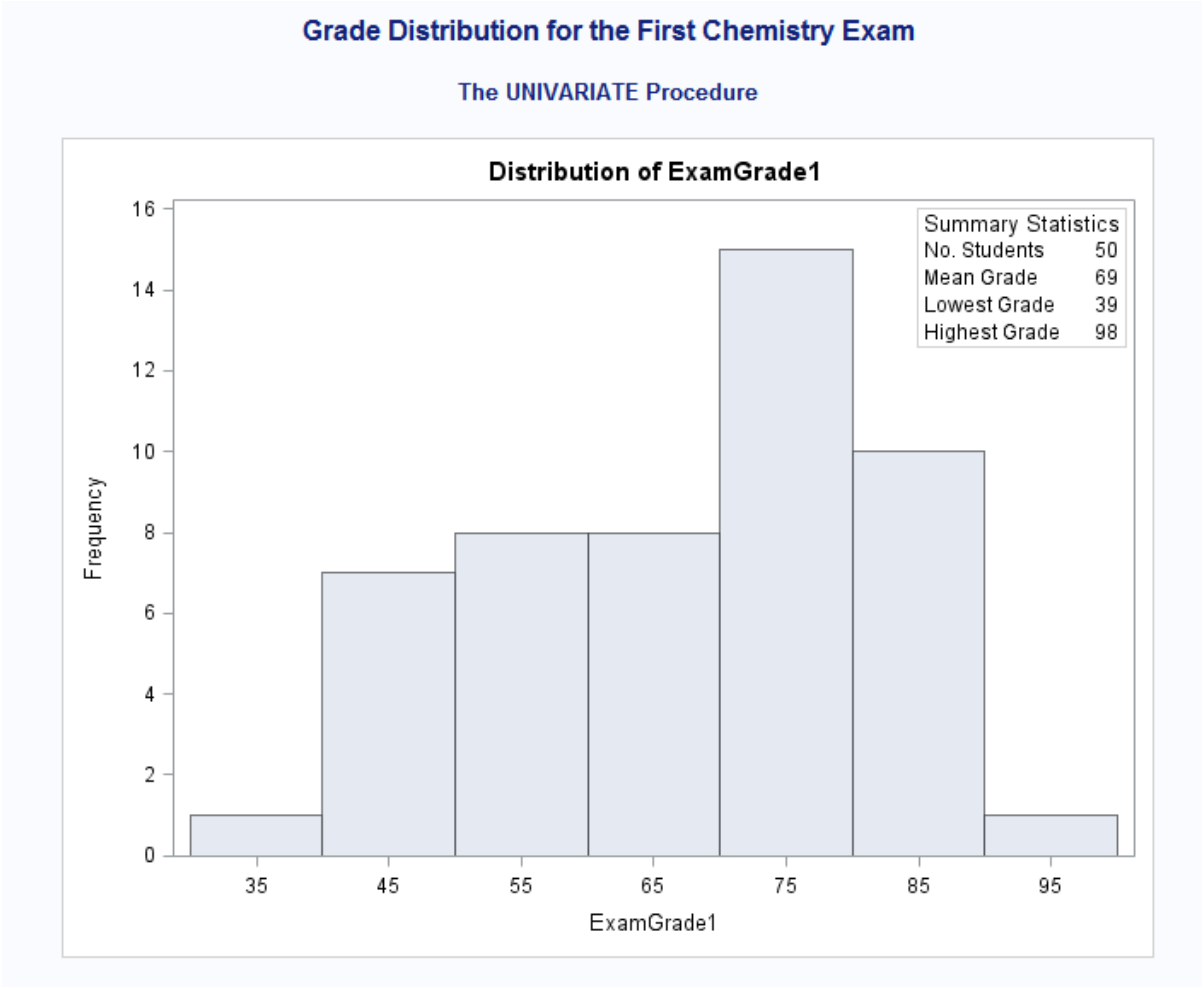
1 統計量のキーワード N、MEAN、MIN、MAX はオブザベーションの数、平均試験成績、最小試験成績、最大試験成績をインセットに表示することを指定します。キーワードはそれぞれ、インセットの統計量を特定するためのカスタマイズされたラベルを割り当てられます。

2 HEADER=オプションは、インセットの上部に表示されるヘッダーテキストを指定します。

3 POSITION=オプションは方位点を使用してインセットを配置します。テーブルがヒストグラムの北東の角(右上隅)に表示されます。

4 FORMAT=オプションは、インセットのすべての統計量に対して 3 のフィールド幅の出力形式を要求します。

図 31.18 *ヒストグラムへのインセットの追加*



ヒストグラムはデータの分布を示します。要約統計量のテーブルがヒストグラムの右上隅に表示され、サンプルサイズ、平均成績、最小値、最大値に関する情報を提供します。

比較ヒストグラムの作成

比較ヒストグラムについて

比較ヒストグラムは、配列または行列として配置される一連の成分ヒストグラムです。PROC UNIVARIATE は均一な横軸および縦軸を使用して成分ヒストグラムを表示します。これにより、比較ヒストグラムを使用して、最大 2 つの分類変数の水準にまたがって数値変数の分布を視覚的に比較できるようになります。

HISTOGRAM ステートメントの CLASS ステートメントを使用して一元比較ヒストグラムまたは二元比較ヒストグラムを作成します。次に CLASS ステートメントの形式を示します。

CLASS *variable-1* <(*variable-option(s)*)> <*variable-2* <(*variable-option(s)*)>> </*options*>;

分類変数は数値または文字のいずれかです。分類変数は連続尺度の値を持つことができますが、通常は変数の水準を定義するいくつかの離散値です。FORMAT ステートメントを使用して、分類変数の値を組み合わせて、分類水準の数を減らすことができます。

1 つの分類変数を指定すると、PROC UNIVARIATE は成分ヒストグラムを上下または左右の配列で表示します。一元比較ヒストグラムを作成するため、PROC UNIVARIATE は分析変数の値を分類変数のフォーマットされた値(水準)で分類します。それぞれの分類水準は個別のヒストグラムを生成します。

2 つの分類変数を指定すると、PROC UNIVARIATE は成分プロットの行列を表示します。二元比較ヒストグラムを作成するため、PROC UNIVARIATE は分析変数の値を分類変数のクロス分類値(水準)で分類します。クロス分類水準の各組み合わせにより別個のヒストグラムが生成されます。class variable-1 の水準は行列の行のラベルで、class variable-2 の水準は行列の列のラベルです。

HISTOGRAM ステートメントでオプションを指定して比較ヒストグラムの外観をカスタマイズできます。たとえば、次の操作を行えます。

- 比較ヒストグラムの行の数を指定します。
- 比較ヒストグラムの列の数を指定します。
- ラベルの背景色、テキストの色などグラフィカルな拡張を指定します。

HISTOGRAM ステートメントで使用可能なキーワードとオプションの完全なリストについては、*Base SAS Procedures Guide* を参照してください。

プログラム

次のプログラムは、CLASS ステートメントを使用して、数値変数 ExamGrade1 の性別およびセクションによる比較ヒストグラムを作成します。

```
proc format;
   value $gendfmt 'M'='Male'
                  'F'='Female'; 1
run;

proc univariate data=grades noprint;
   class Gender 2 Section(order=data); 3
   histogram Examgrade1 / midpoints=45 to 95 by 10 vscale=count vaxis=0 to 6 by 2
                          vaxislabel='Frequency' turnvlabels 4 nrows=2 ncols=3 5
                          cframe=ligr 6 cframeside=gwh cframetop=gwh cfill=gwh; 7
   inset mean(4.1) n / noframe 8 position=(2,65); 9
   format Gender $gendfmt.; 1
```

```
   title 'Grade Distribution for the First Chemistry Exam';
run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 PROC FORMAT は、Gender を文字列でラベル付けするユーザー定義出力形式を作成します。FORMAT ステートメントにより Gender に出力形式が割り当てられます。

2 CLASS ステートメントにより、Gender および Section を分類変数として使用する二元比較ヒストグラムが作成されます。PROC UNIVARIATE は、これらの変数の各水準(値の重複しない組み合わせ)の成分ヒストグラムを生成します。

3 ORDER=オプションにより、入力データセットでの順序に従って Section の値が配置されます。比較ヒストグラムは、Section の水準を曜日に従って表示します(Mon、Wed、Fri)。水準のデフォルトの順序は、Section の内部値の並べ替え順序によって決定されます(Fri、Mon、Wed)。

4 TURNVLABELS オプションにより、縦軸ラベルの文字が回転されて横方向でなく縦方向に表示されます。

5 NROWS=オプションおよび NCOLS=オプションにより、成分ヒストグラムの 2 行 3 列の配置が指定されます。

6 CFRAME=オプションは、軸とフレームによって囲まれる、それぞれの成分ヒストグラムの領域を塗りつぶす色を指定します。CFRAMESIDE=オプションと CFRAMETOP=オプションは、比較ヒストグラムの横側および上部に表示される列ラベルと行ラベルのフレーム領域を塗りつぶす色を指定します。デフォルトでは、これらの領域は塗りつぶしされません。

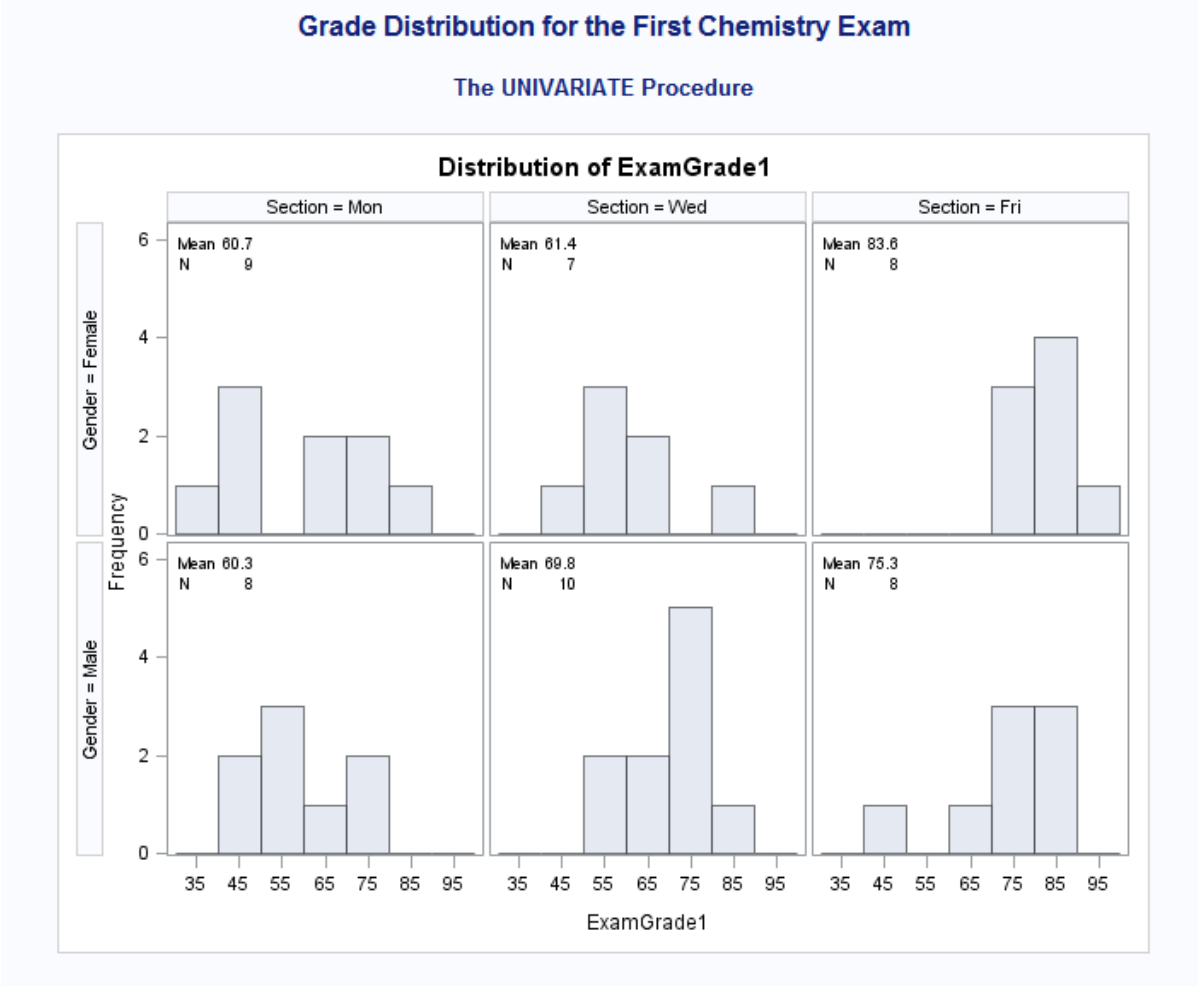
7 CFILL=オプションにより、各成分ヒストグラムの棒を塗りつぶす色が指定されます。デフォルトでは、各棒は塗りつしぶされません。

8 NOFRAME オプションはインセットテーブルの周囲のフレームを非表示にします。

9 POSITION=オプションは、軸のパーセント座標を使用してインセットを配置します。インセットの左下隅の位置は横軸方向へ 2%、縦軸方向へ 65%です。

次の図に比較ヒストグラムを示します。

図 31.19 比較ヒストグラムを使用した Gender と Section による試験成績の検証



比較ヒストグラムは Section と Gender のそれぞれの組み合わせの成分ヒストグラムの 2 行 3 列の行列です。それぞれの成分ヒストグラムは、ExamGrade1 の平均と生徒数を報告する統計量のテーブルを表示します。金曜日セクションの女性と男性の両方が他のセクションの対抗者より高い成績を残したことが容易にわかります。

要約

PROC CHART ステートメント

PROC CHART <DATA=*SAS-data-set*> <*options*>;
chart-type variable(s) </*options*>;

PROC CHART <DATA=SAS-data-set> <options> ;
CHART プロシジャを開始します。PROC CHART ステートメントでは、次のオプションを使用できます。

DATA=SAS-data-set
PROC CHART で使用される SAS データセットの名前を指定します。DATA=が省略されると、PROC CHART は最後に作成されたデータセットを使用します。

LPI=value
PIE グラフ、STAR チャートの比率を指定します。

chart-type variable(s) < /options>;
CHART ステートメントです。

chart-type
次よりいずれかの任意のチャートの種類を指定します。

- BLOCK
- HBAR
- PIE
- VBAR

1つの PROC CHART ステップに任意の数の CHART ステートメントを使用できます。単一の CHART ステートメントに関するオプションのリストを次に示します。

variable(s)
チャート化する変数(チャート変数と呼ばれます)を指定します。

options
オプションのリストを指定します。チャートの種類によって使用できるオプションは制限がある場合があります。

VBAR、HBAR、BLOCK の各ステートメントで次のオプションを使用できます。

GROUP=variable
variable の値それぞれの棒またはブロックのセットを作成します。

SUBGROUP=variable
それぞれのブロックまたは棒を variable の各値を表す文字で比例的に埋めます。VBAR、HBAR、BLOCK、PIE の各ステートメントで次のオプションを使用できます。

DISCRETE
チャート変数の値ごとに棒、ブロックまたはセクションを作成します。

LEVELS=number-of-midpoints
number-of-midpoints を指定します。プロシジャにより中間点が選択されます。

MIDPOINTS=midpoints-list
中間点の値を指定します。

SUMVAR=variable
棒、ブロックまたはセクションのサイズの決定に使用される variable を指定します。

TYPE=SUM|MEAN
作成するチャートの種類を指定します。

SUM
各範囲の Sumvar 変数の値を合計します。PROC CHART は合計を使用してそれぞれの棒、ブロックまたはセクションのサイズを決定します。

MEAN
各範囲の Sumvar 変数の平均値を決定します。その後、PROC CHART は平均値を使用してそれぞれの棒、ブロックまたはセクションのサイズを決定します。

HBAR ステートメントでは、次のオプションを指定できます。

NOSTAT
デフォルトでチャートとともに出力される統計量を非表示にします。

FREQ
度数統計量を要求します。

CFREQ
累積度数統計量を要求します。

PERCENT
パーセント統計量を要求します。

CPERCENT
累積パーセント統計量を要求します。

PROC UNIVARIATE ステートメント

PROC UNIVARIATE *<option(s)>*;

CLASS *variable-1 <(variable-option(s))> <variable-2 <(variable-option(s))>> </option(s)>*;

HISTOGRAM *<variable(s)> </option(s)>*;

INSET *<keyword(s)> </option(s)>*;

PROC UNIVARIATE option(s);
UNIVARIATE プロシジャを開始します。PROC UNIVARIATE ステートメントでは、次のオプションを使用できます。

DATA=SAS-data-set
PROC UNIVARIATE で使用される SAS データセットの名前を指定します。DATA=が省略されると、PROC UNIVARIATE は最後に作成されたデータセットを使用します。

NOPRINT
PROC UNIVARIATE ステートメントによって作成される記述統計量を非表示にします。

CLASS variable-1<(variable-option(s))> <variable-2<(variable-option(s))>>
</ option(s)>;
その値によって成分ヒストグラムの分類水準を決定する、最大 2 つの変数を指定します。CLASS ステートメントの変数は、分類変数と呼ばれます。

CLASS ステートメントに次のオプションを指定できます。

ORDER=DATA | FORMATTED | FREQ | INTERNAL
分類変数の値の表示順序を指定します。

DATA
入力データセットでの順序に従って値を並べ替えます。

FORMATTED
フォーマットされた値の昇順で値を並べ替えます。この順序は、使用している動作環境によって異なります。

FREQ
値を度数カウントの降順で並べ替え、最も多数のオブザベーションがある水準がリストの最初に来ます。

INTERNAL
フォーマットされていない値で値を並べ替えます。PROC SORT を使用するのと同じ順序になります。この順序は、使用している動作環境によって異なります。

HISTOGRAM <variable(s)> </option(s)>;
指定された分析変数に高解像度グラフを使用してヒストグラムと比較ヒストグラムを作成します。HISTOGRAM ステートメントの *variable(s)*を省略すると、プロシジャはヒストグラムを、VAR ステートメントでリストされた各変数について作成するか、または VAR ステートメントを省略した場合は DATA=データセットの各数値変数について作成します。

PROC UNIVARIATE ステートメントでは、次のオプションを使用できます。

CGRID=color
ヒストグラムにグリッドが表示される場合にグリッド線の色を指定します。

GRID
ヒストグラムにグリッドを表示するよう指定します。グリッド線は横線で、縦軸の主目盛に配置されます。

HOFFSET=value
横軸の両端のオフセットを画面のパーセンテージ単位で指定します。

GRID
ヒストグラムにグリッドを表示するよう指定します。グリッド線は横線で、縦軸の主目盛に配置されます。

LGRID=linetype
ヒストグラムにグリッドが表示される場合にグリッドの線の種類を指定します。デフォルトは実線です。

MIDPOINTS=value(s)
連続する中間点と中間点の間の差として、ヒストグラムの棒の幅を決定します。PROC UNIVARIATE は同一の *value(s)*をすべての変数に対して使用します。等間隔の中間点を昇順で記述して使用する必要があります。

VAXIS=value(s)
縦軸の目盛値を指定します。昇順でリストされた等間隔の中間点を使用します。最初の値はゼロで、最後の値は常に最大の棒の高さより大きいか等しい値である必要があります。値は棒と同じ単位でスケール調整する必要があります。

VMINOR=n
縦軸の各主目盛間の補助目盛の数を指定します。PROC UNIVARIATE では補助目盛をラベル付けしません。

VSCALE=scale
縦軸のスケールを指定します。*scale* は次のいずれかです。

COUNT
データ単位ごとのオブザベーション数の単位でデータをスケール調整します。

PERCENT
データ単位ごとのオブザベーションのパーセント単位でデータをスケール調整します。

PROPORTION
データ単位ごとのオブザベーションの比率単位でデータをスケール調整します。

INSET <keyword(s)> </option(s)>;
インセットと呼ばれる要約統計量のボックスまたはテーブルをヒストグラムに直接配置します。

PROC UNIVARIATE ステートメントでは、次のオプションを使用できます。

keyword(s)
インセットに表示する情報を特定する 1 つ以上のキーワードを指定します。PROC UNIVARIATE は、キーワードが要求された順で情報を表示します。キーワードの完全なリストについては、*SAS/GRAPH: Reference* の INSET ステートメントに関する説明を参照してください。

FORMAT=format
インセット内のすべての値の出力形式を指定します。特定の統計量に出力形式を指定する場合、その出力形式が FORMAT=*format* より優先されます。

HEADER=string
string が 40 文字を超えない範囲で、ヘッダーテキストを指定します。

NOFRAME
テキストの周囲に描画される枠を抑制します。

POSITION=position
インセットの位置を決定します。*position* は、方位点キーワード、余白キーワード、または座標ペア(*x*, *y*)です。デフォルトの position は NW で、インセットをディスプレイの左上隅(北西)に配置します。

GOPTIONS ステートメント

GOPTIONS options-list;
グラフィックオプションの値を指定します。グラフィックオプションは、サイズ、色、フォントの種類、塗りつぶしパターン、および記号などのグラフの特徴を制御します。さらに、デバイスエントリで定義されるデバイスパラメータの設定に影響します。デバイスパラメータは、表示の外観、作成される出力の種類、出力先などの特徴を制御します。

FORMAT ステートメント

FORMAT variable format-name;
format-name として指定する特殊なパターンを使用して、*variable* の値を表示できます。

詳細情報

PROC CHART
詳細なドキュメントについては、*Base SAS Procedures Guide* を参照してください。このセクションで説明された機能に加え、PROC CHART を使用して、スターチャートの作成、棒グラフの特定の値の所で参照線を引く、チャートの描画に使用される記号の変更を行うことができます。度数、合計、平均に基づいたチャートだけでなく、累積度数、パーセント、累積パーセントに基づいたチャートも作成できます。

PROC UNIVARIATE
詳細なドキュメントについては、*Base SAS Procedures Guide* を参照してください。

PROC PLOT
変数間のリレーションシップのプロット作成方法の詳細については、“変数間のリレーションシップのプロットについて” (519 ページ)を参照してください。グラフィック表現を用意するにあたり、チャートに適しているデータもあります。また、プロットにより適しているデータもあります。

SAS 出力形式
: 詳細なドキュメントについては、*SAS Formats and Informats: Reference* を参照してください。SAS では、小数値、16 進値、ローマ数字、社会保障番号、日付と時刻、および文字として記述される数値など、多くの出力形式が使用可能です。

PROC FORMAT
: 独自の出力形式の作成方法の詳細なドキュメントについては、*Base SAS Procedures Guide* を参照してください。

SAS/GRAPH ソフトウェア
: 詳細なドキュメントについては、*SAS/GRAPH: Reference* を参照してください。サイトで SAS/GRAPH を所有している場合は、GCHART プロシジャを使用して出力デバイスの高解像度グラフ機能を活用し、色、さまざまなフォントおよびテキストを含むチャートを作成できます。

TITLE ステートメントと FOOTNOTE ステートメント
: レポートのタイトルとフットノートの使用の詳細については、“タイトルとフットノートについて” (446 ページ)を参照してください。

8 部

独自の出力の設計

32 章
SAS ログや出力ファイルに行を書き込む 583

33 章
SAS 出力とそのカスタマイズについて: 基本 599

34 章
SAS 出力とそのカスタマイズについて: The Output Delivery System (ODS) 625

32 章

SAS ログや出力ファイルに行を書き込む

SAS ログや出力ファイルへの行の書き込みについて **583**
目的 583
前提条件 584
PUT ステートメントについて **584**
データセットを作成せずに出力を書き込む **584**
単純なテキストを書き込む **585**
文字列を書き込む 585
変数値を書き込む 586
同じ行へ複数回書き込む 587
固定した行を解除する 588
レポートの書き込み **589**
出力ファイルへの書き込み 589
レポートの設計 590
データ値の書き込み 591
数値データ値の外観の改善 592
各 BY グループの先頭への値の書き込み 593
合計の計算 594
1 ページのレポートのヘッダーとフットノートの書き込み 595
要約 **596**
ステートメント 596
詳細情報 **597**

SAS ログや出力ファイルへの行の書き込みについて

目的

前のセクションでは、SAS データセットにデータ値を格納する方法と、SAS プロシジャを使用してこれらのデータ値に基づいたレポートを作成する方法について学習しました。このセクションでは、次の操作を行う方法を学習します。

- 出力ファイルにデータ値と文字列を配置して出力を設計する
- DATA _NULL_ステートメントを使用してデータセットが作成されないようにする
- プロシジャのかわりに DATA ステップを使用してレポートを作成する

- FILE ステートメントを使用してデータを出力ファイルへ送る

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 1 章, “SAS System について” (3 ページ)
- 3 章, “DATA ステップ処理について” (27 ページ)

PUT ステートメントについて

DATA ステップを使用した出力の作成時、PUT ステートメントを使用して出力をカスタマイズし、SAS ログや他の出力ファイルへテキストを書き込むことができます。PUT ステートメントの形式は次のとおりです。

PUT<*variable*<*format*>> <'*character-string*'>;

variable
書き込む変数の名前を指定します。

format
変数の値の書き込み時に使用される出力形式を指定します。

'*character-string*'
書き込むテキストの文字列を指定します。文字列は引用符で囲むようにしてください。

データセットを作成せずに出力を書き込む

多くの場合、DATA ステップを使用してレポートを書き込む際には追加のデータセットを作成する必要はありません。DATA _NULL_ステートメントを使用すると、SAS は、データセットへオブザベーションを書き込まずに DATA ステップを処理します。DATA _NULL_ステートメントを使用することで、プログラムの効率性を大幅に上げられます。

次に、DATA _NULL_ステートメントの例を示します。

```
data _null_;
```

次のプログラムは、PUT ステートメントを使用して SAS ログへ女性のオリンピックメダリストの情報を書き込みます。プログラムで DATA _NULL_ステートメントが使用されているため、SAS はデータセットを作成しません。

```
data _null_;
   length medalist $ 19;
   input year 1-4  medalist $ 6-24  medal $ 26-31 country $ 33-35 result 37-41;
   put medalist country medal result year;
   datalines;
1984 Lingjuan Li         SILVER CHN 2559
1984 Jin-Ho Kim          BRONZE KOR 2555
1988 Soo-Nyung Kim       GOLD   KOR 2683
     Hee-Kyung Wang      SILVER KOR 2612
```

```
1988 Young-Sook Yun      BRONZE KOR 2593
1992 Youn-Jeong Cho      GOLD   KOR 113
1992 Soo-Nyung Kim       SILVER KOR 105
1992 Natalya Valeyeva    BRONZE URS
1996 Kyung-Wook Kim      GOLD   KOR
1996 Ying He             SILVER CHN
1996 Olena Sadovnycha    BRONZE UKR
2000 Mi-Jin Jun          GOLD   KOR 107
2000 Nam-Soon Kim        SILVER KOR 106
2000 Soo-Nyung Kim       BRONZE KOR 103
;
run;
```

次の出力は結果を示しています。

アウトプット 32.1 *SAS ログへの書き込み*

```
1    data _null_; 2       length medalist $ 19; 3       input year 1-4  medalist $ 6-24  medal
$ 26-31 country $ 33-35 result 37-41; 4       put medalist country medal result year; 5
datalines; Lingjuan Li CHN SILVER 2559 1984 Jin-Ho Kim KOR BRONZE 2555 1984 Soo-Nyung Kim KOR GOLD
2683 1988 Hee-Kyung Wang KOR SILVER 2612 .Young-Sook Yun KOR BRONZE 2593 1988 Youn-Jeong Cho KOR
GOLD 113 1992 Soo-Nyung Kim KOR SILVER 105 1992 Natalya Valeyeva URS BRONZE .1992 Kyung-Wook Kim KOR
GOLD .1996 Ying He CHN SILVER .1996 Olena Sadovnycha UKR BRONZE .1996 Mi-Jin Jun KOR GOLD 107 2000
Nam-Soon Kim KOR SILVER 106 2000 Soo-Nyung Kim KOR BRONZE 103 2000 NOTE: DATA statement used (Total
process time): real time           0.02 seconds cpu time            0.03 seconds 20   ; 21   run;
```

SAS では数値欠損値をピリオドで示します。ログでは、変数 year (年)に 1 つの欠損オブザベーション、および変数 result (結果)に 4 つの欠損オブザベーションが含まれています。

単純なテキストを書き込む

文字列を書き込む

PUT ステートメントは、その最も単純な形式では、ユーザーによって指定される文字列を、SAS ログ、プロシジャの出力ファイルまたは外部ファイルへ書き込みます。この例のように出力先が省略されると、SAS は文字列をログに書き込みます。次の例では、DATA ステップの各反復時に PUT ステートメントを 1 回ずつ実行します。変数 Year および Result に欠損値が検出された場合は、PUT ステートメントがメッセージをログへ書き込みます。

```
data _null_;
   length medalist $ 19;
   input year 1-4  medalist $ 6-24  medal $ 26-31 country $ 33-35 result 37-41;
   if year=. then put '*** Missing Year';
      else
   if result=. then put '*** Missing Results';

   datalines;
1984 Lingjuan Li         SILVER CHN 2559
1984 Jin-Ho Kim          BRONZE KOR 2555
1988 Soo-Nyung Kim       GOLD   KOR 2683
     Hee-Kyung Wang      SILVER KOR 2612
1988 Young-Sook Yun      BRONZE KOR 2593
```

```
1992 Youn-Jeong Cho      GOLD   KOR 113
1992 Soo-Nyung Kim       SILVER KOR 105
1992 Natalya Valeyeva    BRONZE URS
1996 Kyung-Wook Kim      GOLD   KOR
1996 Ying He             SILVER CHN
1996 Olena Sadovnycha    BRONZE UKR
2000 Mi-Jin Jun          GOLD   KOR 107
2000 Nam-Soon Kim        SILVER KOR 106
2000 Soo-Nyung Kim       BRONZE KOR 103
;
run;
```

次の出力は結果を示しています。

アウトプット 32.2 *SAS ログへ文字列を書き込む*

```
22   data _null_; 23      length medalist $ 19; 24      input year 1-4  medalist $ 6-24  medal
$ 26-31 country $ 33-35 result 37-41; 25      if year=. then put '*** Missing Year'; 26         else
27      if result=. then put '*** Missing Results'; 28 29      datalines; *** Missing Year ***
Missing Results *** Missing Results *** Missing Results *** Missing Results NOTE: DATA statement
used (Total process time): real time           0.00 seconds cpu time            0.01 seconds 44   ;
45   run;
```

変数値を書き込む

前の例では、Year の値はデータセット内の 1 つのオブザベーションで欠損しており、変数 Result は 4 つのオブザベーションで欠損しています。どのオブザベーションに欠損値があるか特定するには、文字列と 1 つ以上の変数の値を書き込みます。次のプログラムは、文字列だけでなく、Year と Result の値を書き込みます。

```
data _null_;
   length medalist $ 19;
   input year 1-4  medalist $ 6-24  medal $ 26-31 country $ 33-35 result 37-41;
   if year=. then put '*** Missing Year' medalist country;
      else
   if result=. then put '*** Missing Results' medalist country;
   datalines;
1984 Lingjuan Li         SILVER CHN 2559
1984 Jin-Ho Kim          BRONZE KOR 2555
1988 Soo-Nyung Kim       GOLD   KOR 2683
     Hee-Kyung Wang      SILVER KOR 2612
1988 Young-Sook Yun      BRONZE KOR 2593
1992 Youn-Jeong Cho      GOLD   KOR 113
1992 Soo-Nyung Kim       SILVER KOR 105
1992 Natalya Valeyeva    BRONZE URS
1996 Kyung-Wook Kim      GOLD   KOR
1996 Ying He             SILVER CHN
1996 Olena Sadovnycha    BRONZE UKR
2000 Mi-Jin Jun          GOLD   KOR 107
2000 Nam-Soon Kim        SILVER KOR 106
2000 Soo-Nyung Kim       BRONZE KOR 103
;
run;
```

各文字列の最後の文字はブランクであることに注意してください。これはリスト出力の例です。リスト出力では、SAS は、変数値を書き込んだ後で自動的に 1 つ右のカラム

へ移動します(文字列を書き込んだ後は当てはまりません)。必要なスペースを含める最も単純な方法は、スペースを文字列に含めることです。

SAS は、ポインタを使用してその出力行での位置を追跡しています。この PUT ステートメントの処理の他の説明として、リスト出力では、変数値を書き込んだ後でポインタが 1 つ右のカラムへ移動します(文字列を書き込んだ後は当てはまりません)。このセクションの後半部分では、ポインタを移動して、次のテキスト部分を書き込む場所を制御する方法を学習します。

次の出力は結果を示しています。

アウトプット 32.3 *文字列および変数値を書き込む*

```
46   data _null_; 47      length medalist $ 19; 48      input year 1-4  medalist $ 6-24  medal
$ 26-31 country $ 33-35 result 37-41; 49      if year=. then put '*** Missing Year' medalist
country; 50         else 51      if result=. then put '*** Missing Results' medalist country; 52 53
54      datalines; *** Missing YearHee-Kyung Wang KOR *** Missing ResultsNatalya Valeyeva URS ***
Missing ResultsKyung-Wook Kim KOR *** Missing ResultsYing He CHN *** Missing ResultsOlena Sadovnycha
UKR NOTE: DATA statement used (Total process time): real time           0.00 seconds cpu
time            0.01 seconds 69   ; 70   run;
```

同じ行へ複数回書き込む

デフォルトでは、PUT ステートメントごとに新しい行が開始されます。ただし、複数の PUT ステートメントと 1 つ以上の後置@(アットマーク)を使用すると、同じ行に書き込むことができます。

後置@は、ポインタコントロールの 1 種で、ラインホールド指定子と呼ばれます。ポインタコントロールは SAS がテキストを書き込む場所を指定する 1 つの方法です。次の例では、後置@を使用することで、2 番目の PUT ステートメントの項目を新しい行ではなく同じ行に書き込んでいます。各 PUT ステートメントに後置@があるため、いずれかの PUT ステートメントが実行されると、出力行が更なる書き込みのために固定されます。SAS は、その DATA ステップの同一反復内で以後の PUT ステートメントが実行されるとき、およびその後の反復で PUT ステートメントが実行されるときに、その行に書き込み続けます。

```
data _null_;
   length medalist $ 19;
   input year 1-4  medalist $ 6-24  medal $ 26-31 country $ 33-35 result 37-41;
   if year=. then put '*** Missing Year' medalist country @;
      else
   if result=. then put '*** Missing Results' medalist country @;
   datalines;
1984 Lingjuan Li         SILVER CHN 2559
1984 Jin-Ho Kim          BRONZE KOR 2555
1988 Soo-Nyung Kim       GOLD   KOR 2683
     Hee-Kyung Wang      SILVER KOR 2612
1988 Young-Sook Yun      BRONZE KOR 2593
1992 Youn-Jeong Cho      GOLD   KOR 113
1992 Soo-Nyung Kim       SILVER KOR 105
1992 Natalya Valeyeva    BRONZE URS
1996 Kyung-Wook Kim      GOLD   KOR
1996 Ying He             SILVER CHN
1996 Olena Sadovnycha    BRONZE UKR
2000 Mi-Jin Jun          GOLD   KOR 107
2000 Nam-Soon Kim        SILVER KOR 106
2000 Soo-Nyung Kim       BRONZE KOR 103
```

```
;
run;
```

次の出力は結果を示しています。

アウトプット 32.4 *同じ行へ複数回書き込む*

```
71   data _null_; 72      length medalist $ 19; 73      input year 1-4  medalist $ 6-24  medal
$ 26-31 country $ 33-35 result 37-41; 74      if year=. then put '*** Missing Year' medalist country
@; 75         else 76      if result=. then put '*** Missing Results' medalist country @; 77 78
79      datalines; *** Missing YearHee-Kyung Wang KOR *** Missing ResultsNatalya Valeyeva URS ***
Missing Results Kyung-Wook Kim KOR *** Missing ResultsYing He CHN *** Missing ResultsOlena
Sadovnycha UKR NOTE: DATA statement used (Total process time): real time           0.00 seconds cpu
time            0.01 seconds 94   ; 95   run;
```

出力行の長さが十分にあった場合は、欠損データに関する 3 つのメッセージすべてが単一行に書き込まれます。行の長さが足りない場合は、次の行へ書き込みが続けられます。個別のデータ値または文字列が行に収まらないと判断されると、その項目全体が次の行へ送られます。データ値または文字列が分割されることはありません。

固定した行を解除する

次の例では、入力ファイルに 6 つの欠損値があります。1 つのレコードでは、Year 変数と Result 変数両方の値が欠損しています。それ以外の 4 レコードでは、Year 変数または Result 変数のいずれかの値が欠損しています。

レポートを見やすくするため、すべての欠損変数をオブザベーションごとに別の行で書き込めます。Year と Result の 2 つの変数の値が欠損している場合、2 つの PUT ステートメントが同一行に書き込みます。Year または Result のいずれかの値が欠損している場合、1 つの PUT ステートメントのみがその行に書き込みます。

SAS では出力を書き込む場所を、PUT ステートメントでの後置@の存在、および行の固定を解除する空の PUT ステートメントの存在によって決定します。後置@のある PUT ステートメントを実行すると、SAS はそれ以降の書き込みを現在の出力行に固定します。DATA ステップの現在の反復あるいはそれ以降の反復のいずれかで行が固定されます。後置@のない PUT ステートメントを実行すると、行の固定が解除されます。

メッセージを書き込まずに行の固定を解除するには、空の PUT ステートメントを使用します。

```
put;
```

空の PUT ステートメントは、それ以外の PUT ステートメントと同じ特性を持ちます。デフォルトでは、新しい行に出力を書き込み、そのステートメントに指定されていること(このケースでは何もない)を書き込み、実行が終了したらその行を解放します。後置@が有効な場合、空の PUT ステートメントは現在の行で開始して、何も書き込まずに行を解除します。

次のプログラムでは、同一行に 1 つ以上の項目を書き込む方法を示します。

- 変数 Year の値が欠損している場合、変数 Result でそのオブザベーションの値が欠損している場合に備えて最初の PUT ステートメントが行を固定します。
- 変数 Result の値が欠損している場合、2 番目の PUT ステートメントはメッセージを書き込んでその行の固定を解除します。
- 変数 Result の値は欠損していないが、変数 Year に関するメッセージが書き込まれた場合(year=.)は、空の PUT ステートメントでその行の固定が解除されます。

- 変数 Year と変数 Result のどちらの値も欠損していない場合、行の固定は解除されず、いずれの PUT ステートメントも実行されません。

```
data _null_;
   length medalist $ 19;
   input year 1-4  medalist $ 6-24  medal $ 26-31 country $ 33-35 result 37-41;
   if year=. then put '*** Missing Year' medalist country @;
   if result=. then put '*** Missing Results' medalist country ;
      else if year=. then put;

   datalines;
1984 Lingjuan Li         SILVER CHN 2559
1984 Jin-Ho Kim          BRONZE KOR 2555
1988 Soo-Nyung Kim       GOLD   KOR 2683
     Hee-Kyung Wang      SILVER KOR 2612
1988 Young-Sook Yun      BRONZE KOR 2593
1992 Youn-Jeong Cho      GOLD   KOR 113
1992 Soo-Nyung Kim       SILVER KOR 105
     Natalya Valeyeva    BRONZE URS
1996 Kyung-Wook Kim      GOLD   KOR
1996 Ying He             SILVER CHN
1996 Olena Sadovnycha    BRONZE UKR
2000 Mi-Jin Jun          GOLD   KOR 107
2000 Nam-Soon Kim        SILVER KOR 106
2000 Soo-Nyung Kim       BRONZE KOR 103
;
run;
```

次の出力は結果を示しています。

アウトプット 32.5 *行への 1 回以上の書き込みと行の解除*

```
1    data _null_; 2       length medalist $ 19; 3       input year 1-4  medalist $ 6-24  medal
$ 26-31 country $ 33-35 result 37-41; 4       if year=. then put '*** Missing Year' medalist country
@; 5       if result=. then put '*** Missing Results' medalist country ; 6          else if year=.
then put; 7 8       datalines; *** Missing YearHee-Kyung Wang KOR *** Missing YearNatalya Valeyeva
URS *** Missing ResultsNatalya Valeyeva URS *** Missing ResultsKyung-Wook Kim KOR *** Missing
ResultsYing He CHN *** Missing ResultsOlena Sadovnycha UKR NOTE: DATA statement used (Total process
time): real time           0.46 seconds cpu time            0.03 seconds 23   ; 24   run;
```

レポートの書き込み

出力ファイルへの書き込み

PUT ステートメントは SAS ログへテキスト行を書き込みます。ただし、SAS ログにはプログラムのソースステートメントと SAS からのメッセージも含まれるため、通常、正式なレポートに適した出力先ではありません。

出力レポートの最も単純な出力先は、SAS 出力ファイルであり、これは SAS がプロシジャからの出力を書き込むのと同じ場所です。SAS では自動的にプロシジャ出力ファイルのページ番号などさまざまな設定が定義され、ユーザーはすべての設定を定義するかわりにこれらを活用できます。

行をプロシジャ出力ファイルに送るには、FILE ステートメントを使用します。FILE ステートメントの形式は次のとおりです。

FILE PRINT <*options*>;

PRINT は予約ファイル参照名で、PUT ステートメントが生成する出力を、SAS プロシジャによって生成される出力と同じプリントファイルに出力します。

注: プログラムコード内では、FILE ステートメントは PUT ステートメントより先行することに注意します。

FILE ステートメントの *options* には、出力をカスタマイズするために使用可能なオプションを指定します。このセクションで生成するレポートでは、次のオプションを使用します。

NOTITLES
デフォルトのタイトル行を削除し、その行を書き込み可能にします。デフォルトでは、プロシジャの出力ファイルにはタイトル“The SAS System”が含まれます。レポートで別の説明的なタイトルが作成されるため、NOTITLES オプションを指定してデフォルトタイトルを削除できます。

FOOTNOTES
現在定義されているフットノートをレポートに書き込むかどうか制御します。

注: レポートにフットノートを含めるために FILE ステートメントを使用する場合は、FILE ステートメントに FOOTNOTES オプションを使用し、プログラムには FOOTNOTE ステートメントを含む必要があります。FOOTNOTE ステートメントには、フットノートのテキストを含めます。

注: 空の TITLE ステートメントを使用して、デフォルトのタイトルを削除することもできます: `title;`。この場合、SAS はデフォルトタイトルのかわりに日付とページ番号のみを含む行を書き込みます。その行にそれ以外のテキストを書き込むことはできません。

レポートの設計

レポートの出力先を選択後、レポート作成の次のステップは外観を決めることです。レイアウトを作成し、テキストを配置する行とカラムを決定します。最終レポートの外観を計画することで、レポートの作成に必要な PUT ステートメントを記述できます。このセクションの残りの例では、プログラムを変更して、次に示すような最終レポートを作成する方法を示します。

アウトプット 32.6 新聞の朝刊と夕刊の発行部数レポート

Morning and Evening Newspaper Circulation

State	Year	Thousands of Copies Morning	Evening
Massachusetts	1999	798.4	984.7
	1998	834.2	793.6
	1997	750.3	.
	Total for each category	2382.9	1778.3
	Combined total	4161.2	
Alabama	1999	.	698.4
	1998	463.8	522.0
	1997	583.2	234.9
	1996	.	339.6
	Total for each category	1047.0	1794.9
	Combined total	2841.9	

Preliminary Report

データ値の書き込み

レポートを設計した後、レポートを作成するプログラムの作成を開始できます。次のプログラムに、YEAR、MORNING_COPIES、および EVENING_COPIES の各変数のデータ値を特定の位置に表示する方法を示します。

PUT ステートメントでは、数字が後に続く@はポインタコントロールですが、前述の後置@とは異なります。*@n* 引数はカラムポインタコントロールです。これは SAS にカラム *n* へ移動するよう指示します。この例では、ポインタが指定の場所へ移動し、PUT ステートメントにより、それらのポイントに値が書き込まれます。ポインタコントロールの使用は、単純ですが、カラムにデータ値を書き込む便利な方法です。

```
title;
data _null_;
   input state $ morning_copies evening_copies year;
   file print notitles;
   put @26 year @53 morning_copies @66 evening_copies;
   datalines;
Massachusetts 798.4 984.7 1999
Massachusetts 834.2 793.6 1998
Massachusetts 750.3 .     1997
Alabama       .     698.4 1999
Alabama       463.8 522.0 1998
Alabama       583.2 234.9 1997
Alabama       .     339.6 1996
;
run;
```

次の出力は結果を示しています。

アウトプット 32.7 出力の指定の場所に配置されたデータ値

1999	798.4	984.7
1998	834.2	793.6
1997	750.3	.
1999	.	698.4
1998	463.8	522
1997	583.2	234.9
1996	.	339.6

数値データ値の外観の改善

このレポートの設計では、すべての数値が小数点で整列されています。この結果を出力するには、SAS 出力形式を使用して数値データ値の外観を変更する必要があります。入力データでは、小数値が 0 である 1 つの場合をのぞき、MORNING_COPIES と EVENING_COPIES のすべての値に 1 つの小数点以下桁数が含まれます。リスト出力では、SAS は、値の小数部の 0 を省略するという最も単純な方法で値を書き込みます。フォーマット出力では、PUT ステートメントで出力形式と変数を関連付けることで、各値につき小数点以下桁数を 1 桁表示できます。出力形式を使用すると出力値を整列することもできます。

このプログラムで使用する出力形式は、*w.d* 出力形式と呼ばれています。*w.d* 出力形式により、小数点以下桁数を含む、値全体の書き込みに使用されるカラムの数が指定されます。各値の小数部の書き込みに使用されるカラムの数も指定されます。この例では、出力形式 5.1 により小数点以下 1 桁を含めて 5 カラムを各値の書き込みに使用するよう SAS に指示されます。そのため、必要に応じて小数部の 0 が出力されます。また、出力形式によって、欠損値を示すために SAS が使用するピリオドも小数点に揃えて配置されます。

```
title;
data _null_;
   input state $ morning_copies evening_copies year;
   file print notitles;
   put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
   datalines;
Massachusetts 798.4 984.7 1999
Massachusetts 834.2 793.6 1998
Massachusetts 750.3 .     1997
Alabama       .     698.4 1999
Alabama       463.8 522.0 1998
Alabama       583.2 234.9 1997
Alabama       .     339.6 1996
;
run;
```

次の出力は結果を示しています。

アウトプット 32.8 フォーマットされた数値出力

1999	798.4	984.7
1998	834.2	793.6
1997	750.3	.
1999	.	698.4
1998	463.8	522.0
1997	583.2	234.9
1996	.	339.6

各 BY グループの先頭への値の書き込み

レポート作成の次のステップは、州の名前を出力に追加することです。PUT ステートメントにその他のデータ値とともに州の名前を含めると、州が各行に表示されます。ただし、最終レポートの目的の外観に基づくと、特定の州の最初のオブザベーションについてのみ州の名前を書き込む必要があります。オブザベーションのグループに対してタスクを 1 回実行するには、BY グループ処理のため BY ステートメントの使用が必要です。BY ステートメントの形式は次のとおりです。

BY *by-variable(s)*<NOTSORTED>;

by-variable では、データセットを並べ替える基準となる変数の名前を指定します。オプションの NOTSORTED オプションで、同じ BY 値を持つオブザベーションが、グループ化されてはいるものの、必ずしもアルファベット順または数値順には並べ替えられていないことを示します。

BY グループ処理の場合は、次の操作を行います。

- SAS データセットからのオブザベーションであって、外部ファイルからのものではないことを確認します。
- データが BY グループでグループ化されてはいるものの、必ずしもアルファベット順には並べ替えられていない場合、BY ステートメントで NOTSORTED オプションを使用します。使用例を次に示します。

```
by state notsorted;
```

次のプログラムは、NEWS.CIRCULATION という名前の永久 SAS データセットを作成し、レポートの各 BY グループの最初の行に州の名前を書き込みます。

```
title;
libname news 'SAS-data-library';
data news.circulation;
   length state $ 15;
   input state $ morning_copies evening_copies year;
   datalines;
Massachusetts 798.4 984.7 1999
Massachusetts 834.2 793.6 1998
Massachusetts 750.3 .     1997
Alabama       .     698.4 1999
Alabama       463.8 522.0 1998
Alabama       583.2 234.9 1997
Alabama       .     339.6 1996
;

data _null_;
   set news.circulation;
   by state notsorted;
   file print notitles;
   if first.state then put / @7 state @;
   put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
run;
```

特定の州の最初のオブザベーションでは、PUT ステートメントは州の名前を書き込み、更なる書き込み(年と発行部数)のために行を固定します。次の PUT ステートメントは、年と発行部数を書き込み、固定された行を解除します。最初より後のオブザベーションでは、2 番目の PUT ステートメントのみが処理されます。これは通常どおり、年と発行部数を書き込んで行を解除します。

最初の PUT ステートメントには、ポインタを次の行の先頭に移動するポインタコントロールである、スラッシュ(/)が含まれます。この例では、PUT ステートメントは新しい行への書き込みの準備をします(標準アクション)。その後、スラッシュによりポインタが次の行の先頭に移動されます。その結果、STATE の値を書き込む前に行がスキップされます。出力では、ブランク行によって Massachusetts のデータと Alabama のデータが分けられます。Massachusetts の出力も、ページのさらに 1 行下で開始します。(このブランク行は、後のレポートの拡張で使用します。)

次の出力は結果を示しています。

アウトプット 32.9 *BY グループ処理の効果*

Massachusetts	1999	798.4	984.7
	1998	834.2	793.6
	1997	750.3	.
Alabama	1999	.	698.4
	1998	463.8	522.0
	1997	583.2	234.9
	1996	.	339.6

合計の計算

次のステップは、各州の朝刊発行部数の合計、夕刊発行部数の合計、全体の発行部数の合計を計算することです。合計ステートメントにより合計が集計され、割り当てステートメントにより各州について 0 から集計が開始されます。特定の州の最後のオブザベーションが処理されているとき、割り当てステートメントにより全体の合計が計算され、PUT ステートメントにより合計と追加の説明テキストが書き込まれます。

```
libname news 'SAS-data-library';
title;
data _null_;
   set news.circulation;
   by state notsorted;
   file print notitles;
      /* Set values of accumulator variables to 0 */
      /* at beginning of each BY group.           */
      if first.state then
         do;
            morning_total=0;
            evening_total=0;
            put / @7 state @;
         end;
   put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;

      /* Accumulate separate totals for morning and */
      /* evening circulations.                      */
   morning_total+morning_copies;
   evening_total+evening_copies;

      /* Calculate total circulation at the end of  */
      /* each BY group.                             */

   if last.state then
      do;
         all_totals=morning_total+evening_total;
```

```
            put @52 '------' @65 '------' /
                @26 'Total for each category'
                @52 morning_total 6.1 @65 evening_total 6.1 /
                @35 'Combined total' @59 all_totals 6.1;
        end;
run;
```

次の出力は結果を示しています。

アウトプット 32.10 *各 BY グループの合計の計算と書き込み*

```
Massachusetts      1999                      798.4        984.7
                   1998                      834.2        793.6
                   1997                      750.3            .
                                            ------       ------
                   Total for each category  2382.9       1778.3
                            Combined total          4161.2

Alabama            1999                          .        698.4
                   1998                      463.8        522.0
                   1997                      583.2        234.9
                   1996                          .        339.6
                                            ------       ------
                   Total for each category  1047.0       1794.9
                            Combined total          2841.9
```

合計ステートメントは、合計を集計する際に欠損値を無視することに注意してください。また、デフォルトでは、合計ステートメントはアキュムレータ変数(この場合、MORNING_TOTAL と EVENING_TOTAL)に 0 の初期値を割り当てます。そのため、DO グループの割り当てステートメントは両方の州の最初のオブザベーションについて実行されますが、2 番目の州についてのみ必要です。

1 ページのレポートのヘッダーとフットノートの書き込み

レポートは、タイトル行、列ヘッダーおよびフットノートを除いて完成しました。これは単純な 1 ページのレポートなため、DATA ステップの最初の反復時にのみ実行される PUT ステートメントを使用してヘッダーを書き込むことができます。自動変数_N_は、DATA ステップの反復回数またはループ回数をカウントし、_N_の値が 1 のときに、PUT ステートメントが実行されます。

FILE ステートメントの FOOTNOTES オプションと FOOTNOTE ステートメントによりフットノートが作成されます。次のプログラムは完成版です。

```
libname news 'SAS-data-library';
title;
data _null_;
   set news.circulation;
   by state notsorted;
   file print notitles footnotes;
   if _n_=1 then put @16 'Morning and Evening Newspaper Circulation' //
                     @7  'State' @26 'Year' @51 'Thousands of Copies' /
                     @51 'Morning      Evening';
   if first.state then
      do;
         morning_total=0;
         evening_total=0;
         put / @7 state @;
      end;
```

```
   put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
   morning_total+morning_copies;
   evening_total+evening_copies;
   if last.state then
      do;
         all_totals=morning_total+evening_total;
         put @52 '------' @65 '------' /
             @26 'Total for each category'
             @52 morning_total 6.1 @65 evening_total 6.1 /
             @35 'Combined total' @59 all_totals 6.1;
      end;
   footnote 'Preliminary Report';
run;
```

次の出力は結果を示しています。

アウトプット 32.11 *最終レポート*

Morning and Evening Newspaper Circulation

State	Year	Thousands of Copies Morning	Evening
Massachusetts	1999	798.4	984.7
	1998	834.2	793.6
	1997	750.3	.
	Total for each category	2382.9	1778.3
	Combined total	4161.2	
Alabama	1999	.	698.4
	1998	463.8	522.0
	1997	583.2	234.9
	1996	.	339.6
	Total for each category	1047.0	1794.9
	Combined total	2841.9	

Preliminary Report

ヘッダー用の PUT ステートメントはブランク行を書き込みませんが、ヘッダーの最後の行と Massachusetts の最初のデータとの間にブランク行があることに注意してください。この行は、各 BY グループの最初のオブザベーションの STATE の値を書き込む PUT ステートメントのスラッシュ(/)によるものです。

DATA ステップの最初の反復時に PUT ステートメントを実行すると、特にレポートの長さが 1 ページのみの場合、ヘッダーを簡単に作成できます。

要約

ステートメント

BY *variable-1* <... *variable-n* > <NOTSORTED>;

BY 変数の共通の値を持つすべてのオブザベーションが一緒にグループ化されることを示します。NOTSORTED オプションは、それらの変数はグループ化されます

がそれらのグループが必ずしもアルファベット順または数値順ではないことを示します。

DATA _NULL_;
SAS が出力データセットを作成しないよう指定します。

FILE PRINT <NOTITLES> <FOOTNOTES>;
SAS プロシジャ出力ファイルへ出力するよう指定します。そのファイルへの書き込みを行う PUT ステートメントより前に FILE ステートメントを配置します。NOTITLES オプションは現在有効であるタイトルを非表示にし、その行へのその他のテキストの書き込みをできないようにします。FOOTNOTES オプションは FOOTNOTE ステートメントとともに使用され、ファイルへフットノートを書き込みます。

PUT;
デフォルトでは、新しい行を開始し、以前に固定されていた行を解除します。テキストを何も書き込まない PUT ステートメントを空の PUT ステートメントといいます。

PUT <*variable* <*format*>> <*character string*>;
行を FILE ステートメントで指定した出力先に書き込みます。FILE ステートメントが存在しない場合、PUT ステートメントが SAS ログへ書き込みます。デフォルトでは、各 PUT ステートメントは新しい行を開始して指定の項目を書き込み、行を解除します。DATA ステップには任意の数の PUT ステートメントを含めることができます。

デフォルトでは、SAS は変数または文字列をその行で現在の位置に書き込みます。SAS は、変数値を書き込んだ後は 1 カラム右にポインタを自動的に移動しますが、文字列を書き込んだ後では移動しません。つまり、SAS は変数値の後ろには 1 つブランクを空け、文字列ではブランクを空けません。この出力の形式はリスト出力と呼ばれます。変数名(variable)の後ろに出力形式を指定した場合、SAS はその行での現在の位置から変数の値を書き込みます。また、SAS はユーザーが指定した出力形式も使用します。フォーマットされた値の後のポインタは次のカラムにあります。つまり、SAS は自動的にカラムをスキップすることはありません。PUT ステートメントでの出力形式の使用は、フォーマット出力と呼ばれます。単一の PUT ステートメントにリストとフォーマット出力を組み合わせることができます。

PUT<*@n*> <*variable* <*format*>> <*character-string*> </> <*@*>;
行を FILE ステートメントで指定した出力先に書き込みます。FILE ステートメントが存在しない場合、PUT ステートメントは SAS ログへ書き込みます。@n ポインタコントロールは、ポインタを現在の行のカラム n へ移動します。スラッシュ(/)はポインタを新しい行の先頭へ移動します。(スラッシュを PUT ステートメントの任意の場所で使用して行をスキップできます。)複数のスラッシュは複数の行をスキップします。後置@が存在する場合、PUT ステートメント内の最後の項目にする必要があります。後置@がある PUT ステートメントを実行すると、DATA ステップの同一反復内または後の反復のいずれかにおける、後の PUT ステートメントでの使用ために現在の行が固定されます。後置@のない PUT ステートメントを実行すると、行の固定が解除されます。

TITLE;
SAS 出力のタイトル行を指定します。

詳細情報

ポインタコントロール
ポインタコントロールの詳細については、*SAS Statements: Reference* の PUT ステートメントを参照してください。

ステートメント
このセクションで説明したステートメントの詳細については、*SAS Statements: Reference* を参照してください。

ODS
PUT ステートメントや DATA ステップでの ODS の使用については、*SAS Output Delivery System: User's Guide* を参照してください。

33 章

SAS 出力とそのカスタマイズについて: 基本

SAS 出力とそのカスタマイズの基礎知識について **600**
目的 600
前提条件 600

出力について **600**
プロシジャからの出力 600
DATA ステップアプリケーションからの出力 601
Output Delivery System (ODS)の出力 602

例で使用される入力 SAS データセット **602**

プロシジャ出力の出力先 **603**

出力に情報を付加する **604**
タイトルの追加 604
フットノートの追加 606
変数のラベル付け 608
説明的な出力の作成 609

リスト出力での出力表示の制御 **610**
SAS システムオプションの指定 610
ページの番号付け 611
出力の中央揃え 611
ページと行サイズの指定 611
日付値と時間値の書き込み 612
オプションの選択 612

ページの表示形式の調整 **613**
複数ページのレポートの例で使用される入力データセット 613
中央揃えのタイトルと列ヘッダーの書き込み 614
特定のカラムにタイトルと列ヘッダーを書き込む 616
ヘッダーの部分の変更 617
ページ分割の制御 619

欠損値の表示 **621**
デフォルト値について 621
システムオプションを使用した欠損値の出力のカスタマイズ 621
プロシジャを使用した欠損値の出力のカスタマイズ 622

要約 **623**
ステートメント 623
SAS システムオプション 624

詳細情報 **624**

SAS 出力とそのカスタマイズの基礎知識について

目的

このセクションでは、出力の外観を向上させてより有益なものにすることができるよう、出力について学習します。DATA ステップと PROC ステップの出力について説明します。

このセクションでは、次の操作を行い出力の外観を向上させる方法について説明します。

- タイトル、列ヘッダー、フットノート、およびラベルを追加する
- ヘッダーをカスタマイズする
- ヘッダーの部分を変更する
- ページの番号付けとページ分割の制御を行う
- 日付値と時間値を出力する
- 数値欠損値を文字で表す

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 3 章, “DATA ステップ処理について” (27 ページ)
- 32 章, “SAS ログや出力ファイルに行を書き込む” (583 ページ)

出力について

プロシジャからの出力

SAS プロシジャが呼び出されると、SAS によりデータが分析または処理されます。SAS データセットの読み込み、統計量の計算、結果の出力、新しいデータセットの作成を行えます。SAS プロシジャの実行結果の 1 つに、プロシジャ出力の作成があります。プロシジャ出力の場所は、SAS を実行している方法、動作環境、使用するオプションによって異なります。出力の形式とコンテンツは各プロシジャによって異なります。SORT プロシジャなど一部のプロシジャは、出力を生成しません。

SAS には数多くのプロシジャが用意されており、データの処理に使用できます。たとえば、PRINT プロシジャを使用して、SAS データセット内の各変数の値をリストにするレポートを出力できます。MEANS プロシジャを使用して、すべてのオブザベーション間やオブザベーショングループ内での変数の記述統計量を計算できます。UNIVARIATE プロシジャを使用して数値変数の分布に関する情報を生成できます。データのグラフィック表現には、CHART プロシジャを使用できます。その他多くのプロシジャを SAS を介して利用できます。

DATA ステップアプリケーションからの出力

出力は、通常、プロシジャによって生成されますが、DATA ステップアプリケーションを使用して出力を生成することもできます。DATA ステップを使用すると、次の操作を行えます。

- SAS データセットを作成する
- 外部ファイルへ書き込む
- レポートを作成する

出力を生成するには、DATA ステップ内で FILE ステートメントと PUT ステートメントを一緒に使用します。現在の出力ファイルを特定するには、FILE ステートメントを使用します。その後、PUT ステートメントを使用して、変数の値またはテキスト文字列を含む行を出力ファイルに書き込みます。値はカラム、リスト、またはフォーマットされたスタイルで書き込みできます。

FILE ステートメントと PUT ステートメントを使用して、データのサブセットをターゲットにできます。不要な情報を含む大規模データセットがある場合、この種類の DATA ステッププロセスによって時間とコンピュータリソースを節約できます。DATA ステップの現在の実行で FILE ステートメントが PUT ステートメントより前に実行されるように、コードを記述してください。そうしない場合、データは SAS ログに書き込まれます。

SAS データセットの場合、FILE ステートメントと PUT ステートメントを使用して、別のコンピュータ言語で処理可能な外部ファイルを作成できます。たとえば、高等学校生徒の試験スコアをリストにする SAS データセットを作成できます。その後、試験スコアを分析する FORTRAN プログラムへの入力としてそのファイルを使用できます。既存の FORTRAN プログラムで検索時に予期している、入力 SAS データセットの変数とカラム位置を次のテーブルにリストで示します。

表 33.1 変数とカラム位置

変数	カラム位置
YEAR	10-13
TEST	15-25
GENDER	30
SCORE	35-37

DATA ステップで FILE ステートメントと PUT ステートメントを使用して、FORTRAN プログラムで読み取り可能なデータセットを作成できます。

```
data _null_;
   set out.sats1;
   file 'your-output-file';
   put @10 year @15 test
       @30 gender @35 score;
run;
```

Output Delivery System (ODS)の出力

バージョン 7 より、Output Delivery System (ODS)によってプロシジャ出力がより一段と柔軟になりました。ODS は、さまざまな出力形式で出力を配信し、これらのフォーマットされた出力へのアクセスを簡単にする方法です。ODS には、次の重要な機能があります。

- ODS は、生データと 1 つ以上のテーブル定義を組み合わせて 1 つ以上の出力オブジェクトを生成します。これらのオブジェクトをいずれかの ODS 出力先に送信すると、テーブル定義の指示に従って出力がフォーマットされます。ODS 出力先では、出力データセット、従来のモノスペースの出力、高解像度プリンタ用にフォーマットされた出力、ハイパーテキストマークアップ言語(HTML)でフォーマットされた出力などを生成できます。
- ODS には、DATA ステップおよびプロシジャからの出力の構造を定義するテーブル定義が用意されています。これらの定義を変更、または独自の定義を作成することで出力をカスタマイズできます。
- ODS では、ODS 出力先へ送信する個別の出力オブジェクトを選択できます。たとえば、PROC UNIVARIATE は 5 つの出力オブジェクトを生成します。これらの出力オブジェクトのいずれかから、HTML 出力、出力データセット、従来のリスト出力、またはプリンタ出力を簡単に作成できます。出力先別に異なる出力オブジェクトを送信できます。
- ODS は、**結果**ウィンドウの結果フォルダに各出力オブジェクトへのリンクを保存します。

さらに、ODS を使用することにより、個別のプロシジャおよび DATA ステップから出力をフォーマットする必要がなくなります。プロシジャまたは DATA ステップは、生データと、フォーマット命令を含むテーブル定義の名前を提供します。それで、ODS が出力をフォーマットします。フォーマットは ODS に一元化されるようになったため、新しい ODS 出力先を追加しても、いずれのプロシジャや DATA ステップにも影響しません。新しい出力先が ODS に追加されると、それらは DATA ステップと ODS をサポートするすべてのプロシジャで自動的に利用可能になります。

詳細については、34 章, “SAS 出力とそのカスタマイズについて: The Output Delivery System (ODS)” (625 ページ)を参照してください。

例で使用される入力 SAS データセット

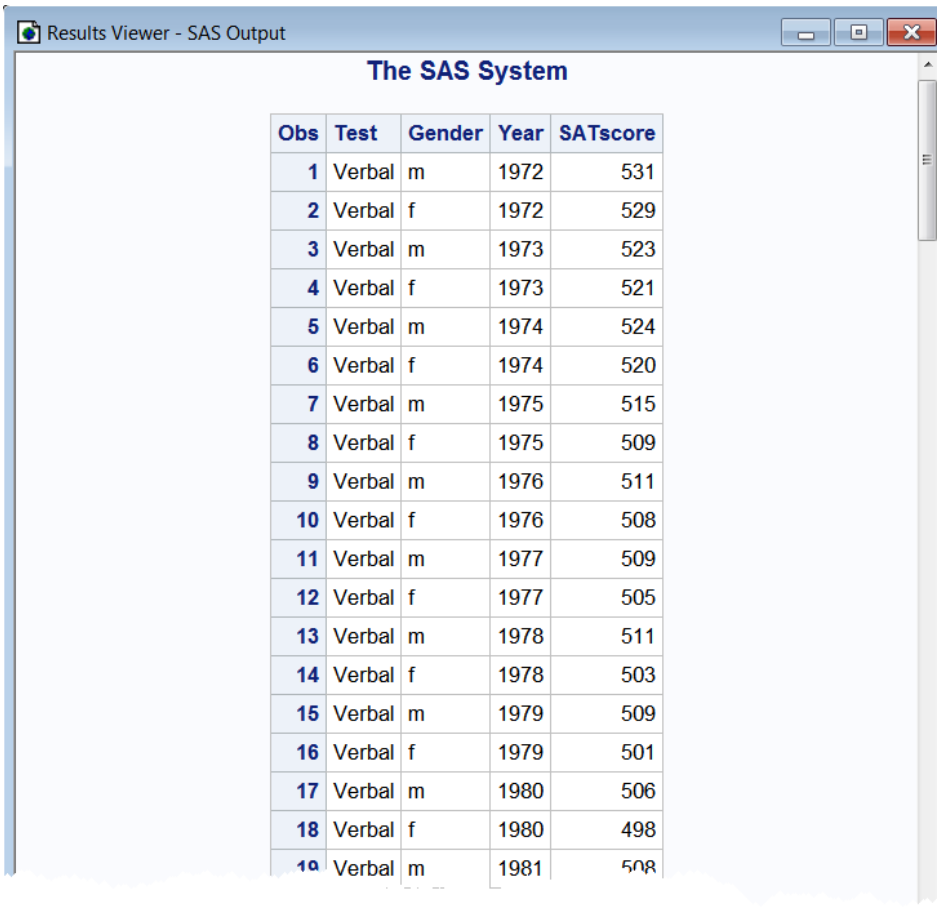
次のプログラムは、1972-1998 年の大学進学予定の高校最上級生の Scholastic Aptitude Test (SAT: 大学進学適性試験)情報を含む SAS データセットを作成します。(DATA ステップ全体を表示するには、“UNIVERSITY_TEST_SCORES データセット” (793 ページ)を参照してください。)この例のデータセットは、ライブラリ参照名 ADMIN で参照される SAS ライブラリに格納されています。データセットは、1972-1998 の選択された年について、試験を受ける全国の生徒の総数に基づいて推定スコアを表示します。スコアは、男性(m)と女性(f)の生徒の、試験の語学と数学の両方の部分について推定されます。

```
options pagesize=60 linesize=80 pageno=1 nodate;
libname admin 'your-data-library';

data admin.sat_scores;
   input Test $ Gender $ Year SATscore @@;
   datalines;
```

```
Verbal m 1972 531  Verbal f 1972 529
Verbal m 1973 523  Verbal f 1973 521
Verbal m 1974 524  Verbal f 1974 520
   ...more SAS data lines...
Math   m 1996 527  Math   f 1996 492
Math   m 1997 530  Math   f 1997 494
Math   m 1998 531  Math   f 1998 496
;

proc print data=admin.sat_scores;
run;
```

次の出力は結果のリストの一部を示しています。

アウトプット 33.1 *ADMIN.SAT_SCORES データセット: HTML 出力の一部*


Results Viewer - SAS Output

The SAS System

Obs	Test	Gender	Year	SATscore
1	Verbal	m	1972	531
2	Verbal	f	1972	529
3	Verbal	m	1973	523
4	Verbal	f	1973	521
5	Verbal	m	1974	524
6	Verbal	f	1974	520
7	Verbal	m	1975	515
8	Verbal	f	1975	509
9	Verbal	m	1976	511
10	Verbal	f	1976	508
11	Verbal	m	1977	509
12	Verbal	f	1977	505
13	Verbal	m	1978	511
14	Verbal	f	1978	503
15	Verbal	m	1979	509
16	Verbal	f	1979	501
17	Verbal	m	1980	506
18	Verbal	f	1980	498
19	Verbal	m	1981	508

プロシジャ出力の出力先

プロシジャ出力の場所は、SAS の起動、実行、終了に使用している方法によって異なります。使用している動作環境および SAS システムオプションの設定によっても異なります。次の表に、動作方法ごとのデフォルトの場所を示します。

表 33.2 プロシジャ出力のデフォルトの場所

動作方法	プロシジャ出力の場所
ウィンドウ環境	**結果ビューア**と**結果**ウィンドウ
対話型ラインモード	端末ディスプレイ上、各ステップの実行ごと
非対話型 SAS プログラム	動作環境によって異なる
バッチジョブ	ラインプリンタまたはディスクファイル

デフォルトでは、ユーザーの Work ディレクトリに SAS が出力を格納します。Windows および UNIX の SAS ウィンドウ環境では、HTML 出力先の開閉後に、現在の作業ディレクトリへ出力されます。SAS セッション中にはいつでも、ODS PREFERENCE ステートメントを使用してデフォルトの動作に戻すことができます。このアクションは、複数のグラフを作成しているけれど、それらを現在の作業ディレクトリには集めたくない場合に役立ちます。

出力に情報を付加する

タイトルの追加

SAS では、出力の各ページの上部に、自動的に次のタイトルが書き込まれます。

```
The SAS System
```

TITLE ステートメントを使用して独自のタイトルを指定し、出力に情報を付加できます。TITLE ステートメントは、指定のタイトルを各ページの上部に書き込みます。次に TITLE ステーメントの形式を示します。

TITLE<*n*> <'*text*'>;

ここで、*n* はタイトルを含める相対行を指定し、*text* はタイトルのテキストを指定します。*n* の値は 1-10 の間です。*n* の指定を省略すると、この値は 1 と見なされます。そのため、最初のタイトル行には TITLE または TITLE1 を指定できます。デフォルトでは、タイトルは中央揃えにされます。

タイトル'SAT Scores by Year, 1972-1998'を出力に追加するには、次の TITLE ステートメントを使用します。

```
title 'SAT Scores by Year, 1972-1998';
```

TITLE ステートメントはグローバルステートメントです。これは、SAS セッション内で、たとえ別の出力が後で生成される場合でも、SAS で最後に作成されたタイトルが、変更または削除されるまで使い続けられるという意味です。TITLE ステートメントは、プログラム内の任意の場所で使用できます。

昇順で番号付けして、ページごとに 10 タイトルまで指定できます。以前のタイトルにサブタイトルを追加する場合は、表示する順番でタイトルに番号を付けます。たとえば、サブタイトル'Separate Statistics by Test Type'を作成するには TITLE3 ステートメントを使用します。タイトル間にブランク行を追加するには、TITLE ステートメントで番号付けする際に番号をスキップします。TITLE ステートメントは次のようになります。

```
title1 'SAT Scores by Year, 1972-1998';
title3 'Separate Statistics by Test Type';
```

タイトル行を変更するには、タイトルのテキストを変更し、すべての TITLE ステートメントを含めてプログラムを再サブミットします。特定の行に TITLE ステートメントが指定されると、その行、およびその行よりあとの番号が付けられたすべての行の、以前の TITLE ステートメントが取り消されることに注意してください。

デフォルトのタイトルを含むすべてのタイトルを削除するには、次を指定します。

```
title;
```

または

```
title1;
```

次の例に、複数の TITLE ステートメントの使用方法を示します。

```
libname admin 'SAS-data-library';

data report;
   set admin.sat_scores;
   if year ge 1995 then output;
   title1 'SAT Scores by Year, 1995-1998';
   title3 'Separate Statistics by Test Type';
run;

proc print data=report;
run;
```

次の出力は結果を示しています。

アウトプット 33.2 *複数の TITLE ステートメントを表示するレポート*

Results Viewer - SAS Output

SAT Scores by Year, 1995-1998

Separate Statistics by Test Type

Obs	Test	Gender	Year	SATscore
1	Verbal	m	1995	505
2	Verbal	f	1995	502
3	Verbal	m	1996	507
4	Verbal	f	1996	503
5	Verbal	m	1997	507
6	Verbal	f	1997	503
7	Verbal	m	1998	509
8	Verbal	f	1998	502
9	Math	m	1995	525
10	Math	f	1995	490
11	Math	m	1996	527
12	Math	f	1996	492
13	Math	m	1997	530
14	Math	f	1997	494
15	Math	m	1998	531
16	Math	f	1998	496

TITLE ステートメントはプログラム内の任意の場所に配置できますが、次のいずれかの場所に配置して、TITLE ステートメントと特定のプロシジャステップを関連付けることができます。

- 出力を生成するステップの前
- プロシジャステートメントの後で、次の DATA ステートメントまたは RUN ステートメントの前か、あるいは次のプロシジャの前

TITLE ステートメントは、変更または削除されるまで、グローバルに適用されることに注意してください。

フットノートの追加

FOOTNOTE ステートメントは TITLE ステートメントと同じガイドラインに従って使用します。FOOTNOTE ステートメントはグローバルステートメントです。これは SAS セッション内で、たとえ別の出力があとで生成される場合でも、SAS で最後に作成されたフットノートが変更または削除されるまで使い続けられるという意味です。FOOTNOTE ステートメントは、プログラム内の任意の場所で使用できます。

フットノートでは、プロシジャ出力または DATA ステップ出力の下部に 10 行までテキストが書き込まれます。次に FOOTNOTE ステーメントの形式を示します。

FOOTNOTE<*n*> <'*text*'>;

ここで *n* はフットノートが配置される相対行を指定し、*text* はフットノートのテキストを指定します。*n* の値は 1-10 の間です。*n* の指定を省略すると、この値は 1 と見なされます。

フットノート'1967 and 1970 SAT scores estimated based on total number of people taking the SAT'を追加するには、次のステートメントをプログラムの任意の場所に指定します。

```
footnote1 '1967 and 1970 SAT scores estimated based on total number';
footnote2 'of people taking the SAT';
```

昇順で番号付けして、ページごとに 10 行までフットノートを指定できます。一連の 1 つのフットノートのテキストを変更してプログラムを再実行すると、そのフットノートのテキストが変更されます。ただし、プログラムで番号付けされた FOOTNOTE ステートメントを実行すると、その行よりあとの番号が付けられたすべてのフットノートが削除されます。

```
footnote;
```

または

```
footnote1;
```

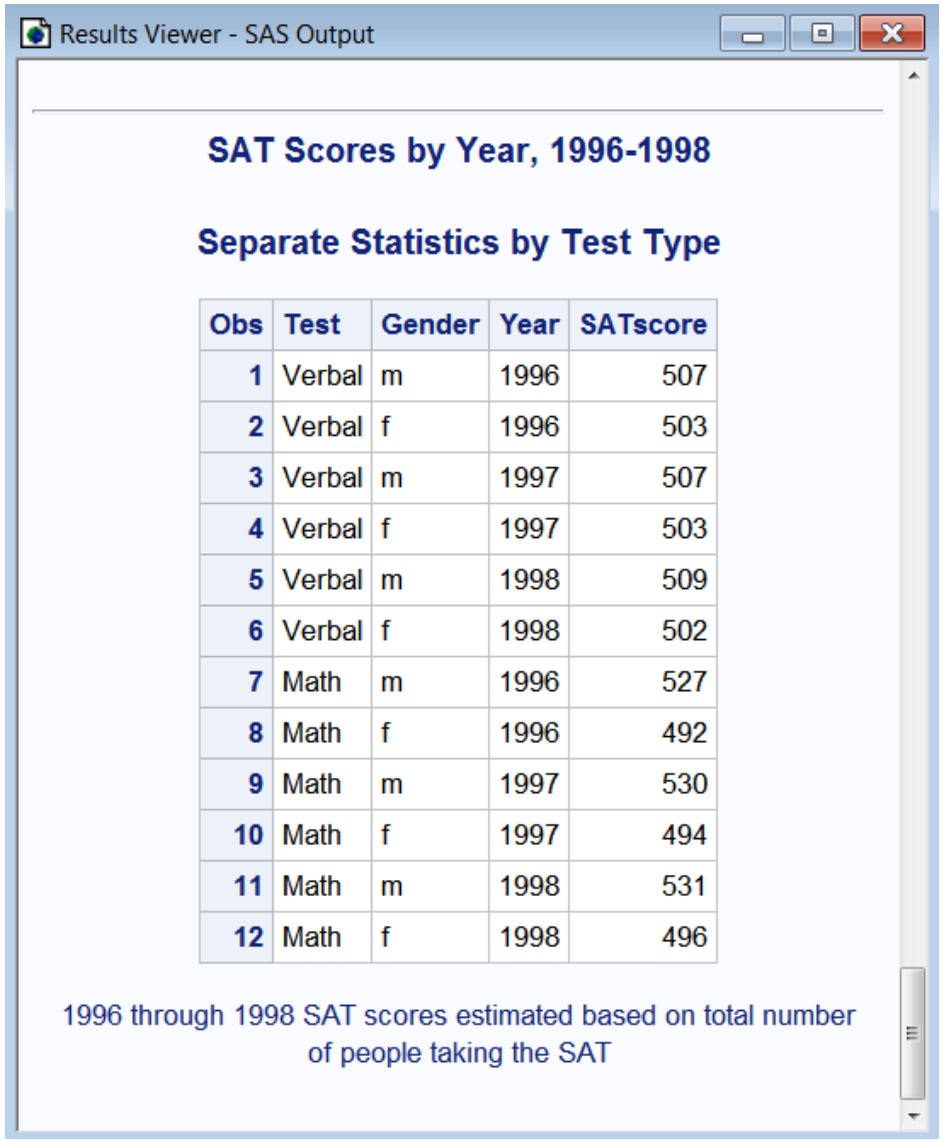
次の例に、複数の FOOTNOTE ステートメントの使用方法を示します。

```
libname admin 'SAS-data-library';

data report;
   set admin.sat_scores;
   if year ge 1996 then output;
   title1 'SAT Scores by Year, 1996-1998';
   title3 'Separate Statistics by Test Type';
   footnote1 '1996 through 1998 SAT scores estimated based on total number';
   footnote2 'of people taking the SAT';
run;

proc print data=report;
```

```
run;
```

次の出力は結果を示しています。

アウトプット 33.3 *フットノートを表示するレポート*



Obs	Test	Gender	Year	SATscore
1	Verbal	m	1996	507
2	Verbal	f	1996	503
3	Verbal	m	1997	507
4	Verbal	f	1997	503
5	Verbal	m	1998	509
6	Verbal	f	1998	502
7	Math	m	1996	527
8	Math	f	1996	492
9	Math	m	1997	530
10	Math	f	1997	494
11	Math	m	1998	531
12	Math	f	1998	496

FOOTNOTE ステートメントはプログラム内の任意の場所に配置できますが、次のいずれかの場所に配置して、FOOTNOTE ステートメントと特定のプロシジャステップを関連付けることができます。

- 前のステップの RUN ステートメントの後ろ
- プロシジャステートメントの後で、次の DATA ステートメントまたは RUN ステートメントの前、あるいは次のプロシジャの前

FOOTNOTE ステートメントは、変更または削除されるまで、グローバルに適用されることに注意してください。

変数のラベル付け

プロシジャ出力では、SAS は自動的に、ユーザーが指定する名前で変数を書き込みます。ただし、DATA ステップ、または、一部のプロシジャではプログラムの PROC ステップのいずれかに LABEL ステートメントを指定して、一部またはすべての変数に対してラベルを指定できます。ラベルの長さは、ブランクを含めて最大 256 文字です。

たとえば、変数 SATscore を語句'SAT Score'と表示するには、次を指定します。

```
label SATscore ='SAT Score';
```

DATA ステップで LABEL ステートメントを指定すると、そのラベルはデータセットに永続的に格納されます。PROC ステップで LABEL ステートメントを指定すると、PROC ステップの期間のみ、ラベルは変数と関連付けられます。いずれの場合も、ラベルが割り当てられている場合、ラベルはおよそすべての SAS プロシジャで書き込まれます。例外は PRINT プロシジャです。LABEL ステートメントを DATA ステップで使用している場合でも PROC ステップで使用している場合でも、PRINT プロシジャでは次のように LABEL オプションを指定する必要があります。

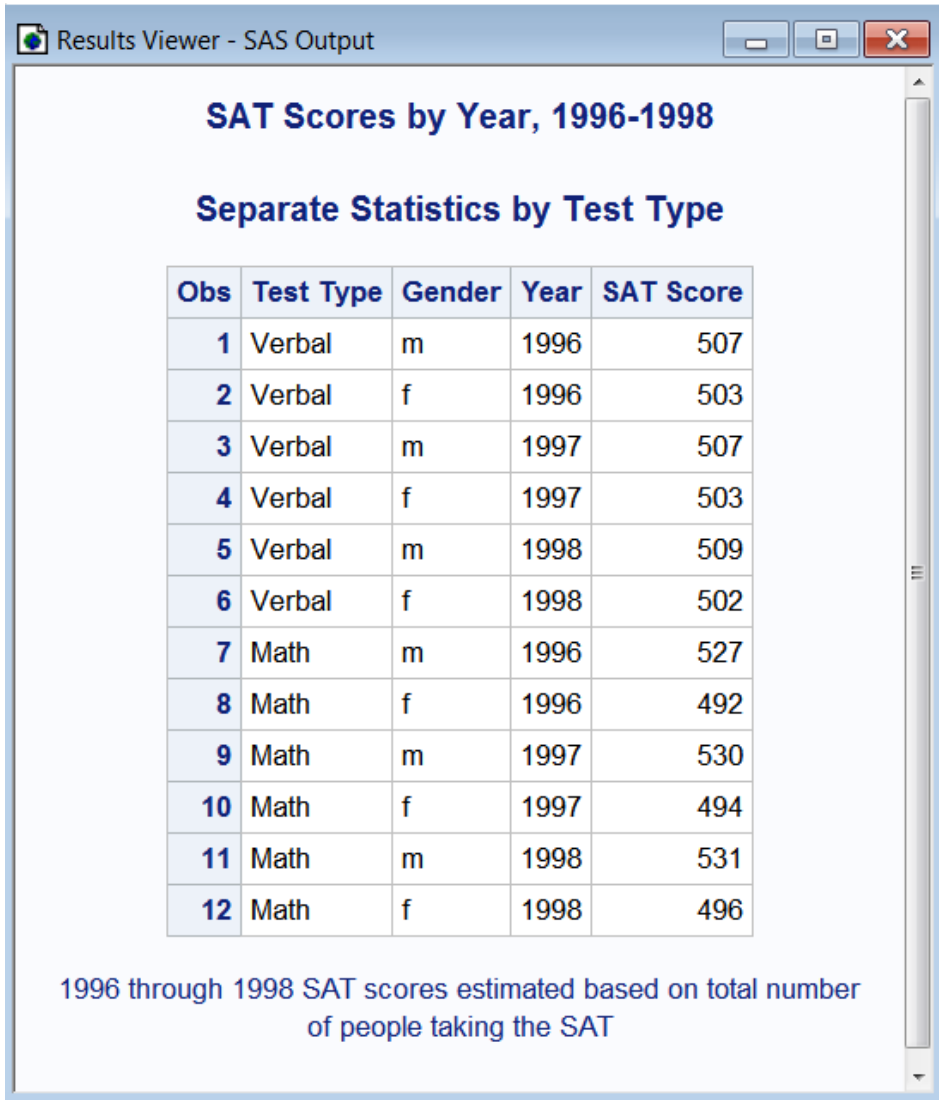
```
proc print data=report label;
run;
```

次の例に、LABEL ステートメントの使用方法を示します。

```
libname admin 'SAS-data-library';

data report;
   set admin.sat_scores;
   if year ge 1996 then output;
   label Test='Test Type'
         SATscore='SAT Score';
   title1 'SAT Scores by Year, 1996-1998';
   title3 'Separate Statistics by Test Type';
   footnote1 '1967 and 1970 SAT scores estimated based on total number';
   footnote2 'of people taking the SAT';
run;

proc print data=report label;
run;
```

次の出力は結果を示しています。

アウトプット 33.4 SAS 出力の変数ラベル



Obs	Test Type	Gender	Year	SAT Score
1	Verbal	m	1996	507
2	Verbal	f	1996	503
3	Verbal	m	1997	507
4	Verbal	f	1997	503
5	Verbal	m	1998	509
6	Verbal	f	1998	502
7	Math	m	1996	527
8	Math	f	1996	492
9	Math	m	1997	530
10	Math	f	1997	494
11	Math	m	1998	531
12	Math	f	1998	496

1996 through 1998 SAT scores estimated based on total number of people taking the SAT

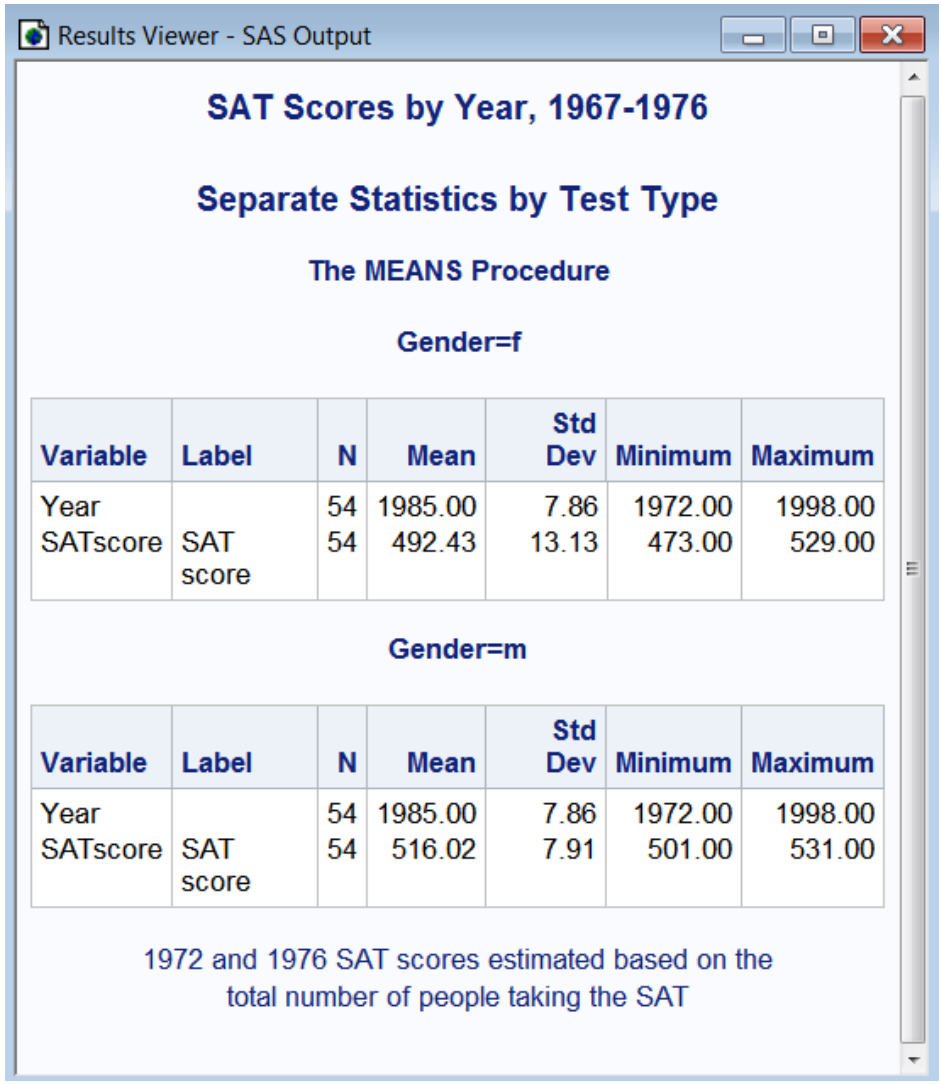
説明的な出力の作成

次の例に、TITLE、LABEL、および FOOTNOTE の各ステートメントと、プロシジャ出力を示します。

```
libname admin 'SAS-data-library';

proc sort data=admin.sat_scores;
   by gender;
run;

proc means data=admin.sat_scores maxdec=2 fw=8;
   by gender;
   label SATscore='SAT score';
   title1 'SAT Scores by Year, 1967-1976';
   title3 'Separate Statistics by Test Type';
   footnote1 '1972 and 1976 SAT scores estimated based on the';
   footnote2 'total number of people taking the SAT';
run;
```

次の出力は結果を示しています。

アウトプット 33.5 *SAS 出力のタイトル、ラベル、フットノート*


Results Viewer - SAS Output

SAT Scores by Year, 1967-1976

Separate Statistics by Test Type

The MEANS Procedure

Gender=f

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
Year		54	1985.00	7.86	1972.00	1998.00
SATscore	SAT score	54	492.43	13.13	473.00	529.00

Gender=m

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
Year		54	1985.00	7.86	1972.00	1998.00
SATscore	SAT score	54	516.02	7.91	501.00	531.00

1972 and 1976 SAT scores estimated based on the total number of people taking the SAT

リスト出力での出力表示の制御

SAS システムオプションの指定

OPTIONS ステートメントで SAS システムオプションを指定して、リスト出力の体裁を整えることができます。システムオプションの指定による変更は、別の OPTIONS ステートメントを発行してオプションが変更されるまで、残りのジョブ、セッション、または SAS プロセスに対して有効のままになります。

SAS システムオプションは、OPTIONS ステートメントや **OPTIONS** ウィンドウ、SAS 起動時や SAS プロセスの初期化時、および構成ファイルで指定できます。デフォルトのオプション設定はサイト間で異なります。自分のサイトの設定を指定するには、OPTIONS プロシジャを実行するか、**OPTIONS** ウィンドウへ移動します。

OPTIONS ステートメントの形式は次のとおりです。

OPTIONS *option(s)*;

ここで、*option* には、変更する 1 つ以上の SAS オプションを指定します。

注: OPTIONS ステートメントは、データ行の中を除き、SAS プログラム内のどの位置にも配置できます。

ページの番号付け

デフォルトでは、出力のページには SAS がページ 1 から開始して番号を付けます。一方、NONUMBER システムオプションを使用してページ番号を非表示にできます。ページ番号を非表示にするには、次の OPTIONS ステートメントを指定します。

```
options nonumber;
```

このオプションは、すべての SAS システムオプションと同様、途中で変更を加えない限り、SAS セッションが終了するまで有効です。オプションを変更するには、次を指定します。

```
options number;
```

PAGENO=システムオプションを使用して、SAS が書き込む次の出力のページの開始ページ番号を指定できます。PAGENO=オプションを使用して、SAS セッションの途中でページ番号を再設定できます。たとえば、次の OPTIONS ステートメントは、次の出力ページ番号を 5 に再設定します。

```
options pageno=5;
```

出力の中央揃え

デフォルトでは、出力と出力のタイトルの両方が中央揃えにされます。ただし、次の OPTIONS ステートメントを指定して、出力を左揃えにできます。

```
options nocenter;
```

NOCENTER オプションは、途中で変更を加えない限り、SAS セッションが終了するまで有効です。オプションを変更するには、次を指定します。

```
options center;
```

ページと行サイズの指定

プロシジャの出力はページと行のサイズに合うように自動的にスケール調整されます。印刷出力の 1 ページの行数と 1 行の文字数は PAGESIZE=と LINESIZE=のシステムオプションの設定によって決定されます。デフォルトの設定はサイトによって異なり、さらにマシン、動作環境、および SAS の実行方法によっても影響を受けます。たとえば、SAS が対話型モードで実行している場合、PAGESIZE=は、デフォルトでデバイスのサイズをユーザーが指定するサイズと見なします。PAGESIZE=オプションと LINESIZE=オプションを再設定して、ページサイズと行サイズの両方を調整できます。

たとえば、次の OPTIONS ステートメントを指定できます。

```
options pagesize=40 linesize=64;
```

PAGESIZE=オプションと LINESIZE=オプションは、途中で変更を加えない限り、SAS セッションが終了するまで有効です。

日付値と時間値の書き込み

デフォルトでは、出力の上部に、ジョブが実行された SAS セッションの開始日付と開始時刻が書き込まれます。この自動記録は、特に、プログラムを多くの回数実行する場合に便利です。ただし、NODATE システムオプションを使用して、これらの値を表示しないように指定することもできます。これを行うには、次の OPTIONS ステートメントを指定します。

```
options nodate;
```

NODATE オプションは、途中で変更を加えない限り、SAS セッションが終了するまで有効です。

オプションの選択

指定に合わせる必要があるシステムオプションを選択します。次の、条件付き IF-THEN/ELSE ステートメントを使用してデータセットをサブセット化するプログラムには、多くの SAS オプションがあります。OPTIONS ステートメントでは、64 の行サイズ、出力の左揃え、出力ページの番号付けを指定し、SAS セッションが開始された日付を表示します。デフォルトで HTML 出力が作成されるため、最初に ODS HTML CLOSE ステートメントを使用して ODS HTML 出力先を閉じる必要があります。HTML 出力を作成しない場合は、HTML 出力先を閉じてシステムリソースを保存します。ODS LISTING 出力先は、デフォルトで閉じられています。LISTING 出力先を開いてリスト出力を作成するために、ODS LISTING ステートメントを指定する必要があります。

```
ods html close;
ods listing;
options linesize=64 nocenter number date;

libname admin 'SAS-data-library';

data high_scores;
   set admin.sat_scores;
   if SATscore < 525 then delete;
run;

proc print data=high_scores;
   title 'SAT Scores: 525 and Above';
run;
ods listing close;
```

次の出力は結果を示しています。

アウトプット 33.6 システムオプションのリスト出力への効果

```
SAT Scores: 525 and Above                                    1 14:30 Thursday, May 2, 2013 Obs
Test     Gender    Year    SATscore 1     Verbal      m       1972       531 2     Verbal
f       1972       529 3     Math        m       1972       527 4     Math        m       1973
525 5     Math        m       1995       525 6     Math        m       1996       527 7
Math        m       1997       530 8     Math        m       1998       531
```

ページの表示形式の調整

複数ページのレポートの例で使用される入力データセット

以降のセクションでは、複数ページのレポートをカスタマイズする方法を学習します。

次のプログラムは、新聞の朝刊と夕刊の発行部数を含む SAS データセットを作成して出力します。各レコードには、州、朝刊発行部数(単位: 千)、夕刊発行部数(単位: 千)、およびデータが表す年が記録されています。デフォルトで HTML 出力が作成されるため、最初に ODS HTML CLOSE ステートメントを使用して ODS HTML 出力先を閉じる必要があります。HTML 出力を作成しない場合は、HTML 出力先を閉じてシステムリソースを保存します。ODS LISTING 出力先は、デフォルトで閉じられています。LISTING 出力先を開いてリスト出力を作成するために、ODS LISTING ステートメントを指定する必要があります。

```
data circulation_figures;
   length state $ 15;
   input state $ morning_copies evening_copies year;
   datalines;
Colorado  738.6 210.2 1984
Colorado  742.2 212.3 1985
Colorado  731.7 209.7 1986
Colorado  789.2 155.9 1987
Vermont   623.4 566.1 1984
Vermont   533.1 455.9 1985
Vermont   544.2 566.7 1986
Vermont   322.3 423.8 1987
Alaska     51.0  80.7 1984
Alaska     58.7  78.3 1985
Alaska     59.8  70.9 1986
Alaska     64.3  64.6 1987
Alabama   256.3 480.5 1984
Alabama   291.5 454.3 1985
Alabama   303.6 454.7 1986
Alabama      .  454.5 1987
Maine        .     .  1984
Maine        .   68.0 1985
Maine     222.7  68.6 1986
Maine     224.1  66.7 1987
Hawaii    433.5 122.3 1984
Hawaii    455.6 245.1 1985
Hawaii    499.3 355.2 1986
Hawaii    503.2 488.6 1987
;
run;


ods html close;
ods listing;
proc print data=circulation_figures;

ods listing close;
```

次の出力は結果を示しています。

アウトプット 33.7 SAS データセット CIRCULATION_FIGURES

```
The SAS System                                1 morning_    evening_ Obs     state       copies
copies     year 1    Colorado      738.6       210.2     1984 2    Colorado      742.2       212.3
1985 3    Colorado      731.7       209.7     1986 4    Colorado      789.2       155.9     1987 5
Vermont       623.4       566.1     1984 6    Vermont       533.1       455.9     1985 7
Vermont       544.2       566.7     1986 8    Vermont       322.3       423.8     1987 9
Alaska         51.0        80.7     1984 10    Alaska         58.7        78.3     1985 11
Alaska         59.8        70.9     1986 12    Alaska         64.3        64.6     1987 13
Alabama       256.3       480.5     1984 14    Alabama       291.5       454.3     1985 15
Alabama       303.6       454.7     1986
```

```
The SAS System                                2 morning_    evening_ Obs     state      copies
copies     year 16    Alabama         .454.5     1987 17    Maine           ..1984 18
Maine           .68.0     1985 19    Maine        222.7        68.6     1986 20    Maine
224.1        66.7     1987 21    Hawaii       433.5       122.3     1984 22    Hawaii       455.6
245.1     1985 23    Hawaii       499.3       355.2     1986 24    Hawaii       503.2       488.6
1987
```

中央揃えのタイトルと列ヘッダーの書き込み

TITLE ステートメントのデフォルトが中央揃えであるため、TITLE ステートメントを使用して中央揃えのタイトルを生成するのは簡単です。列ヘッダーの生成は少し手数がかかります。タイトル全体が中央揃えにされたときにテキストが正しい列に配置されるように、TITLE ステートメントに正しい数のブランクを挿入する必要があります。次の例に、中央揃えの行と列ヘッダーを書き込む方法を示します。タイトルと列ヘッダーは出力の各ページの上部に表示されます。

```
ods html close;
ods listing;

options linesize=80 pagesize=20 nodate;

data report1;
   infile 'your-data-file';
   input state $ morning_copies evening_copies year;
run;

title 'Morning and Evening Newspaper Circulation';
title2;
title3
      'State          Year                      Thousands of Copies';
title4
      '                                          Morning
Evening';

data _null_;
   set report1;
   by state notsorted;
   file print;
   if first.state then
      do;
         morning_total=0;
         evening_total=0;
```

```
            put / @7 state @;
         end;
      put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
      morning_total+morning_copies;
      evening_total+evening_copies;
      if last.state then
         do;
            all_totals=morning_total+evening_total;
            put @52 '------' @65 '------' /
                @26 'Total for each category'
                @52 morning_total 6.1 @65 evening_total 6.1
/
                @35 'Combined total' @59 all_totals 6.1;
         end;
run;
ods listing close;
```

次の出力は結果を示しています。

アウトプット 33.8 *SAS 出力の中央揃えの行と列ヘッダー*

```
Morning and Evening Newspaper Circulation                    1 State          Year
Thousands of Copies Morning      Evening Colorado            1984                      738.6
210.2 1985                         742.2        212.3 1986                       731.7        209.7
1987                        789.2        155.9 ------       ------ Total for each category
3001.7        788.1 Combined total          3789.8 Vermont            1984
623.4        566.1 1985                       533.1        455.9 1986
544.2        566.7 1987                       322.3        423.8 ------       ------ Total for each
category   2023.0       2012.5 Combined total          4035.5
```

```
Morning and Evening Newspaper Circulation                    2 State          Year
Thousands of Copies Morning      Evening Alaska              1984                        51.0
80.7 1985                          58.7         78.3 1986                         59.8         70.9
1987                         64.3         64.6 ------       ------ Total for each category
233.8        294.5 Combined total           528.3 Alabama             1984
256.3        480.5 1985                       291.5        454.3 1986
303.6        454.7 1987                          .454.5 ------       ------ Total for each category
851.4       1844.0 Combined total          2695.4
```

```
Morning and Evening Newspaper Circulation                    3 State          Year
Thousands of Copies Morning      Evening Maine               1984                            ..
1985                           .68.0 1986                       222.7         68.6
1987                        224.1         66.7 ------       ------ Total for each category
446.8        203.3 Combined total           650.1 Hawaii              1984
433.5        122.3 1985                       455.6        245.1 1986
499.3        355.2 1987                       503.2        488.6 ------       ------ Total for each
category   1891.6       1211.2 Combined total          3102.8
```

TITLE ステートメントを使用してタイトルと列ヘッダーを作成するとき、次の点を考慮してください。

- SAS では、デフォルトでタイトル行にページ番号が書き込まれます。そのため、このレポートにはページ番号が表示されています。ページ番号が不要な場合、NONUMBER システムオプションを指定します。
- PUT ステートメントポインタは、最終の TITLE ステートメントのすぐ次の行から開始されます。プロシジャ出力で行われるように、テキストの開始前に行がスキップされることはありません。この例では、TITLE4 ステートメントと各州のデータの最初の行との間のブランク行は、FIRST.STATE グループの PUT ステートメントのスラッシュ(/)によって生成されています。

特定のカラムにタイトルと列ヘッダーを書き込む

特定のカラムにヘッダーをプログラミングする最も簡単な方法は、PUT ステートメントを使用することです。テキストを特定のカラムに配置するために必要なブランクの正確な数を計算するかわりに、ポインタコントロールを使用してポインタを適切なカラムに移動し、テキストを書き込みます。PUT ステートメントを使用してヘッダーを書き込むには、処理されているオブザベーションや DATA ステップの反復にかかわらず、各ページで最初に PUT ステートメントを実行する必要があります。FILE ステートメントで HEADER=オプションを使用して書き込むヘッダーを指定します。

列ヘッダーを指定するには、次の形式で FILE ステートメントを指定します。

FILE PRINT HEADER=*label*;

PRINT は予約ファイル参照名で、任意の PUT ステートメントが生成する出力を、SAS プロシジャによって生成される出力と同じプリントファイルに出力します。*label* 変数は、SAS で新しい出力ページが開始されるたびに実行される、SAS ステートメントのグループを特定するステートメントラベルを定義します。

次のプログラムは、FILE ステートメントの HEADER=オプションを使用して DATA ステップにヘッダールーチンを追加します。ルーチンは、PUT ステートメントのポインタコントロールを使用して、タイトルを書き込み、2 行スキップし、その後特定の場所に列ヘッダーを書き込みます。

```
ods html close;
ods listing;

options linesize=80 pagesize=24;

data _null_;
   set circulation_figures;
   by state notsorted;
   file print notitles header=pagetop;
1
   if first.state then
         do;
            morning_total=0;
            evening_total=0;
            put / @7 state @;
         end;
   put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
   morning_total+morning_copies;
   evening_total+evening_copies;
   if last.state then
      do;
         all_totals=morning_total+evening_total;
         put @52 '------' @65 '------' /
             @26 'Total for each category'
             @52 morning_total 6.1 @65 evening_total 6.1 /
             @35 'Combined total' @59 all_totals 6.1;
      end;
   return; 2
   pagetop: 3
      put @16 'Morning and Evening Newspaper Circulation' //
          @7 'State' @26 'Year' @51 'Thousands of Copies'/
          @51 'Morning      Evening';
```

```
    return; 4
    run;
ods listing close;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 FILE ステートメントの PRINT ファイル参照名でリスト出力が作成されます。NOTITLES オプションによりタイトル行が削除され、それらの行が PUT ステートメントで使用可能になります。HEADER=オプションでは、SAS が新しい出力ページを開始するたびに実行される SAS ステートメントのグループを指すステートメントラベルが定義されます。(HEADER=オプションはプリントファイルの作成にのみ使用できます。)

2 ヘッダールーチンより前に配置されている RETURN ステートメントは、DATA ステップの主要部の終わりを示します。これにより、次の反復のために、実行がステップの最初に戻ります。この RETURN ステートメントがないと、各ページの先頭だけでなく DATA ステップの各反復時にもヘッダールーチンのステートメントが実行されます。

3 pagetop:ラベルによりヘッダールーチンが特定されます。SAS が新しいページを開始するごとに、現在の場所からラベル pagetop:へ移動して、RETURN ステートメントが検出されるまで実行が続けられます。ヘッダールーチンの終わりにある RETURN ステートメントまで実行されると、SAS が新しいページを開始したときに実行されていたステートメントに実行が戻ります。

4 RETURN ステートメントによりヘッダールーチンが終了します。SAS が新しいページを開始したときに実行されていたステートメントに実行が戻ります。

次の出力は結果を示しています。

アウトプット 33.9 *特定の場所に配置されたタイトルと列ヘッダー*

```
Morning and Evening Newspaper Circulation State              Year                       Thousands of
Copies Morning      Evening Colorado           1984                       738.6        210.2
1985                      742.2        212.3 1986                       731.7        209.7
1987                      789.2        155.9 ------       ------ Total for each category
3001.7        788.1 Combined total          3789.8 Vermont            1984
623.4        566.1 1985                        533.1        455.9 1986
544.2        566.7 1987                        322.3        423.8 ------       ------ Total for each
category   2023.0       2012.5 Combined total          4035.5 Alaska
1984                       51.0         80.7 1985                        58.7         78.3
1986                       59.8 70.9
```

```
Morning and Evening Newspaper Circulation State              Year                       Thousands of
Copies Morning      Evening 1987                       64.3         64.6 ------       ------ Total for
each category    233.8        294.5 Combined total           528.3 Alabama
1984                      256.3        480.5 1985                       291.5        454.3
1986                      303.6        454.7 1987                          .454.5 ------       ------
Total for each category    851.4       1844.0 Combined total          2695.4 Maine
1984                         ..1985                          .68.0 1986
222.7         68.6 1987                       224.1         66.7 ------       ------ Total for each
category    446.8        203.3 Combined total           650.1
```

ヘッダーの部分の変更

変数の値を使用して、ページごとに変わるヘッダーを作成できます。たとえば、プロシジャ出力ファイルのデフォルトのページ番号を削除すると、ヘッダーの部分として独自のページ番号を作成できます。デフォルトとは違う方法で数値を書き込むこともできま

す。たとえば、"1"ではなく"ページ 1"と書き込むことができます。ページ番号は、新しいページごとに変わるヘッダーの例です。

次のプログラムは、合計ステートメントを使用してページ番号を作成し、ヘッダールーチンの部分として番号を書き込みます。

```
ods html close;
ods listing;
options linesize=80 pagesize=24;

data _null_;
   set circulation_figures;
   by state notsorted;
   file print notitles header=pagetop;
   if first.state then
         do;
            morning_total=0;
            evening_total=0;
            put / @7 state @;
         end;
   put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
   morning_total+morning_copies;
   evening_total+evening_copies;
   if last.state then
      do;
         all_totals=morning_total+evening_total;
         put @52 '------' @65 '------' /
             @26 'Total for each category'
             @52 morning_total 6.1 @65 evening_total 6.1 /
             @35 'Combined total' @59 all_totals 6.1;
      end;
   return;

   pagetop:
      pagenum+1; 1
      put @16 'Morning and Evening Newspaper Circulation'
              @67 'Page ' pagenum //
2
           @7 'State' @26 'Year' @51 'Thousands of Copies'/
           @51 'Morning      Evening';
   return;
run;
ods listing close;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 この合計ステートメントでは、新しいページが開始されるたびにアキュムレータ変数 PAGENUM へ値 1 を追加します。

2 リテラルの Page と現在のページ番号が各新しいページの上部に出力されます。

次の出力は結果を示しています。

アウトプット 33.10　ヘッダーの部分の変更

```
Morning and Evening Newspaper Circulation          Page 1 State              Year
Thousands of Copies Morning      Evening Colorado           1984                      738.6
210.2 1985                       742.2        212.3 1986                      731.7        209.7
1987                        789.2        155.9 ------       ------ Total for each category
3001.7        788.1 Combined total          3789.8 Vermont            1984
623.4        566.1 1985                       533.1        455.9 1986
544.2        566.7 1987                       322.3        423.8 ------       ------ Total for each
category   2023.0       2012.5 Combined total          4035.5 Alaska
1984                         51.0         80.7 1985                        58.7         78.3
1986                         59.8 70.9
```

```
Morning and Evening Newspaper Circulation          Page 2 State              Year
Thousands of Copies Morning      Evening 1987                        64.3         64.6 ------
------ Total for each category    233.8        294.5 Combined total           528.3 Alabama
1984                        256.3        480.5 1985                       291.5        454.3
1986                        303.6        454.7 1987                          .454.5 ------       ------
Total for each category    851.4       1844.0 Combined total          2695.4 Maine
1984                           ..1985                          .68.0 1986
222.7         68.6 1987                       224.1         66.7 ------       ------ Total for each
category    446.8        203.3 Combined total           650.1
```

ページ分割の制御

アウトプット 33.10 (619 ページ)のレポートでは、Alaska のデータが自動的に 2 ページに分割されています。見やすいページ分割を行うには、特定の州のデータを出力する前に、ページにそのすべてのデータを出力する十分なスペースがあるか知っておく必要があります。

まず、データのグループを出力するために必要な行数を知る必要があります。次に、FILE ステートメントの LINESLEFT=オプションを使用して、その値が現在のページの残っている行の数である変数を作成します。データのグループの書き込みを開始する前に、必要な行の数と変数の値を比較します。使用可能な行より多くの行が必要な場合、_PAGE_ポインタコントロールを使用して、ポインタを新しいページの最初の行に進めます。

レポートでは、1 つの州に必要な行の最大数は 8 です(各州の 4 年の発行部数データと、アンダーライン、各合計、州間のブランク行の 4 行)。次のプログラムは、CKLINES という名前の変数を作成し、各 BY グループの先頭で、その値と 8 とを比較します。値が 8 より少ない場合、その州の書き込みをする前に新しいページが開始されます。

```
ods html close;
ods listing;
options pagesize=24;

data _null_;
   set circulation_figures;
   by state notsorted;
   file print notitles header=pagetop linesleft=cklines;
   if first.state then
         do;
            morning_total=0;
            evening_total=0;
            if cklines<8 then put _page_;
            put / @7 state @;
         end;
```

```
      put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
      morning_total+morning_copies;
      evening_total+evening_copies;
      if last.state then
         do;
            all_totals=morning_total+evening_total;
            put @52 '------' @65 '------' /
                @26 'Total for each category'
                @52 morning_total 6.1 @65 evening_total 6.1 /
                @35 'Combined total' @59 all_totals 6.1;
         end;
      return;

      pagetop:
         pagenum+1;
         put @16 'Morning and Evening Newspaper Circulation'
                 @67 'Page ' pagenum //
             @7 'State' @26 'Year' @51 'Thousands of Copies'/
             @51 'Morning      Evening';
      return;
   run;
   ods listing close;
```

次の出力は結果を示しています。

アウトプット 33.11 *特殊なページ分割の出力*

```
Morning and Evening Newspaper Circulation          Page 1 State              Year
Thousands of Copies Morning      Evening Colorado           1984                      738.6
210.2 1985                       742.2        212.3 1986                      731.7        209.7
1987                        789.2        155.9 ------       ------ Total for each category
3001.7        788.1 Combined total          3789.8 Vermont            1984
623.4        566.1 1985                       533.1        455.9 1986
544.2        566.7 1987                       322.3        423.8 ------       ------ Total for each
category   2023.0       2012.5 Combined total          4035.5
```

```
Morning and Evening Newspaper Circulation          Page 2 State              Year
Thousands of Copies Morning      Evening Alaska             1984                        51.0
80.7 1985                        58.7         78.3 1986                        59.8         70.9
1987                        64.3         64.6 ------       ------ Total for each category
233.8        294.5 Combined total          528.3 Alabama            1984
256.3        480.5 1985                       291.5        454.3 1986
303.6        454.7 1987                          .454.5 ------       ------ Total for each category
851.4       1844.0 Combined total          2695.4
```

```
Morning and Evening Newspaper Circulation          Page 3 State              Year
Thousands of Copies Morning      Evening Maine              1984                          ..
1985                          .68.0 1986                       222.7         68.6
1987                        224.1         66.7 ------       ------ Total for each category
446.8        203.3 Combined total          650.1
```

欠損値の表示

デフォルト値について

次の例では、1972 年の男性の言語と数学のスコアの数値データが欠損しています。1975 年の数学のスコアの性別の文字データが欠損しています。SAS では、データセットの作成時に、デフォルトで、数値欠損値はピリオド、文字欠損値はブランクと置換されます。

```
libname admin 'SAS-data-library';
data admin.sat_scores2;
   input Test $ 1-8 Gender $ 10 Year 12-15 SATscore 17-19;
   datalines;
verbal   m 1972 .
verbal   f 1972 529
verbal   m 1975 515
verbal   f 1975 509
math     m 1972 .
math     f 1972 489
math       1975 518
math       1975 479
;
run;


ods html close;
ods listing;

options pagesize=60 linesize=80 pageno=1 nodate;

proc print data=admin.sat_scores2;
   title 'SAT Scores for Years 1972 and 1975';
run;

ods listing close;
```

次の出力は結果を示しています。

アウトプット 33.12 *欠損値のデフォルト表示*

```
SAT Scores for Years 1972 and 1975                      1 Obs     Test     Gender    Year
SATscore 1     verbal      m       1972         .2     verbal      f       1972       529 3
verbal      m       1975       515 4     verbal      f       1975       509 5     math
m       1972         .6     math        f       1972       489 7     math                1975
518 8     math               1975       479
```

システムオプションを使用した欠損値の出力のカスタマイズ

データセットに数値欠損値が含まれている場合、MISSING=システムオプションを使用して欠損値をデフォルトのピリオドではなく、単一文字を使用して表示できます。使用

する文字を MISSING=オプションの値として指定します。任意の 1 文字を指定できます。

次のプログラムでは、OPTIONS ステートメントの MISSING=オプションにより、PRINT プロシジャで、各数値欠損値に対してピリオドではなく文字 M が表示されるようになります。

```
options missing='M' pageno=1;

libname admin 'SAS-data-library';
data admin.sat_scores2;
   input Test $ 1-8 Gender $ 10 Year 12-15 SATscore 17-19;
   datalines;
verbal   m 1972
verbal   f 1972 529
verbal   m 1975 515
verbal   f 1975 509
math     m 1972
math     f 1972 489
math       1975 518
math       1975 479
;

ods html close;
ods listing;

proc print data=admin.sat_scores2;
   title 'SAT Scores for Years 1972 and 1975';
run;
ods listing close;
```

次の出力は結果を示しています。

アウトプット 33.13 *数値欠損値のカスタマイズされた出力*

```
SAT Scores for Years 1972 and 1975                    1 Obs     Test     Gender    Year
SATscore 1     verbal      m       1972         M 2     verbal      f       1972       529 3
verbal      m       1975       515 4     verbal      f       1975       509 5     math
m       1972         M 6     math        f       1972       489 7     math                1975
518 8     math               1975       479
```

プロシジャを使用した欠損値の出力のカスタマイズ

また、FORMAT プロシジャを使用して、数値欠損値を表すこともできます。この方法では欠損値に出力形式を適用することでカスタマイズします。最初に、FORMAT プロシジャを使用して出力形式を定義し、次に、PROC ステップまたは DATA ステップで FORMAT ステートメントを使用して変数と出力形式を関連付けます。

次のプログラムでは、FORMAT プロシジャを使用して出力形式を定義し、次に、PROC ステップで FORMAT ステートメントを使用して変数 SCORE と出力形式を関連付けています。VALUE ステートメントでは出力形式名にピリオドを後置しませんが、FORMAT ステートメントでは常にピリオドを付けて出力形式を使用することに注意してください。

```
   ods html close;
   ods listing;
```

```
options pageno=1;
libname admin 'SAS-data-library';

proc format;
   value xscore .='score unavailable';
run;

proc print data=admin.sat_scores2;
   format SATscore xscore.;
   title 'SAT Scores for Years 1972 and 1975';
run;

ods listing close;
```

次の出力は結果を示しています。

アウトプット 33.14 *出力形式によって置き換えられた数値欠損値*

```
SAT Scores for Years 1972 and 1975                      1 Obs     Test     Gender    Year
SATscore 1     verbal      m       1972    score unavailable 2     verbal      f
1972                  529 3     verbal      m       1975                  515 4     verbal
f       1975                  509 5     math        m       1972    score unavailable 6
math        f       1972                  489 7     math                1975                  518
8     math                1975                  479
```

要約

ステートメント

FILE *file-specification*;
DATA ステップの PUT ステートメントの出力先に使用する外部ファイルを指定します。

FILE PRINT <HEADER=*label*> <LINESLEFT=*number-of-lines*>;
任意の PUT ステートメントが生成する出力を、SAS プロシジャによって生成される出力と同じプリントファイルに送ります。HEADER=オプションは、SAS で新しい出力ページが開始されるたびに実行する、SAS ステートメントのグループを指定するステートメントラベルを定義します。LINESLEFT=オプションは、その値が現在のページに残っている行の数である変数を定義します。

FOOTNOTE <*n*> <'*text*'>;
出力ページの下部に出力するフットノート行を最大 10 行指定します。変数 *n* はフットノートが配置される相対行を指定し、*text* はフットノートのテキストを指定します。

LABEL variable='*label*';
指定の変数を、ラベルとして指定する説明テキストと関連付けます。ラベルの長さは、ブランクを含めて最大 256 文字です。LABEL ステートメントは、DATA ステップまたは PROC ステップのいずれかで使用できます。

ODS LISTING <*option(s)*>
LISTING 出力先の開閉、管理を行います。

ODS HTML <*option(s)*>
HTML 出力先の開閉、管理を行います。

OPTIONS *option(s)*;
1 つ以上の SAS システムオプションの値を変更します。

TITLE <*n*> <'*text*'>;
プロシジャ出力ファイルとその他の SAS 出力の各ページに出力するタイトル行を最大 10 行指定します。変数 *n* はタイトル行を含める相対行を指定し、*text* はタイトルのテキストを指定します。

SAS システムオプション

NUMBER | NONUMBER
出力の各ページの最初のタイトル行にページ番号を出力するかどうかを制御します。

PAGENO=n
出力の次のページのページ番号を再設定します。

CENTER | NOCENTER
SAS プロシジャ出力を中央揃えにするかどうかを制御します。

PAGESIZE=n
出力ページごとに出力できる行数を指定します。

LINESIZE=n
DATA ステップおよびプロシジャによって使用される SAS ログと標準のプロシジャ出力ファイルについて、プリンタの行の幅を指定します。

DATE | NODATE
SAS ログ、標準のプリントファイル、または PRINT 属性を持つ任意のファイルの、各ページの上部に日付と時間を出力するか制御します。

MISSING='character'
数値変数の値が欠損値の場合に出力する文字を指定します。

詳細情報

ODS
- “Dictionary of ODS Language Statements” (*SAS Output Delivery System: User's Guide*)

SAS 出力
- 32 章, “SAS ログや出力ファイルに行を書き込む” (583 ページ)
- 34 章, “SAS 出力とそのカスタマイズについて: The Output Delivery System (ODS)” (625 ページ)

34 章

SAS 出力とそのカスタマイズについて: The Output Delivery System (ODS)

Output Delivery System (ODS)を使用した SAS 出力のカスタマイズについて . . . **626**
目的 . . . 626
前提条件 . . . 626
例で使用される入力データセット . . . **626**
ODS 出力形式およびその出力先について . . . **627**
出力形式の選択 . . . **629**
フォーマットされた出力の作成 . . . **630**
Web ブラウザ用の HTML 出力の作成 . . . 630
Adobe Acrobat およびその他のアプリケーション用 PDF 出力の作成 . . . 638
RTF および PowerPoint 出力の作成 . . . 639
フォーマットする出力の選択 . . . **642**
出力の識別 . . . 642
プログラム出力の選択と除外 . . . 643
SAS データセットの作成 . . . 645
ODS 出力のカスタマイズ . . . **647**
SAS ジョブレベルでの ODS 出力のカスタマイズ . . . 647
テンプレートを使用した ODS 出力のカスタマイズ . . . 650
ODS 出力へのリンクの格納 . . . **657**
要約 . . . **659**
ODS ステートメント . . . 659
PROC SORT ステートメント . . . 661
PROC TABULATE ステートメント . . . 661
PROC TEMPLATE ステートメント . . . 662
PROC UNIVARIATE ステートメント . . . 662
詳細情報 . . . **662**

Output Delivery System (ODS)を使用した SAS 出力のカスタマイズについて

目的

Output Delivery System (ODS)を使用すると、次のようにさまざまな形式で出力を作成できます。

- HTML ファイル
- PDF ファイル
- RTF ファイル(Microsoft Word 用)
- PowerPoint スライド
- 出力データセット

この章では、上に挙げた出力形式の ODS 出力を作成する方法を学習します。

前提条件

この章を使用する前に、次の概念を理解していることを確認してください。

- 1 章, “SAS System について” (3 ページ)
- 24 章, “SAS 出力と SAS ログの出力指定” (383 ページ)

DATA ステップ処理とプロシジャ出力の作成方法も理解している必要があります。

例で使用される入力データセット

次のプログラムは、オリジナルの PROC TABULATE、PROC UNIVARIATE、および PROC SGPANEL 出力を作成します。これらのプロシジャは、さまざまな国での事務用什器販売の要約統計量を作成します。このプログラム例を実行すると、ODS 出力が作成されます。デフォルトでは、Windows または UNIX の SAS ウィンドウ環境でコードを実行する場合、HTML が作成されます。出力(グラフを含む)は、Work ディレクトリに送られます。この出力は結果ビューアで表示可能です。

注: デフォルトでは、ODS が作成した出力を SAS は Work ディレクトリに格納します。Windows および UNIX の SAS ウィンドウ環境では、HTML 出力先の開閉後に、現在の作業ディレクトリへ出力されます。SAS セッション中にはいつでも、ODS PREFERENCE ステートメントを使用してデフォルトの動作に戻すことができます。このアクションは、複数のグラフを作成しているけれど、それらを現在の作業ディレクトリには集めたくない場合に役立ちます。

このプログラムは次を作成します。

- PROC TABULATE の出力オブジェクト 1 個
- PROC UNIVARIATE の出力オブジェクト 15 個
- PROC SGPANEL の出力オブジェクト 1 個

次のプログラムは、この章で使用する出力を作成します。

```
options nodate nonumber;
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

title;

proc tabulate data=prdsale;
   class region division prodtype;
   classlev region division prodtype;
   var actual;
   keyword all sum;
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         misstext=[label='Missing']
         box=[label='Region by Division and Type'];
run;

title 'Actual Product Sales';
title2 '(millions of dollars)';

proc univariate data=prdsale;
   by Country;
   var actual;
run;

title 'Sales Figures for First Quarter by Product';

proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;
```

注: 例で使用するファイル名は、すべての動作環境において有効とは限りません。各動作環境におけるファイル仕様の違いについては、各動作環境に対応するドキュメントを参照してください。

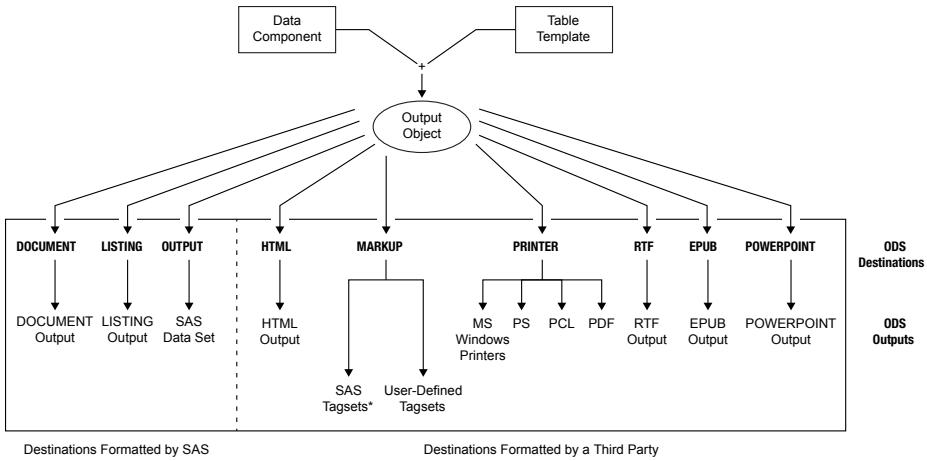
ODS 出力形式およびその出力先について

Output Delivery System (ODS)を使用すると、簡単にアクセス可能なさまざまな形式で出力を作成できます。ODS は、生データと 1 つ以上のテーブルテンプレートを組み合わせて、1 つ以上の出力オブジェクトを生成し、さまざまな種類の表形式出力を作成します。ODS 機能の基本コンポーネントは、出力オブジェクトです。PROC ステップや DATA ステップの実行により、データコンポーネント(生データ)と、フォーマット命令を含むテーブルテンプレート名が提供されます。データコンポーネントとテーブルテンプレートは連携して、出力オブジェクトを形成します。これらのオブジェクトは、PDF、HTML、RTF、LISTING などの、任意のあるいはすべての ODS 出力先に送ることができます。Windows および UNIX の SAS ウィンドウ環境の場合、デフォルトでは、

SAS は ODS を使用して HTML 出力を作成します。バッチモードの場合、デフォルトでは、SAS は LISTING 出力を作成します。ODS 出力先を指定することで、SAS が作成する出力の種類を制御します。

次の図は、出力の概念を示しています。データとテーブルテンプレートにより出力オブジェクトが形成され、出力オブジェクトによりテーブルテンプレートで指定した種類の ODS 出力が作成されます。

図 34.1 *ODS 出力の作成モデル*



次の定義は、上図の用語についての説明です。

データコンポーネント
: ODS をサポートする各プロシジャおよび各 DATA ステップでデータが作成されます。データには、SAS データセットと同様の形式のステップ結果(数値と文字)が含まれています。

テーブルテンプレート
: テーブルテンプレートは、データのフォーマット方法を記述する命令セットです。この記述には以下の項目が含まれますが、これはほんの一部です。

  - 列の順序
  - 列ヘッダーのテキストと順序
  - データの出力形式
  - フォントサイズとフォントの種類

出力オブジェクト
: ODS は、フォーマット命令とデータを組み合わせ、出力オブジェクトを作成します。したがって、出力オブジェクトには、プロシジャまたは DATA ステップの結果と、その結果のフォーマット方法についての情報の両方が含まれます。出力オブジェクトには、名前、ラベルおよびパスが含まれます。

  注: 多くの出力オブジェクトにはフォーマット命令が含まれていますが、含まれていないものもあります。出力オブジェクトがデータのみで構成されている場合もあります。

ODS 出力先
: ODS 出力先には、出力の特定の種類を指定します。ODS は、次を含む、ただしそれだけには限定されない多数の出力先をサポートしています。

  EPUB
  : .epub 拡張子を付けて出力を作成します。.epub 出力形式を使用する電子書籍は、普及しているさまざまな電子書籍リーダーで読み取り可能です。

HTML
埋め込みスタイルシートを含む HTML 4.0 出力を作成します。Web ブラウザを使用して Web 上の出力にアクセスできます。

OUTPUT
SAS データセットを作成します。

POWERPOINT
Microsoft PowerPoint 用の PowerPoint 出力を作成します。

LISTING
従来の SAS 出力(モノスペース形式)を作成します。

PDF
Adobe Acrobat とその他のアプリケーションによる読み取りの出力形式である、PDF 出力を作成します。

PS
Adobe Acrobat とその他のアプリケーションによる読み取りの出力形式である、PDF 出力を作成します。

PRINTER
印刷可能な出力を作成します。

RTF
Microsoft Word 2002 で使用するためにリッチテキスト形式で書き込まれる出力を作成します。

ODS 出力
ODS 出力は、任意の ODS 出力先からのフォーマットされた出力で構成されます。

ODS の詳細については、*SAS Output Delivery System: User's Guide* を参照してください。有効な ODS 出力先については、"Dictionary of ODS Language Statements" (*SAS Output Delivery System: User's Guide*)を参照してください。

出力形式の選択

出力のフォーマットは、プログラムで ODS 出力先を開いたり閉じたりすることによって選択します。1 つ以上の出力先が開いている場合、ODS により出力オブジェクトが送信され、フォーマットされた出力が生成されます。出力先が閉じている場合、ODS により出力オブジェクトは送信されず、出力は生成されません。

デフォルトでは、すべてのプログラムで自動的に HTML 出力が、ユーザーによって明示的に開かれるその他の出力先の出力と共に作成されます。そのため、デフォルトでは、HTML 出力先は開いており、その他の出力先はすべて閉じています。

フォーマットされた出力を作成するには、次のステートメントを使用して 1 つ以上の出力先を開きます。

ODS *destination file-specification(s)*;

destination は有効な ODS 出力先の 1 つです。引数 *file-specification* は出力先を開き、書き込み先ファイルを 1 つ以上指定します。

選択した ODS 出力を表示または印刷するには、HTML 出力先を除いて、ユーザーが開いた出力先をすべて閉じる必要があります。別々のステートメントを使用して出力先を個々に閉じたり、1つのステートメントですべての出力先(HTML 出力先を含む)を閉じたりすることができます。ODS 出力先を閉じるには、次のステートメントを使用します。

ODS *destination* CLOSE;

ODS _ALL_ **CLOSE**;

(デフォルト出力先以外の)ODS 出力を表示する前に、ODS 出力先を閉じる必要があります。出力先を閉じずに例を実行する場合、SAS ログにはファイルが作成されたようにリストされますが、そのファイルにはアクセスできず、ls コマンドを発行してもファイルは表示されません。ファイルにアクセスするには、次のステートメントを SAS セッションで実行します。ODS *destination* CLOSE;

注: 開いている出力先をすべて閉じる ODS _ALL_ CLOSE ステートメントは、SAS リリース 8.2 以降で使用可能です。

HTML 出力を作成する必要がない場合もあります。**ODS HTML CLOSE;** ステートメントをプログラムの最初に使用して HTML 出力先を閉じ、SAS が HTML 出力を作成しないようにします。不要な出力先を閉じると、システムリソースを節約できます。

注: ODS ステートメントはグローバルステートメントであるため、プログラムの最後に HTML 出力先を開くことをお勧めします。現在の SAS セッションで別のプログラムを実行すると、HTML 出力は使用可能になります。HTML 出力先を開くには、**ODS HTML;** ステートメントをプログラムの最後に使用します。

フォーマットされた出力の作成

Web ブラウザ用の HTML 出力の作成

4 種類の HTML 出力ファイルについて

HTML 出力はデフォルトで作成されます。ただし、ODS HTML ステートメントのオプションを使用する場合、またはテーブルファイル、ページファイルやフレームファイルを作成する場合は、ODS HTML ステートメントを使用する必要があります。ODS HTML ステートメントを使用すると、HTML でフォーマットされた出力を作成できます。Internet Explorer、Netscape、または HTML 4 を完全にサポートする任意のその他のブラウザで出力ファイルを参照できます。

ODS HTML ステートメントは、次の 4 種類の HTML ファイルを作成できます。

- DATA ステップまたはプロシジャの結果を含むボディファイル
- ボディファイルの項目へリンクする目次
- ボディファイルの項目へリンクするページ目次
- プロシジャまたは DATA ステップの結果、目次、およびページ目次を表示するフレームファイル

ボディファイルは、デフォルトで、すべての ODS HTML 出力で作成されます。出力へのリンクが不要な場合、目次、ページ目次、およびフレームファイルを作成する必要はありません。

HTML 出力の作成: 最も単純な場合

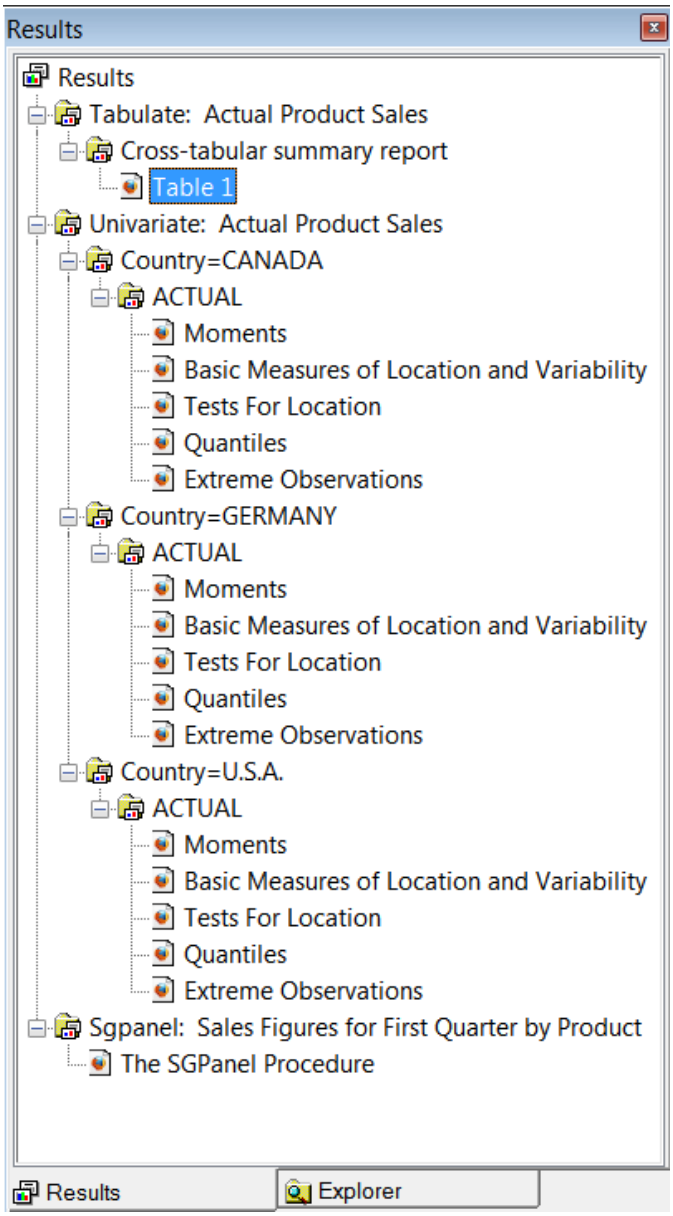
最も単純な種類の HTML 出力を作成する場合、ボディファイルのみ作成する必要があります。次の例では、SORT、MEANS、TABULATE、UNIVARIATE、SGPANEL の各プロシジャが実行されます。これらのファイルには、さまざまな国での事務用什器販売の要約統計量が含まれます。HTML 出力先は開いていて、SAS が自動的に HTML ボディファイルを作成するため、ODS ステートメントは必要ないことに注意してください。

```
options nodate nonumber;
```

```
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

proc tabulate data=prdsale;
   class region division prodtype;
   classlev region division prodtype;
   var actual;
   keyword all sum;
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         misstext=[label='Missing']
         box=[label='Region by Division and Type'];
 run;

title 'Actual Product Sales';
title2 '(millions of dollars)';

proc univariate data=prdsale;
   by Country;
   var actual;
run;

title 'Sales Figures for First Quarter by Product';

proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;
```

結果ビューアウィンドウに出力を表示でき、**結果**ウィンドウで出力オブジェクトを参照できます。名前、種類、説明、テンプレートなどの各出力オブジェクトに関する情報を**テーブルプロパティ**ウィンドウに表示できます。

図 34.2 *結果ウィンドウに表示される出力*



結果ウィンドウでは、次の出力が、**表の作成: 製品売上高実績** ⇨ **クロス集計要約レポート**の下にある出力オブジェクト Table 1 に対応しています。この出力オブジェクトの名前は Table 1 です。

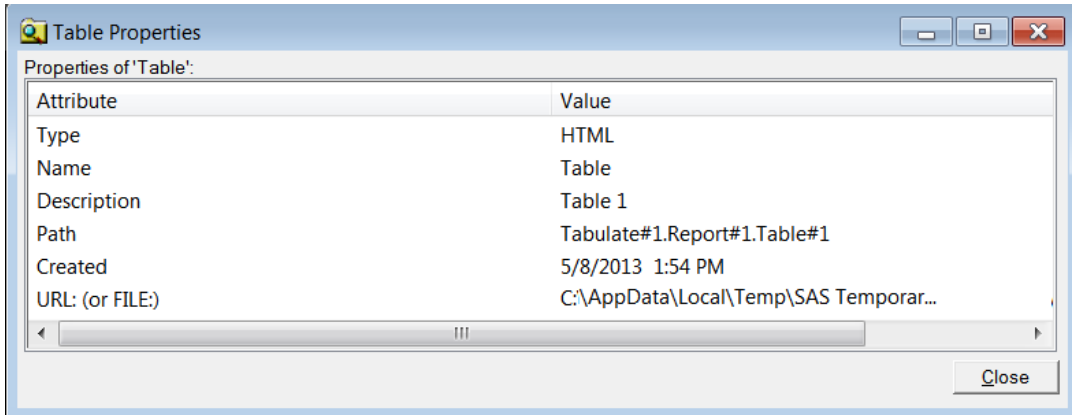
図 34.3 *PROC TABULATE 出力*

Actual Product Sales
(millions of dollars)

Region by Division and Type		Product type		Total
		FURNITURE	OFFICE	
		Actual Sales	Actual Sales	Actual Sales
		Sum	Sum	Sum
Region	Division			
EAST	CONSUMER	$72,570	$108,686	$181,256
	EDUCATION	$73,901	$115,104	$189,005
	Total	$146,471	$223,790	$370,261
WEST	Division			
	CONSUMER	$76,209	$105,020	$181,229
	EDUCATION	$67,945	$110,902	$178,847
	Total	$144,154	$215,922	$360,076
Total	Division			
	CONSUMER	$148,779	$213,706	$362,485
	EDUCATION	$141,846	$226,006	$367,852
	Total	$290,625	$439,712	$730,337

強調表示されている出力オブジェクトで右クリックし、ドロップダウンリストから**プロパティ**を選択して、各出力オブジェクトのプロパティを**テーブルプロパティ**ウィンドウに表示できます。

図 34.4 *PROC TABULATE: Table 1 のプロパティ*



結果ウィンドウでは、次の出力が、**単変量: 製品売上高実績** ⇨ **Country=CANADA** ⇨ **ACTUAL** の下にある出力オブジェクト Moments に対応しています。この出力オブジェクトの名前は Moments です。

図 34.5 *PROC UNIVARIATE 出力*

Actual Product Sales
(millions of dollars)

The UNIVARIATE Procedure
Variable: ACTUAL (Actual Sales)

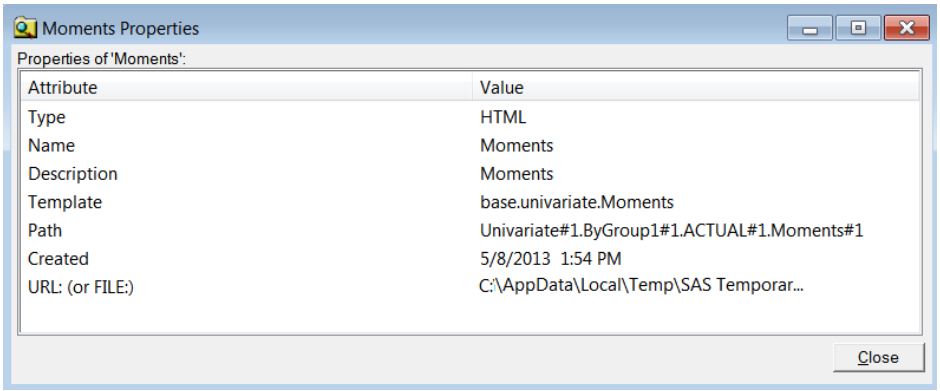
Country=CANADA

Moments			
N	480	Sum Weights	480
Mean	514.5625	Sum Observations	246990
Std Deviation	289.342982	Variance	83719.3614
Skewness	-0.0450336	Kurtosis	-1.1627896
Uncorrected SS	167193366	Corrected SS	40101574.1
Coeff Variation	56.2308723	Std Error Mean	13.2066399

Basic Statistical Measures			
Location		Variability	
Mean	514.5625	Std Deviation	289.34298
Median	513.5000	Variance	83719
Mode	688.0000	Range	997.00000

強調表示されている出力オブジェクトで右クリックし、ドロップダウンリストから**プロパティ**を選択して、各出力オブジェクトのプロパティを**テーブルプロパティ**ウィンドウに表示できます。

図 34.6 *PROC UNIVARIATE: Moments のプロパティ*



Moments Properties

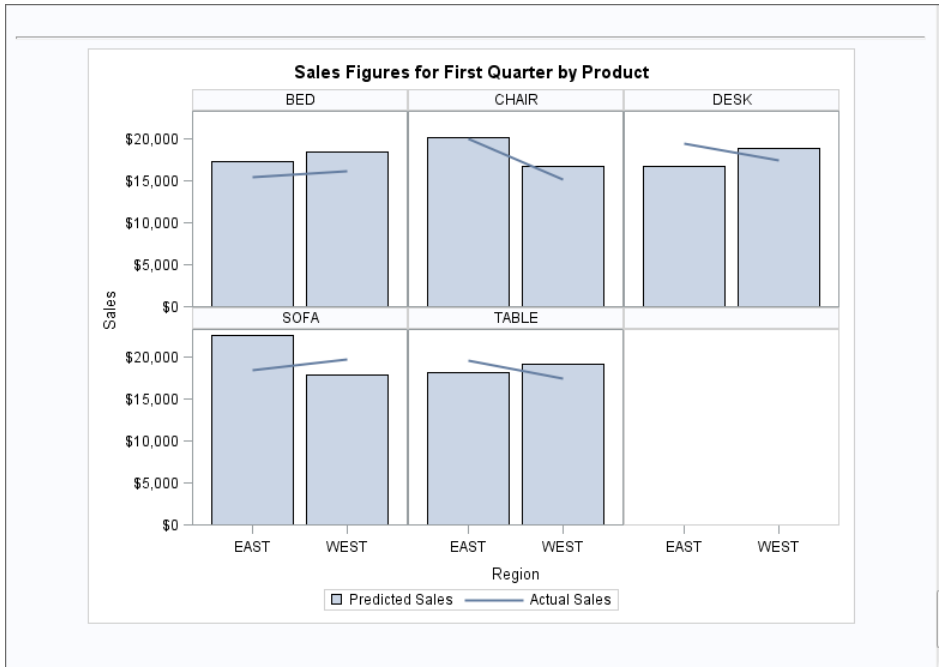
Properties of 'Moments':

Attribute	Value
Type	HTML
Name	Moments
Description	Moments
Template	base.univariate.Moments
Path	Univariate#1.ByGroup1#1.ACTUAL#1.Moments#1
Created	5/8/2013 1:54 PM
URL: (or FILE:)	C:\AppData\Local\Temp\SAS Temporar...

Close

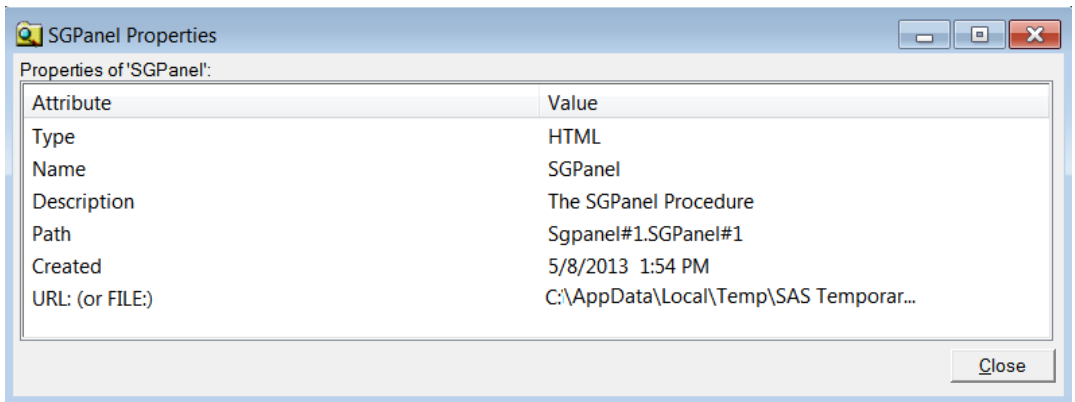
結果ウィンドウでは、次の出力が、**Sgpanel: 第 1 四半期の売上高(製品別)**の下にある出力オブジェクト **SGPanel プロシジャ**に対応しています。この出力オブジェクトの名前は SGPanel です。

図 34.7 *PROC SGPANEL 出力*



強調表示されている出力オブジェクトで右クリックし、ドロップダウンリストから**プロパティ**を選択して、各出力オブジェクトのプロパティを**テーブルプロパティ**ウィンドウに表示できます。

図 34.8 *PROC SGPANEL: SGPanel のプロパティ*



HTML 出力の作成: 結果と目次のリンク付け

ODS HTML 出力先を使用して、目次とページ目次から結果へリンク付けできます。これを行うには、ボディファイル、フレームファイル、目次、およびページ目次の HTML ファイルを作成する必要があります。フレームファイルを表示して目次またはページ目次のリンクを選択すると、プロシジャ結果の選択した部分を含む HTML テーブルがブラウザの上部に表示されます。

次の例は、UNIVARIATE プロシジャから複数の出力ページを作成します。目次またはページ目次のリンクから特定の出力結果(テーブル)にアクセスできます。結果に

は、大学進学予定生徒の平均 SAT スコアの統計量が含まれます。出力は、CLASS ステートメントで Gender の値でグループ化され、そして BY ステートメントで Test の値でグループ化されます。

```
options nodate nonumber;
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

1 ods html file='SalesFig-body.htm'
         contents='SalesFig-contents.htm'
         page='SalesFig-page.htm'
         frame='SalesFig-frame.htm';

2 proc tabulate data=prdsale;
   class region division prodtype;
   classlev region division prodtype;
   var actual;
   keyword all sum;
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         misstext=[label='Missing']
         box=[label='Region by Division and Type'];
run;

title 'Actual Product Sales';
title2 '(millions of dollars)';

3 proc univariate data=prdsale;
   by Country;
   var actual;
run;

title 'Sales Figures for First Quarter by Product';

4 proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;

5 ods html close;
6 ods html;
```

1 ODS HTML ステートメントによって HTML 出力先が開かれ、4 種類のファイルが作成されます。

- ボディファイル(FILE=オプションにより作成)。フォーマットデータを含みます。
- コンテンツファイル。ボディファイルの項目へのリンクがある目次です。
- ページファイル。ボディファイルの項目へのリンクがあるページ目次です。
- フレームファイル。目次、ページ目次、ボディファイルを表示します。

2 TABULATE プロシジャは製品売上実績の要約レポートを作成します。

3 UNIVARIATE プロシジャは、各国の売上実績に関する Moments、Basic Measures、Quantiles、Extreme のオブザベーションテーブルを作成します。

4 SGPANEL プロシジャは、第 1 四半期の売上高のグラフを製品ごとに作成します。

5 ODS HTML CLOSE ステートメントは開いている HTML 出力先を閉じ、出力を表示できるようにします。

6 ODS HTML ステートメントにより HTML 出力先が再度開かれ、次に実行するプログラムでは HTML 出力が生成できるようになります。

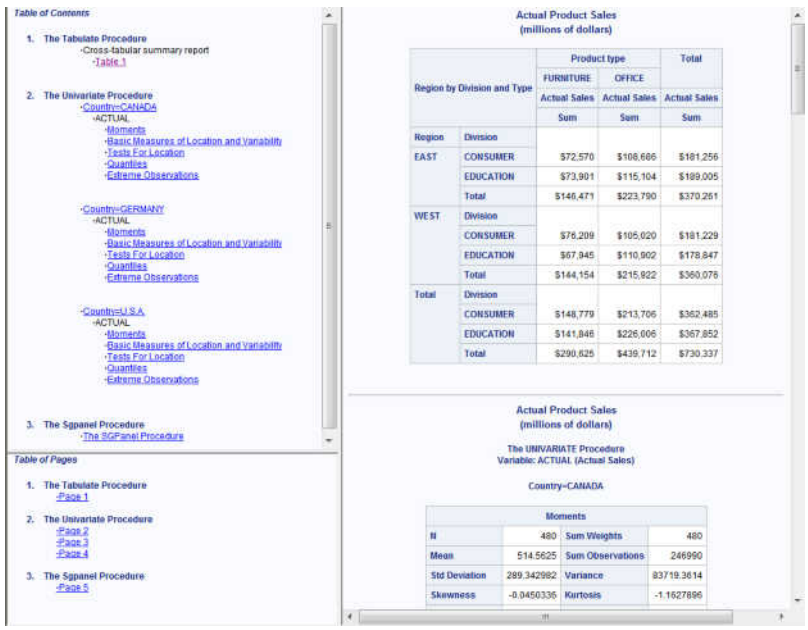
次の SAS ログは、ODS HTML ステートメントにより、4 つの HTML ファイルが作成されたことを示しています。

ログ 34.1 *SAS ログの一部: HTML ファイルの作成*

```
1    options nodate nonumber; 2    proc sort data=sashelp.prdsale out=prdsale; 3        by Country;
4    run; NOTE: There were 1440 observations read from the data set SASHELP.PRDSALE.NOTE: The data set
WORK.PRDSALE has 1440 observations and 10 variables.NOTE: PROCEDURE SORT used (Total process time):
real time           0.03 seconds cpu time            0.03 seconds 5 6    ods html file='SalesFig-
body.htm' 7             contents='SalesFig-contents.htm' 8             page='SalesFig-page.htm'
9             frame='SalesFig-frame.htm'; NOTE: Writing HTML Body file: SalesFig-body.htm NOTE:
Writing HTML Contents file: SalesFig-contents.htm NOTE: Writing HTML Pages file: SalesFig-page.htm
NOTE: Writing HTML Frame file: SalesFig-frame.htm 10 11 12   proc tabulate data=prdsale; 13   class
region division prodtype;
```

次の出力は SalesFig-frame.htm という名前のフレームファイルを示し、これには、目次(左側上)、ページ目次(左側下)、およびボディファイル(右側)が表示されています。目次とページ目次の両方に、ボディファイルの結果へのリンクが含まれます。目次またはページ目次のリンクをクリックすると、ブラウザの上部に該当する結果が表示されます。

図 34.9 *HTML フレームファイル SalesFig-frame.htm の外観*



Adobe Acrobat およびその他のアプリケーション用 PDF 出力の作成

Adobe Acrobat およびその他のアプリケーション用にフォーマットされた出力を作成できます。ただし、ファイルにアクセスする前に、PDF 出力先を閉じる必要があります。

次の例では、SORT、MEANS、TABULATE、UNIVARIATE、SGPANEL の各プロシジャが実行されます。これらのファイルには、さまざまな国での事務用什器販売の要約統計量が含まれます。

```
options nodate nonumber;
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

1 ods html close;
2 ods pdf file='odspdf_output.pdf';

3 proc tabulate data=prdsale;
   class region division prodtype;
   classlev region division prodtype;
   var actual;
   keyword all sum;
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         misstext=[label='Missing']
         box=[label='Region by Division and Type'];
run;

title 'Actual Product Sales';
title2 '(millions of dollars)';

4 proc univariate data=prdsale;
   by Country;
   var actual;
run;

title 'Sales Figures for First Quarter by Product';

5 proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;

6 ods pdf close;
7 ods html;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 HTML 出力先はデフォルトで開いています。リソースを節約するために、ODS HTML CLOSE ステートメントを使用してこの出力先を閉じます。

2 ODS PDF ステートメントにより PDF 出力先を開き、書き込むファイルを指定します。

3 TABULATE プロシジャは、製品売上実績の要約レポートを作成します。

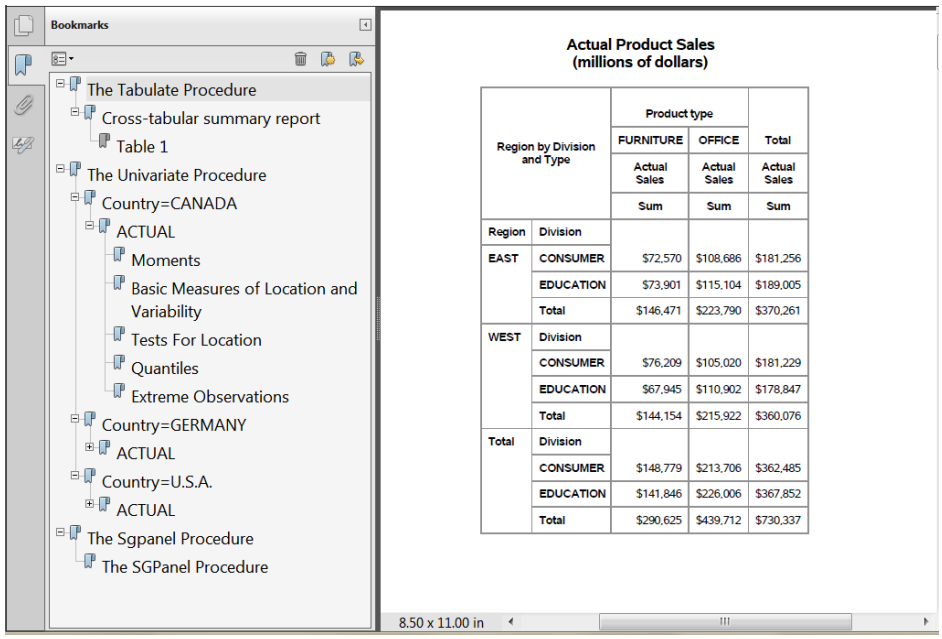
4 UNIVARIATE プロシジャは、各国の売上実績に関する Moments、Basic Measures、Quantiles、Extreme のオブザベーションテーブルを作成します。

5 SGPANEL プロシジャは、第 1 四半期の売上高のグラフを製品ごとに作成します。

6 ODS PDF CLOSE ステートメントは開いている PDF 出力先を閉じ、出力を表示できるようにします。

7 ODS HTML ステートメントにより HTML 出力先が再度開かれ、次に実行するプログラムでは HTML 出力が生成できるようになります。

図 34.10 *ODS 出力: PDF 形式*



Actual Product Sales (millions of dollars)

Region by Division and Type		Product type		Total
		FURNITURE	OFFICE	
		Actual Sales	Actual Sales	Actual Sales
		Sum	Sum	Sum
Region	Division			
EAST	CONSUMER	$72,570	$108,686	$181,256
	EDUCATION	$73,901	$115,104	$189,005
	Total	$146,471	$223,790	$370,261
WEST	Division			
	CONSUMER	$76,209	$105,020	$181,229
	EDUCATION	$67,945	$110,902	$178,847
	Total	$144,154	$215,922	$360,076
Total	Division			
	CONSUMER	$148,779	$213,706	$362,485
	EDUCATION	$141,846	$226,006	$367,852
	Total	$290,625	$439,712	$730,337

RTF および PowerPoint 出力の作成

出力を一度に複数の出力先へ送ることができます。ODS RTF ステートメントは、Microsoft Word 用にフォーマットされた出力を作成します。ODS POWERPOINT ステートメントは、Microsoft PowerPoint 用にフォーマットされた出力を作成します。ファイルにアクセスするには、出力先を閉じる必要があります。ODS _ALL_ CLOSE ステートメントを使用して、開いている出力先をすべて閉じることができます。

次の例では、SORT、MEANS、TABULATE、UNIVARIATE、SGPANEL の各プロシジャが実行されます。これらのファイルには、さまざまな国での事務用什器販売の要約統計量が含まれます。

```
options nodate nonumber;
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

1 ods html close;
2 ods rtf file='odsrtf_output.rtf';
```

```
3 ods powerpoint file='odspp_output.ppt';

4 proc tabulate data=prdsale;
   class region division prodtype;
   classlev region division prodtype;
   var actual;
   keyword all sum;
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         misstext=[label='Missing']
         box=[label='Region by Division and Type'];
run;

title 'Actual Product Sales';
title2 '(millions of dollars)';

5 proc univariate data=prdsale;
   by Country;
   var actual;
run;

title 'Sales Figures for First Quarter by Product';

6 proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;

7 ods _all_ close;
8 ods html;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 HTML 出力先はデフォルトで開いています。リソースを節約するために、ODS HTML CLOSE ステートメントを使用してこの出力先を閉じます。

2 ODS RTF ステートメントは、RTF 出力先を開き、書き込み先のファイルを指定します。

3 ODS POWERPOINT ステートメントは、POWERPOINT 出力先を開き、書き込み先のファイルを指定します。

4 TABULATE プロシジャは、製品売上実績の要約レポートを作成します。

5 UNIVARIATE プロシジャは、各国の売上実績に関する Moments、Basic Measures、Quantiles、Extreme のオブザベーションテーブルを作成します。

6 SGPANEL プロシジャは、第 1 四半期の売上高のグラフを製品ごとに作成します。

7 ODS _ALL_ CLOSE ステートメントは、開いている出力先すべてを閉じ、出力を表示できるようにします。

8 ODS HTML ステートメントにより HTML 出力先が再度開かれ、次に実行するプログラムでは HTML 出力が生成できるようになります。

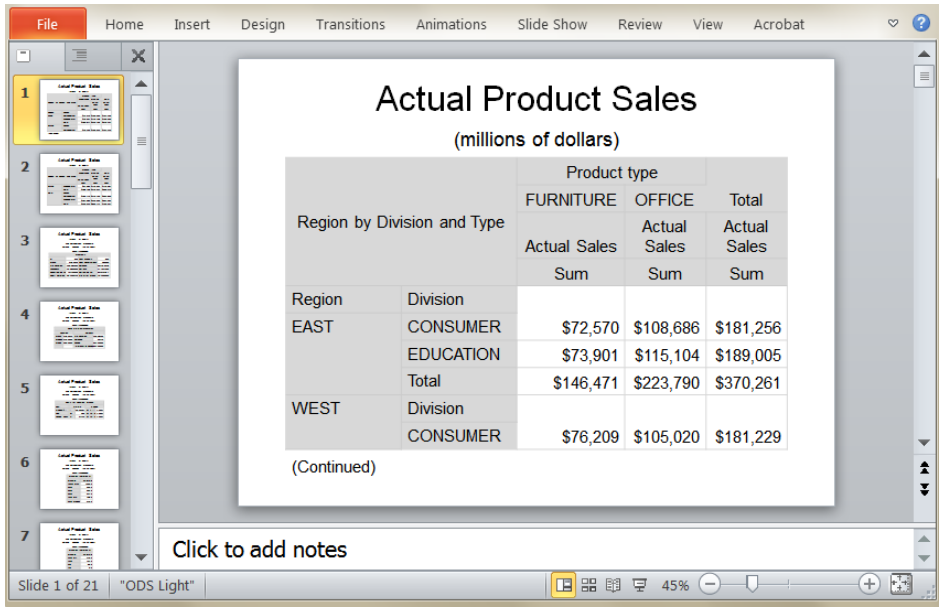
次の出力は、RTF 出力の最初のページを示します。

図 34.11 ODS 出力: RTF 形式

Actual Product Sales
(millions of dollars)

Region by Division and Type		Product type		Total
		FURNITURE	OFFICE	
		Actual Sales	Actual Sales	Actual Sales
		Sum	Sum	Sum
Region	Division			
EAST	CONSUMER	$72,570	$108,686	$181,256
	EDUCATION	$73,901	$115,104	$189,005
	Total	$146,471	$223,790	$370,261
WEST	Division			
	CONSUMER	$76,209	$105,020	$181,229
	EDUCATION	$67,945	$110,902	$178,847
	Total	$144,154	$215,922	$360,076
Total	Division			
	CONSUMER	$148,779	$213,706	$362,485
	EDUCATION	$141,846	$226,006	$367,852
	Total	$290,625	$439,712	$730,337

図 34.12 ODS 出力: PowerPoint 形式



フォーマットする出力の選択

出力の識別

出力オブジェクトの形式でのプログラム出力には、プロシジャまたは DATA ステップの結果と、その結果のフォーマット方法についての情報の両方が含まれます。フォーマットを行う出力オブジェクト選択するには、プログラムが作成する出力オブジェクトの種類を知っておく必要があります。出力オブジェクトを識別するには、ODS TRACE ステートメントを使用できます。次に最も単純な形式の ODS TRACE ステートメントの形式を示します。

ODS TRACE ON | OFF;

ODS TRACE は、プログラムが作成する各出力オブジェクトのレコードを SAS ログへ書き込むかどうかを決定します。ON オプションはトレースレコードをログに書き込み、OFF オプションはトレースレコードの書き込みを抑制します。

トレースレコードには次のコンポーネントがあります。

Name
出力オブジェクトの名前です。

Label
出力オブジェクトのコンテンツを簡単に説明するラベルです。

Template
ODS が出力オブジェクトをフォーマットするために使用するテーブルテンプレートの名前です。

Path
出力オブジェクトの場所を示します。

プログラムの ODS SELECT ステートメントで、出力オブジェクトを名前、ラベルやパスによって参照できます。

次のプログラムでは、TABULATE、UNIVARIATE、および SGPANEL の各プロシジャが実行され、トレースレコードが SAS ログに書き込まれます。

```
ods trace on;

options nodate nonumber;
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

proc tabulate data=prdsale;
   class region division prodtype;
   classlev region division prodtype;
   var actual;
   keyword all sum;
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         misstext=[label='Missing']
         box=[label='Region by Division and Type'];
run;
```

```
title 'Actual Product Sales';
title2 '(millions of dollars)';

proc univariate data=prdsale;
   by Country;
   var actual;
run;

title 'Sales Figures for First Quarter by Product';

proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;

ods trace off;
```

次の出力は ODS TRACE の結果を示しています。

ログ 34.2 *ログの ODS TRACE 部分出力*

```
22   proc univariate data=prdsale; 23   by Country; 24   var actual; 25   run; Output Added:
------------- Name:       Moments Label:      Moments Template:   base.univariate.Moments Path:
Univariate.ByGroup1.ACTUAL.Moments ------------- ......26   title 'Sales Figures for First Quarter by
Product'; 27   proc sgpanel data=prdsale; 28       where quarter=1; 29       panelby product /
novarname; 30       vbar region / response=predict; 31       vline region / response=actual
lineattrs=GraphFit; 32       colaxis fitpolicy=thin; 33       rowaxis label='Sales'; 34   run; NOTE:
PROCEDURE SGPANEL used (Total process time): real time           2.00 seconds cpu time            0.21
seconds Output Added: ------------- Name:       SGPanel Label:      The SGPanel Procedure Path:
Sgpanel.SGPanel ------------- NOTE: There were 360 observations read from the data set
WORK.PRDSALE.WHERE quarter=1; 35 36   ods trace off;
```

プログラム出力の選択と除外

各出力先に関して、ODS は選択リストまたは除外リストを保持しています。選択リストは、出力オブジェクトのリストで、フォーマットされた出力を生成します。除外リストは出力を生成しない出力オブジェクトのリストです。

ODS SELECT ステートメントまたは ODS EXCLUDE ステートメントで出力先を指定して、出力オブジェクトを選択したり除外したりできます。出力先を指定しない場合、ODS はすべての開いている出力先へ出力を送信します。

選択リストと除外リストは、SAS セッションのさまざまなポイント(プロシジャの境界など)で変更およびリセットできます。各プロシジャを、次の PROC ステップまたは DATA ステップによる終了を待機するのではなく、明示的な QUIT ステートメントで終了すると、QUIT ステートメントによって選択リストがリセットされます。

1 つ以上の出力オブジェクトを選択して、開いている ODS 出力先へそれらを送るには、ODS SELECT ステートメントを使用します。次に最も単純な形式の ODS SELECT ステートメントを示します。

ODS SELECT *<ODS-destination> output-object(s)*;

引数 *ODS-destination* は出力形式を特定し、*output-object* で、選択リストに追加する 1 つ以上の出力オブジェクトを指定します。

開いている出力先へ送る出力オブジェクトを 1 つ以上除外するには、ODS EXCLUDE ステートメントを使用します。次に最も単純な形式の ODS EXCLUDE ステートメントを示します。

ODS EXCLUDE *<ODS-destination> output-object(s)*;

引数 *ODS-destination* は出力形式を特定し、*output-object* で、除外リストに追加する 1 つ以上の出力オブジェクトを指定します。

次の例では、TABULATE、UNIVARIATE、SGPANEL の各プロシジャが実行されます。ODS SELECT ステートメントは、PROC UNIVARIATE の ExtremeObs、Quantiles、Moments 出力オブジェクトのみを選択するために、トレースレコードに Name コンポーネントを使用します。それは、ODS SELECT ステートメントと ODS EXCLUDE ステートメントによりプロシジャ境界でリセットが行われているため、PROC TABULATE ステップと PROC SGPANEL ステップの前に ODS EXCLUDE ステートメントを指定してそれらの出力オブジェクトを除外する必要があるからです。

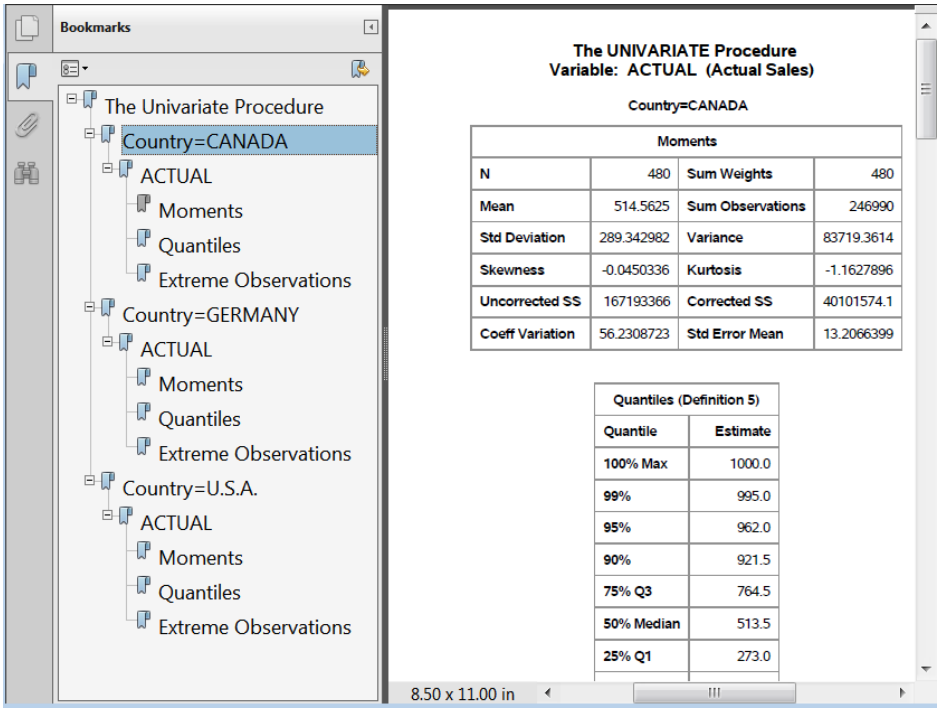
ODS SELECT ステートメントを使用するかわりに、ODS EXCLUDE ステートメントを PROC UNIVARIATE ステップの前に使用することもできます。次の ODS EXCLUDE ステートメントは同じ結果になります。

```
ods exclude BasicMeasures TestsForLocation;

   ods html close;
   options nodate nonumber;
   proc sort data=sashelp.prdsale out=prdsale;
       by Country;
   run;

   ods pdf file='PDFPrdsale.pdf';

   ods exclude table;

   title 'Actual Product Sales';
   title2 '(millions of dollars)';

   proc tabulate data=prdsale;
       class region division prodtype;
       classlev region division prodtype;
       var actual;
       keyword all sum;
       keylabel all='Total';
       table (region all)*(division all),
            (prodtype all)*(actual*f=dollar10.) /
            misstext=[label='Missing']
            box=[label='Region by Division and Type'];
   run;

   title;
   title2;

   ods select ExtremeObs Quantiles Moments;

   proc univariate data=prdsale;
       by Country;
       var actual;
   run;

   title 'Sales Figures for First Quarter by Product';
```

```
ods exclude sgpanel;

proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;

ods pdf close;
ods html open;
```

次の 2 つの表示は PDF 形式の結果を示しています。それらは、PROC UNIVARIATE の ExtremeObs、Quantiles、および Moments テーブルを示しています。

図 34.13 *ODS SELECT ステートメント: PDF 形式*



SAS データセットの作成

ODS では出力オブジェクトから SAS データセットを作成できます。単一の出力データセットを作成するには、次の形式の ODS OUTPUT ステートメントを使用します。

ODS OUTPUT *output-object(s)=SAS-data-set*;

引数 *output-object* には SAS データセットに変換するための 1 つ以上の出力オブジェクトを指定し、*SAS-data-set* には作成するデータセットを指定します。

次のプログラムでは、ODS は、出力先を開き、出力オブジェクト BasicMeasures から SAS データセット MYFILE.MEASURES を作成します。その後 ODS は、出力先を閉じます。

```
data sat_scores;
   input Test $ Gender $ Year SATscore @@;
   datalines;
Verbal m 1972 531  Verbal f 1972 529
Verbal m 1973 523  Verbal f 1973 521
Verbal m 1974 524  Verbal f 1974 520
Verbal m 1975 515  Verbal f 1975 509
Verbal m 1976 511  Verbal f 1976 508
Verbal m 1977 509  Verbal f 1977 505
Verbal m 1978 511  Verbal f 1978 503
Verbal m 1979 509  Verbal f 1979 501
Verbal m 1980 506  Verbal f 1980 498
Verbal m 1981 508  Verbal f 1981 496
Math   m 1985 522  Math   f 1985 480
Math   m 1986 523  Math   f 1986 479
Math   m 1987 523  Math   f 1987 481
Math   m 1988 521  Math   f 1988 483
Math   m 1989 523  Math   f 1989 482
Math   m 1990 521  Math   f 1990 483
Math   m 1991 520  Math   f 1991 482
Math   m 1992 521  Math   f 1992 484
Math   m 1993 524  Math   f 1993 484
Math   m 1994 523  Math   f 1994 487
Math   m 1995 525  Math   f 1995 490
Math   m 1996 527  Math   f 1996 492
Math   m 1997 530  Math   f 1997 494
Math   m 1998 531  Math   f 1998 496
;

proc sort data=sat_scores out=sorted_scores;
   by Test;
run;


   ods html close; 1
   ods output BasicMeasures=measures;
   2

   proc univariate data=sat_scores;
   3
      var SATscore;
      class Gender;
   run;

   ods output close; 4
   ods html; 5
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 HTML 出力先はデフォルトで開いています。リソースを節約するために、ODS HTML CLOSE ステートメントでこの出力先を閉じます。

2 ODS OUTPUT ステートメントにより出力先を開き、出力オブジェクト BasicMeasures から作成する永久データセットを指定します。

3 UNIVARIATE プロシジャにより大学進学予定生徒の平均 SAT スコアの要約統計量が生成されます。出力は CLASS 変数 Gender によってグループ化されます。

4 ODS OUTPUT CLOSE ステートメントで、出力先を閉じます。

5 ODS HTML ステートメントによりデフォルト HTML 出力先が再度開かれ、次に実行するプログラムでは HTML 出力が生成できるようになります。

次の SAS ログは、ODS OUTPUT ステートメントにより、WORK.MEASURES データセットが作成されたことを示しています。

ログ 34.3 *SAS ログの一部: SAS データセットの作成*

```
166 167 168  ods html close; 169  ods output BasicMeasures=measures; 170 171 172  proc univariate
data=sat_scores; 173 174     var SATscore; 175     class Gender; 176  run; NOTE: The data set
WORK.MEASURES has 8 observations and 6 variables.NOTE: PROCEDURE UNIVARIATE used (Total process time):
real time           0.01 seconds cpu time            0.01 seconds 177 178  ods output close; 179  ods
html; NOTE: Writing HTML Body file: sashtml3.htm
```

ODS 出力のカスタマイズ

SAS ジョブレベルでの ODS 出力のカスタマイズ

ODS を使用して、SAS ジョブレベルでの出力のカスタマイズを行うことができます。これを行うには、スタイルテンプレートを使用し、それには、色、フォントの種類、フォントのサイズなどの項目の表示方法が記述されています。スタイルテンプレートにより出力の外観が決定されます。FancyPrinter スタイルテンプレートは、SAS で使用可能なスタイル定義の 1 つです。

使用可能な ODS スタイルテンプレートを表示するには、次のコードを実行します。

```
proc template;
   list styles;
run;
```

アウトプット 34.1 ODS スタイルのサンプルリスト

The SAS System

Listing of: SASHELP.TMPLMST
Path Filter is: Styles
Sort by: PATH/ASCENDING

Obs	Path	Type
1	Styles	Dir
2	Styles.Analysis	Style
3	Styles.BarrettsBlue	Style
4	Styles.BlockPrint	Style
5	Styles.Daisy	Style
6	Styles.Default	Style
7	Styles.Dove	Style
8	Styles.Dtree	Style
9	Styles.EGDefault	Style
10	Styles.FancyPrinter	Style
11	Styles.Festival	Style
12	Styles.FestivalPrinter	Style
13	Styles.Gantt	Style
14	Styles.GrayscalePrinter	Style
15	Styles.HTMLBlue	Style
16	Styles.Harvest	Style
17	Styles.HighContrast	Style
18	Styles.Journal	Style
19	Styles.Journal1a	Style
20	Styles.Journal2	Style

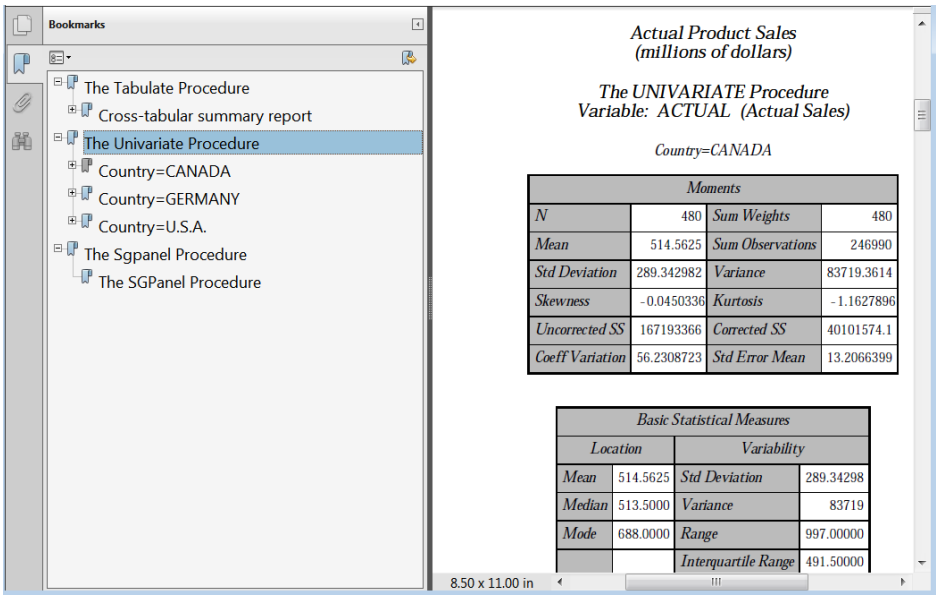
次の例では、プログラム出力のカスタマイズに FancyPrinter スタイルテンプレートを使用しています。ODS PRINTER ステートメントの STYLE=オプションで、プログラムでの FancyPrinter スタイルの使用を指定します。このスタイルでは、プリンタ出力するタイトル、フットノート、変数名を ODS が斜体で書き込みます。PDFTOC=オプションは、PDF ドキュメントで拡張レベルの目次を制御します。PDFTOC=1 オプションは、TOC が拡張レベル 2 であることを示しています。

```
options nodate nonumber;
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

ods html close;
ods pdf file='odspdf_output_custom.pdf' pdftoc=2 style=fancyprinter;
title;

proc tabulate data=prdsale;
```

```
   class region division prodtype;
   classlev region division prodtype;
   var actual;
   keyword all sum;
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         misstext=[label='Missing']
         box=[label='Region by Division and Type'];
 run;

title 'Actual Product Sales';
title2 '(millions of dollars)';

proc univariate data=prdsale;
   by Country;
   var actual;
run;

title 'Sales Figures for First Quarter by Product';

proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;

ods pdf close;
ods html;
```

次の出力は結果を示しています。

図 34.14 プリンタ出力: タイトル、フットノート、変数の斜体での出力



ODS スタイルの詳細については、“Style Templates” (*SAS Output Delivery System: User's Guide*)を参照してください。

テンプレートを使用した ODS 出力のカスタマイズ

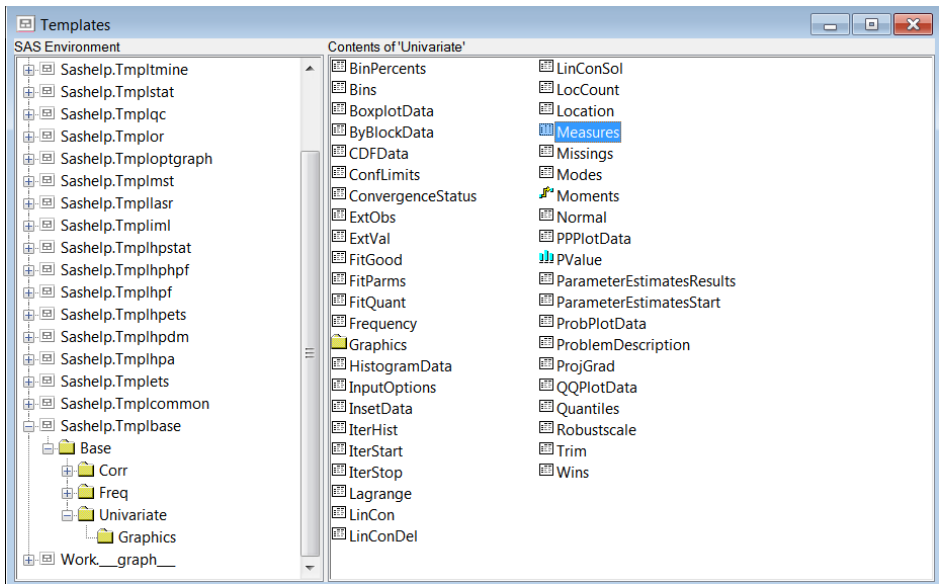
ODS 出力をカスタマイズするもう 1 つの方法は、テーブルテンプレートを使用することです。テーブルテンプレートには、表形式出力のフォーマット方法が記述されています。これにより、テーブルのヘッダーとフットノートの順序、列の順序、出力の外観が決定されます。テーブルテンプレートには 1 つ以上の列、ヘッダーやフットノートを含めることができます。独自のテーブルテンプレートを作成したり、既存のものを変更したりできます。ODS を完全にサポートする多くのプロシジャには、カスタマイズ可能なテーブルテンプレートが用意されています。

SAS ウィンドウ環境で、SAS 提供の SAS スタイルテンプレートを表示するには、次を行います。

1. **結果**ウィンドウで、**結果**フォルダを選択します。右クリックして**テンプレート**を選択し、**テンプレート**ウィンドウを開きます。
2. **Sashelp.Tmplmst** または **Sashelp.Tmplbase** などのディレクトリを展開し、そのディレクトリのコンテンツを表示します。
3. **Base** などのディレクトリをダブルクリックし、そのディレクトリのコンテンツを表示します。

この例では、PROC UNIVARIATE が作成したテンプレートを変更しています。PROC UNIVARIATE という名前のテンプレートを表示するには、**テンプレート**ウィンドウで、**Sashelp.Tmplbase** ⇨ **Base** ⇨ **Univariate** を選択します。Measures テーブルテンプレートを使用すると、出力が変化します。

アウトプット 34.2 *SAS 提供の PROC UNIVARIATE テンプレート*



Measures コードを表示するには、**Measures** をダブルクリックします。**Measures** テンプレートは**テンプレートブラウザ**ウィンドウで開きます。その後、コードを**エディタ**ウィンドウにコピー、貼り付けてカスタマイズできます。

アウトプット 34.3 *SAS 提供の Measure テンプレート*

Template Browser

```
proc template;
   define table Base.Univariate.Measures / store = SASHELP.TMPLBASE;
      notes "Basic measures of location and variability";
      dynamic MeasHdr varname varlabel;
      column LocMeasure LocValue VarMeasure VarValue;
      header h1 h2 h3;
      translate _val_=._ into "";

      define h1;
         text MeasHdr;
         space = 1;
         spill_margin;
      end;

      define h2;
         text "Location";
         end = LocValue;
         start = LocMeasure;
      end;

      define h3;
         text "Variability";
         end = VarValue;
         start = VarMeasure;
      end;

      define LocMeasure;
         space = 3;
         glue = 2;
         style = Rowheader;
         print_headers = OFF;
      end;

      define LocValue;
         space = 5;
         print_headers = OFF;
      end;

      define VarMeasure;
         space = 3;
         glue = 2;
         style = Rowheader;
         print_headers = OFF;
      end;

      define VarValue;
         format = D10.;
         print_headers = OFF;
      end;
   end;
run;
*** END OF TEXT ***
```

TEMPLATE プロシジャを使用して、独自のテーブルテンプレートを作成したり、既存のものを変更したりできます。次に TEMPLATE プロシジャの簡略化された形式を示します。

PROC TEMPLATE;**DEFINE** *table-definition*;**HEADER** *header(s)*;**COLUMN** *column(s)*;**END**;

DEFINE ステートメントは、出力を書き込むためのテンプレートとして機能するテーブルテンプレートを作成します。HEADER ステートメントはヘッダーの順序、COLUMN ステートメントは列の順序を指定します。これら各ステートメントの引数は、出力をフォーマットする、プログラム内のルーチンを指します。END ステートメントによりテーブルテンプレートが終了されます。

次の例に、PROC TEMPLATE を使用した、カスタマイズされた HTML 出力と PDF 出力の作成方法を示します。最初に、SAS 提供の SAS テンプレートコードを、アウトプット 34.3 (651 ページ)に説明されている方法を使用してコピーできます。この例では、SAS プログラムは PROC UNIVARIATE からの Basic Measures 出力テーブル用にカ

スタマイズされたテーブルテンプレートを作成します。カスタマイズされたバージョンでは次のように表示されます。

- "Measures of Variability"セクションが"Measures of Location"セクションより前に表示されます。
- 列ヘッダーが変更されます
- フォントカラーが変更されます
- 統計量が、7.3 出力形式の太字の斜体フォントで、特殊フォントカラーで表示されます

新しいテンプレートを作成します。

```
1 options nodate nonumber;

2 proc template;
  3 define table base.univariate.Measures;
  4 header h1 h2 h3;
  5 column VarMeasure VarValue LocMeasure LocValue;


  6 define h1;
     text "Basic Statistical Measures";
     style=data{color=orange fontstyle=italic};
     spill_margin=on;
     space=1;
  end;

  6 define h2;
     text "Measures of Variability";
     start=VarMeasure;
     end=VarValue;
  end;

  6 define h3;
     text "Measures of Location";
     start=LocMeasure;
     end=LocValue;
  end;

  7 define LocMeasure;
     print_headers=off;
     glue=2;
     space=3;
     style=rowheader;
  end;

  7 define LocValue;
     print_headers=off;
     space=5;
     format=7.3;
     style=data{font_style=italic font_weight=bold color=red};
  end;

  7 define VarMeasure;
     print_headers=off;
     glue=2;
```

```
      space=3;
      style=rowheader;
      style=data{font_style=italic font_weight=bold color=purple};
   end;

   7 define VarValue;
      print_headers=off;
      format=7.3;
      style=data{font_style=italic font_weight=bold color=blue};
   end;
8 end;
9 run;
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 NODATE オプションと NONUMBER オプションは、プリンタ出力に影響します。どのオプションも、HTML 出力には影響しません。

2 PROC TEMPLATE はテーブル作成のプロシジャを開始します。

3 DEFINE ステートメントはテーブルテンプレート base.univariate.Measures を SASUSER に作成します。

SAS 提供の base.univariate.Measures テーブルテンプレートは、SASHELP ライブラリのテンプレートストアに格納されます。(アウトプット 34.2 (650 ページ)を参照してください。)

4 HEADER ステートメントは、後にプログラムで定義されるヘッダーをテーブルテンプレートで使用する順序を決定します。

5 COLUMN ステートメントは、変数を表示する順番を決定します。PROC UNIVARIATE は、変数を指定します。

6 これらの DEFINE ステートメントのまとまりでは、3 つのヘッダーを定義し、各々のヘッダーで使用するテキストを指定しています。デフォルトでは、ヘッダーは全列にわたります。これは H1 の場合です。H2 は変数 VarMeasure と変数 VarValue の範囲です。H3 は LocMeasure と LocValue の範囲になります。

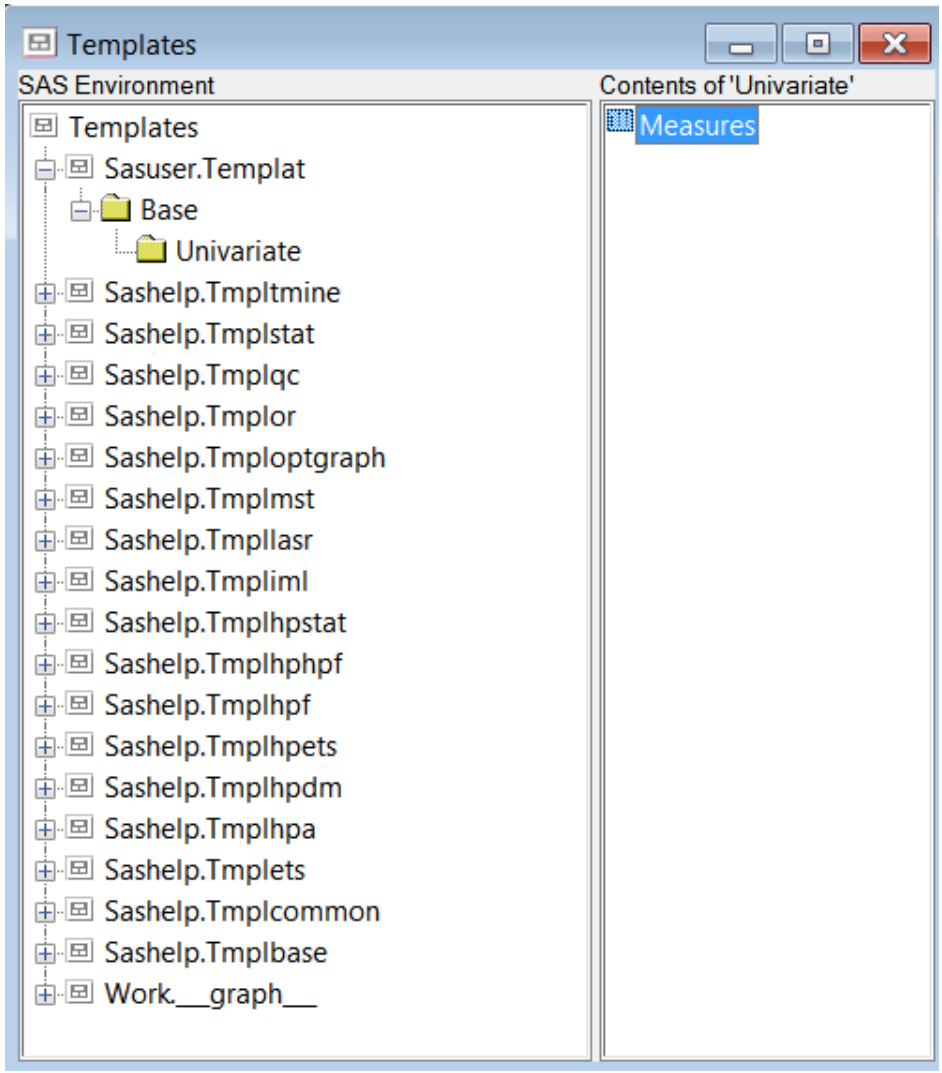
7 これらの DEFINE ステートメントのまとまりでは、4 つの変数のそれぞれに関する特性を指定します。FORMAT=を使用して LocValue と VarValue に 7.3 の出力形式を指定します。また、STYLE=を使用して、これら 2 つの変数に太字で斜体のフォントを指定します。STYLE=オプションは LISTING 出力には影響しません。

8 END ステートメントによりテーブルテンプレートが終了されます。

9 RUN ステートメントによりプロシジャが実行されます。

PROC TEMPLATE で作成したテーブルテンプレートを表示するには、**テンプレート**ウィンドウで、**Sasuser.Templat** ⇨ **Base** ⇨ **Univariate** を選択します。デフォルトでは、ODS は SASHELP 内を探す前に SASUSER でテーブルテンプレートを検索します。PROC UNIVARIATE がこの名前でテーブル定義を呼び出す場合、ODS は SASUSER からのテーブル定義を使用します。

アウトプット 34.4 ユーザー作成の PROC UNIVARIATE テーブルテンプレート



新しいテンプレートを使用した出力を作成します。

```
1 ods select BasicMeasures;

options nodate nonumber;
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

2 ods html file='SalesFig-body.htm'
          contents='SalesFig-contents.htm'
          page='SalesFig-page.htm'
          frame='SalesFig-frame.htm';

3 ods pdf file='odspdf_output_custom2.pdf' ;

title 'Actual Product Sales';
title2 '(millions of dollars)';
```

```
4 proc univariate data=prdsale;
    by Country;
    var actual;
run;

5 ods _all_ close;
6 ods html;

7 proc template;
    delete base.univariate.Measures;
run;
```

1 ODS SELECT ステートメントで基本統計量を含む出力オブジェクトが選択されます。

2 ODS HTML ステートメントは、カスタマイズされたテーブルテンプレートを使用するプログラムを開始します。プログラムが HTML 出力先を開き、書き込み先のファイルを識別します。

3 ODS PDF ステートメントにより PDF 出力先を開き、書き込むファイルを識別します。

4 PROC UNIVARIATE プロシジャでは、BY グループごとの各変数に対して 1 つのオブジェクトを生成します。カスタマイズされたテーブルテンプレートを使用してデータがフォーマットされます。デフォルトでは、ODS は SASHELP 内を探す前に SASUSER でテーブルテンプレートを検索します。PROC UNIVARIATE がこの名前でテーブル定義を呼び出す場合、ODS は SASUSER からのテーブル定義を使用します。

5 ODS _ALL_ CLOSE ステートメントで開いているすべての ODS 出力先を閉じます。

6 ODS HTML ステートメントにより HTML 出力先が再度開かれます。

7 この PROC TEMPLATE ステップでは base.univariate.Measures template を削除します。削除しない場合、削除されるまで、表出力のすべてにそれが適用されます。

プリンタ出力を次に示します。

アウトプット 34.5 *カスタマイズなしの PDF 出力*

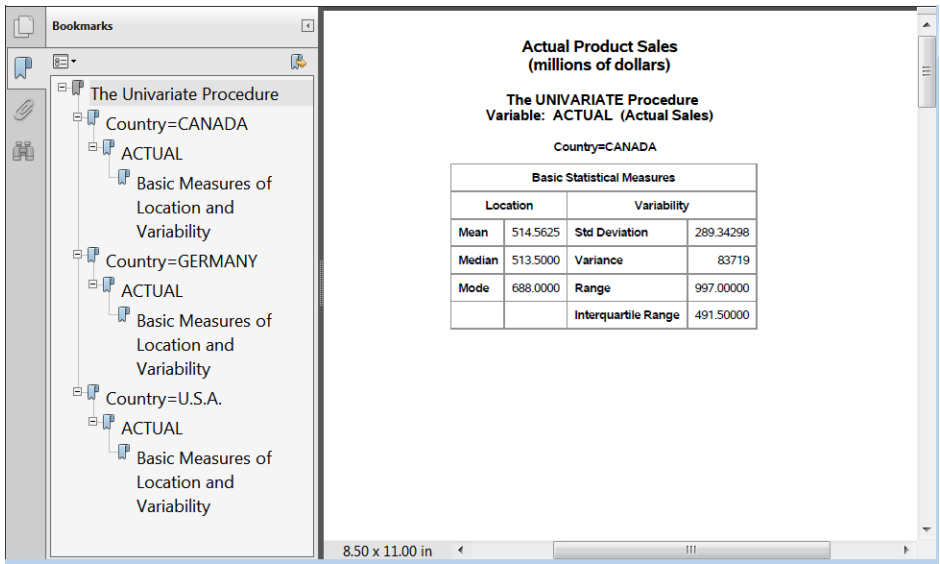
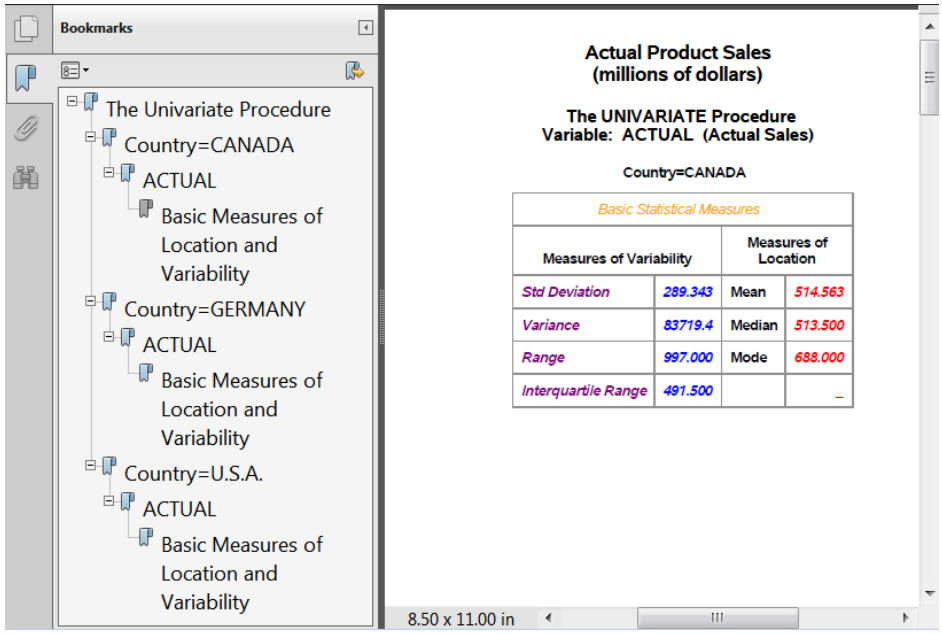
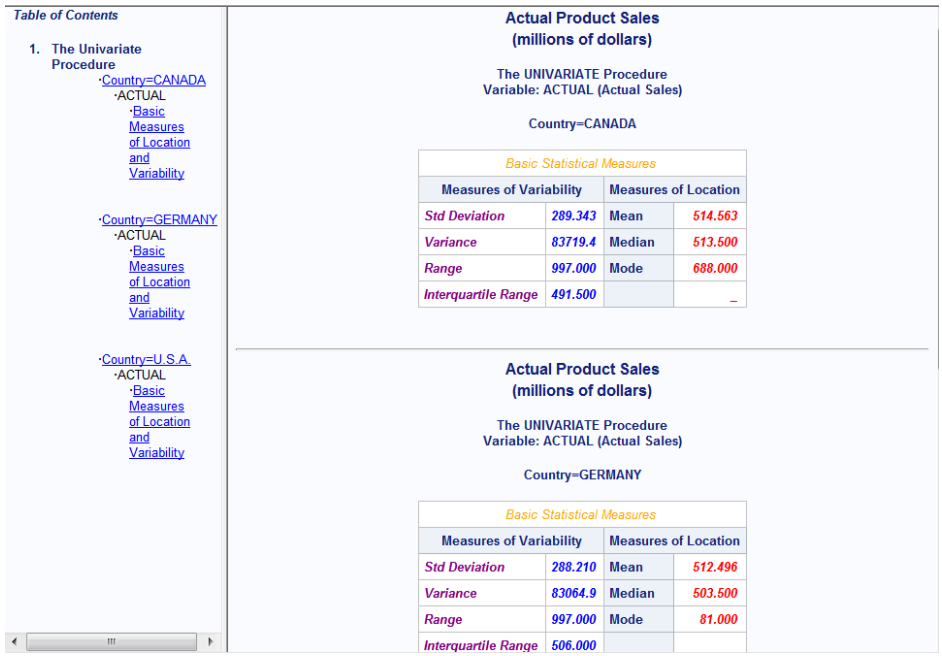


図 34.15 TEMPLATE プロシジャからのカスタマイズされたプリンタ出力



HTML 出力を次に示します。

図 34.16 TEMPLATE プロシジャからのカスタマイズされた HTML 出力



ODS 出力へのリンクの格納

ODS をサポートするプロシジャを実行する場合、**結果**ウィンドウの結果フォルダに、ODS 出力の各部分へのリンクを SAS が自動的に格納します。リンクには、出力を作成した出力先を特定するアイコンが示されます。

次の例では、SAS は UNIVARIATE、TABULATE、SGPANEL の各プロシジャを実行し、HTML、PowerPoint、リッチテキスト形式(RTF)出力を作成します。

```
options nodate nonumber;
proc sort data=sashelp.prdsale out=prdsale;
    by Country;
run;

ods html file='SalesFig-body.htm'
         contents='SalesFig-contents.htm'
         page='SalesFig-page.htm'
         frame='SalesFig-frame.htm';
ods rtf file='odsrtf_output.rtf';
ods powerpoint file='odspp_output.ppt';

proc tabulate data=prdsale;
   class region division prodtype;
   classlev region division prodtype;
   var actual;
   keyword all sum;
   keylabel all='Total';
   table (region all)*(division all),
         (prodtype all)*(actual*f=dollar10.) /
         misstext=[label='Missing']
         box=[label='Region by Division and Type'];
run;

title 'Actual Product Sales';
title2 '(millions of dollars)';

proc univariate data=prdsale;
   by Country;
   var actual;
run;

title 'Sales Figures for First Quarter by Product';

proc sgpanel data=prdsale;
    where quarter=1;
    panelby product / novarname;
    vbar region / response=predict;
    vline region / response=actual lineattrs=GraphFit;
    colaxis fitpolicy=thin;
    rowaxis label='Sales';
run;

ods _all_ close;
```

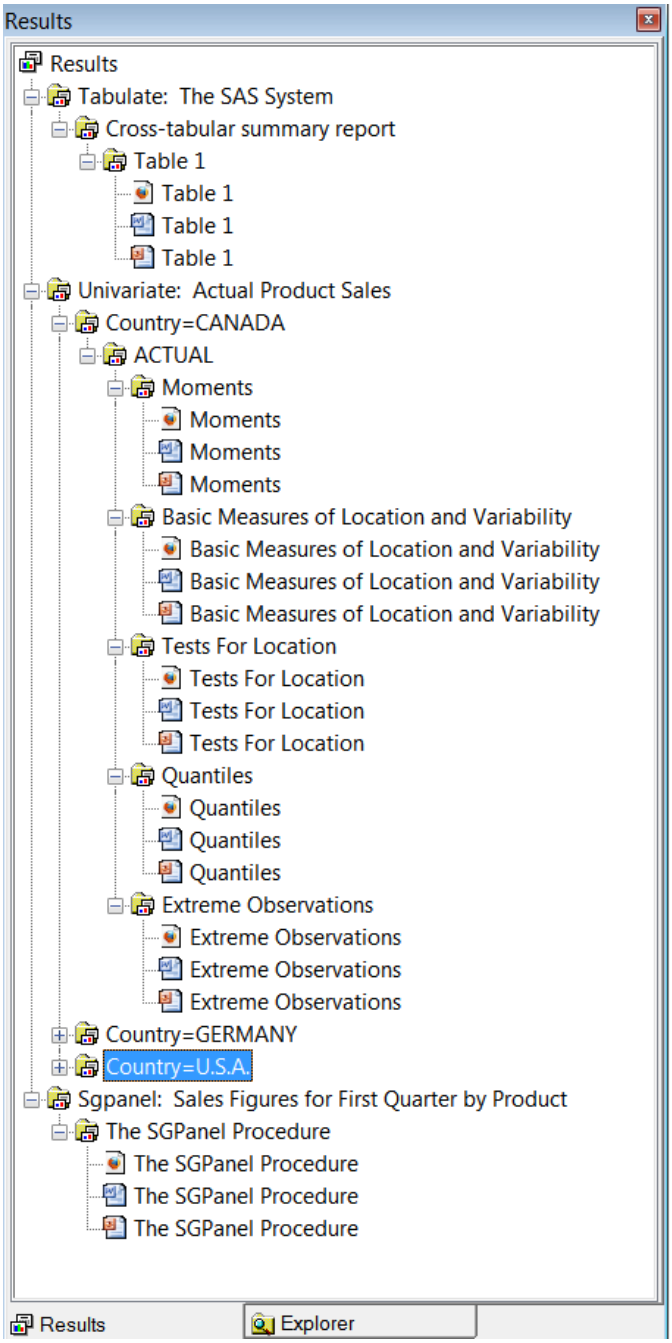
```
ods html;
```

PROC UNIVARIATE、PROC TABULATE、および PROC SGPANEL のそれぞれで**結果**ウィンドウにフォルダが 1 つずつ生成されています。これらのフォルダ内に、変数、BY グループ、および統計量用のフォルダが作成されます。たとえば、PROC UNIVARIATE は各 BY グループに対するフォルダ名 **Country** を **Country=CANADA** などのように作成します。各 BY グループフォルダ内には、それぞれの実績の変数用に **ACTUAL** という名前のフォルダがあります。フォルダ **ACTUAL** 内には、**Moments** などの統計量ごとにフォルダがあります。**Moments** などの一番下のフォルダには、そのプロシジャが作成した出力オブジェクトが含まれます。出力オブジェクトは適切なアイコンで示されます。

各出力オブジェクトのフォルダ内には、各出力へのリンクがあります。リンクの横にあるアイコンは、その出力を作成した ODS 出力先を示します。この例では、Moments 出力は HTML、RTF および POWEPOINT の各出力先へ送られています。

次に示す**結果**ウィンドウには、UNIVARIATE、TABULATE、SGPANEL の各プロシジャが作成したフォルダと出力オブジェクトが表示されています。

図 34.17 結果ウィンドウの外観



要約

ODS ステートメント

ODS EXCLUDE <ODS-destination> *output-object(s)*;
除外リストに追加する 1 つ以上の出力オブジェクトを指定します。

ODS HTML *HTML-file-specification(s)* <STYLE='style-definition'>;
HTML 出力先を開き、その HTML ファイルまたは書き込み先のファイルを指定します。出力先が開くと、Hyper Text Markup Language (HTML)で記述された出力を作成できます。

注: HTML 出力先はデフォルトで開いています。

書き込み先の HTML ファイルは 4 つまで指定できます。これらのファイルの指定の形式を次に示します。

BODY='*body-file-name*'
HTML 出力を含むファイルを指定します。

CONTENTS='*contents-file-name*'
HTML 出力の目次を含むファイルを指定します。コンテンツファイルにはボディファイルへのリンクがあります。

FRAME='*frame-file-name*'
目次、ページ目次、およびボディファイルを統合するファイルを指定します。フレームファイルを開くと、目次かページ目次またはその両方、およびボディファイルが表示されます。FRAME=を指定する場合、CONTENTS=か PAGE=、またはその両方とも指定する必要があります。

PAGE='*page-file-name*'
ボディファイルの各ページの説明とボディファイルへのリンクを含むファイルを指定します。ODS は、プロシジャで新しいページが明示的に要求されるときは、常に、出力の新規ページを生成します。SAS システムオプション PAGESIZE=は、HTML 出力のページには影響しません。

STYLE=オプションを使用して、HTML 表現のスタイルを選択できます。SAS 提供の各スタイルの詳細については、“Style Templates” (*SAS Output Delivery System: User's Guide*)を参照してください。

ODS LISTING;
LISTING 出力先を開きます。

ODS LISTING CLOSE;
LISTING 出力先を閉じ、LISTING 出力を作成しません。

ODS OUTPUT *output-object(s)=SAS-data-set*;
SAS 出力の出力先を開き、1 つ以上の出力オブジェクトを SAS データセットへ変換します。

ODS POWERPOINT *file-specification*;
POWERPOINT 出力先を開き、書き込み先のファイルを指定します。

ODS PDF *file-specification*;
PDF 出力先を開き、書き込み先のファイルを指定します。

ODS RTF *file-specification*;
RTF 出力先を開き、書き込み先のファイルを指定します。出力先が開かれると、RTF 出力を作成できます。

ODS *destination* CLOSE;
指定の *destination* を閉じ、出力を表示できるようにします。

ODS _ALL_ CLOSE;
開いている出力先をすべて閉じます。

ODS SELECT <ODS-destination> *output-object(s)*;
選択リストに追加する 1 つ以上の出力オブジェクトを指定します。

ODS TRACE ON | OFF;
トレースレコードの書き込みをオン/オフにします。トレースをオンにすると、プログラムによって作成される出力オブジェクトの結果リストを使用できるため便利です。

PROC SORT ステートメント

PROC SORT DATA=*SAS-data-set* OUT=*SAS-data-set*;
BY <DESCENDING> *variable-1*<<DESCENDING> *variable-2 …*>;

PROS SORT ステートメント
1 つ以上の文字変数または数値変数の値によって SAS データセットのオブザベーションを順序付けます。DATA=オプションは入力データセットを指定します。OUT=オプションは出力データセットを指定します。

BY *variable(s)*;
並べ替え変数を指定します。

PROC TABULATE ステートメント

PROC TABULATE <*option(s)*>
CLASS *variable(s)* </ *options*>;
CLASSLEV *variable(s)* </ STYLE=<*style-element*>>;
KEYLABEL *keyword-1='description-1'*
<keyword-2='description-2' ...>;
KEYWORD *keyword(s)* </ STYLE=<*style-element*>>;
TABLE <<*page-expression*,> *row-expression*,> *column-expression*</ *table-option(s)*>;
VAR *analysis-variable(s)*</ *option(s)*>;

PROC TABULATE ステートメント
記述統計量を表形式で表示します。

CLASS ステートメント
テーブルの分類変数を識別します。分類変数により、PROC TABULATE で統計量の計算に使用されるカテゴリが決定されます。

CLASSLEV ステートメント
分類変数の水準値ヘッダーのスタイル要素を指定します。

KEYWORD ステートメント
キーワードヘッダーのスタイル要素を指定します。

KEYLABEL ステートメント
PROC TABULATE ステップで使用するキーワードのラベルを指定します。PROC TABULATE は、キーワードの表示にこのラベルを使用します。ラベルが指定されない場合はキーワードが表示されます。

TABLE ステートメント
印刷されるテーブルを記述します。

VAR ステートメント
分析変数として使用する数値変数を識別します。

PROC TEMPLATE ステートメント

PROC TEMPLATE;
DEFINE <*template-type*> *template-name*</ *option(s)*>;
statements-and-attributes;
COLUMN *column(s)*;
HEADER *header-specification(s)*;
END;
DELETE *item-path* / <STORE=*template-store* >
END;

PROC TEMPLATE ステートメント
PROC TEMPLATE テンプレートを開始します。

DELETE ステートメント
指定した項目を削除します。

COLUMN ステートメント
シンボル 1 つをそのテーブルの 1 列として宣言し、それらの列の順序を指定します。

DEFINE ステートメント
テーブルテンプレート内部にテンプレートを作成します。DEFINE ステートメントは、COLUMN ステートメントと HEADER ステートメントを使用して、列ヘッダーとテーブルヘッダーを作成します。

HEADER ステートメント
シンボル 1 つをそのテーブルの 1 ヘッダーとして宣言し、それらのヘッダーの順序を指定します。

PROC UNIVARIATE ステートメント

PROC UNIVARIATE DATA=*SAS-data-set*;
BY *variable(s)*;
VAR *variable(s)*;

PROC UNIVARIATE ステートメント
UNIVARIATE プロシジャを開始し、モーメント(歪度と尖度を含む)、分位点またはパーセント点(中央値など)、度数表、極値に基づいた記述統計量を生成します。

CLASS ステートメント
その値によって分析の分類水準を定義する、最大 2 つの変数を指定します。

VAR ステートメント
分析変数と結果におけるそれらの順序を指定します。

詳細情報

PROC UNIVARIATE
UNIVARIATE プロシジャとその他の基本統計プロシジャの詳細については、*Base SAS(R) 9.3 Procedures Guide: Statistical Procedures* を参照してください。

ODS 出力

- ODS (Output Delivery System)の初歩的内容については、*Getting Started with the SAS Output Delivery System* を参照してください。
- ODS (Output Delivery System)の詳細については、*SAS Output Delivery System: User's Guide* を参照してください。
- 各 SAS スタイル属性の詳細については、“Style Attributes” (*SAS Output Delivery System: User's Guide*)を参照してください。
- ODS スタイルの詳細については、“Style Templates” (*SAS Output Delivery System: User's Guide*)を参照してください。
- 有効な ODS 出力先については、“Dictionary of ODS Language Statements” (*SAS Output Delivery System: User's Guide*)を参照してください。

PROC SGPANEL

SGPANEL プロシジャの詳細については、*SAS ODS Graphics: Procedures Guide* を参照してください。

Base SAS プロシジャ

PROC SORT プロシジャおよび PROC TABULATE プロシジャについては、*Base SAS Procedures Guide* を参照してください。

9 部

SAS ファイルでのデータの格納と管理

35 章
SAS ライブラリの基礎知識 667

36 章
SAS ライブラリの管理 675

37 章
SAS データセットの情報の表示 681

38 章
SAS データセット名と変数属性の変更 693

39 章
SAS データセットのコピー、移動、削除 705

35 章

SAS ライブラリの基礎知識

SAS ライブラリについて **667**
目的 667
前提条件 667
SAS ライブラリ概要 **668**
SAS ライブラリへのアクセス **668**
SAS への SAS ライブラリの場所の指示 668
ライブラリ参照名の割り当て 668
永久ライブラリと一時ライブラリへのライブラリ参照名の使用 669
SAS ライブラリにファイルを格納する **670**
SAS ファイルについて 670
SAS データセットについて 670
その他の SAS ファイルについて 671
SAS ライブラリの SAS データセットを参照する **671**
データセット名について 671
1 レベル名の使用 672
2 レベル名の使用 673
要約 **673**
ステートメント 673
SAS データセットの参照 674
詳細情報 **674**

SAS ライブラリについて

目的

SAS でのデータライブラリの処理方法は、動作環境によって異なります。このセクションでは、SAS ライブラリについての基本概念と SAS プログラムでのライブラリの使用方法を学習します。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

前提条件

このセクションを先に進む前に、次のセクションで説明した概念を理解している必要があります。

- 1 章, “SAS System について” (3 ページ)
- 3 章, “DATA ステップ処理について” (27 ページ)

SAS ライブラリ概要

SAS ライブラリとは、SAS によって認識され、ユニットとして参照および保存できる 1 つ以上の SAS ファイルの集合です。各ファイルは特定の SAS ライブラリのメンバになります。SAS ライブラリを使用して、作業を整理できます。たとえば、SAS プログラムで複数の SAS ファイルを使用する場合に、そのすべてのファイルを同じライブラリに保存できます。ライブラリ内のファイルを整理すると、ファイルの検索やプログラムでのファイルの参照が簡単になります。

大部分の動作環境では、SAS ライブラリは、通常、動作環境でファイルの整理に使用される構成レベルに対応しています。たとえば、ディレクトリベースの動作環境では、SAS ライブラリは同じディレクトリ内の SAS ファイルのグループです。ディレクトリにその他のファイルが含まれている場合でも、SAS ファイルのみが SAS ライブラリの一部です。

z/OS 固有
z/OS 動作環境下では、SAS ライブラリは特別にフォーマットされた z/OS データセットです。この種類のデータセットは、SAS ファイルのみ含むことができます。

SAS ライブラリへのアクセス

SAS への SAS ライブラリの場所の指示

使用している動作環境の種類にかかわらず、SAS ライブラリへアクセスするには、SAS にその場所を指示する必要があります。次の方法でこの操作を行えます。

- 動作環境での物理名を、SAS ライブラリの場所として直接指定します。物理名は使用している動作環境の命名規則に従い、引用符で囲む必要があります。たとえば、SAS ウィンドウ環境では、次の DATA ステートメントにより MYFILE という名前のデータセットが作成されます。

  ```
  data 'c:\my documents\sasfiles\myfile';
  ```

- SAS *ライブラリ参照名* を割り当てます。これは、SAS ライブラリの場所の物理名と一時的に関連付けられた SAS 名です。

ライブラリ参照名の割り当て

SAS ライブラリの場所へライブラリ参照名を割り当てると、その後、動作環境で使用される長い物理名のかわりに、ライブラリ参照名を SAS プログラムで使用してライブラリ内のファイルを参照できます。ライブラリ参照名は、SAS ライブラリの物理的な場所と一時的に関連付けられた SAS 名です。ライブラリ参照名を割り当てるには、いくつかの方法があります。

- LIBNAME ステートメントの使用
- LIBNAME 関数の使用
- SAS **エクスプローラ**ウィンドウから**新規ライブラリ**ウィンドウを使用

- 動作環境コマンドを使用(一部の動作環境)

ライブラリ参照名を割り当てる一般的な方法は、LIBNAME ステートメントを使用して SAS ライブラリと名前を関連付けることです。LIBNAME ステートメントの最も単純な形式を次に示します。

LIBNAME *libref* '*SAS-library*' ;

libref
SAS ライブラリと関連付けられるショートカット名を指定します。この名前は、SAS 名の規則に従う必要があります。ライブラリ参照名の最大長は 8 文字です。

z/OS 固有
z/OS 動作環境下では、ライブラリ参照名も動作環境の命名規則に従う必要があります。

ライブラリ参照名は、ライブラリの動作環境での名前の略語としてとらえてください。ライブラリ参照名は SAS セッションの間のみ持続します。いずれのライブラリについてもライブラリ参照名をセッションごとに変更できます。つまり、SAS を使用するたびに特定のライブラリに対して同じライブラリ参照名を使用する必要はありません。

SAS-library
SAS ライブラリの物理名を指定します。物理名は、動作環境によって認識される名前です。物理名は一重引用符または二重引用符で囲みます。

各動作環境での LIBNAME ステートメントの使用例を次に示します。その他の例については、使用している動作環境に対応する SAS ドキュメントを参照してください。

Windows
```
libname mydata 'c:\my documents\sasfiles';
```

UNIX
```
libname mydata '/u/myid/sasfiles';
```

z/OS
```
libname mydata 'edc.company.sasfiles';
```

LIBNAME ステートメントを使用してライブラリ参照名を割り当てると、割り当てを示す NOTE メッセージが SAS ログに書き込まれます。この NOTE メッセージには動作環境での SAS ライブラリの物理名も含まれます。

永久ライブラリと一時ライブラリへのライブラリ参照名の使用

SAS ライブラリにライブラリ参照名が割り当てられている場合、SAS セッション全体を通じて、ライブラリに格納されている SAS ファイルへのアクセスにライブラリ参照名を使用できます。ライブラリ参照名と SAS ライブラリ間の関連付けは、SAS セッションの間のみ、またはライブラリ参照名を変更するか別の LIBNAME ステートメントで中止するまで持続します。

SAS セッションの開始時、ライブラリ参照名 WORK が特別な SAS ライブラリに自動的に割り当てられます。通常、WORK ライブラリ内のファイルは一時ファイルです。つまり、SAS セッションの開始時に WORK ライブラリは初期化され、セッションの終了時に WORK ライブラリ内のファイルはすべて削除されます。そのため、WORK ライブラリは、後の SAS セッションのために保存しておく必要がない SAS ファイルを格納するのに便利な場所です。セッション終了時の WORK ライブラリのファイルの自動削除により、ディスク領域の無駄を省くことができます。

WORK ライブラリ以外の任意の SAS ライブラリに格納されたファイルは、通常、永久ファイルです。つまり、1 つの SAS セッションから次の SAS セッションへと移っても使用可能です。複数の SAS セッションで使用する場合は、SAS ファイルを永久ライブラリに保存してください。

SAS ライブラリにファイルを格納する

SAS ファイルについて

すべての SAS ファイルは SAS ライブラリに格納されます。SAS ファイルは、SAS によって作成、整理、管理される、特別に構造化されたファイルです。ファイルは、SAS ライブラリに特殊な種類のメンバとして存在します。次に SAS ファイルの例を示します。

- SAS データセット(SAS データファイルまたは SAS データビューとして表示可能)
- SAS カタログ
- SAS/ACCESS ディスクリプタファイル
- コンパイル済み DATA ステッププログラム

注: SAS ステートメントを含むファイルは、通常、SAS ファイルとは見なされません (SAS セッション時に作成されたファイルを含む)。たとえば、ディレクトリベースの動作環境で.sas ファイルは、通常、プログラムを含むテキストファイルですが SAS ファイルとは見なされません。

SAS データセットについて

SAS データセットとは、SAS ライブラリに格納されている SAS ファイルのことで、ディスクリプタ情報で構成されています。ディスクリプタ情報により SAS データセットの属性とそのコンテンツ、およびオブザベーション(行)と変数(列)のテーブルとして構成されるデータ値が特定されます。SAS データセットは、SAS データファイルまたは SAS データビューのいずれかです。

ディスクリプタ情報とオブザベーションが同じ物理的場所にある場合、そのデータセットはメンバタイプ DATA を持つ SAS データファイルです。SAS データファイルにはインデックスを関連付けることができます。インデックスの 1 つの目的は、WHERE 処理のパフォーマンスを最適化することです。基本的に、インデックスには特定の変数の値が昇順で含まれます。インデックスには、それらの値の、SAS データファイルのオブザベーション内での場所に関する情報も含まれます。

ディスクリプタとオブザベーションが個別に格納されている場合、それらはメンバタイプ VIEW を持つ SAS データビューを形成します。SAS データビューのオブザベーションは、SAS データファイル、外部データベース、または外部ファイルに保存できます。ディスクリプタには、データの場所、および処理するオブザベーションと変数に関する情報が含まれています。ビューは、SAS データファイルと同様に使用します。大量のデータのサブセットのみ必要な場合、ビューを使用できます。記憶域の節約に加え、ビューを使用すると、データへの変更が自動的に反映されるため管理が容易になります。SAS データビューには、次の 3 種類があります。

- DATA ステップビュー
- SAS/ACCESS ビュー
- PROC SQL ビュー

注: 通常、SAS データビューは SAS データファイルと同様の動作をします。このドキュメントの他のトピックでは、SAS データセットの 2 つの種類を区別しません。

その他の SAS ファイルについて

SAS データセットに加え、SAS ライブラリには次の種類の SAS ファイルを含めることができます。

SAS カタログ
: 多くの種類の情報を、カタログエントリと呼ばれる個別の単位で格納する SAS ファイルを指定します。各エントリは、エントリ名とエントリタイプによって区別されます。カタログエントリには、キー定義などのシステム情報を含むものがあります。それ以外に、ウィンドウ定義、ヘルプウィンドウ、出力形式、入力形式、マクロ、グラフィック出力に関するアプリケーション情報を含むカタログエントリもあります。SAS カタログは、メンバタイプ CATALOG を持ちます。

SAS/ACCESS ディスクリプタ
: 外部データベースのレイアウトに関する情報を含む SAS ファイルを指定します。SAS はこの情報を使用して、オブザベーションが外部データベースに保存されている SAS データビューを構築します。アクセスディスクリプタはメンバタイプ ACCESS を持ちます。

コンパイル済み DATA ステッププログラム
: コンパイルされて SAS ライブラリに格納されている DATA ステップを含む SAS ファイルを指定します。コンパイル済み DATA ステッププログラムは、メンバタイプ PROGRAM を持ちます。

SAS データセット以外の、すべての SAS ファイルの詳細は、このセクションの範囲外です。SAS ファイルの詳細については、*SAS Language Reference: Concepts* を参照してください。

SAS ライブラリの SAS データセットを参照する

データセット名について

各 SAS データセットは、*libref.filename* の形式の 2 レベル名を持っています。ファイルの 2 レベル名を使用して、いつでもファイルを参照できます。ただし、1 レベル名 (*filename*)を使用してファイルを参照することもできます。デフォルトでは、1 レベル名は、一時 SAS ライブラリのライブラリ参照名 WORK を使用するファイルを参照します。

注: このセクションでは、永久ファイルと一時ファイル、および 1 レベル名と 2 レベル名を区別して説明します。このドキュメントの他のトピック、および大部分の SAS ドキュメントでは、WORK ライブラリ参照名の通常の使用を前提とし、1 レベル名で参照されるファイルを一時ファイルと呼びます。2 レベル名で参照されるファイルは永久ファイルと見なされます。

動作環境の情報
: 一時ファイルと永久ファイルの作成方法については、使用している動作環境のベンダが提供するドキュメントより情報が提供されます。SAS では、WORK ライブラリのファイルは、NOWORKINIT オプションと NOWORKTERM オプションを指定しない限り一時ファイルです。その他すべての SAS ライブラリにあるファイルは永久ファイルです。ただし、使用している動作環境によってはライブラリとファイルの認識が異なる場合があります。たとえば、動作環境によっては、ログオフ時に削除される一時ディレクトリまたは一時 z/OS データセットを作成できる場合があります。使用している動作環境での構造が削除される場合に、SAS ライブラリのすべての

ファイルは削除されるため、動作環境の SAS ライブラリの解釈によって、ライブラリが 1 つのセッションから次のセッションへ保持されるかどうかが決定されます。

1 レベル名の使用

通常、1 レベル名を使用して SAS データセットが参照されると、SAS では、デフォルトで一時ライブラリのライブラリ参照名 WORK が使用されます。たとえば、次のプログラムは、WORK.GRADES という名前の一時 SAS データセットを作成します。

```
data grades;
   infile 'file-specification';
   input Name $ 1-14 Gender $ 15-20 Section $ 22-24 Grade;
run;
```

ただし、予約ライブラリ参照名 USER を割り当てることで、永久 SAS データセットの参照に 1 レベル名を使用できます。USER が割り当てられている場合に 1 レベル名で SAS データセットを参照すると、デフォルトで SAS は、永久 SAS ライブラリのライブラリ参照名 USER を使用します。たとえば、次のプログラムは、USER.GRADES という名前の永久 SAS データセットを作成します。ほかのライブラリ参照名と同様、ライブラリ参照名 USER は割り当てる必要があることに注意してください。

```
libname user 'SAS-library';

data grades;
   infile 'file-specification';
   input Name $ 1-14 Gender $ 15-20 Section $ 22-24 Grade;
run;
```

ここで 1 レベル名で SAS データセットを参照すると、SAS はライブラリ参照名 USER を検索します。SAS ライブラリに USER が割り当てられている場合、USER が 1 レベル名のデフォルトのライブラリ参照名になります。ライブラリ参照名 USER が割り当てられていない場合、SAS では、WORK が 1 レベル名のデフォルトのライブラリ参照名として使用されます。

USER が割り当てられている場合、WORK ライブラリの一時データセットにアクセスするには、2 レベル名を使用する必要があります。たとえば、USER が割り当てられている場合、データセット WORK.GRADES を出力するには、次のように PROC PRINT ステートメントで 2 レベル名を使用する必要があります。

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

USER が割り当てられている場合、1 つのみ変更して、同じプログラムで同じ名前のファイルを異なる SAS ライブラリで使用できます。2 レベル名を指定するかわりに、USER をそれぞれのケースで別々に割り当てます。たとえば、次のプログラムは、*SAS-library-1* の 5 つの SAS データセットを連結し、同じライブラリ内の 1 つの新しい SAS データセット WEEK に投入します。

```
   libname user 'SAS-library-1';

   data week;
      set mon tues wed thurs fri;
   run;
```

LIBNAME ステートメントのライブラリの名前を変更するだけで、同じ名前のファイルを別のライブラリ *SAS-library-2* に結合できます。

```
libname user 'SAS-library-2';

data week;
```

```
   set mon tues wed thurs fri;
run;
```

注: お使いのサイトで、SAS セッションの開始時にライブラリ参照名 USER が自動で割り当てられる場合があります。ライブラリ参照名が割り当てられているかどうかについては、SAS サポート担当者にお問い合わせください。

2 レベル名の使用

使用するライブラリ参照名が WORK、USER、またはユーザーが割り当てたその他のライブラリ参照名のどれであるかにかかわらず、いつでも 2 レベル名を使用して SAS データセットを参照できます。通常、WORK 以外のライブラリ参照名を伴う 2 レベル名は、永久 SAS データセットを参照します。

次のプログラムでは、LIBNAME ステートメントは、SAS 名 INTRCHEM と、*SAS-library* との間の接続を確立します。*SAS-library* は既存のデータセットまたはディレクトリの場所の物理名です。DATA ステップにより、SAS ライブラリ INTRCHEM に SAS データセット GRADES が作成されます。SAS は INPUT ステートメントを使用して *file-specification* 内の生データからデータセットを作成します。

```
libname intrchem 'SAS-library';

data intrchem.grades;
   infile 'file-specification';
   input Name $ 1-14 Gender $ 15-20 Section $ 22-24 Grade;
run;
```

SAS データセット INTRCHEM.GRADES の作成後、その 2 レベル名を使用して読み込みできます。次のプログラムは、ファイル INTRCHEM.GRADES を読み込み、元のデータセットのサブセットである、INTRCHEM.FRIDAY という名前の新しい SAS データセットを作成します。

```
data intrchem.friday;
   set intrchem.grades;
      if Section='Fri';
run;
```

SAS データセット INTRCHEM.FRIDAY を表示するには、次の PROC PRINT ステップを追加します。

```
   proc print data=intrchem.friday;
   run;
```

要約

ステートメント

LIBNAME *libref* '*SAS-library*';
大部分の動作環境で、*libref* と SAS ライブラリを関連付けます。SAS ライブラリの名前は、一重引用符または二重引用符で囲みます。

SAS データセットの参照

任意の SAS データセットを、*libref.filename* の形式の 2 レベル名で参照できます。デフォルトでは、ライブラリ参照名 USER が割り当てられている場合に、1 レベル名を使用して SAS データセットが参照されると、SAS はライブラリ参照名 USER を使用します。USER が割り当てられていない場合、SAS はライブラリ参照名 WORK を使用します。

詳細情報

LIBNAME ステートメント
: LIBNAME ステートメントの、ステートメントのオプションとデフォルトエンジン以外のエンジンの指定に関する情報を含む詳細については、“LIBNAME Statement” (*SAS Statements: Reference*)を参照してください。

動作環境
: 動作環境に固有の情報については、使用している動作環境に対応する SAS ドキュメントを参照してください。

SAS ファイル
: SAS ファイルの詳細情報については、*SAS Language Reference: Concepts* で確認できます。

  PROC SQL ビューの詳細については、*SAS SQL プロシジャユーザーガイド*を参照してください。

SAS ツール
: DATASETS プロシジャを含む、SAS データライブラリの管理に使用可能なツールについて学習するには、36 章, “SAS ライブラリの管理” (675 ページ)を参照してください。

USER ライブラリ参照名
: USER ライブラリ参照名の割り当てに、LIBNAME ステートメントのかわりに使用可能な USER=システムオプションの詳細については、“USER= System Option” (*SAS System Options: Reference*)を参照してください。

  注: ライブラリ参照名を、両方の方法で割り当て、またはいずれかの方法で複数回割り当てた場合、使用されるのは最後の定義です。

WORK ライブラリ
: WORKINIT システムオプションの詳細については、“WORKINIT System Option” (*SAS System Options: Reference*)を参照してください。WORKTERM システムオプションの詳細については、“WORKTERM System Option” (*SAS System Options: Reference*)を参照してください。これらのオプションは、WORK ライブラリが初期化されるタイミングを制御します。

36 章

SAS ライブラリの管理

SAS ライブラリの管理について **675**
目的 675
前提条件 675
ツールの選択 **675**
DATASETS プロシジャについて **676**
PROC DATASETS セッションについて **677**
要約 **678**
プロシジャ 678
ステートメント 678
詳細情報 **679**

SAS ライブラリの管理について

目的

このセクションでは、DATASETS プロシジャを含む、SAS ライブラリの管理に使用可能なツールについて学習します。後続のセクションでは、DATASETS プロシジャの使用方法を説明します。

前提条件

このセクションを使用する前に、35 章, “SAS ライブラリの基礎知識” (667 ページ)で説明した概念を理解している必要があります。

ツールの選択

SAS ファイルの数が増えるにつれ、SAS ライブラリの管理が必要になります。ライブラリの管理には、通常、次のようなルーチンタスクを実行する SAS プロシジャを使用します。

- ライブラリメンバとそのプロパティのリスト

- ファイルの名前変更、削除、移動
- 変数名の変更
- ライブラリとファイルのコピー

SAS プロシジャでは、動作環境コマンドを必要とせずに、SAS ライブラリに対して任意のファイル管理タスクを実行できます。

基本のファイル管理に複数の SAS ツールが使用可能です。これらの機能を単独で、または組み合わせて使用できます。

SAS エクスプローラ
ほとんどのファイル管理タスクのウィンドウが含まれ、SAS プログラムステートメントのサブミットは必要としません。たとえば、新規ライブラリや SAS ファイルを作成したり、既存の SAS ファイルを開いたり、ファイルの移動、コピー、削除など大部分のファイル管理タスクを実行できます。SAS **エクスプローラ**ウィンドウを使用するには、コマンドバーに **libname**、**catalog** または **dir** と入力するか、**ツールバー**メニューからエクスプローラアイコンを選択します。

CATALOG プロシジャ
COPY ステートメントと CONTENTS ステートメントによって、カタログ管理機能を提供します。

COPY プロシジャ
ライブラリのすべてのメンバまたはライブラリ内の個別のファイルをコピーします。

CONTENTS プロシジャ
ライブラリメンバとそのプロパティをリストします。

DATASETS プロシジャ
すべてのライブラリ管理機能が 1 つのプロシジャに統合されています。SAS エクスプローラを使用しない場合や、SAS System をバッチモードまたは対話型ラインモードで使用する場合は、このプロシジャで時間とリソースを節約できます。PROC DATASETS では、SAS ファイルのコピー、削除、変更など、多数の管理タスクを実行できます。

DATASETS プロシジャについて

DATASETS プロシジャは対話型プロシジャです。つまり、プロシジャは RUN ステートメントが実行された後もアクティブなままです。プロシジャを開始した後、予定のタスクがすべて完了するまで、SAS ライブラリ内のファイルの操作を継続できます。1 つのセッションで多数のタスクを完了させる場合、この機能を使用すると時間とリソースを節約できます。

次に、DATASETS プロシジャの知っておくべきいくつかの重要な機能を示します。

- PROC DATASETS ステートメントで入力ライブラリを指定できます。

  DATASETS プロシジャを開始するときに、プロシジャ入力ライブラリと呼ばれる入力ライブラリを指定できます。ファイルのソースとしてライブラリを指定しない場合、SAS により、デフォルトライブラリである一時ライブラリ WORK または USER ライブラリが使用されます。別の入力ライブラリを指定する場合、プロシジャを再び開始する必要があります。

- ステートメントは記述されている順序で実行されます。

  たとえば、SAS データセットのコンテンツを表示し、別のライブラリから 2 番目のデータセットをコピーして、2 つのデータセットのコンテンツを表示して比較することが

できます。前述の順序でタスクを実行するには、そのタスクを実行する SAS ステートメントを同じ順序で指定する必要があります。

- ステートメントのグループは、RUN ステートメントなしで実行できます。

  DATASETS プロシジャの場合に限り、次のプロシジャステートメントは、SAS により暗黙の RUN ステートメントがあるものとして認識され、サブミットするとすぐに実行されます。

  - APPEND ステートメント
  - CONTENTS ステートメント
  - MODIFY ステートメント
  - COPY ステートメント
  - PROC DATASETS ステートメント

  1 つのタスクに関連付けられている複数のステートメントが、前述のステートメントの 1 つを検出するまで読み込まれます。SAS は先行するステートメントのすべてをすぐに実行し、前述のステートメントをもう 1 つ検出するまで読み込みを続けます。最後のタスクを実行するには、RUN ステートメントまたは QUIT ステートメントをサブミットする必要があります。

  注: 対話型ラインモードを実行している場合、この機能を使用して、RUN ステートメントがサブミットされる前にステートメントがすでに実行されたことを示すメッセージを受信できます。

- RUN ステートメントは、PROC DATASETS ステップを停止しません。

  PROC DATASETS ステップを停止するには、QUIT ステートメント、新しい PROC ステートメントまたは DATA ステップをサブミットする必要があります。QUIT ステートメントをサブミットすると、まだ実行されていないステートメントがすべて実行され、プロシジャが終了します。

PROC DATASETS セッションについて

次の例は、通常のセッションにおける PROC DATASETS の動作を示しています。例では、ある SAS ライブラリからのファイルを使用して、別の SAS ライブラリにテストファイルを作成します。データセットがコピーされてそのコンテンツが記述されるため、出力を視覚的にチェックして、変数がテストライブラリの既存のファイルと互換性があるか確認できます。

次のプログラムは、どのステートメントが 1 タスクとして実行されるか示すために、グループにまとめられています。SAS によるタスクとアクションには、プログラムでの出現順序で番号が付けられます。

```
proc datasets library=test89; 1

   copy in=realdata out=test89; 2
   select income88;

   contents data=income88; 3
run;

   modify income88; 4
   rename Sales=Sales88;
```

```
quit; 5
```

次のリストは、前述のプログラムの番号付き項目に対応しています。

1 DATASETS プロシジャを開始し、プロシジャ入力ライブラリとして TEST89 を指定します。

2 データセット INCOME88 を SAS ライブラリ REALDATA から SAS ライブラリ TEST89 にコピーします。SAS はこれらのステートメントを 1 つのタスクとして認識します。SAS で CONTENTS ステートメントが読み込まれると、すぐに COPY ステートメントが実行され、INCOME88 がライブラリ TEST89 にコピーされます。CONTENTS ステートメントが暗黙の RUN ステートメントとして処理されることにより、COPY ステートメントが実行されます。この処理は、SAS をウィンドウ環境で実行しているとよりわかりやすくなります。

3 データセットのコンテンツが記述されます。出力を視覚的にチェックすることにより、変数が既存の SAS データセットと互換性があるか確認できます。SAS が RUN ステートメントを検出すると、INCOME88 のコンテンツが記述されます。前のタスクが実行されているため、プロシジャ入力ライブラリ TEST89 でデータセットが検出されます。

ユーザーはコンテンツを視覚的にチェックしたあと、変数 Sales の名前を変更する必要があると決定します。DATASETS プロシジャがアクティブなままなため、さらにステートメントをサブミットできます。

4 変数 Sales の名前を Sales88 に変更します。

5 DATASETS プロシジャを停止します。最後の 2 つのステートメントが実行され、DATASETS プロシジャを終了します。

要約

プロシジャ

PROC DATASETS <LIBRARY=*libref*>;
プロシジャを開始し、プロシジャによって処理されるライブラリ、つまりプロシジャ入力ライブラリを指定します。LIBRARY=オプションを指定しない場合、デフォルトは、WORK ライブラリまたは USER ライブラリです。PROC DATASETS では、プロシジャのサブミット時に、出力指定リストが作成されます。

ステートメント

QUIT;
先行する未実行のステートメントをすべて実行し、プロシジャを終了します。

RUN;
プロシジャを終了せずに、先行する未実行のステートメントのグループを実行します。

詳細情報

DATASETS プロシジャ
: そのメンバがプライマリデータセットである SAS ライブラリを DATASETS プロシジャを使用して管理する方法について学習するには、次のセクションを参照してください。

- 37 章, “SAS データセットの情報の表示” (681 ページ)。
- 38 章, “SAS データセット名と変数属性の変更” (693 ページ)。
- 39 章, “SAS データセットのコピー、移動、削除” (705 ページ)。

SAS ウィンドウ環境
: SAS ウィンドウ環境を介した SAS ファイルの管理の詳細については、41 章, “SAS ウィンドウ環境の使用” (731 ページ)を参照してください。

動作環境コマンド
: 動作環境コマンドを使用した SAS ファイルの管理の詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

37 章

SAS データセットの情報の表示

SAS データセットの情報の表示について **681**
目的 681
前提条件 682
例で使用される入力データライブラリ **682**
SAS ライブラリのディレクトリリストを表示する **682**
ディレクトリリストについて 682
ライブラリのすべてのファイルのリスト 682
同じメンバタイプを持つファイルのリスト 684
SAS データセットのコンテンツ情報を表示する **684**
DATASETS プロシジャの SAS データセットへの使用 684
1 つのデータセットのコンテンツのリスト 685
ライブラリのすべてのデータセットのコンテンツのリスト 687
異なる形式でコンテンツ情報を表示する **688**
要約 **690**
プロシジャ 690
DATASETS プロシジャステートメント 690
詳細情報 **691**

SAS データセットの情報の表示について

目的

SAS データセットのライブラリを作成すると、動作環境におけるライブラリの保存場所、データセットが作成された方法と作成日時、およびコンテンツの定義方法に関する情報が SAS により作成されて保持されます。DATASETS プロシジャを使用すると、データセットのコンテンツを表示したり、追加のドキュメントを参照したりせずに、この情報を確認できます。

このセクションでは、SAS ライブラリと SAS データセットに関する次の情報を取得する方法について学習します。

- SAS ライブラリに含まれている SAS ファイルの名前と種類
- SAS データセットにある変数の名前と属性
- 動作環境の記憶域パラメータに関する要約情報

- SAS データセットの履歴と構造に関する要約情報

前提条件

このセクションを使用する前に、次のセクションで説明した概念を理解している必要があります。

- 35 章, “SAS ライブラリの基礎知識” (667 ページ)
- 36 章, “SAS ライブラリの管理” (675 ページ)

例で使用される入力データライブラリ

このセクションの例は、米国の天候に関する情報を含む SAS ライブラリを使用します。データセット作成の DATA ステップは、“USCLIM データセット” (797 ページ)に示されています。DATA ステップではカタログエントリは作成されません。図 37.1 (683 ページ) で示されているカタログには、実例のためにエントリが追加されています。

SAS ライブラリのディレクトリリストを表示する

ディレクトリリストについて

ディレクトリリストは、SAS ライブラリにあるファイルのリストです。それぞれのファイルはメンバと呼ばれ、各メンバは、SAS によって割り当てられているメンバタイプを持っています。メンバタイプは、DATA や CATALOG などの SAS ファイルの種類を示します。ステートメントが処理されるとき、SAS は指定されたファイルを検索するだけでなく、ファイルがそのステートメントによって処理可能なメンバタイプを持っていることを確認します。

ディレクトリリストには、2 つの主要な部分があります。

- ディレクトリ情報
- ライブラリメンバとそのメンバタイプのリスト

ライブラリのすべてのファイルのリスト

ライブラリにある全メンバのディレクトリリストを取得するには、LIBRARY=オプションを指定した PROC DATASETS ステートメントのみが必要です。たとえば、次のステートメントは天候情報を含むライブラリのディレクトリリストを**結果**ウィンドウに送信します。LIBNAME ステートメントにより、ライブラリ参照名 USCLIM がこのライブラリへ割り当てられます。

```
libname usclim 'SAS-library';

proc datasets library=usclim;
```

次の出力はディレクトリリストを示しています。

図 37.1 ライブラリ USCLIM のディレクトリリスト

The SAS System

Directory	
Libref	USCLIM
Engine	V9
Physical Name	c:\Users\userid\climate
Filename	c:\Users\userid\climate

#	Name	Member Type	File Size	Last Modified
1	BASETEMP	CATALOG	3830784	04/16/2013 13:22:20
2	HIGHTEMP	DATA	131072	04/16/2013 13:36:54
3	HURRICANE	DATA	131072	04/16/2013 13:36:54
4	LOWTEMP	DATA	131072	04/16/2013 13:36:54
5	REPORT	CATALOG	2077696	04/16/2013 13:22:20
6	TEMPCHNG	DATA	131072	04/16/2013 13:36:54

出力に示されている項目を次にリストします。

Directory
ライブラリの物理名とライブラリ参照名を表示します。一部の動作環境では、追加の情報と別の情報の両方が提供される場合があります。

Name
ファイルに割り当てられている第 2 レベルの SAS メンバ名を含みます。ファイルのメンバタイプがそれぞれ異なる場合、1 つのライブラリに同じ名前のファイルを 2 つ持つことができます。

Member Type
SAS ファイルのメンバタイプを示します。最も一般的なメンバタイプは、DATA と CATALOG です。たとえば、ライブラリ USCLIM には、タイプ CATALOG の 2 つのカタログとタイプ DATA の 4 つのデータセットが含まれています。

File Size
ファイルのサイズを示します。

Last Modified
ファイルが最後に変更された日付を示します。

注: PROC DATASETS を最初に実行した後、DATASETS プロシジャが続いて実行されます。PROC DATASETS ステートメントを再実行せずに、さらにプロシジャステートメントを実行できます。DATASETS プロシジャを終了するには、`QUIT;` ステートメントを実行します。

同じメンバタイプを持つファイルのリスト

ディレクトリリストに特定のタイプの SAS ファイルのみを表示するには、PROC DATASETS ステートメントで MEMTYPE=オプションを使用します。次のステートメントはデータセットに関する情報のみを含む USCLIM のリストを生成します。

```
proc datasets library=usclim memtype=data;
```

次の出力は、USCLIM に格納されているデータセット(メンバタイプ DATA)に関する情報を示します。

図 37.2 ライブラリ USCLIM のデータセットのみのディレクトリ

The SAS System

Directory	
Libref	USCLIM
Engine	V9
Physical Name	c:\Users\userid\climate
Filename	c:\Users\userid\climate

#	Name	Member Type	File Size	Last Modified
1	HIGHTEMP	DATA	131072	04/16/2013 13:36:54
2	HURRICANE	DATA	131072	04/16/2013 13:36:54
3	LOWTEMP	DATA	131072	04/16/2013 13:36:54
4	TEMPCHNG	DATA	131072	04/16/2013 13:36:54

注: このドキュメントの例では、PROC DATASETS を使用した SAS データセットのみの管理に焦点を当てています。MEMTYPE=を指定することで、その他のメンバタイプもリストに表示できます。たとえば、MEMTYPE=CATALOG では、SAS カタログのみが表示されます。

SAS データセットのコンテンツ情報を表示する

DATASETS プロシジャの SAS データセットへの使用

オブザベーションを表示せずに SAS データセットのコンテンツを確認するには、DATASETS プロシジャで CONTENTS ステートメントを使用します。CONTENTS ステートメントとそのオプションを使用して、データセットに関する詳細情報、および変数とその属性のリストを得ることができます。

1 つのデータセットのコンテンツのリスト

SAS ライブラリ USCLIM には、4 つのデータセットが含まれています。データセット TEMPCHNG には、気温が著しく変化するデータが含まれます。次のプログラムは、データセット TEMPCHNG の変数を表示します。

```
proc datasets library=usclim memtype=data;
   contents data=tempchng;
run;
```

CONTENTS ステートメントからの出力には、TEMPCHNG データセットに関する情報が生成されます。DATA=オプションには、データセットの名前を指定します。次の出力は、CONTENTS ステートメントからの結果を示しています。

図 37.3 データセット TEMPCHNG に関する CONTENTS ステートメントからの出力

The SAS System

The DATASETS Procedure

Data Set Name	USCLIM.TEMPCHNG	Observations	5
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	04/05/2013 09:56:16	Observation Length	56
Last Modified	04/05/2013 09:56:16	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1167
Obs in First Data Page	5
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\Users\userid\climate\tempchng.sas7bdat
Release Created	9.0401B0
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat
2	Date	Num	8	DATE9.	DATE7.
6	Diff	Num	8		
4	End_f	Num	8		
5	Minutes	Num	8		
3	Start_f	Num	8		
1	State	Char	13		$CHAR13.

CONTENTS ステートメントからの出力は動作環境によって異なることに注意してください。

次のリストでは、CONTENTS ステートメントを PROC DATASETS と使用する際に必要となる場合がある情報を記述しています。

DATASETS プロシジャのヘッダー
フィールド名を含みます。そのデータセットに該当するものがない場合、フィールドは空になります。フィールド名を次に示します。

Data Set Name
データセットに割り当てられる 2 レベル名です。

Member Type
ライブラリメンバの種類です。

Engine
データセットの読み書きに SAS が使用するアクセス方法です。

Created
データセットが作成された日付です。

Last Modified
データセットが最後に変更された日付です。

Protection
データセットが READ、WRITE、または ALTER 操作にパスワード保護されているかどうかを示します。

Data Set Type
メンバタイプ DATA のファイルにのみ適用されます。このフィールドに情報がある場合、データセットに、SAS 統計プロシジャで使用する特殊なオブザベーションと変数が含まれていることを示します。

Label
LABEL=データセットオプションで指定される、データセットを特定するための説明的な情報です。

Data Representation
特定の動作環境にデータが格納される形式です。

Encoding
コード化文字セットのコード値へのマッピングを指定します。文字セット内の各文字が、固有の数値表現にマップされます。

Observations
現在データセットにあるオブザベーションの総数です。

Variables
データセットの変数の数を示します。

Indexes
データセットのインデックスの数です。

Observation Length
各オブザベーションのバイト単位の長さです。

Deleted Observations
削除としてマークされたオブザベーションの数です(該当する場合)。

Compressed
データが固定長レコードか、または可変長レコードか示します。データセットが圧縮済みの場合、追加のフィールドで、新しいオブザベーションがデータセットの末尾に追加されるか、またはデータセット内の未使用領域に書き込まれるか

が示されます。また、データセットがシーケンシャルアクセスのみでなく、オブザベーション番号によってランダムにアクセス可能かどうかが示されます。

Sorted
データセットが並べ替え済みかどうかを示します。

Engine/Host Dependent Information
ファイルの読み書きのメカニズムであるエンジンに関する情報と、動作環境によるデータセットの格納方法をリストで表示します。エンジンによって、このセクションの出力は異なります。詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

Alphabetical List of Variables and Attributes
データセット内のすべての変数名をアルファベット順でリストにし、変数の定義時に割り当てられた属性を記述します。次に、属性の説明を示します。

#
オブザベーションの変数の論理位置です。これは変数の定義時に割り当てられた番号です。

Variable
変数の名前です。

Type
変数が文字または数値かを示します。

Len
変数のバイト単位の長さです。

Format
変数の出力形式です。

Informat
変数の入力形式です。

さらに、該当する場合に、出力には次の情報を記述するテーブルが表示されます。

- インデックス付けされた変数のインデックス
- 定義されているすべての一貫性制約
- 並べ替え情報

ライブラリのすべてのデータセットのコンテンツのリスト

DATA=オプションでキーワード _ALL_ を指定して、ライブラリ内のすべてのデータセットのコンテンツをリストで表示できます。次のステートメントは、ライブラリのディレクトリリストとディレクトリ内の各データセットのコンテンツリストを生成します。

```
   contents data=_all_;
run;
```

ディレクトリリストのみを作成するには、NODS オプションを CONTENTS ステートメントに追加します。次のステートメントはディレクトリリストを生成しますが、個別のデータセットのコンテンツリストを非表示にします。プロシジャ入力ライブラリのディレクトリリストを生成する場合は、次の形式を使用します。

```
   contents data=_all_ nods;
run;
```

別のライブラリのディレクトリリストを生成する場合は、ライブラリ参照名を含めます。次の例は、ライブラリ STORM を指定します。

```
   contents data=storm._all_ nods;
```

```
run;
```

異なる形式でコンテンツ情報を表示する

デフォルト以外の様式のコンテンツリストを生成するには、CONTENTS ステートメントで VARNUM オプションまたは SHORT オプションを使用します。たとえば、次のステートメントは、変数が定義された順序(つまり、データセット内の論理位置)での変数の名前のリストを生成します。

```
   contents data=tempchng varnum;
run;
```

CONTENTS ステートメントでデータセット TEMPCHNG が指定され、含まれている VARNUM オプションにより、変数はその論理位置の順序でリストされます。(デフォルトでは、CONTENTS ステートメントは変数をアルファベット順のリストにします。)

次の出力は、コンテンツを変数番号の順序で示します。

図 37.4 データセット TEMPCHNG の変数番号順コンテンツ

The SAS System

The DATASETS Procedure

Data Set Name	USCLIM.TEMPCHNG	Observations	5
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	04/05/2013 09:56:16	Observation Length	56
Last Modified	04/05/2013 09:56:16	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1167
Obs in First Data Page	5
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\Users\userid\climate\tempchng.sas7bdat
Release Created	9.0401B0
Host Created	W32_7PRO

Variables in Creation Order

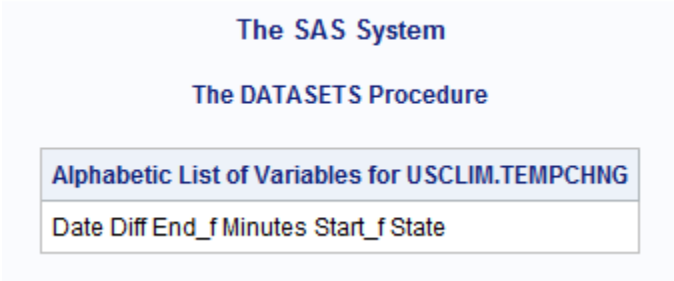
#	Variable	Type	Len	Format	Informat
1	State	Char	13		$CHAR13.
2	Date	Num	8	DATE9.	DATE7.
3	Start_f	Num	8		
4	End_f	Num	8		
5	Minutes	Num	8		
6	Diff	Num	8		

コンテンツリストのすべての情報を必要としない場合、CONTENTS ステートメントで SHORT オプションを使用して、簡易版を要求できます。次のステートメントは、簡易版を要求し、QUIT ステートメントを発行することで DATASETS プロシジャを終了します。

```
   contents data=tempchng short;
run;
quit;
```

次の出力は、TEMPCHNG データセットの変数名のリストを表示します。

図 37.5 データセット TEMPCHNG の変数名のみのリスト


The SAS System

The DATASETS Procedure

Alphabetic List of Variables for USCLIM.TEMPCHNG
Date Diff End_f Minutes Start_f State

要約

プロシジャ

PROC DATASETS <LIBRARY=*libref* <MEMTYPE=*mtype(s)*>>;

MEMTYPE=オプションでは、特定の種類(複数指定可)の SAS ファイルに処理を制限し、指定したメンバタイプの SAS ファイルに限定してライブラリディレクトリをリストします。

DATASETS プロシジャステートメント

CONTENTS <DATA=<*libref*>.*SAS-data-set*> <NODS> <SHORT> <VARNUM> ;

ライブラリにある特定の SAS データセットのコンテンツを記述します。デフォルトのデータセットは、そのジョブまたはセッションで最後に作成されたデータセットです。PROC DATASETS の CONTENTS ステートメントの場合、DATA=が指定されるときのデフォルトのライブラリ参照名はプロシジャ入力ライブラリです。ただし、CONTENTS プロシジャの場合、デフォルトのライブラリ参照名は WORK または USER のいずれかです。

SAS 出力にライブラリのディレクトリリストのみを生成するには、DATA=オプションで NODS オプションにキーワード_ALL_を指定して使用します。つまり、NODS オプションにより個別のファイルのコンテンツが非表示にされます。DATA=オプションで 1 つの SAS データセットのみ指定する場合、NODS オプションを使用できません。

SHORT オプションは、SAS データセットの変数名のアルファベット順リスト、インデックス情報、一貫性制約情報、並べ替え情報のみ生成します。

VARNUM オプションは、変数が定義された順序(つまり、データセット内の論理位置)での変数名のリストを生成します。デフォルトでは、CONTENTS ステートメントは変数をアルファベット順のリストにします。

詳細情報

CATALOG プロシジャ
: CATALOG プロシジャを使用して、カタログに関するコンテンツ情報を取得できます。詳細については、“CATALOG” (*Base SAS Procedures Guide*)を参照してください。

DATASETS プロシジャ
: DATASETS プロシジャは、SAS ファイルを管理するユーティリティプロシジャです。DATASETS プロシジャおよび CONTENTS ステートメントの詳細については、“DATASETS” (*Base SAS Procedures Guide*)を参照してください。

CONTENTS プロシジャ
: CONTENTS プロシジャでは、SAS データセットのコンテンツを表示したり、SAS ライブラリのディレクトリを書き込んだりします。詳細については、“CONTENTS” (*Base SAS Procedures Guide*)を参照してください。

ウィンドウ環境
: SAS データセットに関する情報の取得にウィンドウ環境を使用する際の詳細については、41 章, “SAS ウィンドウ環境の使用” (731 ページ)を参照してください。

38 章

SAS データセット名と変数属性の変更

SAS データセット名と変数属性の変更について **693**
目的 693
前提条件 694
例で使用される入力データライブラリ **694**
SAS データセットの名前の変更 **694**
変数属性の変更 **695**
変数属性の変更方法について 695
変数の名前の変更 696
出力形式の割り当て、変更、または削除 696
ラベルの割り当て、変更、または削除 699
要約 **702**
DATASETS プロシジャステートメント 702
詳細情報 **703**

SAS データセット名と変数属性の変更について

目的

SAS では、新しいデータセットを作成することなしに、データセット名と変数の属性を変更できます。このセクションでは、DATASETS プロシジャでステートメントを使用して次の操作を行う方法を学習します。

- データセットの名前の変更
- 変数の名前の変更
- 変数の出力形式の変更
- 変数ラベルの変更

このセクションでは、DATASETS プロシジャを使用したデータセットの変更に焦点を当てています。ただし、説明されるステートメントとオプションのいくつかを使用して、その他の種類の SAS ファイルを変更することもできます。

注: DATASETS プロシジャを使用して、オブザベーションの値の変更、変数の作成や削除、変数の種類や長さの変更を行うことはできません。これらの変更は、DATA ステップステートメントと関数を使用して行われます。

前提条件

このセクションを使用する前に、次のセクションで説明した概念を理解している必要があります。

- 35 章, “SAS ライブラリの基礎知識” (667 ページ)
- 36 章, “SAS ライブラリの管理” (675 ページ)
- 37 章, “SAS データセットの情報の表示” (681 ページ)

例で使用される入力データライブラリ

このセクションの例は、米国の天候に関する情報を含む SAS ライブラリを使用します。SAS ライブラリにデータセットを作成する DATA ステップは、“CLIMATE、PRECIP および STORM データセット” (798 ページ)に示されています。

SAS データセットの名前の変更

効率的なライブラリ管理を行うために、データセットの名前の変更が必要とされる場合が多くあります。たとえば、データセットをアーカイブするとき、または新しいデータ値を追加するときにデータセットの名前を変更します。

DATASETS プロシジャの CHANGE ステートメントを使用して、同じライブラリ内の 1 つ以上のデータセットの名前を変更できます。CHANGE ステートメントの構文は、次のとおりです。

CHANGE *old-name=new-name*;

old-name
SAS データセットの現在の名前を示します。

new-name
データセットに指定する予定の名前を示します。

この例は、米国の天候に関する情報を含む SAS ライブラリ USCLIM 内の、2 つのデータセットの名前を変更します。次のプログラムは、DATASETS プロシジャを開始します。次に、データセット HIGHTEMP の名前を USHIGH に、データセット LOWTEMP の名前を USLOW に変更します。

```
libname usclim 'SAS-library';

proc datasets library=usclim;
   change hightemp=ushigh lowtemp=uslow;
run;
```

これらのステートメントが処理される際に、メッセージが SAS ログに送信されます。メッセージにより、データセットの名前が変更されたことが確認できます。

ログ 38.1 *ライブラリ USCLIM 内のデータセットの名前の変更*

```
38   proc datasets library=usclim; 39      change hightemp=ushigh lowtemp=uslow;
40   run; NOTE: Changing the name USCLIM.HIGHTEMP to USCLIM.USHIGH
(memtype=DATA).NOTE: Changing the name USCLIM.LOWTEMP to USCLIM.USLOW
(memtype=DATA).
```

次のプログラムでは DATASETS プロシジャが実行されます。これにより変更結果を確認できます。

```
proc datasets library=usclim;
run;
```

次の出力はライブラリについての情報を示しています。

図 38.1 *ライブラリ USCLIM 内のデータセットの名前の変更*

The SAS System

Directory	
Libref	USCLIM
Engine	V9
Physical Name	c:\Users\userid\climate
Filename	c:\Users\userid\climate

#	Name	Member Type	File Size	Last Modified
1	BASETEMP	CATALOG	3830784	04/16/2013 13:22:20
2	HURRICANE	DATA	131072	04/16/2013 13:36:54
3	REPORT	CATALOG	2077696	04/16/2013 13:22:20
4	TEMPCHNG	DATA	131072	04/16/2013 13:36:54
5	USHIGH	DATA	131072	04/16/2013 13:36:54
6	USLOW	DATA	131072	04/16/2013 13:36:54

変数属性の変更

変数属性の変更方法について

SAS データセットの変数にはそれぞれ、名前、種類、長さ、出力形式、入力形式、ラベルなどの属性があります。これらの属性を使用して、変数を特定したり、SAS における変数の使用方法を定義したりできます。

DATASETS プロシジャで MODIFY ステートメントと従属ステートメントを使用して、特定の属性の割り当て、変更、または削除を行うことができます。たとえば、MODIFY ステートメントと従属ステートメントを使用して次のタスクを実行できます。

- 変数の名前の変更
- 出力形式の割り当て、変更、削除による値の書き込み方法や表示方法の変更
- ラベルの割り当て、変更、または削除

注: MODIFY ステートメントを使用して変数の種類や長さなどの固定属性を変更することはできません。

変数の名前の変更

1 つ以上の一致変数名があるデータセットの結合の前などに、変数の名前の変更が必要な場合があります。.DATASETS プロシジャで MODIFY ステートメントとその従属 RENAME ステートメントを使用して、1 つ以上の変数の名前を変更できます。ステートメントの構文は、次のとおりです。

MODIFY *SAS-data-set*;

RENAME *old-name=new-name*;

SAS-data-set
名前を変更する変数が含まれる SAS データセットの名前を示します。

old-name
変数の現在の名前を示します。

new-name
変数に指定する予定の名前を示します。

この例は、SAS ライブラリ USCLIM にあるデータセット HURRICANE の 2 つの変数の名前を変更します。次のステートメントは変数名 State を Place に、変数名 Deaths を USDeaths に変更します。DATASETS プロシジャはすでにアクティブなため、PROC DATASETS ステートメントは必要ありません。

```
   modify hurricane;
   rename State=Place Deaths=USDeaths;
run;
```

次の出力に示すように、SAS ログメッセージにより、変数の名前が Place と USDeaths に変更されたことが確認できます。これらの変数に割り当てられているその他の属性はすべて、変更されずにそのままです。

ログ 38.2 *データセット HURRICANE の変数の名前の変更*

```
48      modify hurricane; 49      rename State=Place Deaths=USDeaths; NOTE:
Renaming variable State to Place.NOTE: Renaming variable Deaths to USDeaths.50
run; NOTE: MODIFY was successful for USCLIM.HURRICANE.DATA.
```

出力形式の割り当て、変更、または削除

SAS では、多くの SAS プロシジャによって出力に使用される、出力形式を割り当てて保存できます。出力形式を割り当て、変更または削除して、値の書き込み方法や表示方法を変更できます。DATASETS プロシジャで MODIFY ステートメントとその従属 FORMAT ステートメントを使用して、変数の出力形式を変更できます。変数の出力形式を、SAS 出力形式またはユーザーが定義して保存した出力形式のいずれかに変更

できます。また、出力形式を削除することもできます。これらのステートメントの構文は、次のとおりです。

MODIFY *SAS-data-set*;

FORMAT *variable(s)* <*format*>;

SAS-data-set
出力形式を変更する変数が含まれる SAS データセットの名前を示します。

variable
出力形式を割り当て、変更、削除する 1 つ以上の変数の名前を示します。

format
変数に指定する予定の出力形式を示します。出力形式を指定しない場合、指定の変数と関連付けられている出力形式がすべて削除されます。

出力形式の割り当てや変更を行う場合、次の規則に従ってください。

- 変数の名前を出力形式の前に記述します。
- 複数の変数に出力形式を割り当てる場合は、複数の変数の名前または略記法の変数リストを記述します。
- 句読文字を使用してリストの項目を区切らないでください。

次の FORMAT ステートメントでは、同一の FORMAT ステートメントにいくつかの変数と出力形式を含める方法を示します。

```
format Date1-Date5 date9. Cost1 Cost2 dollar4.2 Place $char25.;
```

変数 Date1 から Date5 までは略記法形式で記述されており、出力形式 DATE9.が 5 つの変数すべてに割り当てられています。変数 Cost1 と Cost2 は共通の出力形式の前に個別に記述されています。出力形式$CHAR25.が変数 Place に割り当てられています。

変数から出力形式を削除する場合、次の 2 つの規則が適用されます。

- 変数の名前のみを記述します。
- 同一の FORMAT ステートメントを使用して出力形式の割り当てや変更を行う場合、変数名をリストの最後に配置します。

次の例では、データセット HURRICANE の出力形式を変更する方法を示します。

```
      contents data=hurricane;
         modify hurricane;
         format Date monyy7. Millions;
      contents data=hurricane;
   run;
```

この例では、次の変更が行われます。

- 変数 Date の出力形式を月、日、および年の綴りどおりから月と年の略記法に変更
- 変数 Millions の出力形式の削除
- データセット HURRICANE の変更前と変更後のコンテンツの表示

FORMAT ステートメントはメッセージを SAS ログに送信しないため、変更が行われたことを確認する場合は、CONTENTS ステートメントを使用する必要があります。

2 つの CONTENTS ステートメントからの次の出力は、変更前と変更後のデータセットのコンテンツを示しています。変数 Date の出力形式は、WORDDATE18.から MONYY7.へ変更され、変数 Millions の出力形式は削除されています。

図 38.2 *データセット HURRICANE の変数の出力形式の変更: 変更前*

The SAS System

The DATASETS Procedure

Data Set Name	USCLIM.HURRICANE	Observations	6
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	04/16/2013 13:36:54	Observation Length	48
Last Modified	04/17/2013 07:25:49	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1361
Obs in First Data Page	6
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\Users\userid\climate\hurricane.sas7bdat
Release Created	9.0401B0
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
2	Date	Num	8	WORDDATE18.	DATE9.	
4	Millions	Num	8	DOLLAR6.		Damage
5	Name	Char	8			
1	Place	Char	14		$CHAR14.	
3	USDeaths	Num	8			

図 38.3 データセット HURRICANE の変数の出力形式の変更: 変更後

The SAS System

The DATASETS Procedure

Data Set Name	USCLIM.HURRICANE	Observations	6
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	04/16/2013 13:36:54	Observation Length	48
Last Modified	04/17/2013 07:37:02	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1361
Obs in First Data Page	6
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\Users\userid\climate\hurricane.sas7bdat
Release Created	9.0401B0
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
2	Date	Num	8	MONYY7.	DATE9.	
4	Millions	Num	8			Damage
5	Name	Char	8			
1	Place	Char	14		$CHAR14.	
3	USDeaths	Num	8			

ラベルの割り当て、変更、または削除

ラベルは、テーブル、プロットおよびグラフ内の変数を特定する説明的な情報です。通常は、ユーザーによって変数の作成時にラベルが割り当てられます。ユーザーによってラベルが割り当てられない場合、SAS では変数名がラベルとして使用されます。ただし、CONTENTS の出力では、ラベルが割り当てられていない場合、フィールドはブランクのままです。MODIFY ステートメントとその従属 LABEL ステートメントを使用して、ラベルの割り当て、変更、削除を行うことができます。これらのステートメントの構文は、次のとおりです。

MODIFY *SAS-data-set*;

LABEL *variable*=<'*label*'>;

SAS-data-set
ラベルを変更する変数が含まれる SAS データセットの名前を示します。

variable
ラベルを割り当て、変更、削除する変数の名前を示します。

label
変数に指定する予定の、1-256 文字のラベルを示します。ラベルが存在する場合にラベルを指定しないと、SAS によって現在のラベルが削除されます。

LABEL ステートメントを使用する場合、次の規則に従ってください。

- ラベルのテキストは、一重引用符または二重引用符で囲みます。ラベルに一重引用符が使用されている場合(例: アポストロフィ)、二重引用符でテキストを囲みます。
- ラベルはブランクを含め、256 文字以下に制限します。
- ラベルを削除するには、*variable*=' 'というように、ラベルのテキストにブランクを使用します。

SAS データセット HURRICANE では、次のステートメントは、変数 Millions のラベルを変更し、変数 Place にラベルを割り当てます。LABEL ステートメントは SAS ログにメッセージを送信しないため、変更が行われたことを確認するために CONTENTS ステートメントが指定されています。QUIT ステートメントにより、DATASETS プロシジャが終了されます。

```
      contents data=hurricane;
         modify hurricane;
            label Millions='Damage in Millions' Place='State Hardest Hit';
      contents data=hurricane;
   run;
   quit;
```

2 つの CONTENTS ステートメントからの次の出力は、変更前と変更後のデータセットのコンテンツを示しています。

図 38.4 *データセット HURRICANE の変数ラベルの変更: 変更前*

The SAS System

The DATASETS Procedure

Data Set Name	USCLIM.HURRICANE	Observations	6
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	04/16/2013 13:36:54	Observation Length	48
Last Modified	04/17/2013 07:37:02	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1361
Obs in First Data Page	6
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\Users\userid\climate\hurricane.sas7bdat
Release Created	9.0401B0
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
2	Date	Num	8	MONYY7.	DATE9.	
4	Millions	Num	8			Damage
5	Name	Char	8			
1	Place	Char	14		$CHAR14.	
3	USDeaths	Num	8			

図 38.5 データセット HURRICANE の変数ラベルの変更: 変更後

The SAS System

The DATASETS Procedure

Data Set Name	USCLIM.HURRICANE	Observations	6
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	04/16/2013 13:36:54	Observation Length	48
Last Modified	04/17/2013 08:00:42	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	2
First Data Page	1
Max Obs per Page	1361
Obs in First Data Page	6
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\Users\userid\climate\hurricane.sas7bdat
Release Created	9.0401B0
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
2	Date	Num	8	MONYY7.	DATE9.	
4	Millions	Num	8			Damage in Millions
5	Name	Char	8			
1	Place	Char	14		$CHAR14.	State Hardest Hit
3	USDeaths	Num	8			

要約

DATASETS プロシジャステートメント

CHANGE *old-name=new-name*;
old-name で指定される SAS データセットの名前を、*new-name* で指定される名前に変更します。1 つの CHANGE ステートメントを使用して、同じライブラリの複数のデータセットの名前を変更できます。新しい名前はすべて有効な SAS 名である必要があります。

MODIFY *SAS-data-set*;
変更する SAS データセットを特定します。MODIFY ステートメントで使用可能ないくつかの従属ステートメントを次に示します。

FORMAT *variable(s)* <*format*>;
1 つ以上の変数に対する出力形式の割り当て、変更、または削除を行います。*variable* 引数と *format* 引数を使用して変数と出力形式を指定します。出力形式の前に複数の変数を記述して、複数の変数に同じ出力形式を指定できます。出力形式を削除する場合は、*format* を指定しません。

LABEL *variable*=<'*label*'>;
variable で指定される変数のラベルを割り当て、変更、削除します。ラベルを削除するには、引用符の内側にブランクを配置します。

RENAME *old-name*=*new-name*;
old-name で指定される 1 つ以上の変数の名前を、*new-name* で指定される名前に変更します。1 つの RENAME ステートメントを使用して、同じデータセットの複数の変数の名前を変更できます。すべての名前は有効な SAS 名である必要があります。

詳細情報

入力形式と出力形式
入力形式と出力形式は、データの読み込みと表示に使用可能です。詳細については、*SAS Formats and Informats: Reference* を参照してください。

LABEL ステートメント
LABEL ステートメントでは、変数に説明ラベルが割り当てられます。詳細については、“LABEL Statement” (*SAS Statements: Reference*)を参照してください。

MODIFY ステートメント
DATASETS プロシジャの MODIFY ステートメントには、入力形式を変更したり、変数のインデックスの作成や削除を行う追加のステートメントがあります。詳細については、“DATASETS” (*Base SAS Procedures Guide*)を参照してください。

変数の名前の変更
RENAME=データセットオプションを使用、または DATA ステップで RENAME ステートメントを使用して、変数の名前を変更できます。詳細については、“RENAME= Data Set Option” (*SAS Data Set Options: Reference*)および“RENAME Statement” (*SAS Statements: Reference*)を参照してください。

変数
DATA ステップでの変数の作成方法および削除方法について学習するには、6 章, “SAS データセットから作成する” (91 ページ)を参照してください。

39 章

SAS データセットのコピー、移動、削除

SAS データセットのコピー、移動、削除について **705**
目的 705
前提条件 706
例で使用される入力データライブラリ **706**
SAS データセットのコピー **706**
プロシジャ入力ライブラリからのコピー 706
その他のライブラリからのコピー 708
特定の SAS データセットのコピー **710**
コピーするデータセットの選択 710
データセットのコピーからの除外 710
SAS ライブラリと SAS データセットの移動 **711**
ライブラリの移動 711
特定のデータセットの移動 712
SAS データセットの削除 **713**
削除するデータセットの指定 713
保存するデータセットの指定 714
SAS ライブラリのすべてのファイルの削除 **714**
要約 **715**
プロシジャ 715
DATASETS プロシジャステートメント 715
詳細情報 **716**

SAS データセットのコピー、移動、削除について

目的

SAS データセットのコピー、移動、および削除は、最も頻繁に行われるライブラリ管理タスクです。たとえば、これらのタスクを実行して、テストファイルの作成、バックアップの作成、ファイルのアーカイブ、および未使用のファイルを削除します。DATASETS プロシジャを使用して、SAS ライブラリ内のすべてのファイル、またはライブラリ内の特定のファイルに対して操作を行うことができます。

このセクションでは、DATASETS プロシジャを使用して次の操作を行う方法を学習します。

- ライブラリ全体のコピー
- 特定の SAS データセットのコピー
- 特定の SAS データセットの移動
- 特定の SAS データセットの削除
- ライブラリのすべてのファイルの削除

このセクションでは、DATASETS プロシジャを使用したデータセットのコピー、移動、削除に焦点をあてています。説明されるステートメントとオプションを使用して、その他の種類の SAS ファイルをコピー、移動、削除することもできます。

前提条件

このセクションを使用する前に、次のセクションで説明した概念を理解している必要があります。

- 35 章, “SAS ライブラリの基礎知識” (667 ページ)
- 36 章, “SAS ライブラリの管理” (675 ページ)
- 38 章, “SAS データセット名と変数属性の変更” (693 ページ)

例で使用される入力データライブラリ

このセクションの例は、米国とその他の国々の気象統計量の収集および格納に使用されるサンプルデータセットを含む、5 つの SAS ライブラリを使用します。ライブラリのライブラリ参照名は、PRECIP、USCLIM、CLIMATE、WEATHER、および STORM です。次の LIBNAME ステートメントにより、ライブラリ参照名が割り当てられます。

```
libname precip 'SAS-library-1';
libname usclim 'SAS-library-2';
libname climate 'SAS-library-3';
libname weather 'SAS-library-4';
libname storm 'SAS-library-5';
```

注: 各 LIBNAME ステートメントについて、*SAS-library* はその SAS ライブラリの場所のそれぞれの物理名です。すべてまたはいくつかの SAS データセットをあるライブラリから別のライブラリへコピーするには、入力ライブラリと出力ライブラリが異なる物理場所に存在する必要があります。

SAS ライブラリ CLIMATE、PRECIP および STORM にデータセットを作成する DATA ステップは、“CLIMATE、PRECIP および STORM データセット” (798 ページ)に示されています。SAS ライブラリ USCLIM にデータセットを作成する DATA ステップは、“USCLIM データセット” (797 ページ)に示されています。

SAS データセットのコピー

プロシジャ入力ライブラリからのコピー

DATASETS プロシジャで COPY ステートメントを使用して、すべてまたは一部の SAS データセットをあるライブラリから別のライブラリへコピーできます。データセットのコピ

一時、SAS により、ディスクリプタ情報を含めて各ファイルのコンテンツが複製され、各ライブラリのディレクトリ内の情報が更新されます。

注意:
処理中、SAS により自動的に入力ライブラリから同じ名前の出力データセットにデータが書き込まれます。重複したデータセット名がある場合、コピーを開始する前に警告メッセージは発せられません。 ライブラリに対して変更を行う前に、重複しているデータセット名がないか視覚的に確認するために、入力ライブラリと出力ライブラリのディレクトリリストを取得することが重要です。

プロシジャ入力ライブラリ(PROC DATASETS ステートメントによって指定される)からファイルをコピーするには、COPY ステートメントを使用します。COPY ステートメントの構文は、次のとおりです。

COPY OUT=*libref* <options>;

libref
ファイルのコピー先の SAS ライブラリのライブラリ参照名を指定します。出力ライブラリを指定する必要があります。

たとえば、ライブラリ PRECIP には降雪量と降雨量のデータセットが含まれ、ライブラリ CLIMATE には気温のデータセットが含まれています。次のプログラムはコンテンツをリストで表示するため、操作を行う前に視覚的に比較できます。

```
proc datasets library=precip;
   contents data=_all_ nods;
   contents data=climate._all_ nods;
run;
```

PROC DATASETS ステートメントによりプロシジャが開始され、プロシジャ入力ライブラリ PRECIP が指定されます。1 番目の CONTENTS ステートメントは、ライブラリ PRECIP のディレクトリリストを生成します。2 番目の CONTENTS ステートメントは、ライブラリ CLIMATE のディレクトリリストを生成します。

次の SAS 出力は 2 つのディレクトリリストを示しています。

図 39.1 *コピー前のディレクトリ PRECIP の確認*

The SAS System

The DATASETS Procedure

Directory	
Libref	PRECIP
Engine	V9
Physical Name	c:\Users\userid\precip
Filename	c:\Users\userid\precip

#	Name	Member Type	File Size	Last Modified
1	RAIN	DATA	131072	04/17/2013 08:26:48
2	SNOW	DATA	131072	04/17/2013 08:26:48

図 39.2 コピー前のディレクトリ CLIMATE の確認

The SAS System

The DATASETS Procedure

Directory	
Libref	CLIMATE
Engine	V9
Physical Name	c:\Users\userid\climate2
Filename	c:\Users\userid\climate2

#	Name	Member Type	File Size	Last Modified
1	HIGHTEMP	DATA	131072	04/17/2013 08:26:48
2	LOWTEMP	DATA	131072	04/17/2013 08:26:48

ディレクトリ内に重複する名前はないため、COPY ステートメントを発行して、目的とする出力結果を得ることができます。

```
   copy out=climate;
run;
```

次の SAS ログは、ライブラリ PRECIP のデータセットがライブラリ CLIMATE へコピーされるときのメッセージを示します。これで、データセット RAIN とデータセット SNOW が、PRECIP ライブラリと CLIMATE ライブラリに 1 つずつの 2 つになりました。

ログ 39.1 コピー時に SAS ログへ送信されるメッセージ

```
143     copy out=climate; 144  run; NOTE: Copying PRECIP.RAIN to CLIMATE.RAIN
(memtype=DATA).NOTE: There were 5 observations read from the data set
PRECIP.RAIN.NOTE: The data set CLIMATE.RAIN has 5 observations and 4
variables.NOTE: Copying PRECIP.SNOW to CLIMATE.SNOW (memtype=DATA).NOTE: There
were 3 observations read from the data set PRECIP.SNOW.NOTE: The data set
CLIMATE.SNOW has 3 observations and 4 variables.
```

その他のライブラリからのコピー

別の PROC DATASETS ステートメントを使用することなしに、プロシジャ入力ライブラリ以外のライブラリからコピーできます。これを行うには、COPY ステートメントで IN= オプションを使用して、プロシジャ入力ライブラリを上書きします。オプションの構文は、次のとおりです。

COPY OUT=*libref-1* IN=*libref-2*;

libref-1
ファイルのコピー先の SAS ライブラリのライブラリ参照名を指定します。

libref-2
ファイルのコピー元の SAS ライブラリのライブラリ参照名を指定します。

IN=オプションは、複数のライブラリを出力ライブラリにコピーする場合に便利なツールです。各入力ライブラリに対して、PROC DATASETS ステートメントを繰り返すことなしに、1 つの COPY ステートメントを使用できます。

たとえば、次のステートメントは、PRECIP、STORM、CLIMATE、USCLIM の各ライブラリを、ライブラリ WEATHER にコピーします。プロシジャ入力ライブラリは PRECIP で、これは前の PROC DATASETS ステートメントで指定されています。

```
   copy out=weather;
   copy in=storm out=weather;
   copy in=climate out=weather;
   copy in=usclim out=weather;
run;
```

次の SAS ログは、これらのライブラリからのデータセットがライブラリ WEATHER にまとめられたことを示しています。

ログ 39.2 *4 つのライブラリのライブラリ WEATHER へのコピー*

```
142     copy out=weather; NOTE: Copying PRECIP.RAIN to WEATHER.RAIN
(memtype=DATA).NOTE: There were 5 observations read from the data set
PRECIP.RAIN.NOTE: The data set WEATHER.RAIN has 5 observations and 4
variables.NOTE: Copying PRECIP.SNOW to WEATHER.SNOW (memtype=DATA).NOTE: There
were 3 observations read from the data set PRECIP.SNOW.NOTE: The data set
WEATHER.SNOW has 3 observations and 4 variables.143     copy in=storm
out=weather; NOTE: Copying STORM.HURRICANE to WEATHER.HURRICANE
(memtype=DATA).NOTE: There were 6 observations read from the data set
STORM.HURRICANE.NOTE: The data set WEATHER.HURRICANE has 6 observations and 5
variables.NOTE: Copying STORM.TORNADO to WEATHER.TORNADO (memtype=DATA).NOTE:
There were 5 observations read from the data set STORM.TORNADO.NOTE: The data
set WEATHER.TORNADO has 5 observations and 4 variables.144     copy in=climate
out=weather; NOTE: Copying CLIMATE.HIGHTEMP to WEATHER.HIGHTEMP
(memtype=DATA).NOTE: There were 6 observations read from the data set
CLIMATE.HIGHTEMP.NOTE: The data set WEATHER.HIGHTEMP has 6 observations and 4
variables.NOTE: Copying CLIMATE.LOWTEMP to WEATHER.LOWTEMP (memtype=DATA).NOTE:
There were 5 observations read from the data set CLIMATE.LOWTEMP.NOTE: The data
set WEATHER.LOWTEMP has 5 observations and 4 variables.NOTE: Copying
CLIMATE.RAIN to WEATHER.RAIN (memtype=DATA).NOTE: There were 5 observations read
from the data set CLIMATE.RAIN.NOTE: The data set WEATHER.RAIN has 5
observations and 4 variables.NOTE: Copying CLIMATE.SNOW to WEATHER.SNOW
(memtype=DATA).NOTE: There were 3 observations read from the data set
CLIMATE.SNOW.NOTE: The data set WEATHER.SNOW has 3 observations and 4 variables.
145     copy in=usclim out=weather; 146  run; NOTE: Copying USCLIM.BASETEMP to
WEATHER.BASETEMP (memtype=CATALOG).NOTE: Copying USCLIM.HURRICANE to
WEATHER.HURRICANE (memtype=DATA).NOTE: There were 6 observations read from the
data set USCLIM.HURRICANE.NOTE: The data set WEATHER.HURRICANE has 6
observations and 5 variables.NOTE: Copying USCLIM.REPORT to WEATHER.REPORT
(memtype=CATALOG).NOTE: Copying USCLIM.TEMPCHNG to WEATHER.TEMPCHNG
(memtype=DATA).NOTE: There were 5 observations read from the data set
USCLIM.TEMPCHNG.NOTE: The data set WEATHER.TEMPCHNG has 5 observations and 6
variables.NOTE: Copying USCLIM.USHIGH to WEATHER.USHIGH (memtype=DATA).NOTE:
There were 6 observations read from the data set USCLIM.USHIGH.NOTE: The data
set WEATHER.USHIGH has 6 observations and 5 variables.NOTE: Copying USCLIM.USLOW
to WEATHER.USLOW (memtype=DATA).NOTE: There were 7 observations read from the
data set USCLIM.USLOW.NOTE: The data set WEATHER.USLOW has 7 observations and 5
variables.
```

特定の SAS データセットのコピー

コピーするデータセットの選択

大規模な SAS ライブラリから、いくつかのデータセットのみをコピーするには、SELECT ステートメントと COPY ステートメントを使用します。キーワード SELECT の後に、データセット名を(複数の場合は、名前と名前の間にブランクを入れて)記述するか、または、使用可能な場合は略記法のメンバリスト(YRDATA1-YRDATA5 など)を使用します。

たとえば、次のステートメントは、データセット HURRICANE をライブラリ USCLIM からライブラリ STORM へコピーします。入力プロシジャライブラリは PRECIP であるため、USCLIM 入力ライブラリを指定するために COPY ステートメントに IN=オプションを含めます。

```
copy in=usclim out=storm;
   select hurricane;
run;
```

次の SAS ログは、データセット HURRICANE のみが、ライブラリ STORM へコピーされたことを示します。

ログ 39.3 *データセット HURRICANE のライブラリ STORM へのコピー*

```
147     copy in=usclim out=storm; 148        select hurricane; 149  run; NOTE:
Copying USCLIM.HURRICANE to STORM.HURRICANE (memtype=DATA).NOTE: There were 6
observations read from the data set USCLIM.HURRICANE.NOTE: The data set
STORM.HURRICANE has 6 observations and 5 variables.
```

データセットのコピーからの除外

いくつかのデータセットを除いてライブラリ全体をコピーするには、EXCLUDE ステートメントと COPY ステートメントを使用します。キーワード EXCLUDE の後に、除外するデータセット名を(複数の場合は、名前と名前の間にブランクを入れて)記述します。また、使用可能な場合は略記法のメンバリスト(YRDATA1-YRDATA5 など)を使用することもできます。

次のステートメントは、ライブラリ PRECIP のファイルを、データセット SNOW を除き、USCLIM へコピーします。プロシジャ入力ライブラリは PRECIP であるため、IN=オプションは必要ありません。

```
copy out=usclim;
   exclude snow;
run;
```

次の SAS ログは、データセット RAIN が USCLIM にコピーされ、データセット SNOW はライブラリ PRECIP にのみに残ることを示します。

ログ 39.4 *ライブラリ USCLIM へのコピーからのデータセット SNOW の除外*

```
150     copy out=usclim; 151        exclude snow; 152  run; NOTE: Copying
PRECIP.RAIN to USCLIM.RAIN (memtype=DATA).NOTE: There were 5 observations read
from the data set PRECIP.RAIN.NOTE: The data set USCLIM.RAIN has 5 observations
and 4 variables.
```

SAS ライブラリと SAS データセットの移動

ライブラリの移動

COPY ステートメントには MOVE オプションが用意されており、これを使用して、SAS データセットを入力ライブラリ(プロシジャ入力ライブラリまたは IN=オプションで指定される入力ライブラリのいずれか)から出力ライブラリ(OUT=オプションで指定される)へ移動できます。MOVE オプションを使用すると、SAS によりまずファイルが出力ライブラリへコピーされ、その後入力ライブラリからファイルが削除されることに注意してください。

次のステートメントは、ライブラリ PRECIP のすべてのデータセットをライブラリ CLIMATE へ移動します。

```
   copy out=climate move;
run;
```

次の SAS ログは、PRECIP 内のデータセットが CLIMATE へ移動されたことを示します。

ログ 39.5 *ライブラリ PRECIP 内のデータセットのライブラリ CLIMATE への移動*

```
153     copy out=climate move; 154  run; NOTE: Moving PRECIP.RAIN to
CLIMATE.RAIN (memtype=DATA).NOTE: There were 5 observations read from the data
set PRECIP.RAIN.NOTE: The data set CLIMATE.RAIN has 5 observations and 4
variables.NOTE: Moving PRECIP.SNOW to CLIMATE.SNOW (memtype=DATA).NOTE: There
were 3 observations read from the data set PRECIP.SNOW.NOTE: The data set
CLIMATE.SNOW has 3 observations and 4 variables.
```

MOVE オプションを使用してファイルを移動した後、CONTENTS ステートメントによる PRECIP のディレクトリリストで、ライブラリにメンバがないことが確認できます。次のステートメントの出力に示されるように、ライブラリ PRECIP に含まれるデータセットはもうありません。そのため、データセット RAIN とデータセット SNOW は、ライブラリ CLIMATE に含まれるものが唯一のものです。

```
   contents data=_all_ nods;
run;
```

次の出力は SAS ログと、続けてライブラリ PRECIP のディレクトリリストを示しています。

ログ 39.6 *CONTENTS ステートメントの SAS ログ*

```
155     contents data=_all_ nods; 156  run; WARNING: No matching members in
directory.
```

図 39.3 データセットがないことを示すライブラリ PRECIP のディレクトリリスト

The SAS System

The DATASETS Procedure

Directory	
Libref	PRECIP
Engine	V9
Physical Name	c:\Users\userid\precip
Filename	c:\Users\userid\precip

注: SAS ライブラリ PRECIP からデータセットは削除されましたが、ライブラリ参照名は割り当てられたままです。動作環境でライブラリへ割り当てられた名前は、すべてのファイルがあるライブラリから別のライブラリへ移動されても削除されません。

特定のデータセットの移動

SELECT ステートメントと EXCLUDE ステートメントを使用して、1 つ以上の SAS データセットを移動できます。たとえば、次のステートメントは、データセット HURRICANE をライブラリ USCLIM からライブラリ STORM へ移動します。

```
copy in=usclim out=storm move;
   select hurricane;
run;
```

次の出力は結果を示しています。

ログ 39.7 データセット HURRICANE のライブラリ USCLIM からライブラリ STORM への移動

```
157     copy in=usclim out=storm move; 158         select hurricane; 159  run;
NOTE: Moving USCLIM.HURRICANE to STORM.HURRICANE (memtype=DATA).NOTE: There were
6 observations read from the data set USCLIM.HURRICANE.NOTE: The data set
STORM.HURRICANE has 6 observations and 5 variables.
```

同様に、次の例では、EXCLUDE ステートメントを使用して、データセット SNOW を除くすべてのファイルをライブラリ CLIMATE からライブラリ USCLIM へ移動します。

```
copy in=climate out=usclim move;
   exclude snow;
run;
```

次の出力は結果を示しています。

ログ 39.8 *SNOW を除くすべてのデータセットのライブラリ CLIMATE からライブラリ USCLIM への移動*

```
160     copy in=climate out=usclim move; 161        exclude snow; 162  run;
NOTE: Moving CLIMATE.HIGHTEMP to USCLIM.HIGHTEMP (memtype=DATA).NOTE: There were
6 observations read from the data set CLIMATE.HIGHTEMP.NOTE: The data set
USCLIM.HIGHTEMP has 6 observations and 4 variables.NOTE: Moving CLIMATE.LOWTEMP
to USCLIM.LOWTEMP (memtype=DATA).NOTE: There were 5 observations read from the
data set CLIMATE.LOWTEMP.NOTE: The data set USCLIM.LOWTEMP has 5 observations
and 4 variables.NOTE: Moving CLIMATE.RAIN to USCLIM.RAIN (memtype=DATA).NOTE:
There were 5 observations read from the data set CLIMATE.RAIN.NOTE: The data set
USCLIM.RAIN has 5 observations and 4 variables.
```

SAS データセットの削除

削除するデータセットの指定

SAS ライブラリから 1 つ以上のデータセットを削除するには、DELETE ステートメントを使用します。複数のデータセットを削除する場合は、DELETE キーワードの後に、データセット名を名前と名前の間にブランクを入れて記述します。また、使用可能な場合は略記法のメンバリストを使用することもできます(YRDATA1-YRDATA5 など)。

注意:
プログラムステートメントがサブミットされるとすぐに、SAS ライブラリのファイルが削除されます。削除操作の開始前に操作の確認を求められないため、プログラムをサブミットする前に、行うファイルの削除操作を確認してください。

たとえば、次のプログラムは、USCLIM をプロシジャ入力ライブラリとして指定します。その後、データセット RAIN をライブラリから削除します。

```
proc datasets library=usclim;
   delete rain;
run;
```

次の出力は、SAS により DELETE ステートメントが処理されるときに、SAS ログにメッセージが送信されることを示しています。

ログ 39.9 *ライブラリ USCLIM からのデータセット RAIN の削除*

```
163  proc datasets library=usclim; 164     delete rain; 165  run; NOTE: Deleting
USCLIM.RAIN (memtype=DATA).
```

RAIN データセットの削除後にライブラリ USCLIM のコンテンツをリストで表示するには、次のプログラムを実行します。

```
proc datasets library=usclim;
run;
quit;
```

保存するデータセットの指定

いくつかを除きすべてのデータセットを削除するには、SAVE ステートメントを使用して、保存するデータセットの名前を記述します。データセット名を(複数の場合は、名前と名前の間にブランクを入れて)記述するか、または、使用可能な場合は略記法のメンバリスト(YRDATA1-YRDATA5 など)を使用します。

次のステートメントは、ライブラリ USCLIM から TEMPCHNG を除くすべてのデータセットを削除します。

```
proc datasets library=usclim;
   save tempchng;
run;
```

次の出力は、保存操作の SAS ログを示しています。SAS によりログにメッセージが送信され、SAVE ステートメントで指定されたデータセットが保存され、ライブラリのその他メンバがすべて削除されたことが確認できます。

ログ 39.10 *データセット TEMPCHNG を除くライブラリ USCLIM のすべてのメンバの削除*

```
171  proc datasets library=usclim; 172     save tempchng; 173  run; NOTE: Saving
USCLIM.TEMPCHNG (memtype=DATA).NOTE: Deleting USCLIM.BASETEMP
(memtype=CATALOG).NOTE: Deleting USCLIM.HIGHTEMP (memtype=DATA).NOTE: Deleting
USCLIM.LOWTEMP (memtype=DATA).NOTE: Deleting USCLIM.REPORT
(memtype=CATALOG).NOTE: Deleting USCLIM.USHIGH (memtype=DATA).NOTE: Deleting
USCLIM.USLOW (memtype=DATA).
```

SAS ライブラリのすべてのファイルの削除

SAS ライブラリのすべてのファイルを一度に削除するには、PROC DATASETS ステートメントで KILL オプションを使用します。

注意:

KILL オプションは、ステートメントがサブミットされた直後に、ライブラリのすべてのメンバを削除します。 削除操作の確認を求められないため、プログラムをサブミットする前に、行うファイルの削除操作を確認してください。

たとえば、次のプログラムは、ライブラリ WEATHER のすべてのデータセットを削除し、DATASETS プロシジャを終了します。

```
proc datasets library=weather kill;
run;
quit;
```

SAS ログの出力を次に示します。

ログ 39.11 *ライブラリ WEATHER のすべてのメンバの削除*

```
174  proc datasets library=weather kill; NOTE: Deleting WEATHER.BASETEMP
(memtype=CATALOG).NOTE: Deleting WEATHER.HIGHTEMP (memtype=DATA).NOTE: Deleting
WEATHER.HURRICANE (memtype=DATA).NOTE: Deleting WEATHER.LOWTEMP
(memtype=DATA).NOTE: Deleting WEATHER.RAIN (memtype=DATA).NOTE: Deleting
WEATHER.REPORT (memtype=CATALOG).NOTE: Deleting WEATHER.SNOW
(memtype=DATA).NOTE: Deleting WEATHER.TEMPCHNG (memtype=DATA).NOTE: Deleting
WEATHER.TORNADO (memtype=DATA).NOTE: Deleting WEATHER.USHIGH
(memtype=DATA).NOTE: Deleting WEATHER.USLOW (memtype=DATA).175  run; 176  quit;
```

注: SAS ライブラリからすべてのデータセットとカタログが削除されましたが、セッションの間はライブラリ参照名は割り当てられたままです。動作環境でライブラリへ割り当てられた名前は、ライブラリに含まれているファイルが削除されても削除されません。

要約

プロシジャ

PROC DATASETS LIBRARY=*libref* <KILL>;
プロシジャを開始し、後続のステートメントのプロシジャ入力ライブラリを指定します。KILL オプションは、ライブラリからすべてのメンバとメンバタイプを削除します。

DATASETS プロシジャステートメント

COPY OUT=*libref* <IN=*libref*> <MOVE>;
PROC DATASETS ステートメントで指定されるプロシジャ入力ライブラリから、OUT=オプションで指定される出力ライブラリへファイルをコピーします。IN=オプションは、別の入力ライブラリを指定します。MOVE オプションは、入力ライブラリからのファイルを出力ライブラリへコピーした後、削除します。

次に示すステートメントを、COPY ステートメントと一緒に使用できます。

EXCLUDE *SAS-data-set*;
コピー処理から除外する SAS データセットを指定します。このステートメントに記述しないファイルが、出力ライブラリへコピーされます。

SELECT *SAS-data-set*;
出力ライブラリへコピーする SAS データセットを指定します。

DELETE *SAS-data-set*;
このステートメントで指定される SAS データセットのみ削除します。

SAVE *SAS-data-set*;
このステートメントで指定されるメンバ以外の、ライブラリのすべてのメンバを削除します。

詳細情報

CATALOG プロシジャ
CATALOG プロシジャでは、SAS カタログのエントリが管理されます。詳細については、“CATALOG” (*Base SAS Procedures Guide*)を参照してください。

DATASETS プロシジャ
DATASETS プロシジャは、SAS ファイルを管理するユーティリティプロシジャです。詳細については、“DATASETS” (*Base SAS Procedures Guide*)を参照してください。

10 部

SAS 環境の基礎知識

40 章
SAS 環境について 719

41 章
SAS ウィンドウ環境の使用 731

42 章
SAS 環境のカスタマイズ 771

40 章

SAS 環境について

SAS 環境について **719**
目的 719
前提条件 720
動作環境による違い 720

SAS セッションの開始 **720**

SAS 処理モードの選択 **721**
処理モードとカテゴリ 721
SAS ウィンドウ環境における処理 722
ラインモードでの対話的な処理 725
バッチモードでの処理 726
非対話的な処理 726

要約 **727**
コマンド 727
オプション 727
システムオプション 728
ステートメント 728
コマンド 728

詳細情報 **729**
動作環境情報 729
ウィンドウ環境コマンド 729
ドキュメント 729

SAS 環境について

目的

このセクションでは、SAS プログラムを実行するさまざまな方法について学習します。さらに重要なこととして、SAS が実行可能なさまざまなモード、および実行するジョブの種類によってどのモードが最適であるかについて説明します。

このセクションでは、デフォルトの処理モードである SAS ウィンドウ環境について紹介します。

前提条件

このセクションの説明を理解するためには、“DATA ステップ処理の概要” (109 ページ)で説明されている、DATA ステッププログラミングの基本を理解している必要があります。

動作環境による違い

SAS の外観は動作環境ごとに異なりますが、メニューから使用可能な操作の大部分は同じです。

動作環境間の最も大きな違いの 1 つは、メニュー項目の選択方法です。

使用しているワークステーションがマウスを装備していない場合、次にマウス操作に相当するキーボード操作を示します。

表 40.1 マウス操作および相当するキーボード操作

マウス操作	相当するキーボード操作
項目をダブルクリック	項目の隣にあるスペースに、**s** または **x** を入力してから、Enter を押す
項目を右クリック	項目の隣にあるスペースに、**?**を入力してから、Enter を押す

このドキュメントの例では、Microsoft Windows 環境における SAS ウィンドウの外観を示しています。通常、その他の動作環境での対応するウィンドウでは同様の結果を得られます。使用している動作環境でドロップダウンメニューが表示されない場合、コマンドプロンプトにグローバルコマンド PMENU を入力します。

SAS セッションの開始

SAS セッションを開始するには、SAS を起動する必要があります。動作環境のプロンプトで、SAS コマンドを実行します。ほとんどの場合、SAS コマンドは次に示すとおりです。

```
sas
```

注: SAS コマンドは、サイトによって異なる場合があります。詳細については、SAS サポート担当者にお問い合わせください。

SAS セッションは、SAS システムオプションを指定して起動することにより、カスタマイズ可能です。設定はセッションが終了するまで有効なままです。たとえば、LINESIZE= システムオプションを使用して、SAS ログとプリントファイルの行サイズを指定できます。いくつかのシステムオプションは初期化時にのみ指定でき、その他のシステムオプションは SAS セッション時に指定できます。詳細については、“SAS セッションと SAS プログラムの起動時のカスタマイズ” (773 ページ)を参照してください。

SAS 処理モードの選択

処理モードとカテゴリ

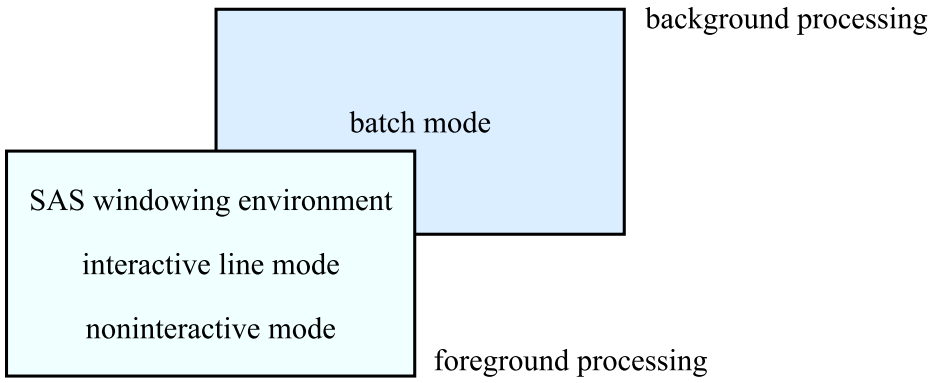
処理モードとカテゴリの概要

SAS の実行に使用可能な 4 つのモードはすべて、次の 2 つのカテゴリの 1 つに属しています。

- フォアグラウンド処理
- バックグラウンド処理

次の図に、4 つのそれぞれのモードと、モードが属する処理の種類を示します。処理要件の変更に応じて、ある処理モードから別の処理モードに変更した方が有用である場合があります。

図 40.1 フォアグラウンド処理またはバックグラウンド処理時の SAS の実行モード



フォアグラウンド処理について

フォアグラウンド処理には、バッチモードを除いて、SAS を実行可能なすべての方法が含まれています。フォアグラウンド処理はすぐに開始しますが、プログラムの実行時に、現在のワークステーションセッションが占有され、他のタスクの実行には使用できなくなります。[1] フォアグラウンド処理では、ワークステーションディスプレイ、ファイル、プリンタ、またはテープを出力先にできます。

次の項目に 1 つ以上該当する場合は、フォアグラウンド処理の使用を検討してください。

- SAS プログラミングを学習中である。
- プログラムの稼働テスト中である。
- 高速なターンアラウンドが必要である。
- ごく小規模なデータファイルを処理しようとしている。
- 対話型アプリケーションを使用している。

1 ワークステーション環境では、別のウィンドウに切り替えて作業を継続できます。

バックグラウンド処理について

SAS をバックグラウンドで実行する唯一の方法はバッチ処理です。動作環境によりすべての作業が調整されるため、プログラムを実行すると同時に、ワークステーションセッションを使用して他の作業を行うことができます。ただし、動作環境によりプログラムの実行がスケジュールされ優先順位を割り当てられるため、実行まで入力キュー(動作環境の実行予定ジョブのリスト)で待機する必要がある場合もあります。プログラムが最後まで実行されると、出力を参照、削除、または印刷できます。

サイトによってはバックグラウンド処理が要求される場合があります。さらに、次の項目を考慮してください。

- 熟達した SAS ユーザーであり、初心者ほど多くのエラーはない。
- すでにテスト済みで修正済みのプログラムを実行している。
- 高速なターンアラウンドより、コンピュータリソースの使用を最小化するほうが重要である。
- 大規模なデータファイルを処理しようとしている。
- 実行時間の長いプログラムである。
- テープを使用している。

これらの項目に 1 つ以上該当する場合は、バックグラウンド処理を選択してください。

SAS ウィンドウ環境における処理

SAS ウィンドウ環境における処理の概要

SAS ウィンドウ環境は、一連のウィンドウによって構成されるグラフィカルユーザーインターフェイス(GUI)であり、これを使用してファイルやフォルダの整理、プログラムの編集や実行、プログラム出力の表示、およびプログラムと SAS セッションに関するメッセージを表示できます。

対話型グラフィカルな機能であるため、単一のセッションを使用して、プログラムの準備やサブミットを行い、出力やメッセージを参照した後で、必要に応じて、プログラムの変更および再サブミットができます。ウィンドウからウィンドウへの移動、セッションの中断、そして中断したときと同じ場所へ戻ることもできます。

全般的な特徴

SAS ウィンドウ環境は、SAS セッションのデフォルトの環境です(サイトの環境がカスタマイズされていない場合)。

注: デフォルトの環境であるため、このドキュメントの多くのトピックでは、タスクを SAS ウィンドウ環境において実行するものとして説明しています。

SAS ウィンドウ環境で最もよく使用される 5 つのウィンドウは、エクスプローラ、結果、エディタ、ログ、およびアウトプットです。

エクスプローラ
フォルダ、サブフォルダ、および個別の項目の階層システムです。SAS へのプライマリグラフィカルインターフェイスを提供し、これを使用してユーザーは次の操作を行えます。

- カタログ、テーブル、ライブラリ、および動作環境のファイルなどのデータにアクセスして操作できます。
- SAS プログラミングウィンドウを開きます。
- Output Delivery System (ODS)にアクセスします。
- カスタマイズしたフォルダを作成し、定義します。

エクスプローラを使用して、ライブラリとファイルのショートカットの表示や設定、ライブラリメンバとカタログエントリの表示や設定、または SAS ファイルを開いたり編集したりできます。

SAS ウィンドウ環境の起動時、エクスプローラは現在使用可能なライブラリのリストを示す単一ペインのウィンドウとして表示される場合があることに注意してください。**表示** ⇨ **ツリーの表示**を選択するか、または TREE コマンドを発行して、**エクスプローラ**ウィンドウにナビゲーションツリーを追加できます。

エディタまたはプログラムエディタ
: SAS ステートメントの入力、編集、およびサブミット、さらに SAS ソースファイルを保存するための領域を提供します。

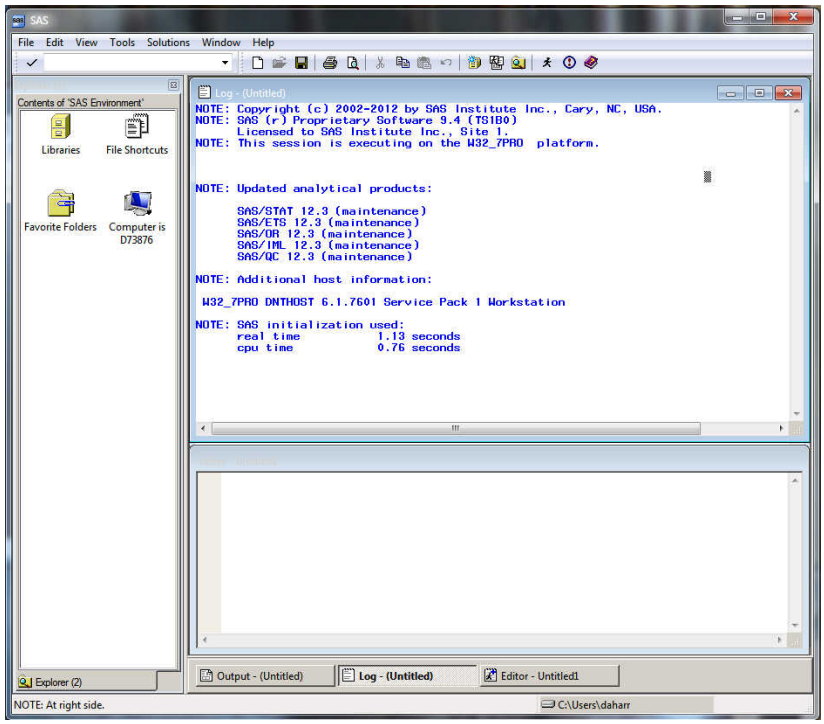
ログ
: SAS ログを参照したり、スクロールしたりできます。SAS ログには SAS セッションで発生する出来事に関するメッセージが表示されます。

アウトプット
: プロシジャ出力を参照したり、スクロールしたりできます。

結果
: プロシジャ出力のインデックスを参照したり、操作したりできます。

プログラムエディタ、ログ、およびアウトプットの各ウィンドウはまとめて、プログラミングウィンドウと呼ばれることがあります。

図 40.2 *SAS ウィンドウ環境: SAS エクスプローラ、ログおよびエディタの各ウィンドウ(Windows オペレーティングシステム)*



SAS ウィンドウ環境では、次の操作が可能な追加のウィンドウも使用できます。

- オンラインヘルプのアクセス
- 一部の SAS システムオプションの表示と変更
- ファンクションキー設定の表示と変更
- テキスト情報の作成と格納

これらのウィンドウとウィンドウ環境におけるタスクの実行の詳細については、“SAS ウィンドウ環境の使用について” (732 ページ)を参照してください。

SAS ウィンドウ環境の起動

SAS ウィンドウ環境を起動するには、有効にする任意のシステムオプションを後に続けた SAS コマンドを実行します。SAS ウィンドウ環境は、SAS のデフォルト動作方法として設定されていますが、使用しているサイトによってはデフォルト設定でない場合があります。

SAS ウィンドウ環境がデフォルト動作方法でない場合、SAS コマンドに DMSEXP オプションを指定します。または、システムオプションの設定を含む構成ファイルに DMSEXP オプションを含めることもできます。構成ファイルの詳細については、“SAS セッションと SAS プログラムの起動時のカスタマイズ” (773 ページ)を参照してください。

システムでその他のコマンドオプションを指定するのと同様に、SAS コマンドにオプションを指定します。

次の表に、さまざまな動作環境下で SAS ウィンドウ環境を起動して DMSEXP オプションを指定する方法を示します。

表 40.2 DMSEXP オプションを使用した SAS の開始

動作環境	コマンド
z/OS	sas options ('dmsexp')
Windows	sas -dmsexp
UNIX	sas -dmsexp

その他のシステムでのコマンドオプションの指定方法の詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

SAS ウィンドウ環境セッションの終了

SAS ウィンドウ環境セッションは BYE コマンドまたは ENDSAS コマンドを使用して終了できます。SAS コマンド行で BYE または ENDSAS を指定してから、ENTER キーまたは RETURN キーを押してコマンドを実行します(使用しているオペレーティングシステムによって異なります)。

プログラムエディタウィンドウで ENDSAS ステートメントを使用してセッションを終了することもできます。データ行に次のステートメントを入力し、サブミットして実行します。

```
endsas;
```

SAS ウィンドウ環境セッションの中断

SAS セッションから動作環境に戻ることが必要な場合があります。SAS セッションを終了したくない場合、X コマンドを発行して、動作環境へエスケープできます。単にコマンド行で次のコマンドを実行します。

```
x
```

その後、動作環境から、適切な動作環境コマンドを実行して、同じ SAS セッションのエスケープした時点へ戻ることができます。たとえば、z/OS 動作環境下では、動作環境コマンドは Return または End です。

単一の動作環境コマンドを実行するには、次の形式の X コマンドを使用します。

X *operating-environment-command*

または、コマンドに埋め込みブランクが含まれる場合は、次を使用します。

X *'operating-environment-command'*

たとえば、多くのシステムでは、次を指定して現在の時刻を表示できます。

```
x time
```

コマンドの実行後、適切な操作を実行して SAS セッションへ戻れます。

その他の動作環境における SAS セッションの中断の詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

ラインモードでの対話的な処理

全般的な特徴

ラインモードの処理では、プログラミングステートメントが一度に 1 行ずつ入力されるため、DATA ステップと PROC ステップは、RUN ステートメントが入力された後、または別のステップ境界が検出された後に実行されます。プログラムのメッセージと出力がモニタに表示されます。

プログラムステートメントの変更は、最初の入力時にのみ可能で、ENTER キーまたは RETURN キーを押した後は変更不可であるため、エントリは注意して入力する必要があります。

SAS のラインモードでの起動

ラインモードで SAS を起動するには、有効にする任意のシステムオプションを後に続けた SAS コマンドを実行します。NODMS システムオプションにより、対話型ラインモードのセッションが有効化されます。サイトで NODMS がデフォルトのシステムオプションでない場合、SAS コマンドでオプションを指定するか、起動時に有効にされるシステムオプションの設定を含むファイルである構成ファイルに、NODMS の指定を含めます。

次の表に、さまざまな動作環境下の SAS コマンドでの NODMS システムオプションの指定方法を示します。

表 40.3 *NODMS システムオプションの指定*

動作環境	コマンド
z/OS	sas options ('nodms')
UNIX	sas -nodms

Run ステートメントを使用したラインモードでのプログラムの実行

ラインモードでは、新しいステップ境界が検出されるまで DATA ステップは実行されません。これは、RUN DATA または PROC ステートメントが入力された後に起こります。言い換えると、ウィンドウ環境で **`DATA X; X=1;`** をサブミットしても、次の RUN DATA または PROC ステートメントがサブミットされるまで実行されません。

各行の先頭に数字と疑問符が表示されている SAS プロンプトで、ステートメントをさらに入力します。DATALINES ステートメントを使用すると、大なり記号(>)が疑問符に置き換わり、データ行が求められていることを示します。

ラインモードを使用している場合、プログラミングのヒントとして、各 DATA ステップまたは PROC ステップを、次のステップのプログラミングステートメントの入力を開始する前に実行するようにすると、ログを見やすくできます。END ステートメントまたはデータ行の終了を示すセミコロンのいずれかを入力すると、ステップはすぐに実行されます。

ラインモード SAS セッションの終了

セッションを終了するには、SAS プロンプトに **endsas;** と入力し、Enter キーを押します。セッションが終了し、動作環境に戻ります。

ラインモード SAS セッションの中断

ラインモードでは、次のステートメントを実行して、動作環境へエスケープできます。

```
x;
```

適切な動作環境コマンドを実行して SAS セッションへ戻ることができます。単一の動作環境コマンドを実行するには、次の形式の X ステートメントを使用します。

X *operating-environment-command*;

または、コマンドに埋め込みブランクが含まれる場合は、次を使用します。

X *'operating-environment-command'*;

たとえば、多くのシステムでは、次を指定して現在の時刻を表示できます。

```
x time;
```

この形式の X コマンドを使用すると、コマンドが実行され、SAS セッションに戻ります。

バッチモードでの処理

バッチモードでプログラムを実行する最初のステップは、次を含むファイルの準備をすることです。

- プログラムの管理に使用している動作環境によって必要とされる制御言語ステートメント
- プログラムの実行に必要な SAS ステートメント

次に、ファイルを動作環境へサブミットすると、動作環境によりプログラムが実行され、ワークステーションセッションは解放されて他の作業が可能になります。プログラムが実行されるまで、いずれの方法でもプログラムを表示したり変更したりできないため、これはバックグラウンド処理と呼ばれます。ログと出力は、動作環境の制御言語で指定される出力先に送られます。指定がない場合、それらはデフォルト出力先に送られます。バッチ処理の例については、使用している動作環境に対応する SAS ドキュメントを参照してください。

非対話的な処理

全般的な特徴

非対話型処理は、対話型処理とバッチ処理それぞれの特徴の一部を備えています。非対話的に処理を行う場合は、外部ファイルに保存されている SAS プログラムステートメントを実行します。SAS コマンドを使用して、動作環境へプログラムステートメントをサブミットします。

注: SAS コマンドは各動作環境下で異なった方法で実装されています。たとえば、z/OS の下では、そのコマンドは通常 CLIST になります。

対話型処理でのように、処理はすぐに開始され、現在のワークステーションセッションが占有されます。ただし、バッチ処理と同様に、プログラムと対話はできません。

注: これに対するいくつかの例外については、使用している動作環境に対応する SAS ドキュメントを参照してください。

ログ出力やプロシジャ出力は、プログラムの実行が完了するとすぐに表示できます。ログ出力とリスト出力はワークステーションへ出力され、SAS ウィンドウ環境とは異なり、出力をファイルに明示的に保存する必要があります。プログラムを修正または変更する必要があると判断した場合、エディタを使用して必要な変更を行いプログラムを再サブミットする必要があります。

非対話型モードでのプログラムの実行

非対話型モードでプログラムを実行する場合、対話型モードで行う場合のように、SAS セッションを開始しません。SAS セッションを開始するかわりに、SAS プログラムを実行します。最初のステップは、バッチジョブで行うのと同様に、SAS ステートメントをファイルに入力することです。次に、システムプロンプトで SAS コマンドを、ファイルの完全名と任意の指定するシステムオプションを後に続けて指定します。

次の例は、z/OS 動作環境にある区分データセット your-userid.UGWRITE.TEXT のメンバ TEMP 内の SAS ステートメントを実行します。

```
sas input(ugwrite.text(temp))
```

INPUT オペランドが指しているファイルには、非対話型セッションのための SAS ステートメントが含まれていることに注意してください。

その他の動作環境での非対話型モードの使用方法の詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。サイトに固有の情報については、SAS サイト担当者にお問い合わせください。

ログと出力の参照

ログと出力情報は、ワークステーションディスプレイに表示されるか、またはファイルへ送られます。デフォルトの処理は、動作環境によって異なります。いずれの場合も、ディスプレイ内か、または目的のファイルを開いて情報を参照できます。

詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

要約

コマンド

OPTIONS
ウィンドウ環境を使用する場合のオプション設定を表示します。

オプション

PROC OPTIONS *options*;
すべての SAS システムオプションの現在の値をリストで表示します。

システムオプション

DMS | NODMS
起動時に、SAS セッションで SAS プログラミングウィンドウを有効化するかどうかを指定します。

LINESIZE=*n*
SAS LISTING 出力の幅を指定します。

VERBOSE
起動時に、構成ファイル内およびコマンド行上のすべてのオプションのリストを表示します。

ステートメント

DATALINES;
直後にデータが続くことを SAS に伝えます。

ENDSAS;
現在の DATA ステップまたは PROC ステップの終了時に SAS ジョブまたは SAS セッションを終了します。

OPTIONS *option*;
1 つ以上のシステムオプションを、サイトに設定されているデフォルト値から変更します。

RUN;
それ以前に入力された SAS ステップを実行します。

X *'operating-environment-command'*;
SAS セッション内から動作環境コマンドを発行するために使用します。oerating-environment-command にコマンドを指定します。コマンドを省略すると、動作環境のサブモードに入ります。

コマンド

BYE
SAS セッションを終了します。

ENDSAS
SAS セッションを終了します。

EXPLORER
エクスプローラウィンドウを起動します。

PMENU
ウィンドウのドロップダウンメニューを有効化します。

X <'*operating-environment-command*'>
動作環境コマンドが実行された後、適切な操作を行って SAS へ戻るためのプロンプトが出されます。コマンドを省略すると、動作環境のサブモードに入ります。

詳細情報

動作環境情報

特定のオプションとプリファレンスのカスタマイズの詳細については、使用している動作環境に対応するドキュメントを参照してください。

ウィンドウ環境コマンド

SAS ウィンドウ環境で使用可能なすべてのコマンドのリストについては、SAS オンラインヘルプを参照してください。

ヘルプ ⇨ SAS ヘルプとドキュメント

ドキュメント

SAS ウィンドウ環境のその他の使用例については、*Getting Started with the SAS System* を参照してください。

41 章

SAS ウィンドウ環境の使用

SAS ウィンドウ環境の使用について **732**
目的 732
前提条件 732
動作環境による違い 732

データの整理 **733**
データの整理の概要 733
ライブラリとライブラリメンバの探索 733
ライブラリ参照名の割り当て 734
ライブラリの割り当てに関する問題の管理 735

オンラインヘルプの検索 **736**
SAS オンラインヘルプシステムへのアクセス 736
ウィンドウのヘルプへのアクセス 736

SAS ウィンドウ環境で使用できるコマンドの種類 **736**
SAS ウィンドウ環境コマンドの種類の概要 736
コマンド行のコマンドの使用 736
メニューの使用 737
行コマンドの使用 738
ファンクションキーの使用 738

SAS ウィンドウの操作 **739**
ウィンドウを開く 739
ウィンドウの管理 741
ウィンドウのスクロール 741
例: ウィンドウのスクロール 742
ウィンドウでの色の変更と強調表示 742
テキストの検索と変更 743
テキストの切り取り、貼り付け、格納 744

テキストの操作 **744**
SAS テキストエディタ 744
テキストの移動と再配置 744
カラム番号と行番号の表示 746
テキストの大文字または小文字変換 746
テキストの結合と分割 748

ファイルの操作 **748**
ファイルの検索方法 748
ファイル固有のコマンドの発行 750
ファイルを開く 750
ファイルショートカットの割り当て 751

既存のファイルショートカットの変更 752
ファイルの印刷 752

SAS プログラムでの作業 753
エディタウィンドウ 753
アウトプットウィンドウ 755
ログウィンドウ 756
その他のエディタの使用 757
プログラムの作成とサブミット 757
プログラムの保存 758
プログラムのデバッグ 758
プログラムを開く 758
プログラムの編集 759
ファイルショートカットへのプログラムの割り当て 759

出力の操作 759
出力の操作の概要 759
出力形式の設定 760
SAS 出力の種類へのデフォルトのビューアの割り当て 761
結果ウィンドウでの出力の操作 762
出力テンプレートの操作 765
出力の印刷 768

要約 768
ステートメント 768
ウィンドウ 768
コマンド 769
プロシジャ 770

詳細情報 770

SAS ウィンドウ環境の使用について

目的

このセクションでは、データの整理方法、ヘルプへのアクセス方法、適切なコマンドの検索および使用方法を含む、SAS ウィンドウ環境について学習します。

さらに、SAS ウィンドウ環境を使用して、ファイル、SAS プログラム、および SAS 出力を操作する方法を学習します。

前提条件

このセクションを先に進む前に、“SAS 環境について” (719 ページ)に説明されている概念を理解している必要があります。

動作環境による違い

SAS の外観は動作環境ごとに異なりますが、メニューから使用可能な操作の大部分は同じです。

動作環境間の最も大きな違いの 1 つは、メニュー項目の選択方法です。

使用しているワークステーションがマウスを装備していない場合、次にマウス操作に相当するキーボード操作を示します。

表 41.1 *マウス操作および相当するキーボード操作*

マウス操作	相当するキーボード操作
項目をダブルクリック	項目の隣にあるスペースに、**s** または **x** を入力してから、Enter を押す
項目を右クリック	項目の隣にあるスペースに、**?**を入力してから、Enter を押す

このドキュメントの例では、Microsoft Windows 環境における SAS ウィンドウの外観を示しています。通常、その他の動作環境での対応するウィンドウでは同様の結果を得られます。使用している動作環境でドロップダウンメニューが表示されない場合、コマンドプロンプトにグローバルコマンド PMENU を入力します。

データの整理

データの整理の概要

SAS ウィンドウ環境を使用して、データの整理、およびファイルの検索とアクセスを簡単に行うことができます。このセクションでは、ウィンドウを使用して次の操作を行う方法を学習します。

- ライブラリとライブラリメンバの探索
- ライブラリ参照名の割り当て

ライブラリとライブラリメンバの探索

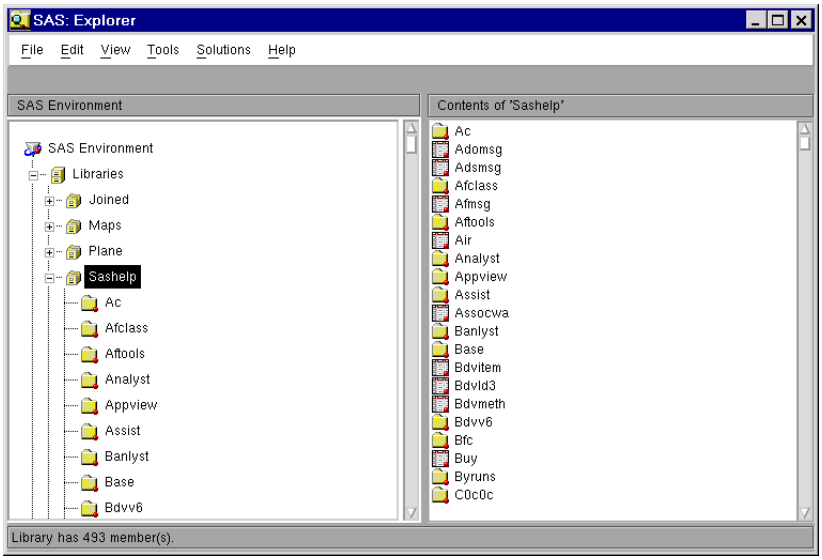
多くのホストでは、SAS ウィンドウ環境は、デフォルトで**エクスプローラ**ウィンドウで開きます。このウィンドウがデフォルトで表示されない場合は、EXPLORER コマンドを発行して起動できます。エクスプローラを使用して、現在使用可能なライブラリを表示し、そのコンテンツを探索できます。

- 使用可能なライブラリのリストを表示するには、ライブラリフォルダを選択して、メニューから**開く**を選択します。
- ライブラリのコンテンツを探索するには、特定のライブラリを選択して、メニューから**エクスプローラ**を選択します。
- ライブラリメンバのコンテンツを探索するには、特定のライブラリメンバを選択して、メニューから**開く**を選択します。

注: エクスプローラツリービューがオンになっている場合、ツリーノードを展開したり折りたたんだりして、ライブラリとライブラリメンバを探索できます。+記号と-記号のように見える、ツリーノードの展開アイコンを選択して、ツリーノードを展開/折りたたみできます。エクスプローラツリービューは、**エクスプローラ**ウィンドウから**表示** ⇨ **ツリーの表示**を選択して切り替えられます。

次の表示画面は、展開したツリーノードを示しています。

図 41.1 ツリービューがオンになっている SAS エクスプローラウィンドウ



ライブラリ参照名の割り当て

SAS セッションでの作業を続行する前に、ライブラリ参照名を割り当てて、作業する SAS ファイルの永久記憶域の場所を用意します。

1. **エクスプローラ**ウィンドウから、**ライブラリ**フォルダを選択します。
2. **ファイル** ⇨ **新規**を選択します。

   ライブラリの作成ウィンドウが開きます。
3. ライブラリの名前を入力します。
4. エンジンの種類を選択します。
5. 動作環境のディレクトリパス名を入力するか、ディレクトリを参照して選択します。
6. エンジンによっては、必要に応じてその他のフィールドを入力し、指定するオプションがある場合は入力します。

   選択するエンジンが決まっていない場合、(自動的に選択される)デフォルトエンジンを使用します。

   デフォルトエンジンを使用すると、新しいライブラリの指定パスに存在するデータセットすべてに使用されるエンジンが SAS により選択されます。データセットが存在しない場合には、Base SAS Engine が割り当てられます。
7. **OK** を選択します。**エクスプローラ**ウィンドウの**ライブラリ**フォルダの下に、新しいライブラリが表示されます。

注: SAS の起動時に新しいライブラリを自動的に割り当てるには、**ライブラリの作成**ウィンドウの**起動時に有効**チェックボックスを選択します。

動作環境に応じて、次のライブラリの割り当て方法を使用できます。

メニュー
ファイル ⇨ **新規**

(**エクスプローラ**ウィンドウからのみ使用可能)

コマンド
DMLIBASSIGN (いずれのウィンドウからも使用可能)

ツールバー
ライブラリの作成(いずれのウィンドウからも使用可能)

ライブラリの割り当てに関する問題の管理

SAS レジストリに格納されているいずれかの永久ライブラリ割り当てが起動時に失敗した場合、次の NOTE メッセージが SAS ログに表示されます。

```
NOTE: One or more library startup assignments were not restored.
```

次のエラーは、ライブラリの割り当てに関する問題で一般的な原因です。

- ライブラリの依存関係が見つからない
- SAS レジストリでライブラリの割り当ての必須フィールド値が失われている
- SAS レジストリでライブラリの割り当ての必須フィールド値が無効である

  たとえば、ライブラリ名は 8 文字に制限されており、エンジン値は実際のエンジン名に一致している必要があります。
- ライブラリ参照名の暗号化されたパスワードデータが SAS レジストリで変更されている

注意:
SAS レジストリエディタで、多くのライブラリ割り当てエラーを修正できます。 ライブラリ参照名や SAS レジストリエディタに習熟していない場合は、ヘルプを依頼してください。SAS レジストリエディタの使用ではエラーが発生しやすく、起動時にライブラリを割り当てられなくなるおそれがあります。

SAS レジストリエディタでライブラリの割り当てエラーを修正するには、次の操作を実行します。

1. **ソリューション** ⇨ **アクセサリ** ⇨ **レジストリエディタ**を選択するか、REGEDIT コマンドを発行します。
2. オペレーティングシステムに応じて次のいずれかのパスを選択し、必要に応じてキーとキーの値を変更します。

   ```
   CORE\OPTIONS\LIBNAMES
   ```

   または

   ```
   CORE\OPTIONS\LIBNAMES\CONCATENATED
   ```

   または

   ```
   CORE\LIBNAMES
   ```

たとえば、永久連結ライブラリのキーが正の整数以外に名前変更されていると判断した場合は、準拠するようにキーを再び名前変更できます。処理を開始するには、キーを選択して、ポップアップメニューで**名前の変更**を選択します。

オンラインヘルプの検索

SAS オンラインヘルプシステムへのアクセス

SAS オンラインヘルプにアクセスするには、**ヘルプ** ⇨ **SAS ヘルプとドキュメント**を選択します。

ウィンドウのヘルプへのアクセス

次のいずれの方法でも、個別のウィンドウでヘルプにアクセスできます。

- ウィンドウのコマンド行から HELP コマンドを発行します。
- ウィンドウのヘルプボタンを選択します(存在する場合)。
- ツールバーのヘルプアイコンを選択します。
- ヘルプが必要なウィンドウから、**ヘルプ** ⇨ **このウィンドウの使い方**を選択します。

SAS ウィンドウ環境で使用できるコマンドの種類

SAS ウィンドウ環境コマンドの種類の概要

SAS ウィンドウ環境コマンドには特定の種類があります。使用するコマンドの種類は、実行が必要なタスクや個人用プリファレンスによって異なります。コマンドは次の形式です。

- コマンド行のコマンド
- メニューコマンド
- 行コマンド(テキスト編集ウィンドウ内)
- キーボードのファンクションキー

SAS ウィンドウ環境で発行可能な特定のコマンドの詳細については、“SAS ウィンドウの操作” (739 ページ)を参照してください。SAS テキストエディタで使用可能な特定のコマンドの詳細については、“テキストの操作” (744 ページ)を参照してください。

コマンド行のコマンドの使用

コマンド行のコマンドは、次の 2 つの場所で入力できます。

- コマンド行(有効になっている場合)
- **コマンド**ウィンドウ(使用可能な場合)

コマンド行が有効になっている場合、コマンド行にカーソルを置いて、コマンドを入力できます。特定のウィンドウのコマンド行のオン/オフを、**ツール** ⇨ **オプション** ⇨ **コマンド行をオンにする**または**ツール** ⇨ **オプション** ⇨ **コマンド行をオフにする**を選択して切り替えできます。

コマンドウィンドウ(使用している動作環境で使用可能な場合)には、テキスト領域があります。カーソルをこの領域に配置して、コマンドを発行できます。

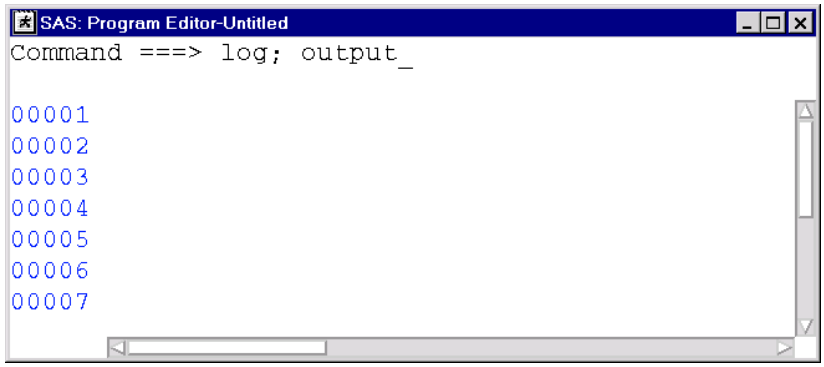
コマンドを実行するには、コマンド行にコマンドを入力し、使用している動作環境に応じて、ENTER キーを押します。単純な 1 語のコマンド、セミコロンで区切った複数のコマンド、またはコマンドの後にオプションを続けて入力できます。

たとえば、**エディタ**ウィンドウから移動して、ログとアウトプットの両方のウィンドウを開く場合、**エディタ**ウィンドウのコマンド行で次を指定します。

```
log; output
```

次の表示画面は、コマンド行に入力した **log; output** を示しています。

図 41.2 *コマンド行でのコマンドの入力*



次に、ENTER または RETURN を押して両方のコマンドを実行します。ログウィンドウとアウトプットウィンドウが表示されます。**アウトプット**ウィンドウを開くコマンドが最後に実行されたため、このウィンドウがアクティブウィンドウです。

メニューの使用

SAS ウィンドウ環境のウィンドウでは、コマンド行のかわりにメニューを表示できます。その場合、通常はコマンドを入力して完了するタスクを、メニューを選択することで実行できます。

お使いの動作環境でドロップダウンメニューを使用するようデフォルト設定されていない場合、コマンド行で PMENU コマンドを発行して、メニューをサポートするすべてのウィンドウで有効にします。

メニューやメニュー項目は、マウスでポイントアンドクリックして選択できます。一部の動作環境では、カーソルをメニュー項目上に移動し、ENTER または RETURN を押すことでも選択できます。選択した項目に応じて、次の 3 つの内いずれかの処理が行われます。

- コマンドの実行
- メニューの表示
- ダイアログボックスの表示

多くのケースでは、項目をダブルクリックする場合と、項目を右クリックする場合とでは表示されるメニューが異なります。項目を選択しても予想した結果が得られない場合、もう一方を試してみてください。

ワークステーションでマウスを装備していないその他の動作環境については、マウス操作に相当するキーボード操作を次に示します。

表 41.2 マウス操作に相当するキーボード操作の表

マウス操作	相当するキーボード操作
ダブルクリック	項目の隣にあるスペースに、**s** または **x** を入力してから、Enter を押す
右クリック	項目を右クリックするかわりに、項目の隣にあるスペースに、**?**を入力してから、Enter を押す

行コマンドの使用

行コマンドは、テキストのコピー、移動、削除などの、テキストの編集を行う 1 つ以上の文字です。行コマンドは、エディタや SAS NOTEPAD などのテキスト編集ウィンドウの番号付き部分に入力して実行できます。

行コマンドは通常、画面表示の番号付き部分に入力するか、ファンクションキーを使用して実行されますが、コロンを前に付けてコマンド行で実行することもできます。

注: NUMBERS コマンドを発行して、テキスト編集ウィンドウで行番号のオン/オフを切り替えます。

行コマンドの詳細については、“テキストの操作” (744 ページ)を参照してください。

ファンクションキーの使用

キーボードには、あらかじめデフォルト値が割り当てられているファンクションキーがあります。**KEYS** ウィンドウでそれらの値を参照または変更できます。**KEYS** ウィンドウを開くには、**ツール** ⇨ **オプション** ⇨ **KEYS** を選択するか、KEYS コマンドを発行します。

KEYS ウィンドウにあるキーの設定を変更するには、古い値の上に新しい値を上書きします。新しい設定はすぐに有効となり、END コマンドを実行して **KEYS** ウィンドウを閉じると永続的に保存されます。

ファンクションキーを使用してキー設定をカスタマイズし、特定の SAS セッションにおけるニーズを満たすことができます。たとえば、多数のプログラムをサブミットして、**エディタ**ウィンドウと**アウトプット**ウィンドウとの間を移動する必要があるとします。そのため、出力の表示を終了するたびに、コマンド行に PGM コマンドと ZOOM コマンドを入力して ENTER または RETURN を押す必要があります。ファンクションキーのいずれかをこのアクションを実行するショートカットとして定義するには、次のコマンドを不要な値に上書きするかまたは値がない箇所に入力します。

```
pgm; zoom
```

その後は、ファンクションキーを押すたびにコマンドが実行され、時間を節約できます。ファンクションキーを使用して行コマンドを実行することもできます。コマンド行から行コマンドを発行するのと同じように、単に行コマンドをコロンで先行します。

SAS ウィンドウの操作

ウィンドウを開く

SAS ウィンドウ環境には、数多くのウィンドウが用意されており、これを使用してタスクを完了できます。コマンドを入力して、ウィンドウを開きます。コマンド実行方法の詳細については、“SAS ウィンドウ環境で使用できるコマンドの種類” (736 ページ)を参照してください。

次のコマンドを使用して、ウィンドウを開いてアクティブにできます。

表 41.3 *ウィンドウコマンド*

ウィンドウコマンド	ウィンドウ名
AF C=library.catalog.entry.type	ビルド
DMFILEASSIGN	ファイルショートカットの割り当て
DMLIBASSIGN	ライブラリの作成
EDOP	エディタオプション
EXPFIND	検索
EXPLORER	エクスプローラ
FOOTNOTES	FOOTNOTES
FSBROWSE	FSBrowse
FSEDIT	FSEdit
FSFORM formname	FSForm
FSVIEW	FSView
HELP	ヘルプ
KEYS	KEYS
LOG	ログ
NOTEPAD、NOTE	NOTEPAD
ODSRESULTS	結果
ODSTEMPLATES	テンプレート

ウィンドウコマンド	ウィンドウ名
OPTIONS	オプション
OUTPUT、LISTING、LIST、LST	アウトプット
PROGRAM、PGM、PROG	プログラムエディタ
REGEDIT	レジストリエディタ
REPOSMGR	リポジトリマネージャ
SASENV	エクスプローラ(コンテンツのみ表示)
SETPASSWORD	パスワード
TITLES	TITLES
VAR	プロパティ

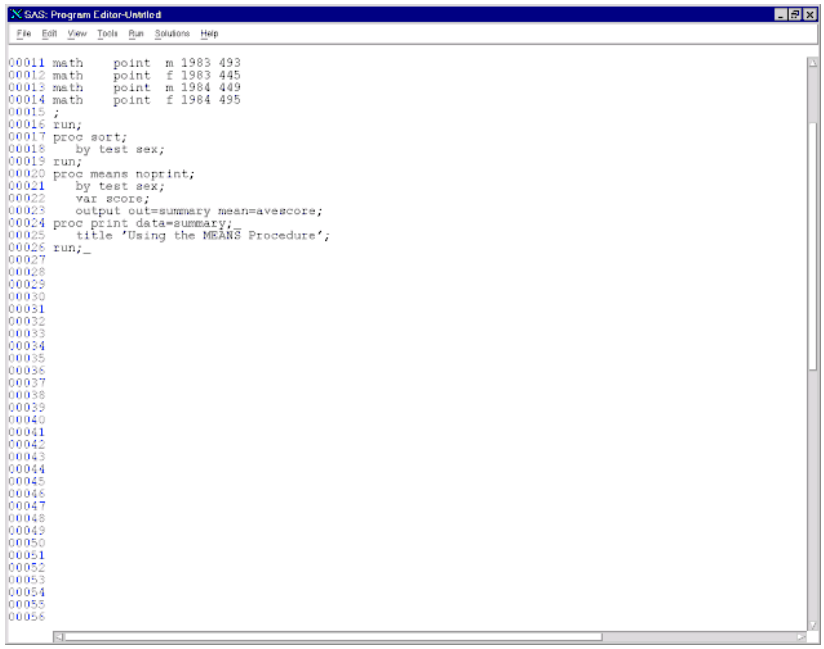
ウィンドウコマンドは、いずれのコマンドプロンプトでも使用できます。複数のウィンドウコマンドを組み合わせて使用すると便利です。

たとえば、**ログ**ウィンドウから、次のコマンド文字列を使用して、アクティブウィンドウを変更し、そのウィンドウのサイズを最大化して、語 paint を print に変更します。

```
pgm; zoom; change paint print
```

次のディスプレイは、カーソルがすぐにエディタへ移動し、(ZOOM コマンドによって)エディタがディスプレイ全体に最大化されてていることを示しています。語 paint は print に変更され、カーソルはこの文字列の最後の文字の後に配置されます。

図 41.3 *一連のウィンドウ呼び出しコマンドの実行*



ウィンドウの管理

ウィンドウの管理コマンドを使用すると、より効率的にウィンドウにアクセスして使用できます。次のリストに、ウィンドウの管理で頻繁に使用するコマンドを示します。

BYE
SAS セッションを終了します。

CLEAR
アクティブウィンドウからすべてのテキストを削除します。

END
ウィンドウを閉じます。エディタでは、このコマンドは SUBMIT コマンドのように動作します。

NEXT
次に開いているウィンドウにカーソルを移動し、そのウィンドウをアクティブにします。

PREVWIND
以前に開いているウィンドウにカーソルを移動し、そのウィンドウをアクティブにします。

RECALL
テキストエディタウィンドウ(エディタや SAS NOTEPAD など)からサブミットされたステートメントをテキストエディタに戻します。

ZOOM
ウィンドウをディスプレイ全体に拡大します。再度実行すると、ウィンドウを元のサイズに戻します。このコマンドはすべての動作環境で使用できるわけではありません。

ウィンドウのスクロール

スクロールコマンドを使用して、テキスト内を移動できます。コマンドの名前はその機能を表しています。次のコマンドがあります。

BACKWARD
ウィンドウのコンテンツを後方向に移動します。

FORWARD
ウィンドウのコンテンツを前方向に移動します。

LEFT
ウィンドウのコンテンツを左に移動します。

RIGHT
ウィンドウのコンテンツを右に移動します。

TOP
カーソルをウィンドウの最初の行の先頭文字に移動します。

BOTTOM
テキストの最後の行を表示します。

HSCROLL、VSCROLL
HSCROLL は、LEFT コマンドまたは RIGHT コマンド使用時の左右への移動量を決定します。VSCROLL は、FORWARD コマンドまたは BACKWARD コマンド使用時の前方または後方への移動量を決定します。

必要に応じて、HSCROLL コマンドおよび VSCROLL コマンドと一緒に次のオプションを使用します。HALF がデフォルトのスクロール幅です。

PAGE
ウィンドウに表示されるすべてです。

HALF
ウィンドウに表示される半分です。

MAX
ウィンドウに表示される左右上下の最大部分です。

n
n 行または n カラムで、ユーザーが指定する数です。

CURSOR
HSCROLL と組み合わせて使用している場合、カーソルは LEFT コマンドの実行時にはディスプレイの左に、RIGHT コマンドの実行時はディスプレイの右に移動します。

注: このオプションは、編集可能なウィンドウでのみ有効です。

VSCROLL と組み合わせて使用している場合、カーソルは FORWARD コマンドの実行時にはディスプレイの上方に、BACKWARD コマンドの実行時はディスプレイの下方に移動します。

例: ウィンドウのスクロール

自動水平スクロール値を 5 文字スペースに設定するには、次を指定します。

```
hscroll 5
```

この後、LEFT コマンドまたは RIGHT コマンドを実行すると、指定の方向へ 5 文字スペース移動します。自動垂直スクロール値をページ半分に設定する場合、次を指定します。

```
vscroll half
```

この後、FORWARD コマンドを実行すると、前のページ半分がディスプレイに残り、新しいページ半分がスクロールされて表示されます。

特定の行数を前進または後進する必要がある場合、FORWARD コマンドでスクロール幅を使用して、デフォルトのスクロール値を一時的に上書きします。スクロール値は、BACKWARD、FORWARD、LEFT、RIGHT の各コマンドで指定できます。

ウィンドウでの色の変更と強調表示

ディスプレイがカラーをサポートする場合、SAS で用意されている簡単な方法を使用して環境をカスタマイズできます。COLOR コマンドを使用して SAS ウィンドウ環境の色を変更できます。SYNCONFIG コマンドを使用して、SAS コードの配色を変更することもできます。ウィンドウ環境の色を変更するには、単に COLOR コマンドと、続けて変更するフィールド要素やウィンドウ要素と目的の色を指定します。点滅や色の反転などの、属性の強調表示も変更できます。

F たとえば、ウィンドウの境界を赤に変更する場合、次を指定します。

```
color border red
```

これで境界は赤に変更されます。

ほかに利用可能な色は、青、緑、青緑色、ピンク、黄色、白、黒、赤紫、灰色、茶色、オレンジです。指定の色が使用不可の場合、SAS はその最も近い色へマッチしようとします。

一部の色の選択肢は、特定のウィンドウでのみ有効です。

詳細については、**SASColor** ウィンドウのオンラインヘルプを参照してください。**SASColor** ウィンドウには、SASCOLOR コマンドを使用してアクセスできます

エディタウィンドウや NOTEPAD などの、コードを入力するウィンドウのテキストの配色も変更できます。SAS 言語の別々の要素をそれぞれの色で表示できるため、コードの解析がより簡単になり便利です。コードの配色を変更するには、SYNCONFIG コマンドを使用します。SYNCOLOR コマンドを使用して、これらのウィンドウの色分けのオン/オフを切り替えます。

コードの作成と編集を行うウィンドウの配色の変更に関する詳細は、SYNCONFIG コマンドを発行して使用できるオンラインヘルプを参照してください。

テキストの検索と変更

文字列を検索して変更することは頻繁にあります。FIND コマンドとその後に文字列を指定して文字列を検索できます。その後、カーソルは検索する文字列の最初の出現箇所に移動します。CAPS ON が有効になっている場合、文字列を引用符で囲むのを忘れないでください。

文字列を変更するには、CHANGE コマンドに続けて 1 スペースを挿入して現在の文字列、さらに続けて 1 スペースを挿入して新しい文字列を指定します。埋め込みブランクや特殊文字を含む文字列はすべて引用符で囲むようにしてください。FIND コマンドと CHANGE コマンドでは両方とも、文字列は任意の長さにできます。

FIND コマンドと CHANGE コマンドのどちらも次のオプションを指定可能で、特殊な文字列を検索したり、変更したりできます。

- ALL
- FIRST
- ICASE
- LAST
- NEXT
- PREFIX
- PREV
- SUFFIX
- WORD

一緒に使用可能なオプションの詳細については、*SAS System Options: Reference* を参照してください。オプション ALL は、検索や変更を、指定した文字列のすべての出現に対して行うことに注意してください。次の例では、host のすべての出現が operating environment に変更されます。

```
change host 'operating environment' all
```

FIND コマンドで以前に指定された文字列の検索を再開するには、RFIND コマンドを指定します。CHANGE コマンドで以前に指定された文字列の変更を継続するには、RCHANGE コマンドを指定します。文字列の前の出現を検索するには、BFIND または FIND PREV コマンドを指定します。BFIND コマンドでは、PREFIX、SUFFIX、および WORD の各オプションを使用できます。

テキストの切り取り、貼り付け、格納

切り取りと貼り付け機能を使用して、次の操作を行えます。

- 操作するテキストを特定します。
- テキストのコピーを、ペーストバッファと呼ばれる一時記憶域に格納します。
- テキストを挿入します。
- 現在のすべてのペーストバッファの名前をリストで表示するか、削除します。

次のコマンドを使用して、テキストの操作/格納が行えます。

MARK
　切り取りまたは貼り付けを行うテキストを特定します。

CUT
　選択したテキストをディスプレイから削除し、ペーストバッファに格納します。

STORE
　選択したテキストをコピーして、ペーストバッファに格納します。

PASTE
　ペーストバッファに格納されたテキストをカーソルのある場所に挿入します。

テキストの操作

SAS テキストエディタ

SAS テキストエディタは、Base SAS、SAS/FSP、および SAS/AF の各ソフトウェアのエディタウィンドウと SAS NOTEPAD ウィンドウで使用できる編集機能です。編集ウィンドウのコマンド行、およびコードが表示されている任意の行からテキストを編集できます。

このセクションでは、SAS テキストエディタを使用した一般的なテキスト編集タスクの実行に使用可能なコマンドについて説明します。SAS ウィンドウ環境コマンドの詳細については、“SAS ウィンドウ環境で使用できるコマンドの種類” (736 ページ)を参照してください。

テキストの移動と再配置

テキストの単一行の移動、削除、挿入およびコピーの一部の基本操作はすでに説明済みです。テキストのブロックの操作も規則は同様で、単に編集する最初と最後の行にダブルレターを使用します。

たとえば、次のリストをアルファベット順にするには、テキストのブロックを移動する必要があります。例の行 5 と行 6 の MM (移動)ブロックコマンド、および行 1 の B 行コマンドに注意してください。

```
b 001  c signifies the line command copy
00002  d signifies the line command delete
00003  i signifies the line command insert
00004  m signifies the line command move
mm 05  a signifies the line command after
mm 06  b signifies the line command before
```

```
00007  r signifies the line command repeat
```

Enter を押して変更を実行します。次に結果を示します。

```
00001  a signifies the line command after
00002  b signifies the line command before
00003  c signifies the line command copy
00004  d signifies the line command delete
00005  i signifies the line command insert
00006  m signifies the line command move
00007  r signifies the line command repeat
```

さらにいくつかコマンドをマスタすると、テキストエディタ内ではるかに複雑な操作を実行できます。いくつかのコマンドは、使用してテキストの位置を調整できます。テキストを左揃えにするには JL (左揃え)コマンドを、右揃えにするには JR (右揃え)コマンド、中央揃えにするには JC (中央揃え)を指定します。テキストブロックの位置を調整するには、JJL、JJR、および JJC コマンドを使用します。たとえば、次のテキストを中央揃えにするとします。

```
00001 Study of Advertising Responses
00002 Topnotch Hotel Website
00003 Conducted by Global Information, Inc.
```

その場合は、JJC ブロックコマンドを最初と最後の行に追加して、Enter を押します。

次の行コマンドのセットを実行して、指定したスペースの数だけテキストを左右にシフトすることもできます。

>[n]
: ユーザーが指定したスペースの数だけ、テキストを右にシフトします。デフォルトは 1 スペースです。

<[n]
: ユーザーが指定したスペースの数だけ、テキストを左にシフトします。デフォルトは 1 スペースです。

テキストのブロックを左にシフトするには、次のコマンドをブロックの最初と最後の行番号の上に指定します。

```
<<[n]
```

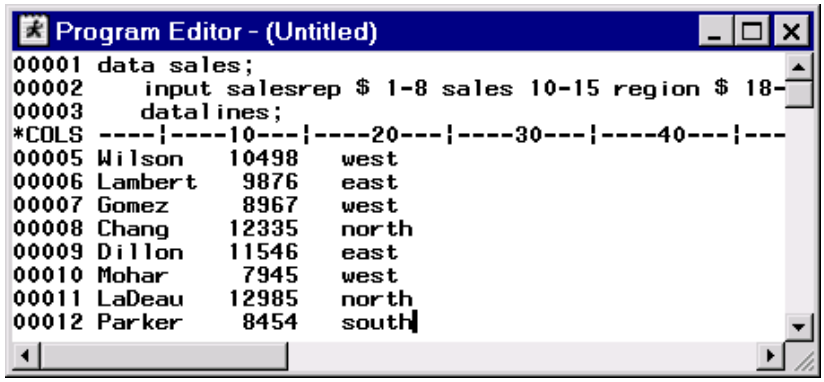
テキストのブロックを右にシフトするには、次のコマンドを使用して指定します。

```
>>[n]
```

カラム番号と行番号の表示

テキストエディタでカラム番号を表示するには、COLS 行コマンドを指定します。次の図に示すように、このコマンドは、カラムモードで INPUT ステートメントを記述している場合に特に便利です。

図 41.4 COLS コマンドの実行



COLS 行コマンドまたはその他の保留中の行コマンドをすべて削除するには、コマンド行で RESET コマンドを実行します。COLS コマンドを指定した行で D (削除)行コマンドを実行しても、同様の結果を得ることができます。

NUMBERS コマンドは、エディタウィンドウや SAS NOTEPAD ウィンドウのデータ行に番号を表示します。データ行に番号を追加するには、次のコマンドを指定します。

```
numbers on
```

番号を削除するには、次を指定します。

```
numbers off
```

また、NUMBERS コマンドを引数なしで使用する場合、コマンドを 1 度実行すると番号付けがオンに、再度実行すれば番号付けがオフになります。

テキストの大文字または小文字変換

概要

テキストの大文字/小文字変換には、2 種類のタスクを実行するコマンドが 2 セットあります。

表 41.4 テキストの大文字/小文字の変更

コマンド	アクション
CAPS	デフォルトの変更
CU 行コマンド、CL 行コマンド	既存テキストの大文字/小文字の変更

デフォルトの変更

テキストを入力する際のデフォルトの大文字/小文字設定を変更するには、CAPS コマンドを使用します。CAPS コマンドを実行後、入力するテキストは、ENTER または

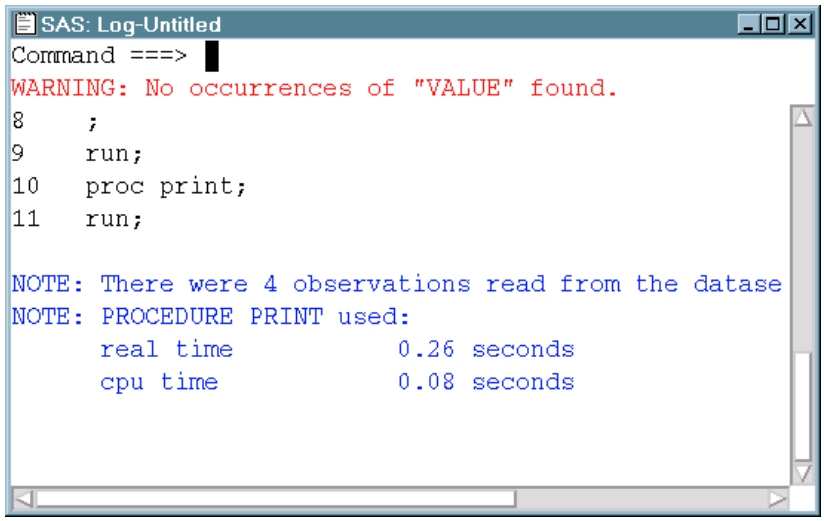
RETURN を押すと同時に大文字に変換されます。一部の動作環境下では、CAPS ON が有効な場合、入力または変更される文字はカーソルがその行から移動されると大文字に変換されます。FIND、RFIND、BFIND コマンドで指定される文字列は、文字列が引用符で囲まれていない限り、大文字で入力されたものとして見なされます。

たとえば、語 value を**ログ**ウィンドウで検索する場合、コマンド行で次を指定します。

```
find value
```

CAPS コマンドがすでに指定されていた場合、SAS は、語 value ではなく、語 VALUE を検索します。次に示すように、VALUE は見つからなかったことを示すメッセージが表示されます。

図 41.5 *CAPS ON が有効になっている FIND コマンドの結果*



しかし、次のコマンドを指定すると SAS は語 value を検索して、それを検出します。

```
find 'value'
```

設定 CAPS ON は、セッションの終了か、またはユーザーがそれをオフにするまで有効です。次を指定して、CAPS コマンドを実行できます。

```
caps on
```

自動の大文字変換を止めるには、次を指定します。

```
caps off
```

また、CAPS コマンドは切り替えスイッチのように使用でき、1 度実行するとコマンドがオンに、再度実行すればコマンドがオフになります。

既存のテキストの大文字/小文字の変更

すでに入力済みのテキストを大文字や小文字に変換するには、行コマンド CU と CL を使用します。1 行のテキストを、大文字に変換するには CU (大文字変換)コマンドを、小文字に変換するには CL (小文字変換)コマンドを実行します。

次の例では、CU 行コマンドと CL 行コマンドはそれぞれ大文字または小文字に変換される、1 行のテキストをどちらも選択しています。

```
   00001 Study of Gifted Seventh Graders
   cu002 Burns County Schools, North Carolina
   cl003 Conducted by Educomp, Inc.
```

Enter を押してコマンドを実行します。テキスト行は、次のように変換されます。

```
00001 Study of Gifted Seventh Graders
00002 BURNS COUNTY SCHOOLS, NORTH CAROLINA
00003 conducted by educomp, inc.
```

テキストのブロックの場合、2 つの選択肢があります。1 番目の選択肢では、CCU ブロックコマンドを実行してテキストのブロックを大文字に変換し、CCL ブロックコマンドを実行してテキストのブロックを小文字に変換します。変換するテキストの最初と最後の行、両方に、ブロックコマンドを配置します。2 番目の選択肢では、次に示すように数値の引数を指定して、大文字/小文字変換する行の数を指定します。

```
cu3 1 Study of Gifted Seventh Graders
00002 Burns County Schools, North Carolina
00003 Conducted by Educomp, Inc.
```

Enter を押してコマンドを実行します。次に示すように、3 行のテキストが大文字変換されます。

```
00001 STUDY OF GIFTED SEVENTH GRADERS
00002 BURNS COUNTY SCHOOLS, NORTH CAROLINA
00003 CONDUCTED BY EDUCOMP, INC.
```

テキストの結合と分割

複数の行コマンドを使用して、個々のテキストを結合したり分割したりできます。TC (テキスト接続)コマンドを使用して、2 行のテキストを接続できます。たとえば、次の行を結合するには、TC 行コマンドを次のように入力します。2 番目の行は、最初の行の最後の語と 2 番目の行の最初の語との間にスペースを作成するために、わざとカラム 2 で開始されていることに注意してください。

```
tc001 This study was conducted by
00002  Educomp, Inc., of Annapolis, Md.
```

Enter を押してコマンドを実行します。行は次のように表示されます。

```
00001 This study was conducted by Educomp, Inc., of Annapolis, Md.
```

逆に、TS (テキスト分割)コマンドは、カーソルの現在の位置から後のテキストを新しい行の先頭にシフトします。

TC 行コマンド、TS 行コマンドなどのいずれの行コマンドもファンクションキーを使用して実行できますが、コロンを前に付けて定義する必要があります。

ファイルの操作

ファイルの検索方法

概要

SAS ウィンドウ環境には、次を含め、ファイルやライブラリメンバを検索する方法が多数あります。

- **エクスプローラ**ウィンドウを使用する方法
- **検索**ウィンドウを使用する方法

エクスプローラを使用したファイルの検索

多くの動作環境では、SAS ウィンドウ環境が開くと**エクスプローラ**ウィンドウもデフォルトで開きます。**エクスプローラ**ウィンドウがデフォルトで表示されない場合、EXPLORER コマンドを発行してこのウィンドウを開くことができます。

- **エクスプローラ**ウィンドウのコンテンツのみのビューでファイルを検索するには、**ライブラリ**フォルダまたは**ファイルショートカット**フォルダを選択し、ポップアップメニューから**開く**を選択します。このプロセスを、目的のファイルを特定するまで、サブフォルダでも続けます。
- **エクスプローラ**ウィンドウのツリービューでファイルを検索するには、ツリーの展開アイコン(+アイコンと−アイコン)を使用して目的のファイルをウィンドウに表示します。

注: **エクスプローラ**ウィンドウの異なるレベル間を移動するには、次に示す特定のナビゲーションツールを使用すると便利です。

メニュー
表示 ⇨ **1 つ上のレベルへ**

コマンド
UPLEVEL

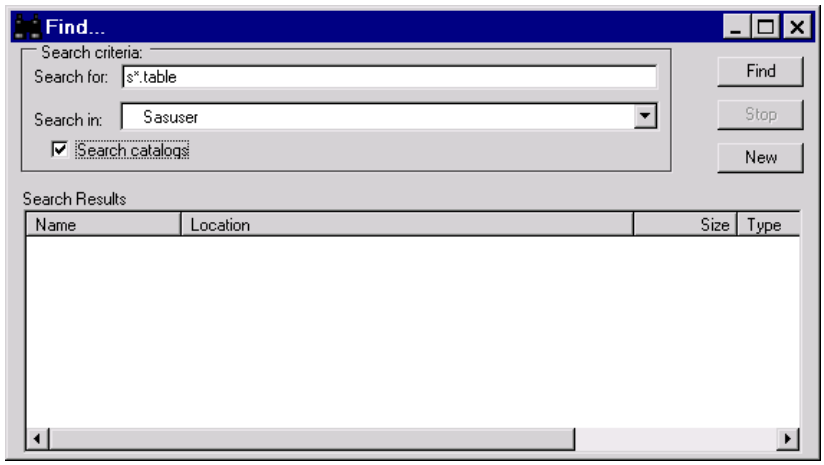
エクスプローラウィンドウビュー選択の詳細については、“エクスプローラウィンドウのカスタマイズ ” (781 ページ)を参照してください。

検索ウィンドウを使用したファイルの検索

検索ウィンドウを使用して、SAS ライブラリに存在する表現(テキスト文字列やライブラリメンバなど)を検索できます。デフォルトの検索は、ライブラリ内のカタログを除いてすべてを検索しますが、検索のチェックボックスを選択して、ライブラリのカタログも含めるようにできます。

次の表示画面は、**検索対象**と**検索場所**フィールドに検索情報を入れた状態の**検索**ウィンドウを示しています。

図 41.6 *検索ウィンドウ*



ファイルを検索するには、次の操作を実行します。

1. **エクスプローラ**ウィンドウから**ツール** ⇨ **検索**を選択して**検索**ウィンドウを開きます。

   あるいは、EXPFIND または EXPFIND <library-name>コマンドを発行します。EXPFIND コマンドを発行すると、SASUSER がデフォルトのライブラリになります。

EXPFIND WORK コマンドを発行すると、WORK がデフォルトのライブラリになります。

2. **検索対象**フィールドに、検索する表現を入力します。ワイルドカード文字を使用できます。
3. **検索場所**ドロップダウンリストから、検索するライブラリを選択します。
4. 選択したライブラリのカタログまで検索範囲を広げるには、**カタログ内**をクリックします。

   カタログの検索は、ライブラリ内にあるカタログのサイズと数に応じて、検索時間が大幅に長くなる場合があります。
5. **検索**をクリックします。

例: 検索ウィンドウを使用したファイルの検索

特定の文字で始まり、特定のライブラリに存在する、TABLE ファイルを検索できます。文字 S で始まり、SASHELP ライブラリに存在するファイルの場合を次に示します。

1. **ツール** ⇨ **検索**を選択して**検索**ウィンドウを開きます。
2. **検索対象**フィールドに `s*.table` と入力します。
3. **検索場所**ドロップダウンリストから、**SASHELP** を選択します。
4. **検索**をクリックします。

ファイル固有のコマンドの発行

SAS ウィンドウ環境には、ファイルを特定した後、ファイルに対して発行可能な多数のコマンドがあります。使用可能なコマンドは、操作しているファイルの種類によって異なります。

1. 操作するファイルを検索します。詳細については、“ファイルの検索方法” (748 ページ)を参照してください。
2. ファイルを選択して、ファイルを右クリックします。ファイル固有のコマンドのリストが表示され、選択を行うことができます。

動作環境の情報

z/OS 動作環境を使用している場合、項目の隣の選択フィールドに`?`を入力してポップアップメニューを開くことができます。その他に、項目の隣の選択フィールドに `s` または `x` を入力することも可能です。

ファイルを開く

SAS ウィンドウ環境では、さまざまな方法を使用してファイルを開くことができます。

エクスプローラから SAS ファイルを開くには、次の操作を行います。

1. 開く対象のファイルが表示されるまで、ライブラリと目的のライブラリメンバを開いていきます。
2. ファイルを選択して、ポップアップメニューから**開く**を選択します。

   ファイルの種類によっては、**エディタで開く**を選択できる場合もあります。

注: 一部のケースでは、ポップアップメニューから **SAS ノートパッドに表示**も選択可能で、ファイルを SAS **NOTEPAD** ウィンドウで開くことができます。

ファイルショートカットがあるファイルを開くには、次の操作を行います。

1. **ファイルショートカット**フォルダを開きます。
2. ファイルショートカットを選択して、ポップアップメニューから**開く**を選択します。

ファイルショートカットの割り当て

ファイルショートカット参照は、外部ファイル(.sas プログラムファイルや.dat テキストファイルなど)へのエイリアスを提供します。ファイルショートカットは、ファイル参照名(fileref)と同じです。ドラッグアンドドロップ機能がサポートされる動作環境では、**エクスプローラ**ウィンドウから**エディタ**ウィンドウへファイルショートカットをドラッグしてコンテンツを表示できます。

ファイルショートカットを割り当てるには、次の操作を実行します。

1. **エクスプローラ**ウィンドウから、**ファイルショートカット**フォルダを選択します。
2. **ファイル** ⇨ **新規**を選択します。
3. **ファイルショートカットの割り当て**ウィンドウの**名前**フィールドに、ファイルショートカットの名前を入力します。
4. ファイルショートカットに使用する手法またはデバイスを選択します。

   手法ドロップダウンリストで選択可能な手法またはデバイスは、動作環境に応じて異なります。DISK 手法がデフォルトの手法です(動作環境で使用可能な場合)。
5. SAS の起動時に毎回ショートカットを自動的に割り当てるには、**起動時に有効**チェックボックスを選択します。このオプションは、すべてのファイルショートカット手法で使用できるわけではありません。

   ファイルショートカットの起動時の有効化を停止するには、SAS **エクスプローラ**ウィンドウでそのファイルショートカットを選択し、ポップアップメニューで**削除**を選択します。
6. メソッド情報領域のフィールドに、ファイルショートカットを作成するファイルの名前と場所などを入力します。実際のファイルを検索するには、**参照**を選択します。この領域で使用可能なフィールドは、選択した手法やデバイスの種類によって異なります。

   注: 新しい手法の種類を選択すると、メソッド情報のフィールドに入力済みのエントリはすべて消去されます。
7. **OK** を選択して、新しいファイルショートカットを作成します。SAS **エクスプローラ**ウィンドウのファイルショートカットフォルダにファイルショートカットが表示されます。

使用している動作環境に応じて、次の方法を使用してファイルショートカットを作成できます。

メニュー
ファイル ⇨ **新規**

エクスプローラウィンドウの**ファイルショートカット**にマウスが置かれている場合。

コマンド
DMFILEASSIGN<*file-shortuct-name*><METHOD=><AUTO=>Yes | No

file-shortcut-name
既存のファイルショートカット参照を指定します。

METHOD= *method-name*
ファイルショートカットの割り当てウィンドウが開くときに使用される手法を指定します。

AUTO= Yes|No
ファイルショートカットの割り当てウィンドウが開く場合にそのウィンドウの**起動時に有効**チェックボックスの状態を設定します。

ポップアップ
ファイルショートカットの作成(**エクスプローラ**ウィンドウの**ファイルショートカット**フォルダを開いている場合)。

ツールバー
新規(**エクスプローラ**ウィンドウの**ファイルショートカット**にマウスが置かれている場合)。

既存のファイルショートカットの変更

必要に応じて、既存のファイルショートカット参照を変更できます。

コマンド行からの操作を次に示します。

1. 次のコマンドを発行します。

   DMFILEASSIGN *file-shortcut-name*

   ファイルショートカットの割り当てウィンドウが表示されます。選択したファイルショートカットに固有の情報が、フィールドに含まれています。

2. 必要に応じて、**ファイルショートカットの割り当て**ウィンドウのフィールドを編集します。

SAS **エクスプローラ**からの操作を次に示します。

1. **ファイルショートカット**フォルダを右クリックして、**開く**を選択します。または、フォルダをダブルクリックして開きます。
2. 変更するファイルショートカット参照を右クリックして、**変更**を選択します。
3. 必要に応じて、**ファイルショートカットの割り当て**ウィンドウのフィールドを編集します。

動作環境の情報
z/OS 動作環境を使用している場合、項目の隣の選択フィールドに**?**を入力してポップアップメニューを開くことができます。その他に、項目の隣の選択フィールドに **s** または **x** を入力することも可能です。

ファイルの印刷

さまざまな方法を使用してファイルを印刷できます。多くの場合、印刷機能は、操作しているファイルの種類と使用している動作環境によって異なります。

ファイルを印刷可能な一般的な方法を次に示します。

エクスプローラからの印刷
目的のファイルを**エクスプローラ**ウィンドウで検出します。そのファイル上で右クリックして、**印刷**を選択します。

テキストエディタからの印刷
エディタや SAS NOTEPAD などのテキストエディタでファイルを開きます。テキストエディタの印刷コマンドを使用します。

ファイルの印刷の詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

SAS プログラムでの作業

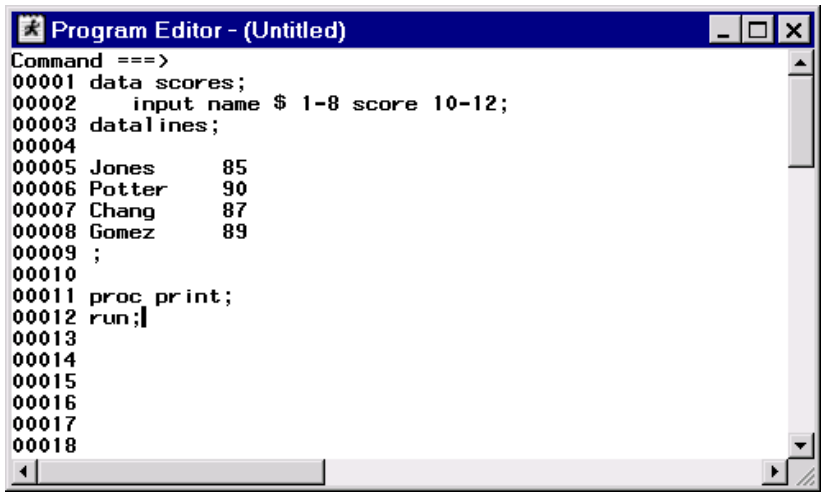
エディタウィンドウ

概要

SAS プログラムを処理するとき、通常、SAS プログラミングウィンドウ(エディタ、ログ、アウトプットの各ウィンドウ)を使用します。これらのプログラミングウィンドウのうち、エディタが最もよく使用されるウィンドウです。これを使用して次の操作を行えます。

- SAS プログラムを定義するプログラムステートメントの入力とサブミット。
- テキストの編集。
- プログラムのファイルへの保存。
- すでに作成済みのファイルからコンテンツのコピー。
- 別ファイルへのコンテンツのコピー。

図 41.7 行番号がオンになっているエディタウィンドウ



注: ここに表示する**プログラムエディタ**ウィンドウには行番号が含まれています。行番号は、プログラムの作成時や編集時に便利です。行番号のオン/オフを切り替えるには、NUMBERS コマンドを発行します。

コマンド行のコマンドとエディタ

エディタでプログラムを開発するときに、便利なコマンドが多数あります。次のコマンドをコマンド行から実行できます。

TOP
エディタの先頭へスクロールします。

BOTTOM
テキストの最後の行へスクロールします。

BACKWARD
テキストの先頭に向かって逆方向へスクロールします。

FORWARD
テキストの末尾に向かって順方向へスクロールします。

LEFT
ウィンドウの左へスクロールします。

RIGHT
ウィンドウの右へスクロールします。

ZOOM
ウィンドウのサイズを大きくします。このコマンドを再度発行すると、ウィンドウを前のサイズに戻すことができます。

UNDO
最新にサブミットされたテキスト編集コマンドの効果をキャンセルします。UNDO コマンドを実行し続けていくと、以前のコマンドが最後のものから遡って元に戻されます。

SUBMIT
現在の SAS ウィンドウ環境セッションのステートメントブロックをサブミットします。

RECALL
現在の SAS ウィンドウ環境セッションで最新にサブミットされたステートメントブロックを**プログラムエディタ**ウィンドウに戻します。RECALL コマンドを実行し続けていくと、以前のステートメントが最後のものより遡って元に戻されます。

CLEAR
指定されたとおりにウィンドウをクリアします。次の例で示すように、別のウィンドウから適切なオプションを指定した CLEAR コマンドを実行し、エディタ、ログ、アウトプットの各ウィンドウをクリアできます。`clear pgm clear log clear output`

CAPS
すべての入力を大文字に変換します。

FIND
指定された文字列を検索します。文字列に埋め込みブランクまたは特殊文字が含まれている場合、文字列を引用符で囲みます。

CHANGE
指定された文字列を別の文字列に変更します。コマンドキーワードの後に最初の文字列、1 スペース、2 番目の文字列と続けます。埋め込みブランクと特殊文字の規則が適用されます。たとえば、次を指定するとします。`change 'operating system' platform`

この CHANGE コマンドは、最初に検索された operating system と語 platform とを置換します。最初の文字列には埋め込みブランクが含まれているため、引用符で囲む必要があることに注意してください。

注: より有用と思われるコマンド行のコマンドの一部をここに掲載しています。**プログラムエディタ**ウィンドウでは、ほとんどすべての SAS コマンドが有効です。その他のコマンド行のコマンドの詳細については、“SAS ウィンドウの操作” (739 ページ)を参照してください。

行コマンドとエディタ

プログラムエディタウィンドウの最左部分には、番号が付いたフィールドがあります。このフィールドが行コマンドを入力する場所です。これらのコマンドは 1 つ以上の文字で示され、行を移動、コピー、削除、位置調整、または挿入できます。

いくつかの一般的な行コマンドを、以下に示します。

- M -- 1 行のテキストを移動します。
- C -- 1 行のテキストをコピーします。
- D -- 1 行のテキストを削除します。
- I -- 1 行のテキストを挿入します。

一部の行コマンドは使用する場合、場所も指定する必要があります。たとえば、エディタ内で行の番号付きフィールドに **M** と入力した場合は、テキスト行の移動先の場所を指定する必要があります。場所の指定には、**A** (後)行コマンドと **B** (前)行コマンドを使用できます。

行の番号付きフィールドに **A** を入力した場合、ENTER キーを押すと、移動対象のテキスト行は、A でマークされた行の後に配置されます。行の番号付きフィールドに **B** を入力した場合、ENTER キーを押すと、移動対象のテキスト行は、B でマークされた行の前に配置されます。

次の例では、行コマンドを使用して、**プログラムエディタ**ウィンドウで 1 行のテキストを新しい場所へ移動する方法を示しています。次の行をアルファベット順にするには、最初の行を最後の行の後に配置する必要があります。これを行うには、M 行コマンドと A 行コマンドを使用します。

```
m 001 Lincoln f Wake Ligon  135
00002 Andrews f Wake Martin 140
00003 Black   m Wake Martin 149
a 004 Jones   m Wake Ligon  142
```

Enter を押すと、**プログラムエディタ**ウィンドウで行が次のように表示されます。

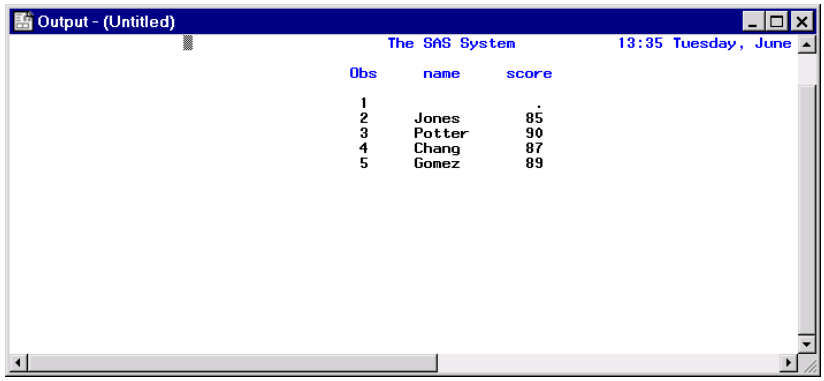
```
00001 Andrews f Wake Martin 140
00002 Black   m Wake Martin 149
00003 Jones   m Wake Ligon  142
00004 Lincoln f Wake Martin 135
```

プログラムエディタウィンドウでプログラムのステートメントの編集に使用できる、行コマンドや行コマンドの組み合わせは他にも多数あります。詳細については、“テキストの操作” (744 ページ)を参照してください。

アウトプットウィンドウ

現在の SAS セッションからのプロシジャ出力を、**アウトプット**ウィンドウで表示してスクロールできます。プログラムのサブミットの結果は、出力を生成する PROC ステップが含まれている場合、通常、**アウトプット**ウィンドウに表示されます。

図 41.8 サブミットされたプロシジャの結果を表示するアウトプットウィンドウ



プログラムエディタウィンドウのところで前述したコマンド行のコマンドのほとんどが**アウトプット**ウィンドウで使用できます。SAS セッション内のすべての出力が前の出力に追加されるため、CLEAR コマンドは、**アウトプット**ウィンドウで特に便利です。出力が蓄積されないようにするには、次のプログラムをサブミットする前に CLEAR コマンドを実行してください。他の任意のウィンドウからは、次を指定して**アウトプット**ウィンドウをクリアできます。

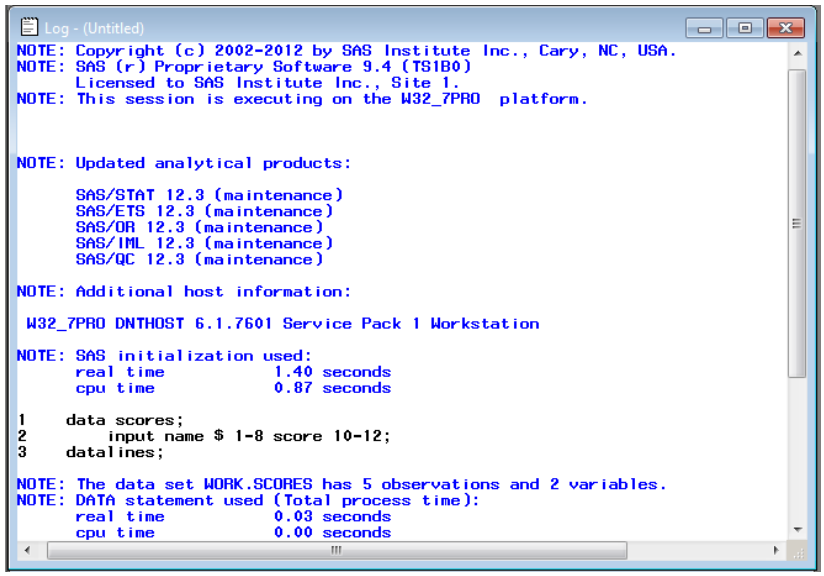
```
clear output
```

ログウィンドウ

ログウィンドウを使用して、次を行うことができます。

- プログラミングエラーがいつ発生したかの認識
- それらのエラーの修正に必要な情報の取得
- エラー修正のためにユーザーが行った手順のフィードバックの受信

図 41.9 *SAS セッションに関する情報を表示するログウィンドウ*



ログウィンドウには、ユーザーがサブミットした SAS ステートメントとそのプログラムに関する SAS からのメッセージが表示されます。ほとんどの動作環境下では、**ログ**ウィンドウに次の情報が表示されます。

- プログラムが実行された時間
- プログラムが実行された SAS のリリース
- コンピュータインストールの詳細とそのサイト番号
- 指定された出力データセットのオブザベーションと変数の数
- 各ステップで使用されたコンピュータリソース

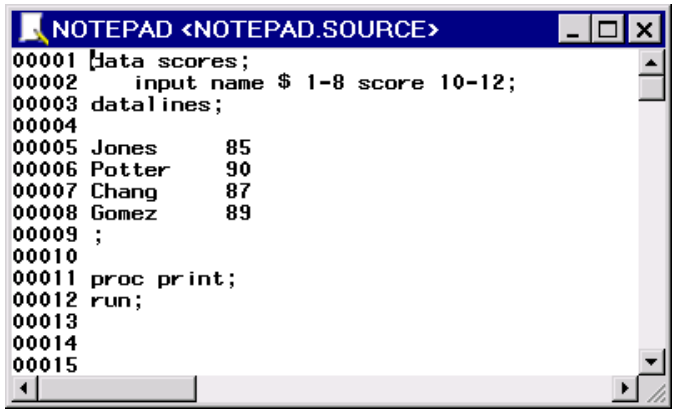
エディタウィンドウやアウトプットウィンドウと同様に、**ログ**ウィンドウでコマンド行のコマンドを使用できます。詳細については、“エディタウィンドウ” (753 ページ)を参照してください。

その他のエディタの使用

NOTEPAD ウィンドウ

エディタは SAS プログラムを記述するために設計されていますが、**NOTEPAD** ウィンドウを使用して SAS プログラムの作成や編集を行うこともできます。NOTEPAD はテキストエディタで、SAS プログラムの作成、編集、保存、サブミットに使用できます。コードを操作する別の場所として、NOTEPAD は便利です。NOTEPAD を開くには、NOTEPAD コマンドまたは NOTES コマンドを発行します。

図 41.10 行番号がオンになっている SAS NOTEPAD ウィンドウ



注: ここに表示する **NOTEPAD** ウィンドウには行番号が含まれています。プログラムを作成したり編集するとき、行番号は有用です。NOTEPAD で行番号のオン/オフを切り替えるには、NUMBERS コマンドを発行します。

複数の NOTEPAD を開いている場合、NOTEPAD ウィンドウと、**プログラムエディタ**ウィンドウ、複数の SAS セッション、およびその他のアプリケーションとの間でテキストの切り取り、コピー、貼り付けを行うことができます。

注: NOTEPAD からプログラムをサブミットするには、**実行** ⇨ **サブミット**を選択するか、または NOTESUBMIT コマンドを発行するかのいずれかで行うことが必要です。

注: このドキュメントで説明しているプログラム情報には、デフォルトのエディタとしてエディタウィンドウを使用します。

プログラムの作成とサブミット

SAS プログラムを作成してサブミットするには、次の操作を行います。

1. エディタでプログラムのテキストを入力します。
2. コマンド行に `submit` と入力して Enter を押します。

   使用している環境でプログラムのサブミットが割り当てられている、ファンクションキー、メニューコマンド、またはツールバー項目も使用できます。

   注: **NOTEPAD** ウィンドウからプログラムをサブミットする場合は、SUBMIT コマンドのかわりに NOTESUBMIT コマンドを使用する必要があります。

プログラムの保存

プログラムを保存するには、次の操作を行います。

1. **プログラムエディタ**ウィンドウで、プログラムの作成または編集を行います。
2. コマンド行の場合は、FILE コマンドの後にファイル参照名または実際のファイル名を続けて発行します。実際のファイル名を使用する場合は、ファイル名を引用符で囲んでください。

FILE コマンドは、**プログラムエディタ**ウィンドウのコンテンツをクリアしません。プログラムのコピーを 1 つ保存し、その後**プログラムエディタ**ウィンドウで作業を続けられます。

プログラムを既存のファイル参照名またはファイル名で保存しようとすると、ダイアログボックスが表示されます。ダイアログボックスでは、次の操作を選択できます。

- 既存のファイルのコンテンツを新しいファイルで上書き
- 新しいファイルを既存のファイルに追加
- FILE コマンドのキャンセル

更新したバージョンでファイルを頻繁に置換したい場合もあります。ダイアログボックスを非表示にするには、FILE コマンドでファイル参照名や完全ファイル名の後に、REPLACE オプションを追加します。**プログラムエディタ**ウィンドウのテキストを既存のファイルの末尾に追加するには、FILE コマンドでファイル参照名や完全ファイル名の後に APPEND オプションを指定します。

注: プログラムを SAS オブジェクトまたは動作環境に固有のファイルとして格納することもできます。プログラムの作成または編集後に、**ファイル** ⇨ **オブジェクトとして保存**または**ファイル** ⇨ **名前を付けて保存**をそれぞれのケースに応じて選択します。

プログラムのデバッグ

ユーザーは、プログラムのサブミット後に**ログ**ウィンドウに表示される情報を使用してプログラムのデバッグを行うことができます。プログラムに問題があり、SAS セッション終了後に調べる必要がある場合は、**ログ**ウィンドウのコンテンツを外部ファイルに保存します。

ログウィンドウのコンテンツを外部ファイルに保存するには、次の操作を行います。

1. **ログ**ウィンドウが開いていない場合は、開きます。
2. コマンド行から、ファイル参照名または実際のファイル名を続けた FILE コマンドを実行します。ファイル名を使用する場合は、名前を引用符で囲んでください。

FILE コマンドは、現在表示されている情報を削除せずに、**ログ**ウィンドウの情報のコピーを保存します。既存のファイル参照またはファイルの名前を指定すると、ダイアログボックスが表示され次の 3 つの選択肢が示されます。既存ファイルのコンテンツを新しいファイルで上書きする、新しいファイルを既存ファイルへ追加する、またはコマンドのキャンセルです。

プログラムを開く

SAS プログラムを開く方法はいくつかあります。このセクションでは、最もよく使用される 2 つの方法について説明します。

プログラムエディタウィンドウから SAS プログラムを開くには、次の操作を行います。

1. **ファイル** ⇨ **開く**を選択します。
2. **開く**ウィンドウを使用して、目的の SAS プログラムファイルを検索します。

コマンドを使用して SAS プログラムを開くには、次の操作を行います。

1. **プログラムエディタ**ウィンドウが開いていない場合は、開きます。
2. コマンド行で、FILE コマンドの後に、割り当てられたファイル参照名または実際のファイル名を続けて指定します。実際のファイル名は一重引用符または二重引用符で囲みます。

   デフォルトでは、プログラムは、既存のプログラムステートメントの末尾に追加されます。

注: プログラムステートメントがすでにエディタに存在する場合は、B (前)行コマンドや A (後)行コマンドを使用して、プログラムの追加先の場所を決定できます。行コマンドの詳細については、"行コマンドの使用" (738 ページ)を参照してください。

プログラムエディタウィンドウにすでにあるテキストを、これから開くプログラムで置換する場合、INCLUDE コマンドでファイル参照名またはファイル名の後に REPLACE オプションを指定します。

プログラムの編集

プログラムを編集するには、次の操作を行います。

1. **プログラムエディタ**ウィンドウで既存のプログラムを開きます。
2. 既存のプログラムステートメントを編集するか、または新しいステートメントをプログラムに追加します。

   必要に応じて、コマンド行のコマンドと行コマンドを使用します。
3. プログラムを保存します。

ファイルショートカットへのプログラムの割り当て

プログラムをファイルショートカットへ割り当て、以降のファイルの検索と操作をより容易にできます。ファイルショートカットの詳細については、"ファイルショートカットの割り当て" (751 ページ)を参照してください。

出力の操作

出力の操作の概要

SAS Output Delivery System (ODS)を使用して、SAS プロシジャ出力を管理できます。ODS を完全にサポートするプロシジャでは、次の操作を行えます。

- 生成する生データと 1 つ以上のテーブル定義を組み合わせて、フォーマットされた結果を含む 1 つ以上の出力オブジェクトを生成します
- **結果**ウィンドウの結果フォルダに各出力オブジェクトへのリンクを保存します
- HTML 出力、LISTING 出力、場合によっては SAS/GRAPH 出力などのさまざまな種類のファイル出力を生成できます

- プロシジャ出力から出力データセットを生成できます
- ユーザーがプロシジャを実行する際にいつでも使用可能なテーブル定義を作成することでプロシジャ出力をカスタマイズする方法を提供します

SAS ウィンドウ環境では、結果、テンプレート、プリファレンス、および SAS レジストリエディタの各ウィンドウを介して ODS の多くの機能を使用できます。**結果**ウィンドウには、SAS によって生成されるプロシジャ出力へのポインタがあります。**テンプレート**ウィンドウでは、プロシジャ出力と関連付けが可能なすべてのテーブル、列ヘッダー、およびスタイルテンプレートを管理できます。

結果的に、**プリファレンス**ウィンドウと SAS レジストリエディタは、SAS が生成するプロシジャ出力の種類の設定に使用できます。

このセクションでは、ODS の SAS ウィンドウ環境と関連する部分のみについて説明します。ODS の詳細については、24 章, “SAS 出力と SAS ログの出力指定” (383 ページ)および *SAS Output Delivery System: User's Guide* を参照してください。

出力形式の設定

概要

使用している動作環境に応じて、SAS 出力を 1 つ以上の出力形式(種類)で生成できます。リスト出力がデフォルトの種類です。その他の出力の種類には、HTML、出力データセット、および PostScript があります。プロシジャ出力へのポインタは、**結果**ウィンドウに表示されます。

出力の種類を設定するには、**プリファレンス**ウィンドウ(動作環境で使用可能な場合)、SAS レジストリエディタのいずれか、またはその両方を使用します。

プリファレンスウィンドウを使用した出力の種類の設定

使用している動作環境で**プリファレンス**ウィンドウがサポートされる場合、次に示す手順で出力の種類を設定できます。

1. **ツール** ⇨ **オプション** ⇨ **プリファレンス**を選択するか、DLGPREF コマンドを発行して**プリファレンス**ウィンドウを開きます。
2. **結果**タブを選択します。
3. 生成する出力の種類に一致するチェックボックスを選択または選択解除します。

   HTML 出力の生成を選択した場合、次を選択して出力の詳細を定義できます。

   - HTML スタイル

     スタイルボックスをクリックして 1 つのスタイルを強調表示します。スタイルは、特に出力の色とフォントを定義します。

   - 出力の保存先フォルダ

     現在のセッション期間のみ HTML 出力を保存するには、**WORK フォルダを使用する**を選択します。出力は、ユーザーの現在のセッション終了時に削除されます。

     SAS セッション終了時に削除されないフォルダに HTML 出力を保存するには、**フォルダ**テキストボックスにパスを入力します。

   - **すぐに結果ビューアを表示する**チェックボックス

     選択した場合、HTML 出力が生成されるたびに、ブラウザが自動的に開き出力がロードされます。

SAS レジストリエディタを使用した出力の種類の設定

SAS レジストリエディタを使用して出力の種類を設定するには、次の操作を行います。

1. **ソリューション** ⇨ **アクセサリ** ⇨ **レジストリエディタ**を選択するか、REGEDIT コマンドを発行して、SAS レジストリエディタを開きます。
2. 左側のツリーから、ODS フォルダを展開します。
3. Preferences フォルダを展開します。
4. 目的の出力の種類を選択します。
5. 右側で、値キーを選択し、ポップアップメニューから**変更**を選択します。
6. 表示されるダイアログボックスで、必要に応じて、**値のデータ**フィールドを編集します。

   このフィールドが 1 に設定されている場合、その出力の種類が生成されます。このフィールドが 0 に設定されている場合、その出力の種類は生成されません。

SAS 出力の種類へのデフォルトのビューアの割り当て

SAS で出力を生成すると、出力ポインタが**結果**ウィンドウに表示されます。生成する出力の各種類にデフォルトのビューアを割り当てることができます。デフォルトのビューアが割り当てられると、**結果**ウィンドウの出力ポインタをダブルクリックして、デフォルトのビューアで出力を開くことができます。たとえば、PostScript 出力ポインタをダブルクリックすると、Ghostview が開き PostScript 出力がロードされるようにできます。

動作環境の情報
Windows 動作環境では、デフォルトのビューアはユーザーの Windows レジストリからの情報を使用して自動的に設定されます。

SAS 出力の種類へデフォルトのビューアを割り当てるには、次の操作を行います。

1. **エクスプローラ**ウィンドウから、**ツール** ⇨ **オプション** ⇨ **エクスプローラ**を選択します。
2. **エクスプローラオプション**ウィンドウの上部にあるドロップダウンメニューから、**ホストファイル**を選択します。
3. 登録されているファイルの種類をスクロールして、使用するファイルの種類を検索します。
4. 目的のファイルの種類を選択して、**編集**を選択します。
5. **アクションの編集**ウィンドウで**追加**を選択して、そのファイルの種類用のアクション名とアクションコマンドを入力します。

   たとえば、PostScript ファイルの種類のデフォルトビューアとして Ghostview を設定するには、次のアクション名とアクションコマンドを追加します。

   アクション名
   &Edit

   アクションコマンド
   x ghostview '%s' &
6. **アクションの編集**ウィンドウで **OK** を選択します。
7. 前のステップで指定したアクションを選択して、**標準設定**を選択します。

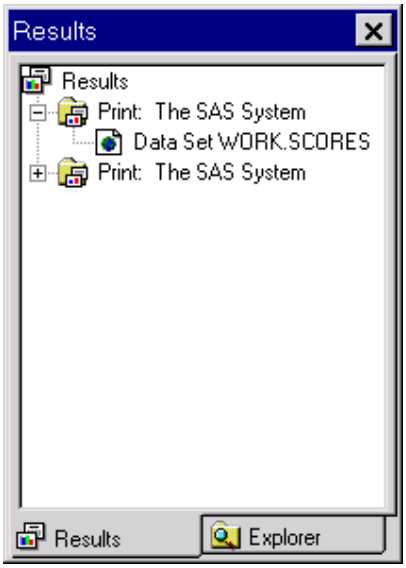
動作環境の情報
Windows 動作環境では、デフォルトのビューアはユーザーの Windows レジストリからの情報を使用して自動的に設定されます。

結果ウィンドウでの出力の操作

概要

結果ウィンドウは、SAS が生成するプロシジャ出力または DATA ステップ出力へのポインタを提供します。このウィンドウは、SAS セッションの起動時にデフォルトで開かれます。**結果**ウィンドウは、**表示** ⇨ **結果**を選択するか、または ODSRESULTS コマンドを発行して開くこともできます。

図 41.11 ツリービューの結果ウィンドウ



結果ウィンドウを使用して、次の操作を行えます。

- ポインタから出力へ移動する。
- 結果ポインタを削除する。
- 結果ポインタの名前を変更する。
- リスト出力を別の形式にして保存する。
- 最初の出力ポインタ項目をすばやく表示する。
- 結果プロパティを表示する。

結果ウィンドウのビューのカスタマイズ

結果ウィンドウは、次の 3 つのビューのいずれかで表示できます。

- ツリー
- コンテンツのみ
- エクスプローラ

ツリービュー(デフォルト)では、ナビゲーション用のツリーのみがあります。コンテンツのみのビューでは、ツリーはオフにされ、コンテンツがフォルダとして表示されます。エクスプローラビューでは、**結果**ウィンドウは 2 つのペインで表示されます。1 つはツリーペインで、もう 1 つはコンテンツペインです。

ツリービューペインのオン/オフを切り替えるには、**結果**ウィンドウから TREE コマンドを発行します。コンテンツペインのオン/オフを切り替えるには、**結果**ウィンドウから

CHILD コマンドを発行します。**結果**ウィンドウの**表示**メニューから、**ツリーの表示**、**コンテンツの表示**などのコマンドを選択しても同じ処理を実行できます

注: デフォルトでは、ツリーペインでの出力ポインタは名前別ではなく、ラベル別にリストされています。ラベルは、通常、出力名より説明的です。SAS システムオプション LABEL を使用して、この設定を変更できます。

結果ポインタを使用した出力の移動

SAS がプロシジャまたは DATA ステップを実行すると、**結果**ウィンドウに出力へのポインタが配置されます。**結果**ウィンドウでのポインタ使用方法の詳細については、"ツリービューの結果ウィンドウでの移動" (763 ページ)、"コンテンツのみのビューの結果ウィンドウでの移動" (763 ページ)、または"エクスプローラビューの結果ウィンドウでの移動" (764 ページ)を参照してください。

ツリービューの結果ウィンドウでの移動

ツリービューでは、出力ポインタがプロシジャの階層で表示されます。SAS 出力を操作するには、次を実行します。

1. 表示するプロシジャ出力に一致するフォルダを特定します。
2. フォルダの隣にある展開アイコン(+アイコンまたは−アイコン)を使用して、そのコンテンツを開いたり非表示にしたりします。

   または次を実行します。

   - フォルダをダブルクリックして、展開または折りたたみます。
   - フォルダを選択して、ポップアップメニューから**開く**を選択します。
3. 目的のポインタを特定したら、ポインタをダブルクリックするか、ポインタを選択してポップアップメニューから**開く**を選択します。

   目的の出力が表示されます。

動作環境の情報
z/OS 動作環境を使用している場合、項目の隣の選択フィールドに`?`を入力してポップアップメニューを開くことができます。その他に、項目の隣の選択フィールドに `s` または `x` を入力することも可能です。

次の方法を使用してツリービュー内を移動することもできます。

メニュー
表示 ⇨ **1 つ上のレベルへ**

コマンド
UPLEVEL

ツールバー
1 つ上のレベルへアイコン

キー
動作環境によっては、矢印キーや Backspace キーを使用した移動もできます。

コンテンツのみのビューの結果ウィンドウでの移動

コンテンツのみのビューでは、出力ポインタはプロシジャの階層で、階層の最上位レベルから表示されます。階層内をドリルダウンまたはロールアップして目的の出力を検索できます。

フォルダを開くと、現在のウィンドウのコンテンツが選択したフォルダのコンテンツと置換されます。SAS 出力を操作するには、次を実行します。

1. 表示するプロシジャ出力に一致するフォルダを特定します。

2. フォルダを選択して、ポップアップメニューから**開く**を選択します。

   フォルダをダブルクリックして開くこともできます。

3. 目的のポイントを特定したら、ポイントをダブルクリックするか、ポイントを選択してポップアップメニューから**開く**を選択します。

   目的の出力が表示されます。

動作環境の情報
z/OS 動作環境を使用している場合、項目の隣の選択フィールドに`?`を入力してポップアップメニューを開くことができます。その他に、項目の隣の選択フィールドに `s` または `x` を入力することも可能です。

エクスプローラビューの結果ウィンドウでの移動
エクスプローラビューには、2 つのウィンドウペインがあります。左ペインには、表示可能なプロシジャ出力の階層ビュー(ツリービュー)があります。右ペインには、現在フォーカスされている項目のコンテンツ(コンテンツビュー)が表示されます。

結果ポインタの削除
結果ポインタが存在するプロシジャフォルダを削除して、その結果ポインタを削除できます。**結果**ウィンドウのプロシジャフォルダを削除すると、そのフォルダ内に存在するすべての出力ポインタが削除されます。

注: リスト出力ポインタを含むプロシジャフォルダを削除すると、実際のリスト出力が**アウトプット**ウィンドウから削除されます。フォルダ内に他の出力ポインタが存在する場合(HTML など)、ポインタのみが削除され、実際の出力は使用可能なまま残ります。

プロシジャ出力を削除するには、次の操作を行います。

1. **結果**ウィンドウで、削除するプロシジャに対応するプロシジャフォルダを選択します。
2. ポップアップメニューから**削除**を選択します。
3. **はい**を選択して削除を確定します。

ヒント 削除するプロシジャフォルダを選択して、**編集** ⇨ **削除**を選択しても、出力ポインタを削除できます。

結果ポインタの名前変更
結果ポインタの名前を変更するには、次の操作を行います。

1. 名前を変更するポインタを選択します。
2. ポップアップメニューから、**名前の変更**を選択します。
3. 新しい名前(と必要に応じて説明)を入力して、**OK** を選択します。

ヒント 名前を変更するポインタを選択して、**編集** ⇨ **名前の変更**を選択しても、結果ポインタの名前を変更できます。

リスト出力の別の形式での保存
結果ウィンドウからリスト出力をファイルに保存するには、次の操作を行います。

1. **結果**ウィンドウのツリーを展開して、目的のリスト出力ポインタを検索します。
2. リスト出力ポインタを選択して、ポップアップメニューから**名前を付けて保存**を選択します。

アウトプットウィンドウからリスト出力をファイルに保存するには、次の操作を行います。

1. **アウトプット**ウィンドウにアクセスします。
2. コマンド行で、FILE コマンドの後にファイル参照名または実際のファイル名を続けて指定します。ファイル名を使用する場合は、ファイル名を引用符で囲んでください。

注: FILE コマンドは、現在表示されている情報を削除せずに、**アウトプット**ウィンドウの情報のコピーを保存します。

リスト出力をカタログオブジェクトとして保存するには、次の操作を行います。

1. **結果**ウィンドウのツリーを展開して、目的のリスト出力項目を検出します。
2. リスト出力項目を選択して、ポップアップメニューから**オブジェクトとして保存**を選択します。

最初の出力ポインタ項目の表示

最初の出力ポインタ項目を表示するには、次の操作を行います。

1. 目的の結果ポインタを選択します。
2. ポップアップメニューから、**表示**を選択します。

   選択した結果ポインタでリストされる、最初の出力ポインタ項目が表示されます。たとえば、プロシジャでリスト出力と HTML 出力を生成し、リスト出力が最初に生成された場合、リスト出力が表示されます。

結果プロパティの表示

結果ウィンドウのフォルダ、出力ポインタ、または出力ポインタ項目(リスト出力や HTML 出力など)のプロパティを表示できます。

1. **結果**ウィンドウで、目的のフォルダ、出力ポインタ、または出力ポインタ項目を選択します。
2. ポップアップメニューから、**プロパティ**を選択します。

出力テンプレートの操作

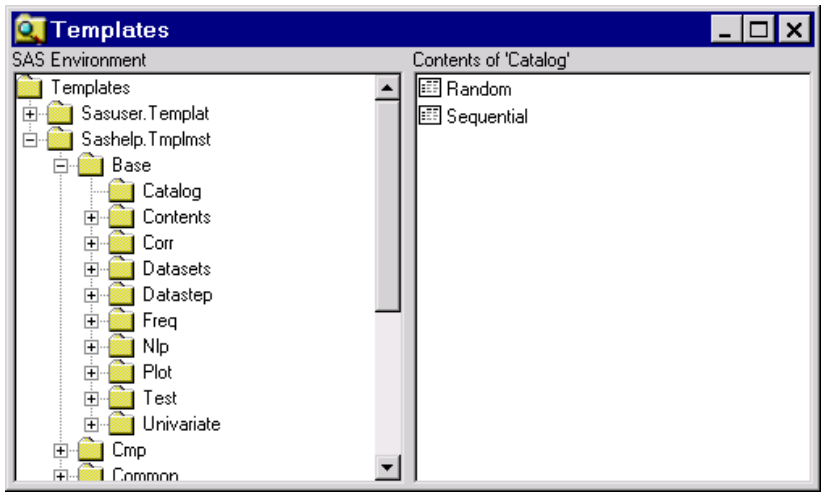
出力テンプレートの操作の概要

テンプレートには、Output Delivery System (ODS)によりプロシジャの結果の期待されたレイアウトを決定するのに使用される詳細情報が含まれています。

テンプレートウィンドウを使用して、現在 SAS で使用可能なすべてのテンプレートを管理できます。特に、**テンプレート**ウィンドウを使用して、次の操作を行えます。

- PROC TEMPLATE ソースコードの参照。
- PROC TEMPLATE ソースコードの編集。
- テンプレートのプロパティの表示。

図 41.12 エクスプローラビューのテンプレートウィンドウ



表示 ⇨ **テンプレート**を**結果**ウィンドウから選択するか、ODSTEMPLATES コマンドを発行して、**テンプレート**ウィンドウを開くことができます。

PROC TEMPLATE を使用して、テンプレートの作成や編集を行うことができます。

注: SAS 提供のテンプレートは、SASHELP に格納されています。PROC TEMPLATE を使用して作成されたテンプレートは、SASUSER、または ODS PATH ステートメントで指定されたライブラリに格納されます。

テンプレートウィンドウのビューのカスタマイズ

テンプレートウィンドウは、次の 3 つのビューのいずれかで表示されます。

- エクスプローラ
- ツリー
- コンテンツのみ

エクスプローラビュー(これがデフォルトです)では、ツリーペインとコンテンツペインの 2 つのペインがある**テンプレート**ウィンドウが表示されます。ツリービューでは、ナビゲーション用のツリーのみがあります。コンテンツのみのビューでは、ツリーはオフにされています。

コンテンツペインのオン/オフを切り替えるには、**テンプレート**ウィンドウから CHILD コマンドを発行します。ツリーペインのオン/オフを切り替えるには、**テンプレート**ウィンドウから TREE コマンドを発行します。

詳細については、“ツリービューの結果ウィンドウでの移動” (763 ページ)、“コンテンツのみのビューの結果ウィンドウでの移動” (763 ページ)、または“エクスプローラビューの結果ウィンドウでの移動” (764 ページ)を参照してください。

エクスプローラビューのテンプレートウィンドウでの移動

エクスプローラビューには、2 つのウィンドウペインがあります。左ペインには、表示可能なテンプレートの階層ビュー(ツリービュー)があります。右ペインには、現在フォーカスされているテンプレートのコンテンツ(コンテンツビュー)が表示されます。

テンプレートを選択し、ポップアップメニューから**エクスプローラ**を選択することで、エクスプローラビューから追加のテンプレートウィンドウを開くことができます。

ツリービューのテンプレートウィンドウでの移動

ツリービューでは、テンプレートは階層で表示されます。テンプレートを操作するには、次を実行します。

1. 表示するテンプレートを含むフォルダを特定します。
2. フォルダの隣にある展開アイコン(+アイコンまたは−アイコン)を使用して、そのコンテンツを開いたり非表示にしたりします。

   または次の操作を行えます。

   - フォルダをダブルクリックして、展開または折りたたみます。
   - フォルダを選択して、ポップアップメニューから**開く**を選択します。
3. 表示するテンプレートをダブルクリックするか、またはテンプレートを選択して、ポップアップメニューから**開く**を選択します。

   ブラウザウィンドウにテンプレートコードが表示されます。

動作環境の情報

z/OS 動作環境を使用している場合、項目の隣の選択フィールドに**?**を入力してポップアップメニューを開くことができます。その他に、項目の隣の選択フィールドに **s** または **x** を入力してダブルクリックすることも可能です。

コンテンツのみのビューのテンプレートウィンドウでの移動

コンテンツのみのビューでは、テンプレートはフォルダとして表示されます。フォルダを開くと、現在のウィンドウのコンテンツが選択したフォルダのコンテンツと置換されます。このビューでテンプレートを操作するには、次を実行します。

1. 表示するテンプレートを含むフォルダを特定します。
2. フォルダを選択して、ポップアップメニューから**開く**を選択します。

   フォルダをダブルクリックして開くこともできます。
3. 表示するテンプレートをダブルクリックするか、またはテンプレートを選択して、ポップアップメニューから**開く**を選択します。

   ブラウザウィンドウにテンプレートコードが表示されます。

動作環境の情報

z/OS 動作環境を使用している場合、項目の隣の選択フィールドに**?**を入力してポップアップメニューを開くことができます。その他に、項目の隣の選択フィールドに **s** または **x** を入力してダブルクリックすることも可能です。

PROC TEMPLATE ソースコードの参照

PROC TEMPLATE ソースコードを参照するには、次の操作を行います。

1. **テンプレート**ウィンドウで、目的のテンプレートを特定します。
2. テンプレートを選択して、ポップアップメニューから**開く**を選択します。

   テンプレートコードがブラウザウィンドウに表示されます。

PROC TEMPLATE ソースコードの編集

PROC TEMPLATE ソースコードを編集するには、次の操作を行います。

1. **テンプレート**ウィンドウで、目的のテンプレートを特定します。
2. テンプレートを選択して、ポップアップメニューから**編集**を選択します。テンプレートコードが編集ウィンドウで表示されます。

3. 必要に応じて、テンプレートコードを変更します。
4. **実行** ⇨ **サブミット**を選択して、変更したテンプレートコードをサブミットします。

注: 構文エラーがある場合は、編集したテンプレートのコードをサブミットするとエラーが**ログ**ウィンドウに表示されます。

注: PROC TEMPLATE の追加情報は、*SAS Output Delivery System: User's Guide* に記載されています。

テンプレートのプロパティの表示

テンプレートのプロパティを表示するには、次の操作を行います。

1. **テンプレート**ウィンドウで、目的のテンプレートを特定します。
2. テンプレートを選択し、ポップアップメニューから**プロパティ**を選択します。

プロパティダイアログボックスにテンプレートの種類、パス、サイズ、説明、修正日時が表示されます。この情報は、**テンプレート**ウィンドウがアクティブなときに、**表示** ⇨ **詳細**を選択して、表示することもできます。

出力の印刷

出力の印刷に使用する方法は、生成する出力の種類と使用している動作環境によって異なります。SAS ウィンドウ環境のウィンドウには、その特定のウィンドウのコンテンツの印刷に使用できる印刷オプションを持つメニューが用意されています。この機能は、動作環境によって異なりますが、すべての動作環境で使用可能です。

HTML 出力を生成する場合、出力を Web ブラウザで開いて、Web ブラウザの印刷コマンドを使用して Web ブラウザから出力を印刷できます。

印刷の詳細については、使用している動作環境に対応する SAS ドキュメントと動作環境のドキュメントを参照してください。

要約

ステートメント

ODS PATH location(s)
PROC TEMPLATE によって作成された定義を検索する場所と検索する順序を指定します。

<libname.>item-store <READ | UPDATE | WRITE>

item-store
スタイルテンプレートまたはテーブル定義、あるいはその両方を含むアイテムストアを特定します。

ウィンドウ

ファイルショートカットの割り当て
ファイルショートカット参照を作成または編集できます。このウィンドウを開くには、DMFILEASSIGN コマンドを発行します。

検索
SAS ライブラリにある表現を検索できます。このウィンドウを開くには、エクスプローラから**ツール** ⇨ **検索**を選択するか、EXPFIND コマンドを発行します。

ログ
実行済みのプログラムに関する情報を確認できます。このウィンドウを開くには、**表示** ⇨ **ログ**を選択するか、LOG コマンドを発行します。

アウトプット
リスト出力を表示できます。このウィンドウを開くには、**表示** ⇨ **アウトプット**を選択するか、OUTPUT コマンドを発行します。

エディタ
SAS プログラムステートメントを入力、編集、サブミット、保存できます。このウィンドウを開くには、**表示** ⇨ **エディタ**を選択するか、PGM コマンドを発行します。

結果
SAS を使用して生成したプロシジャ出力へのポインタを提供します。このウィンドウを開くには、**表示** ⇨ **結果**を選択するか、ODSRESULTS コマンドを発行します。

SAS NOTEPAD
SAS プログラムステートメントを入力、編集、サブミット、保存できます。このウィンドウを開くには、NOTEPAD コマンドまたは NOTES コマンドを発行します。

SAS レジストリエディタ
SAS レジストリを編集し、SAS ウィンドウ環境のさまざまな側面をカスタマイズできます。このウィンドウにアクセスするには、REGEDIT コマンドを発行します。

テンプレート
現在使用可能な出力テンプレートを管理できます。このウィンドウにアクセスするには、**表示** ⇨ **テンプレート**を**結果**ウィンドウ内から選択します。

コマンド

AUTOEXPAND
ツリーノードが選択されたとき、またはプロシジャ出力が生成されたときに、ツリー階層を自動的に展開します。

AUTOSYNC
アウトプットウィンドウの使用可能な最初の出力へワンクリックで自動的に移動できます。

CHILD
コンテンツペインのオン/オフを切り替えます。

CLEAR
すべての SAS 出力ポインタを削除します。

DELETESELS
現在フォーカスされている項目を削除します。

注: 出力ポインタがリスト出力に関連付けられている場合、リスト出力も削除されます。

DESELECT_ALL
コンテンツペインの表示可能時に、選択されている項目をすべて選択解除します。

DETAILS
コンテンツペインの表示可能時に、項目の詳細のオンとオフを切り替えます。

DMOPTLOAD
DMOPTSAVE によって保存されているシステムオプション設定をロードします。

DMOPTSAVE
以降の SAS セッションでロードできるようにシステムオプション設定をすべて保存します。

FIND
指定される文字列との一致を検索します。

LARGEVIEW
コンテンツペインの表示可能時に、(一部の動作環境で)大きいアイコンを表示します。

PMENU
ウィンドウのメニューをオンにします。

PRINT
対象の SAS リスト出力を印刷します。

REFRESH
ウィンドウのコンテンツを更新します。

RENAMESELS
現在フォーカスされている、出力ポインタの名前を変更します。

SELECT_ALL
コンテンツペインの表示可能時に、すべての項目を選択します。

SMALLVIEW
コンテンツペインの表示可能時に、(一部の動作環境で)小さいアイコンを横方向に表示します。

TREE
ツリービュー(階層ビュー)のオン/オフを切り替えます。

UPLEVEL
フォーカスを階層の 1 つ上のレベルに移動します。

プロシジャ

PROC TEMPLATE を使用してテンプレート情報を設定します。

詳細情報

SAS 言語要素の詳細については、
SAS Language Elements by Name, Product, and Category (**support.sas.com**)を参照してください。

印刷と SAS Output Delivery System の詳細については、
SAS Output Delivery System: User's Guide を参照してください。

使用の開始に役立つ例については、
Getting Started with the SAS System を参照してください。

42 章

SAS 環境のカスタマイズ

SAS 環境のカスタマイズについて **771**
目的 771
前提条件 772
動作環境による違い 772

現在のセッションのカスタマイズ **772**
カスタマイズ方法 772
SAS セッションと SAS プログラムの起動時のカスタマイズ 773
SAS システムオプションを使用したカスタマイズ 774

セッション間で引き継がれる設定のカスタマイズ **776**
セッション間で引き継がれる設定のカスタマイズの概要 776
SAS レジストリエディタを使用した SAS セッションと SAS アプリケーションのカスタマイズ 776
プリファレンスウィンドウを使用した SAS セッションのカスタマイズ 780
DMOPTSAVE コマンドと DMOPTLOAD コマンドを使用したシステムオプション設定の保存 780

SAS ウィンドウ環境のカスタマイズ **781**
エクスプローラウィンドウのカスタマイズ 781
エディタのカスタマイズ 784
フォントのカスタマイズ 784
色のカスタマイズ 785
SAS ウィンドウ環境プリファレンスの設定 785

要約 **785**
コマンド 785
プロシジャ 785
ステートメント 786
システムオプション 786
ウィンドウ 786

詳細情報 **787**

SAS 環境のカスタマイズについて

目的

このセクションでは、SAS で次の種類のカスタマイズを行う方法を学習します。

- 現在のセッションの間のみ有効なカスタマイズ
- セッション間で引き継がれるカスタマイズ
- デフォルトの SAS 環境である、SAS ウィンドウ環境へ適用可能なカスタマイズ

前提条件

このセクションを使用するには、SAS ウィンドウ環境を理解している必要があります。SAS ウィンドウ環境の詳細については、“SAS ウィンドウ環境の使用について” (732 ページ)を参照してください。

動作環境による違い

SAS の外観は動作環境ごとに異なりますが、メニューから使用可能な操作の大部分は同じです。

動作環境間の最も大きな違いの 1 つは、メニュー項目の選択方法です。

使用しているワークステーションがマウスを装備していない場合、次にマウス操作に相当するキーボード操作を示します。

表 42.1 マウス操作および相当するキーボード操作

マウス操作	相当するキーボード操作
項目をダブルクリック	項目の隣にあるスペースに、`s` または `x` を入力してから、Enter を押す
項目を右クリック	項目の隣にあるスペースに、`?`を入力してから、Enter を押す

このドキュメントの例では、Microsoft Windows 環境における SAS ウィンドウの外観を示しています。通常、その他の動作環境での対応するウィンドウでは同様の結果を得られます。使用している動作環境でドロップダウンメニューが表示されない場合、コマンドプロンプトにグローバルコマンド PMENU を入力します。

現在のセッションのカスタマイズ

カスタマイズ方法

SAS を理解するにつれて、SAS の設定方法を選好したくなります。ユーザーの希望する作業スタイルに適するよう SAS を設定するために使用可能な、多数のオプションがあります。変更可能な設定には、次のものがあります。

- ウィンドウの色とフォントの属性
- ライブラリとファイルショートカット
- 出力の表示形式
- ファイルの処理機能
- システム変数の使用

現在の SAS セッションは、次の方法でカスタマイズできます。

- SAS セッションまたは SAS プログラムの起動時
- SAS システムオプションの使用
- ドロップダウンメニューオプションの使用

SAS セッションと SAS プログラムの起動時のカスタマイズ

起動に関するオプションの自動的な設定

一部のシステムオプションは SAS 起動時にのみ指定可能です。これらのシステムオプションは、次に影響します。

- SAS とオペレーティングシステムとの対話方法
- 使用しているハードウェア
- セッションまたはプログラムの設定方法

注: いつでも指定可能な他のシステムオプションもあります。詳細については、“SAS システムオプションを使用したカスタマイズ” (774 ページ)を参照してください。

通常、いずれの起動に関するオプションも、サイトでの SAS インストール時にデフォルトで設定されています。ただし、起動に関するオプションは、SAS を起動するたびにコマンド行で指定できます。

SAS の実行ごとに使用するオプションを指定する必要をなくすには、そのオプションを構成ファイルで指定します。SAS を起動するたびに、SAS はこのファイルを検索して、ファイルに含まれるカスタマイズ設定を使用します。独自のものを作成する前に、デフォルトの構成ファイルを確認してください。

注: 構成ファイルと SAS コマンドの両方でオプションを指定した場合、そのオプションは連結されます。構成ファイルにもあるオプションを SAS コマンドで指定すると、SAS コマンドからの設定が構成ファイルの設定に優先します。

構成ファイルに記載されているオプションとシステム起動時のコマンド行のオプションすべての現在の設定を表示するには、SAS コマンドで VERBOSE システムオプションを使用します。

SAS ステートメントを自動的に実行する

SAS の起動時に SAS システムオプションを自動的に設定できるように、特殊な自動実行ファイルを作成して、SAS の起動時にステートメントを自動的に実行することもできます。SAS を起動するたびに、SAS はこの特殊ファイルを検索して、ファイルに含まれるステートメントをすべて実行します。

このファイルを使用して日常的に使用するステートメントを実行することで時間を節約できます。たとえば、次のステートメントを追加できます。

- 定期的に使用するシステムオプションを含む OPTIONS ステートメント
- 定期的に使用するファイルショートカットとライブラリを定義する FILENAME ステートメントと LIBNAME ステートメント

SAS システムオプションを使用したカスタマイズ

OPTIONS ステートメントとオプションウィンドウの使用

SAS システムオプションにより、グローバル SAS 設定が決定されます。たとえば、グローバルオプションは次に影響を与えます。

- SAS 出力の表示形式
- SAS によるファイルの処理方法
- SAS データセットからのオブザベーションの処理方法
- システム変数の使用方法

前のセクションでは、起動時に設定する必要がある一部の起動に関するオプションについて説明しました。しかしながら、いつでも指定可能なシステムオプションも多数あります。これらのシステムオプションは、OPTIONS ステートメントや**オプション**ウィンドウで設定できます。

システムオプション設定は、再び変更されるまで、または現在のセッションが終了するまで有効のままであることに注意してください。

システムオプション設定の表示方法はいくつかあります。最も一般的な 2 つの方法を次に示します。

- **オプション**ウィンドウ(コマンド行で OPTIONS を入力します)
- OPTIONS プロシジャ

OPTIONS プロシジャを使用してシステムオプションの完全なリストを取得するには、次のステートメントをサブミットします。

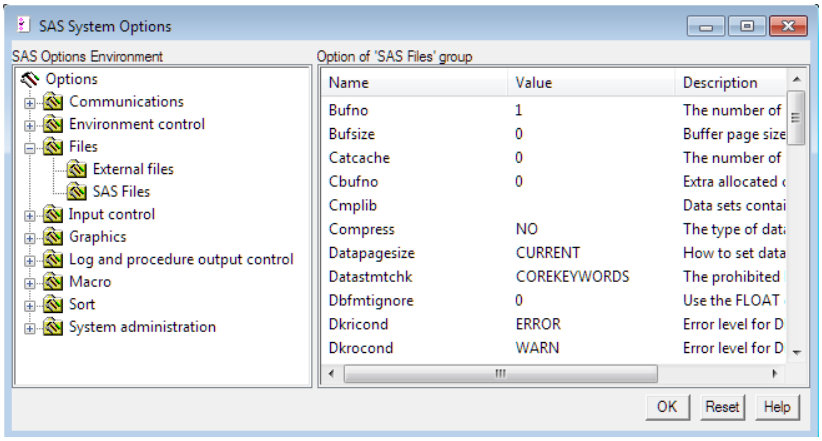
```
proc options;
run;
```

オプションウィンドウでは、オプションは機能によってグループ化されています。ウィンドウの左側には、使用可能なオプショングループが配置されたツリーがあります。オプショングループを展開してサブグループを表示できます。

z/OS 固有

z/OS ユーザーは、マウスを使用して、またはグループ名やサブグループ名の前に **s** または **x** を入力してグループとサブグループを展開できます。サブグループを選択すると、そのサブグループの個別のオプションがウィンドウの右側に表示されます。

図 42.1　SAS オプションウィンドウ



オプションウィンドウを開くには、次のタスクのいずれかを実行します。

- OPTIONS コマンドを発行します。
- **ツール** ⇨ **オプション** ⇨ **システム**を選択します。

各グループまたはサブグループのオプションはアルファベット順で表示され、動作環境に固有のオプション(これもアルファベット順です)が続きます。

SAS オプションウィンドウでのオプションの検索

オプションは、さまざまな方法で検索できます。

- 目的のオプションがウィンドウの右側に表示されるまで、ウィンドウの左側にあるオプションのグループとサブグループを展開します。
- オプションのグループまたはサブグループを選択して、ポップアップメニューから**オプションの検索**を選択します。**オプションの検索**ウィンドウで、検索するオプションの名前を入力し、**OK** を選択します。

SAS オプションウィンドウでのオプションの設定

1. **オプション**ウィンドウで、設定するオプションを検索します。
2. **オプション**ウィンドウの右側からオプションを選択します。
3. ポップアップメニューから**値の変更**または**デフォルトに設定**を選択します。z/OS ユーザーは、オプション名の前に **s** または **x** を入力して、ポップアップメニューにアクセスします。
   - **値の変更**を選択すると、ダイアログボックスが表示され、オプション値を編集できます。
   - **デフォルトに設定**を選択すると、オプション値がデフォルトの SAS システム値にリセットされます。
4. **OK** を選択して、変更を保存します。すべての編集されたオプションを以前の値に戻すには、**リセット**を選択します。

注: ポップアップメニューのすべての項目が淡色表示(使用不可を示す)されている場合、オプションは、起動時に関するオプションで、SAS セッションの起動時にのみ設定できます。

セッション間で引き継がれる設定のカスタマイズ

セッション間で引き継がれる設定のカスタマイズの概要

前のセクションでは、現在の SAS セッションの期間のみ有効であるカスタマイズを行う方法について説明しました。このセクションでは、セッション間で引き継がれるカスタマイズを行う方法について説明します。

次のウィンドウのいずれかを使用して、セッション間で引き継がれるカスタマイズを行うことができます。

- SAS レジストリエディタ
- **プリファレンス**ウィンドウ
- **オプション**ウィンドウ

SAS レジストリエディタを使用した SAS セッションと SAS アプリケーションのカスタマイズ

SAS レジストリについて

SAS レジストリには、特定の SAS セッションと SAS アプリケーションに関する情報が格納されています。システムオプションとは異なり、SAS レジストリに対するカスタマイズは 1 つの SAS セッションが終了しても有効のままです。SAS レジストリのカスタマイズは、PROC REGISTRY または SAS レジストリエディタのいずれかを使用して行うことができます。

このセクションでは、PROC REGISTRY のグラフィカルな代替である、SAS レジストリエディタの使用方法を示します。PROC REGISTRY の詳細については、*Base SAS Procedures Guide* を参照してください。

注意:

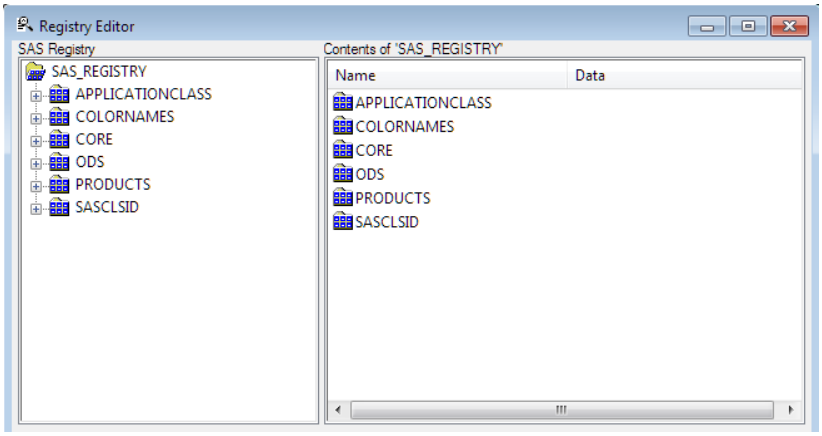
SAS レジストリに対する変更は、注意深く計画して行ってください。 多くの場合、SAS レジストリの編集について責任者が指定されていることが望ましいです。不適切な SAS レジストリの編集は、SAS セッションのパフォーマンスに悪影響を与えることがあります。

データを格納する SAS レジストリエディタの値は、キーとサブキーに存在します。キーとサブキーはフォルダのような外観で、SAS レジストリエディタの左側のツリーに表示されます。キーにサブキーがある場合、ツリーにある+アイコンや-アイコンを使用して展開または折りたたむことができます。キーまたはサブキーに値がある場合、値はウィンドウの右側に表示されます。

動作環境の情報

z/OS 動作環境では、+アイコンまたは-アイコンを、カーソルをその上に置いて Enter キーを押すことで選択できます。

図 42.2 SAS レジストリエディタ



SAS セッションと SAS アプリケーションをカスタマイズするには、SAS レジストリエディタを使用して、キーとキー値の追加、変更、名前の変更、削除を行います。

SAS レジストリエディタを使用して、次のタスクも実行できます。

- レジストリファイルのインポート(任意のキーから開始)
- レジストリのコンテンツをエクスポート(任意のキーから開始)
- レジストリファイルの登録解除

SAS レジストリエディタを開く

SAS レジストリエディタを開くには、**ソリューション** ⇨ **アクセサリ** ⇨ **レジストリエディタ**を選択するか、REGEDIT コマンドを発行します。

SAS レジストリエディタでの情報の検索

SAS レジストリエディタで、特定のキー、キー値の名前、およびキー値データなどの特定の情報を検索できます。

1. 検索を開始するキーを選択します。
2. ドロップダウンメニューを開き、**検索**を選択します。
3. **レジストリエディタの検索**ウィンドウで、検索する文字列フィールドに検索文字列を入力します。
4. 検索を実行する場所に応じて、**キー**、**値の名前**または**値のデータ**のチェックボックスの 1 つ以上をチェックします。
5. **検索**を選択して、検索を開始します。

SAS レジストリエディタでのキーの設定

SAS レジストリエディタでキーを追加、変更、名前の変更、または削除できます。たとえば、出力の印刷時に、SAS で新しい用紙の種類を使用できるようにするとします。そのためには、その用紙の種類を表す新しいキーを作成する必要があります。また、この新しい用紙の種類のキー値を作成して設定する必要もあります。詳細については、“SAS レジストリエディタでの新しいキー値の設定” (778 ページ)を参照してください。

注: キーを追加すると、新しいキーは最後に選択したキーのサブキーになります。

SAS レジストリエディタでキーを設定するには、次の操作を行います。

1. SAS レジストリエディタの左側にあるキーを展開または折りたたんで(+アイコンと−アイコンを使用)、目的のキーを特定します。

2. キーを選択した状態で、ドロップダウンメニューから処理(**新しいキーの作成**、**名前の変更**、**削除**など)を選択します。ダイアログボックスが表示され、追加情報の入力や処理の確認を行うことができます。

注意:
削除処理では、削除するキーの下にあるすべてのサブキーと値(存在する場合)が削除されます。

SAS レジストリエディタでの新しいキー値の設定

新しいキーを作成したら、そのキーに値を追加します。値の追加では、値のデータと値の名前の割り当ても行います。

注: 新しいキーに類似した既存のキーがある場合、そのキーのサブキーとキー値を確認します。確認を行うと、新しいキーのサブキーとキー値の決定に役立ちます。

新しいキーの値を追加するには、次の操作を行います。

1. SAS レジストリエディタの左側にある新しいキーを選択します。
2. ポップアップメニューから処理(**文字列値の作成**、**バイナリ値の作成**、**倍精度実数値の作成**など)を選択します。
3. ダイアログボックスで、新しいキー値の名前と値を入力します。
4. **OK** を選択して、処理を完了します。

SAS レジストリエディタでの既存のキー値の編集

1. SAS レジストリエディタの左側にあるキーを選択します。
2. キーにサブキーがある場合、+アイコンを選択してキーをさらに展開します。
3. SAS レジストリエディタの右側で編集するキー値を選択します。
4. ポップアップメニューから、適切な処理(**変更**、**名前の変更**、**削除**など)を選択します。ダイアログボックスが表示され、追加情報の入力や処理の確認を行うことができます。

レジストリファイルのインポート

レジストリファイルをインポートして、すばやく SAS レジストリに値を入力して修正を行うことができます。レジストリファイルは、テキストエディタで作成されるテキストファイルです。レジストリファイル構文の詳細については、“REGISTRY” (*Base SAS Procedures Guide*)を参照してください。

1. **ファイル** ⇨ **レジストリファイルのインポート**を選択します。
2. インポートするファイルを選択して、**OK** を選択します。

インポート時にエラーが発生した場合、メッセージがステータスバーに表示され、エラーが**ログ**ウィンドウに表示されます。SAS レジストリエディタのオプション**フルステータスをログに出力する**を使用している場合、レジストリのすべての変更はログに送信されます。詳細については、“レジストリエディタのオプションの設定” (779 ページ)を参照してください。

レジストリファイルのエクスポート

SAS レジストリのすべてまたは一部を、ファイルへエクスポート(またはコピー)できます。

1. ファイルのエクスポートを開始するキーを、既存のレジストリで選択します。ルートキーを選択すると、選択したルートキーから始まるツリー全体がエクスポートされます。

2. **ファイル** ⇨ **レジストリファイルのエクスポート**を選択します。
3. 既存のレジストリを保存するファイルへのフルパスを入力するか、または参照してファイルを選択し **OK** を選択します。

エクスポート時にエラーが発生した場合、メッセージがステータスバーに表示され、エラーが**ログ**ウィンドウに表示されます。**フルステータスをログに出力する** SAS レジストリエディタオプションを使用している場合、レジストリのすべての変更はログに送信されます。

インポート済みレジストリファイルのアンインストール

アンインストール機能は、インポートしたレジストリファイルを読み込み、ファイル内に見つかったキーをレジストリから削除します。この処理時にエラーが発生した場合、メッセージがステータスバーに表示され、エラーが**ログ**ウィンドウに表示されます。

注: SAS には ROOT キーのセットが付属しています。ROOT キーは、アンインストール処理時に削除されません。

1. **ファイル** ⇨ **レジストリファイルのアンインストール**を選択します。
2. SAS レジストリからアンインストールする外部レジストリファイルを選択して、**OK** を選択します。アンインストールが完了すると、メッセージ行にメッセージが表示されます。

レジストリエディタのオプションの設定

1. SAS レジストリエディタが開いていない場合は、開きます。
2. **レジストリエディタ**ウィンドウから、**ツール** ⇨ **オプション** ⇨ **レジストリエディタ**を選択します。
3. レジストリビューの選択グループボックスで、レジストリエディタのビューを選択します。
   - 重ね合わせで表示するモードでは、レジストリ内の任意の場所のデータを変更できます。HKEY_USER_ROOT が HKEY_SYSTEM_ROOT に重ねられます。重ね合わせで表示するモードの上位ルートは、SAS REGISTRY として表示されます。
   - すべて表示するモードでは、レジストリへの 2 つの主なエントリポイント HKEY_SYSTEM_ROOT と HKEY_USER_ROOT に含まれるすべてのエントリが、レジストリエディタに表示されます。通常、HKEY_SYSTEM_ROOT ツリーは SASHELP ライブラリに格納されており、HKEY_USER_ROOT は SASUSER ライブラリに格納されています。
4. 次より適切なチェックボックスを選択または選択解除します。

   HKEY_SYSTEM_ROOT を書き込みアクセスでオープンする
   ユーザーが SASHELP への書き込みアクセス権を持っている場合、レジストリを書き込みアクセスで開きます。

   フルステータスをログに出力する
   レジストリファイルのインポート時またはアンインストール時に行われた変更をすべてログに書き込みます。通常、**ログ**ウィンドウにはエラーのみ表示されます。

   16 進フォーマットでの符号なし整数値を表示する
   符号なし整数値を HEX 形式または DECIMAL 形式で値リストに表示します。

レジストリエディタのオプションウィンドウの設定をすべてデフォルト値に戻すには、**全オプションをリセットする**を選択します。

プリファレンスウィンドウを使用した SAS セッションのカスタマイズ

プリファレンスウィンドウには、アクセスして SAS のプリファレンスを設定できる一連のタブがあります。プリファレンスを使用して SAS 環境をカスタマイズおよび制御できます。たとえば、**全般**タブを使用して起動ロゴを選択したり、**結果**タブを使用して出力プリファレンスを制御したり、**編集**タブを使用して、カーソルでテキストを挿入するか上書きするかなどのエディタプリファレンスを設定したりできます。

プリファレンスウィンドウの設定は、SAS セッション間で有効のまま引き継がれます。

プリファレンスウィンドウにアクセスするには、**ツール** ⇨ **オプション** ⇨ **プリファレンス**を選択するか、または DLGPREF コマンドを発行します。

動作環境の情報

プリファレンスウィンドウは、一部の動作環境では使用できません。また、一部のプリファレンス設定は動作環境に固有です。プリファレンス設定の詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

DMOPTSAVE コマンドと DMOPTLOAD コマンドを使用したシステムオプション設定の保存

SAS セッション間で引き継がれるシステムオプション設定を保存するには、グローバルコマンド DMOPTSAVE と DMOPTLOAD を使用するのが最も簡単な方法です。作業スタイルに最適な方法でシステムオプションを設定した後、コマンド行に **DMOPTSAVE** と入力して、Enter を押します。これで、現在のシステムオプション設定が以降も使用できるよう保存されます。その後、別の SAS セッションを起動したときに保存済みの設定を取得するには、コマンド行に **DMOPTLOAD** と入力して Enter を押します。これにより、システムオプション設定が変更されて、DMOPTSAVE コマンド発行時に有効だったシステムオプションに戻ります。

DMOPTSAVE コマンドと DMOPTLOAD コマンドには、次に示す他の便利な機能もあります。

- パラメータを発行して、システムオプション設定の別のセットに名前を付けて保存先を制御します。
- システムオプション設定はデフォルトでデータセットとして保存されるため、保存済みのシステムオプション設定を SAS エクスプローラを使用して表示できます。
- パラメータを発行して、システムオプション設定をレジストリキーに保存することもできます。

DMOPTSAVE コマンドをパラメータなしで発行すると、SAS により、システムオプション設定を含むデータセット(myopts)がデフォルトのライブラリに保存されます。デフォルトのライブラリは、通常、現在のユーザープロファイルがあるライブラリです。ほとんどの場合、これは SASUSER ライブラリです。

これらのコマンドの使用に関する詳細は、SAS オンラインヘルプを参照してください。

SAS ウィンドウ環境のカスタマイズ

エクスプローラウィンドウのカスタマイズ

エクスプローラウィンドウのカスタマイズ方法

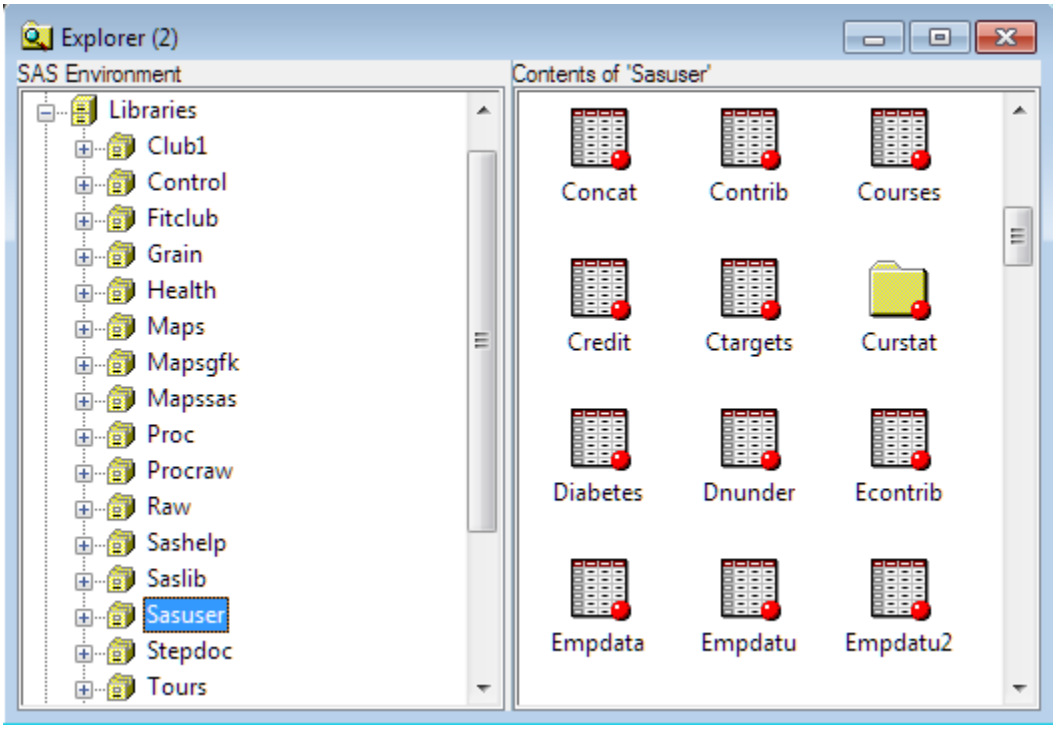
エクスプローラウィンドウを、次に示す方法でカスタマイズできます。

- コンテンツのみのビューまたはエクスプローラビューを選択します。
- コンテンツビューでの項目の表示形式を変更します。
- フォルダ(動作環境ファイルへのアクセスを追加するフォルダを含む)を追加および削除します。
- メンバ、エントリ、および動作環境ファイルのタイプを表示します。
- ポップアップメニューの項目を追加します。
- メンバ、エントリ、および動作環境ファイルのタイプを非表示にします。

コンテンツのみのビューまたはエクスプローラビューの選択

エクスプローラウィンドウは、エクスプローラビューまたはコンテンツのみのビューのいずれかで表示できます。エクスプローラビューでは、**エクスプローラ**ウィンドウに左右の 2 つの部分、左側にフォルダをリスト表示するツリービュー、右側にツリービューで選択されたフォルダのコンテンツを表示するコンテンツビューがあります。

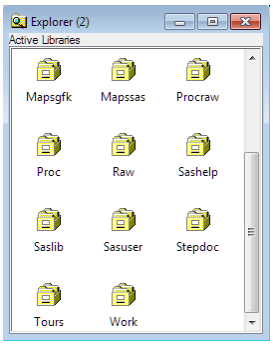
図 42.3 エクスプローラビューが有効にされたエクスプローラウィンドウ



コンテンツのみのビューでは、**エクスプローラ**ウィンドウは、お使いの SAS 環境のコンテンツを表示する単一ペインのウィンドウです。同じウィンドウ内の前のコンテンツが、

フォルダを開くと、そのフォルダのコンテンツによって置き換えられます。コンテンツのみのビューでは、プルダウンメニューアクションとポップアップメニューアクション、およびツールバー項目(ツールバーが使用可能な場合)を使用して、**エクスプローラ**ウィンドウをナビゲートします。

図 42.4 *コンテンツのみのビューが有効にされたエクスプローラウィンドウ*



動作環境の情報
ほとんどの動作環境では、エクスプローラは、デフォルトでコンテンツのみのビューで表示されます。

動作環境に応じて、次に示す方法で 2 つのビューを切り替えできます。

メニュー:
表示 ⇨ **ツリーを表示**

コマンド:
TREE

ツールバー
ツリーツールボタンを切り替えます

コンテンツビューでの項目の表示形式の変更

表示メニューから項目を選択して、**エクスプローラ**ウィンドウのコンテンツビューでのファイルの表示形式を決定できます。次に選択可能な項目を示しますが、動作環境によっては使用できない項目もあります。

大きいアイコン
各ファイルを大きなアイコンで表示します。

小さいアイコン
各ファイルを小さなアイコンで表示します(PC ホストでのみ使用可能)。

一覧
ファイルを左揃えのリストで表示します。

詳細
ファイルを詳細情報の列(ファイルサイズやタイプなど)とともにリスト表示します。

表示メニューから選択するかわりに、動作環境で次に示すコマンドを使用できます。

DETAILS
ファイルを詳細情報の列(ファイルサイズやタイプなど)とともにリスト表示します。

LARGEVIEW
各ファイルを大きなアイコンで表示します。

SMALLVIEW
動作環境に応じて、このコマンドは、ファイルのリスト、または各ファイルを小さなアイコンで表示します。

フォルダの追加と削除

エクスプローラウィンドウにライブラリフォルダとファイルショートカットフォルダがデフォルトで表示されます。これらのフォルダをオフにしたり、マイ コンピュータ、お気に入りフォルダ、結果などのその他のフォルダをオンにしたりできます。

1. **エクスプローラ**ウィンドウから、**ツール** ⇨ **オプション** ⇨ **エクスプローラ**を選択します。
2. ウィンドウ上部のドロップダウンリストから、**初期化**を選択します。
3. 追加または削除するフォルダを選択して、**追加**または**削除**を選択します。変更に応じて、説明フィールドがオンまたはオフに変更されます。

動作環境の情報

お気に入りフォルダウィンドウでは、エクスプローラから、動作環境固有のファイルへアクセスできます。この機能は、z/OS 動作環境では使用できません。

メンバ、エントリ、および動作環境ファイルのタイプの表示の有効化

よく使用されるメンバ、カタログエントリ、および動作環境ファイルは、**エクスプローラ**ウィンドウに登録されて表示されます。登録されているタイプには、少なくとも 1 つのアイコンが定義され、ポップアップメニューアクションが定義されている必要があります。未定義のタイプは、**エクスプローラ**ウィンドウに表示されず、関連付けられているアクションもありません。

未定義のタイプを追加(登録)するには、次の操作を行います。

1. **エクスプローラ**ウィンドウから、**ツール** ⇨ **オプション** ⇨ **エクスプローラ**を選択します。
2. ウィンドウ上部のドロップダウンリストから、カテゴリ(メンバ、カタログエントリ、ホストファイルなど)を選択します。登録されているタイプがウィンドウに表示されます。
3. カテゴリの未定義のタイプを表示するには、**未定義タイプを表示する**チェックボックスを選択します。
4. タイプを選択して、**編集**を選択します。
5. **アイコンの選択**を選択します。
6. **アイコンの選択**ダイアログボックスで、上部のドロップダウンリストからカテゴリを選択して、アイコンを選択し、**OK** を選択して、ダイアログボックスを閉じます。
7. そのタイプのアクションを追加して(必要な場合)、**OK** を選択します。タイプへのアクションの追加の詳細については、“メンバ、エントリ、または動作環境ファイルのタイプへのメニューアクションの追加” (783 ページ)を参照してください。タイプが登録されているタイプリストに追加されます。

メンバ、エントリ、または動作環境ファイルのタイプへのメニューアクションの追加

任意のカタログエントリ、メンバ、または動作環境ファイルのタイプへメニューアクションを追加できます。

1. **エクスプローラ**ウィンドウから、**ツール** ⇨ **オプション** ⇨ **エクスプローラ**を選択します。
2. ウィンドウ上部のドロップダウンリストから、カテゴリ(メンバ、カタログエントリ、ホストファイルなど)を選択します。登録されているタイプがウィンドウに表示されます。
3. 編集する登録されているタイプを選択します。
4. **編集**を選択します。
5. そのエントリの**オプション**ダイアログボックスで、**追加**を選択します。

6. アクションの名前(これがその項目のポップアップメニューに表示されるアクションになります)とアクションコマンドを入力します。アクションコマンドの例を確認するには、登録されているタイプのコマンドを参照します。
7. **OK** を選択します。

注: アクションセクションのアンパサンド(&)の直後の文字は、そのアクションの実行に使用されるショートカットキーを示します。

メンバ、エントリ、およびホストファイルのタイプの非表示

メンバ、カタログエントリ、およびホストファイルを非表示にして、**エクスプローラ**ウィンドウに表示されないようにできます。

1. **エクスプローラ**ウィンドウから、**ツール** ⇨ **オプション** ⇨ **エクスプローラ**を選択します。
2. ウィンドウ上部のドロップダウンリストから、カテゴリ(メンバ、カタログエントリ、ホストファイルなど)を選択します。登録されているタイプがウィンドウに表示されます。
3. ビューから削除する、登録されているタイプを選択します。
4. **削除**を選択します。プロンプトが表示されたら **OK** を選択して、削除を確定します。

登録されているタイプは、削除されると、未定義タイプの表示ビューに移動します。登録されているタイプに追加し直すには、アイコンを再定義する必要があります。

エディタのカスタマイズ

エディタの全般オプションとテキスト編集オプションをカスタマイズできます。たとえば、プログラムを編集するときに行コマンドを使用する場合、プログラムエディタに行番号を常に表示するようにできます。

エディタをカスタマイズするには、次の操作を行います。

1. SAS プログラミングウィンドウ(**プログラムエディタ**、**ログ**、**アウトプット**、**NOTEPAD** などのウィンドウ)を選択します。
2. **ツール** ⇨ **オプション** ⇨ **エディタ**を選択します。
3. ドロップダウンリストから、編集するオプションのカテゴリを選択します。
4. オプショングループボックスから、オプションを選択して、ポップアップメニューから**変更**を選択します。
5. 表示されるダイアログボックスで、オプションの名前、値、あるいは両方を編集します。

フォントのカスタマイズ

フォントウィンドウでは、SAS ウィンドウ環境のデフォルトのフォント情報を設定します。**フォント**ウィンドウにアクセスするには、DLGFONT コマンドを発行するか、**ツール** ⇨ **オプション** ⇨ **フォント**を選択します。

フォントウィンドウは、ホスト固有です。詳細については、使用しているホストのドキュメントを参照してください。

色のカスタマイズ

注: **SASColor** ウィンドウで行われた変更は、影響を受ける SAS ウィンドウが閉じて再度開かれると反映されます。

NOTEPAD やプログラムエディタなどの、編集ウィンドウのデフォルトの色も、SYNCONFIG コマンドを使用して変更できます。このコマンドは、SAS 言語とプログラムの要素の色も制御し、SAS プログラムの解析とその作用の理解を容易にします。SYNCONFIG を使用すると、**配色の編集**ウィンドウが開き、いくつかの配色から選択できます。既定の配色を変更することもできます。

SAS ウィンドウ環境プリファレンスの設定

プリファレンスウィンドウを使用して、使いやすいように SAS ウィンドウ環境のそれぞれの部分をカスタマイズできます。詳細については、“プリファレンスウィンドウを使用した SAS セッションのカスタマイズ” (780 ページ)を参照してください。

要約

コマンド

DLGFONT
SAS ウィンドウ環境のフォントの制御に使用される、**フォント**ウィンドウを開きます。

DLGPREF
一部の動作環境で**プリファレンス**ウィンドウを開きます。

OPTIONS
SAS システムオプションウィンドウを開きます。

PMENU
ウィンドウ環境でメニューバーをオンにします。

REGEDIT
レジストリエディタウィンドウを開きます。

SASCOLOR
背景や境界などのウィンドウ要素の色を変更する、**SASColor** ウィンドウを開きます。

SYNCONFIG
エディタ、NOTEPAD、プログラムエディタの各ウィンドウの配色の編集に使用される、**配色の編集**ウィンドウを開きます。

プロシジャ

PROC OPTIONS <SHORT | LONG>;
すべての SAS システムオプションの現在の値をリストで表示します。SHORT オプションと LONG オプションにより、SAS システムオプションのリスト表示の形式が決定されます。

注: **SAS システムオプション**ウィンドウを使用しても、すべての SAS システムオプションの現在の値を表示できます。

PROC REGISTRY <options>;
SAS レジストリを管理します。

注: SAS レジストリエディタを使用して SAS レジストリを管理することもできます。

ステートメント

OPTIONS *option-1* <*...option-n*>;
1 つ以上の SAS システムオプションの値を変更します。

システムオプション

VERBOSE | NOVERBOSE
構成ファイルで指定されたすべてのシステムオプションの設定を、ワークステーションまたはバッチログのいずれかに表示するかを制御します。

ウィンドウ

エディタオプションウィンドウ
プログラムエディタなどの SAS ウィンドウ環境の固有のウィンドウのオプションを設定できます。**エディタオプション**ウィンドウを開くには、変更するウィンドウへ移動して**ツール** ⇨ **オプション** ⇨ **エディタ**を選択するか、EDOPT コマンドを発行します。

エクスプローラオプションウィンドウ
エクスプローラウィンドウのオプションを設定できます。このウィンドウを開くには、**ツール** ⇨ **オプション** ⇨ **エクスプローラオプション**を選択するか、EXPOPTS コマンドを発行します。

フォントウィンドウ
SAS ウィンドウ環境で使用するデフォルトのフォントを選択できます。このウィンドウにアクセスするには、DLGFONT コマンドを発行します。

注: このウィンドウは、使用している動作環境に固有です。

プリファレンスウィンドウ
SAS System プリファレンスを設定できます。このウィンドウにアクセスするには、DLGPREF コマンドを発行します。

注: このウィンドウは、使用している動作環境に固有です。

SASColor ウィンドウ
SAS ウィンドウの異なるウィンドウ要素のデフォルト色を変更できます。このウィンドウにアクセスするには、SASCOLOR コマンドを発行します。

SAS レジストリエディタ
SAS レジストリを編集し、SAS ウィンドウ環境のさまざまな側面をカスタマイズできます。このウィンドウにアクセスするには、REGEDIT コマンドを発行します。

SAS システムオプションウィンドウ
現在の SAS システムオプションの表示や変更を行うことができます。このウィンドウにアクセスするには、OPTIONS コマンドを発行します。

詳細情報

カスタマイズ
: 動作環境固有のオプションとプリファレンスのカスタマイズの詳細については、使用している動作環境に対応する SAS ドキュメントを参照してください。

SAS プロシジャ
: SAS プロシジャの詳細については、*Base SAS Procedures Guide* を参照してください。

ステートメントとオプション
: このセクションで説明されたステートメントとオプションの詳細については、*SAS Statements: Reference* および *SAS System Options: Reference* を参照してください。

11 部

付録

付録 1
選択例の完全な DATA ステップ *791*

付録 2
DATA ステップデバッガコマンド *801*

付録 1

選択例の完全な DATA ステップ

選択例の完全な DATA ステップ **791**

CITY データセット **792**
CITY データセットを作成するための DATA ステップ 792

UNIVERSITY_TEST_SCORES データセット **793**
UNIVERSITY_TEST_SCORES データセットを作成するための DATA ステップ 793

YEAR_SALES データセット **794**
YEAR_SALES データセットを作成するための DATA ステップ 794

HIGHLOW データセット **795**
HIGHLOW データセットを作成するための DATA ステップ 795

GRADES データセット **796**
GRADES データセットを作成するための DATA ステップ 796

USCLIM データセット **797**
USCLIM.HIGHTEMP データセットを作成するための DATA ステップ 797
USCLIM.HURRICANE データセットを作成するための DATA ステップ 797
USCLIM.LOWTEMP データセットを作成するための DATA ステップ 797
USCLIM.TEMPCHNG データセットを作成するための DATA ステップ 798
USCLIM.BASETEMP および USCLIM.REPORT カタログに関する注意事項 798

CLIMATE、PRECIP および STORM データセット **798**
CLIMATE.HIGHTEMP データセットを作成するための DATA ステップ 798
CLIMATE.LOWTEMP データセットを作成するための DATA ステップ 798
PRECIP.RAIN データセットを作成するための DATA ステップ 799
PRECIP.SNOW データセットを作成するための DATA ステップ 799
STORM.TORNADO データセットを作成するための DATA ステップ 799

選択例の完全な DATA ステップ

このドキュメントでは、各セクションで使用されるデータセットの作成方法が示されています。しかしながら、入力データが長いものであるか、データセットの実際の内容がセクションで重要でない場合、データセットの作成に使用される DATA ステップや生データはセクション内ではなく、この付録に記載されています。

セクション内で詳細が与えられない DATA ステップのみが、ここに掲載されています。

CITY データセット

CITY データセットを作成するための DATA ステップ

```
data city;
   input Year 4. @7 ServicesPolice comma6.
         @15 ServicesFire comma6. @22 ServicesWater_Sewer comma6.
         @30 AdminLabor comma6. @39 AdminSupplies comma6.
         @45 AdminUtilities comma6.;
   ServicesTotal=ServicesPolice+ServicesFire+ServicesWater_Sewer;
   AdminTotal=AdminLabor+AdminSupplies+AdminUtilities;
   Total=ServicesTotal+AdminTotal;
   label               Total='Total Outlays'
              ServicesTotal='Services: Total'
             ServicesPolice='Services: Police'
               ServicesFire='Services: Fire'
         ServicesWater_Sewer='Services: Water & Sewer'
                  AdminTotal='Administration: Total'
                  AdminLabor='Administration: Labor'
               AdminSupplies='Administration: Supplies'
              AdminUtilities='Administration: Utilities' ;
   datalines;
1980  2,819   1,120    422   391      63     98
1981  2,477   1,160    500   172      47     70
1982  2,028   1,061    510   269      29     79
1983  2,754     893    540   227      21     67
1984  2,195     963    541   214      21     59
1985  1,877     926    535   198      16     80
1986  1,727   1,111    535   213      27     70
1987  1,532   1,220    519   195      11     69
1988  1,448   1,156    577   225      12     58
1989  1,500   1,076    606   235      19     62
1990  1,934     969    646   266      11     63
1991  2,195   1,002    643   256      24     55
1992  2,204     964    692   256      28     70
1993  2,175   1,144    735   241      19     83
1994  2,556   1,341    813   238      25     97
1995  2,026   1,380    868   226      24     97
1996  2,526   1,454    946   317      13     89
1997  2,027   1,486  1,043   226       .     82
1998  2,037   1,667  1,152   244      20     88
1999  2,852   1,834  1,318   270      23     74
2000  2,787   1,701  1,317   307      26     66
;
```

UNIVERSITY_TEST_SCORES データセット

UNIVERSITY_TEST_SCORES データセットを作成するための DATA ステップ

DATA ステップの生データを使用して、次のデータセットが作成されます。

- OUT.UNIVERSITY_TEST_SCORES
- OUT.UNIVERSITY_TEST_SCORES2
- OUT.UNIVERSITY_TEST_SCORES3
- OUT.UNIVERSITY_TEST_SCORES4
- OUT.UNIVERSITY_TEST_SCORES5
- OUT_ERROR1
- OUT.ERROR2
- OUT.ERROR3

```
libname out 'your-data-library';

data out.university_test_scores;
   input Test $ Gender $ Year TestScore;
   datalines;
Verbal m 2005 504
Verbal f 2005 496
Verbal m 2006 504
Verbal f 2006 497
Verbal m 2007 501
Verbal f 2007 497
Verbal m 2008 505
Verbal f 2008 502
Verbal m 2009 507
Verbal f 2009 503
Verbal m 2010 507
Verbal f 2010 503
Verbal m 2011 509
Verbal f 2011 502
Math   m 2005 521
Math   f 2005 484
Math   m 2006 524
Math   f 2006 484
Math   m 2007 523
Math   f 2007 487
Math   m 2008 525
Math   f 2008 490
Math   m 2009 527
Math   f 2009 492
Math   m 2010 530
Math   f 2010 494
Math   m 2011 531
Math   f 2011 496
```

```
;
```

YEAR_SALES データセット

YEAR_SALES データセットを作成するための DATA ステップ

```
data year_sales;
   input Month $ Quarter $ SalesRep $14. Type $ Units Price @@;
   AmountSold=Units*price;
   datalines;
01 1 Hollingsworth Deluxe    260 49.50 01 1 Garcia        Standard   41 30.97
01 1 Hollingsworth Standard  330 30.97 01 1 Jensen        Standard  110 30.97
01 1 Garcia        Deluxe    715 49.50 01 1 Jensen        Standard  675 30.97
02 1 Garcia        Standard 2045 30.97 02 1 Garcia        Deluxe     10 49.50
02 1 Garcia        Standard   40 30.97 02 1 Hollingsworth Standard 1030 30.97
02 1 Jensen        Standard  153 30.97 02 1 Garcia        Standard   98 30.97
03 1 Hollingsworth Standard  125 30.97 03 1 Jensen        Standard  154 30.97
03 1 Garcia        Standard  118 30.97 03 1 Hollingsworth Standard   25 30.97
03 1 Jensen        Standard  525 30.97 03 1 Garcia        Standard  310 30.97
04 2 Garcia        Standard  150 30.97 04 2 Hollingsworth Standard  260 30.97
04 2 Hollingsworth Standard  530 30.97 04 2 Jensen        Standard 1110 30.97
04 2 Garcia        Standard 1715 30.97 04 2 Jensen        Standard  675 30.97
05 2 Jensen        Standard   45 30.97 05 2 Hollingsworth Standard 1120 30.97
05 2 Garcia        Standard   40 30.97 05 2 Hollingsworth Standard 1030 30.97
05 2 Jensen        Standard  153 30.97 05 2 Garcia        Standard   98 30.97
06 2 Jensen        Standard  154 30.97 06 2 Hollingsworth Deluxe     25 49.50
06 2 Jensen        Standard  276 30.97 06 2 Hollingsworth Standard  125 30.97
06 2 Garcia        Standard  512 30.97 06 2 Garcia        Standard 1000 30.97
07 3 Garcia        Standard  250 30.97 07 3 Hollingsworth Deluxe     60 49.50
07 3 Garcia        Standard   90 30.97 07 3 Hollingsworth Deluxe     30 49.50
07 3 Jensen        Standard  110 30.97 07 3 Garcia        Standard   90 30.97
07 3 Hollingsworth Standard  130 30.97 07 3 Jensen        Standard  110 30.97
07 3 Garcia        Standard  265 30.97 07 3 Jensen        Standard  275 30.97
07 3 Garcia        Standard 1250 30.97 07 3 Hollingsworth Deluxe     60 49.50
07 3 Garcia        Standard   90 30.97 07 3 Jensen        Standard  110 30.97
07 3 Garcia        Standard   90 30.97 07 3 Hollingsworth Standard  330 30.97
07 3 Jensen        Standard  110 30.97 07 3 Garcia        Standard  465 30.97
07 3 Jensen        Standard  675 30.97 08 3 Jensen        Standard  145 30.97
08 3 Garcia        Deluxe    110 49.50 08 3 Hollingsworth Standard  120 30.97
08 3 Hollingsworth Standard  230 30.97 08 3 Jensen        Standard  453 30.97
08 3 Garcia        Standard  240 30.97 08 3 Hollingsworth Standard  230 49.50
08 3 Jensen        Standard  453 30.97 08 3 Garcia        Standard  198 30.97
08 3 Hollingsworth Standard  290 30.97 08 3 Garcia        Standard 1198 30.97
08 3 Jensen        Deluxe     45 49.50 08 3 Jensen        Standard  145 30.97
08 3 Garcia        Deluxe    110 49.50 08 3 Hollingsworth Standard  330 30.97
08 3 Garcia        Standard  240 30.97 08 3 Hollingsworth Deluxe     50 49.50
08 3 Jensen        Standard  453 30.97 08 3 Garcia        Standard  198 30.97
08 3 Jensen        Deluxe    225 49.50 09 3 Hollingsworth Standard  125 30.97
09 3 Jensen        Standard  254 30.97 09 3 Garcia        Standard  118 30.97
09 3 Hollingsworth Standard 1000 30.97 09 3 Jensen        Standard  284 30.97
09 3 Garcia        Standard  412 30.97 09 3 Jensen        Deluxe    275 49.50
09 3 Garcia        Standard  100 30.97 09 3 Jensen        Standard  876 30.97
09 3 Hollingsworth Standard  125 30.97 09 3 Jensen        Standard  254 30.97
```

```
09 3 Garcia        Standard 1118 30.97 09 3 Hollingsworth Standard  175 30.97
09 3 Jensen        Standard  284 30.97 09 3 Garcia        Standard  412 30.97
09 3 Jensen        Deluxe    275 49.50 09 3 Garcia        Standard  100 30.97
09 3 Jensen        Standard  876 30.97 10 4 Garcia        Standard  250 30.97
10 4 Hollingsworth Standard  530 30.97 10 4 Jensen        Standard  975 30.97
10 4 Hollingsworth Standard  265 30.97 10 4 Jensen        Standard   55 30.97
10 4 Garcia        Standard  365 30.97 11 4 Hollingsworth Standard 1230 30.97
11 4 Jensen        Standard  453 30.97 11 4 Garcia        Standard  198 30.97
11 4 Jensen        Standard   70 30.97 11 4 Garcia        Standard  120 30.97
11 4 Hollingsworth Deluxe    150 49.50 12 4 Garcia        Standard 1000 30.97
12 4 Jensen        Standard  876 30.97 12 4 Hollingsworth Deluxe    125 49.50
12 4 Jensen        Standard 1254 30.97 12 4 Hollingsworth Standard  175 30.97
;
```

HIGHLOW データセット

HIGHLOW データセットを作成するための DATA ステップ

```
data highlow;
   input Year @7 DateOfHigh:date9. DowJonesHigh @26 DateOfLow:date9. DowJonesLow;
   format LogDowHigh LogDowLow 5.2 DateOfHigh DateOfLow date9.;
   LogDowHigh=log(DowJonesHigh);
   LogDowLow=log(DowJonesLow);
datalines;
1968  03DEC1968  985.21  21MAR1968  825.13
1969  14MAY1969  968.85  17DEC1969  769.93
1970  29DEC1970  842.00  06MAY1970  631.16
1971  28APR1971  950.82  23NOV1971  797.97
1972  11DEC1972 1036.27  26JAN1972  889.15
1973  11JAN1973 1051.70  05DEC1973  788.31
1974  13MAR1974  891.66  06DEC1974  577.60
1975  15JUL1975  881.81  02JAN1975  632.04
1976  21SEP1976 1014.79  02JAN1976  858.71
1977  03JAN1977  999.75  02NOV1977  800.85
1978  08SEP1978  907.74  28FEB1978  742.12
1979  05OCT1979  897.61  07NOV1979  796.67
1980  20NOV1980 1000.17  21APR1980  759.13
1981  27APR1981 1024.05  25SEP1981  824.01
1982  27DEC1982 1070.55  12AUG1982  776.92
1983  29NOV1983 1287.20  03JAN1983 1027.04
1984  06JAN1984 1286.64  24JUL1984 1086.57
1985  16DEC1985 1553.10  04JAN1985 1184.96
1986  02DEC1986 1955.57  22JAN1986 1502.29
1987  25AUG1987 2722.42  19OCT1987 1738.74
1988  21OCT1988 2183.50  20JAN1988 1879.14
1989  09OCT1989 2791.41  03JAN1989 2144.64
1990  16JUL1990 2999.75  11OCT1990 2365.10
1991  31DEC1991 3168.83  09JAN1991 2470.30
1992  01JUN1992 3413.21  09OCT1992 3136.58
1993  29DEC1993 3794.33  20JAN1993 3241.95
1994  31JAN1994 3978.36  04APR1994 3593.35
1995  13DEC1995 5216.47  30JAN1995 3832.08
```

```
1996  27DEC1996 6560.91  10JAN1996 5032.94
1997  06AUG1997 8259.31  11APR1997 6391.69
1998  23NOV1998 9374.27  31AUG1998 7539.07
1999  31DEC1999 11497.12 22JAN1999 9120.67
2000  14JAN2000 11722.98 07MAR2000 9796.04
2001  21MAY2001 11337.92 21SEP2001 8235.81
2002  19MAR2002 10635.25 09OCT2002 7286.27
2003  31DEC2003 10453.92 11MAR2003 7524.06
2004  28DEC2004 10854.54 25OCT2004 9749.99
2005  04MAR2005 10940.55 20APR2005 10012.36
2006  27DEC2006 12510.57 20JAN2006 10667.39
2007  09OCT2007 14164.53 05MAR2007 12050.41
2008  02MAY2008 13058.20 10OCT2008 8451.19
;
run;
```

GRADES データセット

GRADES データセットを作成するための DATA ステップ

```
data grades;
   input Name &$14. Gender :$2. Section :$3. ExamGrade1 @@;
   datalines;
Abdallah       F Mon  46 Anderson       M Wed  75
Aziz           F Wed  67 Bayer          M Wed  77
Bhatt          M Fri  79 Blair          F Fri  70
Bledsoe        F Mon  63 Boone          M Wed  58
Burke          F Mon  63 Chung          M Wed  85
Cohen          F Fri  89 Drew           F Mon  49
Dubos L        M Mon  41 Elliott        F Wed  85
Farmer         F Wed  58 Franklin       F Wed  59
Freeman        F Mon  79 Friedman       M Mon  58
Gabriel        M Fri  75 Garcia         M Mon  79
Harding        M Mon  49 Hazelton       M Mon  55
Hinton         M Fri  85 Hung           F Fri  98
Jacob          F Wed  64 Janeway        F Wed  51
Jones          F Mon  39 Jorgensen      M Mon  63
Judson         F Fri  89 Kuhn           F Mon  89
LeBlanc        F Fri  70 Lee            M Fri  48
Litowski       M Fri  85 Malloy         M Wed  79
Meyer          F Fri  85 Nichols        M Mon  58
Oliver         F Mon  41 Park           F Mon  77
Patel          M Wed  73 Randleman      F Wed  46
Robinson       M Fri  64 Shien          M Wed  55
Simonson       M Wed  62 Smith N        M Wed  71
Smith R        M Mon  79 Sullivan       M Fri  77
Swift          M Wed  63 Wolfson        F Fri  79
Wong           F Fri  89 Zabriski       M Fri  89
;
```

USCLIM データセット

USCLIM.HIGHTEMP データセットを作成するための DATA ステップ

```
libname usclim 'SAS-data-library';

data usclim.hightemp;
   input State $char14. City $char14. Temp_f Date date9. Elevation;
   datalines;
Arizona       Parker         127 07jul1905 345
Kansas        Alton          121 25jul1936 1651
Nevada        Overton        122 23jun1954 1240
North Dakota  Steele         121 06jul1936 1857
Oklahoma      Tishomingo     120 26jul1943 6709
Texas         Seymour        120 12aug1936 1291
;
```

USCLIM.HURRICANE データセットを作成するための DATA ステップ

```
libname usclim 'SAS-data-library';

data usclim.hurricane;
   input @1 State $char14. @16 Date date7. Deaths Millions Name $;
   format Date worddate18. Millions dollar6.;
   informat State $char14. Date date9.;
   label Millions='Damage';
   datalines;
Mississippi    14aug1969 256 1420 Camille
Florida        14jun1972 117 2100 Agnes
Alabama        29aug1979 5   2300 Frederick
Texas          15aug1983 21  2000 Alicia
Texas          03aug1980 28  300  Allen
North Carolina 27aug2011 6   450  Irene
;
```

USCLIM.LOWTEMP データセットを作成するための DATA ステップ

```
libname usclim 'SAS-data-library';

data usclim.lowtemp;
   input State $char14. City $char14. Temp_f Date date9. Elevation;
   datalines;
Alaska        Prospect Creek -80 23jan1971 1100
Colorado      Maybell        -60 01jan1979 5920
Idaho         Island Prk Dam -60 18jan1943 6285
Minnesota     Pokegama Dam   -59 16feb1903 1280
North Dakota  Parshall       -60 15feb1936 1929
South Dakota  McIntosh       -58 17feb1936 2277
Wyoming       Moran          -63 09feb1933 6770
;
```

USCLIM.TEMPCHNG データセットを作成するための DATA ステップ

```
libname usclim 'SAS-data-library';

data usclim.tempchng;
   input @1 State $char13. @15 Date date7. Start_f End_f Minutes;
   Diff=End_f-Start_f;
   informat State $char13. Date date7.;
   format Date date9.;
   datalines;
North Dakota  21feb1918 -33 50  720
South Dakota  22jan1943 -4  45  2
South Dakota  12jan1911 49  -13 120
South Dakota  22jan1943 54  -4  27
South Dakota  10jan1911 55  8   15
;
```

USCLIM.BASETEMP および USCLIM.REPORT カタログに関する注意事項

カタログ USCLIM.BASETEMP と USCLIM.REPORT は、SAS データセットとカタログの両方の DATASETS プロシジャによる処理方法を示すのに使用されています。これらのカタログのコンテンツは、この本の趣旨において重要なものではありません。ほとんどの場合において、SAS/AF、SAS/FSP、またはその他の SAS 製品を使用してカタログエントリを作成します。このセクションの例は、これらのカタログがなくてもテストできます。

CLIMATE、PRECIP および STORM データセット

CLIMATE.HIGHTEMP データセットを作成するための DATA ステップ

```
libname climate 'SAS-data-library';

data climate.hightemp;
   input Place $ 1-13 Date date9. Degree_f Degree_c;
   datalines;
South Africa 21jan2010 122 50
Israel       21jun1942 129 54
Argentina    02jan1920 120 49
Saskatchewan 05jul1937 113 45
India        18jun1905 124 51
Poland       29jul1921 104 40
;
```

CLIMATE.LOWTEMP データセットを作成するための DATA ステップ

```
libname climate 'SAS-data-library';

data climate.lowtemp;
```

```
   input Place $ 1-13 Date date9. Degree_f Degree_c;
   datalines;
Antarctica    21jul83 -129 -89
Siberia       06feb33 -90  -68
Greenland     09jan54 -87  -66
Yukon         03feb47 -81  -63
Alaska        23jan71 -80  -67
;
```

PRECIP.RAIN データセットを作成するための DATA ステップ

```
libname precip 'SAS-data-library';

data precip.rain;
   input Place $ 1-12 @13 Date date9. Inches Cms;
   format Date date9.;
   datalines;
La Reunion  15mar1952 74 188
Taiwan      10sep1963 49 125
Australia   04jan1979 44 114
Texas       25jul1979 43 109
Canada      06oct1964 19 49
;
```

PRECIP.SNOW データセットを作成するための DATA ステップ

```
libname precip 'SAS-data-library';

data precip.snow;
   input Place $ 1-12 @13 Date date7. Inches Cms;
   format Date date9.;
   datalines;
Colorado    14apr21 76 193
Alaska      29dec55 62 158
France      05apr69 68 173
;
```

STORM.TORNADO データセットを作成するための DATA ステップ

```
libname storm 'SAS-data-library';

data storm.tornado;
   input State $ 1-12 @13 Date date7. Deaths Millions;
   format Date date9. Millions dollar6.;
   label Millions='Damage in Millions';
   datalines;
Iowa        11apr65 257 200
Texas       11may70 26  135
Nebraska    06may75 3   400
Connecticut 03oct79 3   200
Georgia     31mar73 9   115
;
```

付録 2

DATA ステップデバッガコマンド

ディクショナリ **801**
BREAK 801
CALCULATE 803
DELETE 804
DESCRIBE 805
ENTER 806
EXAMINE 807
GO 808
HELP 809
JUMP 809
LIST 810
QUIT 811
SET 811
STEP 812
SWAP 813
TRACE 813
WATCH 814

ディクショナリ

BREAK

プログラムの実行を実行ステートメントで一時停止します。

カテゴリ: デバッグリクエストの操作

別名: B

構文

BREAK *location* <AFTER *count*> <WHEN *expression* > <DO *group* >

必須引数

location
ブレークポイントを設定する場所を指定します。*Location* は次のうちの1つになります。

label
ステートメントラベルです。ブレークポイントはラベルの次のステートメントに設定されます。

line-number
ブレークポイントを設定するプログラムの行番号です。

*
現在の行

オプション引数

AFTER *count*
ステートメントが *count* 回実行される毎に、ブレークポイントを有効にします。カウントは連続します。DO ループ内でステートメントに AFTER オプションが適用されると、カウントはループの反復から次の反復へと継続されていきます。デバッガは反復の開始前に *count* の値を 1 にリセットしません。

BREAK コマンドに AFTER と WHEN の両方がある場合、AFTER が最初に評価されます。AFTER カウントが満たされたら、WHEN 式が評価されます。

ヒント AFTER オプションは DO ループのデバッグに便利です。

WHEN *expression*
式が正しい場合に、ブレークポイントを有効にします。

DO *group*
DO ステートメントと END ステートメントに囲まれたデバッガコマンド群です。DO *group* の構文は以下になります。

DO; *command-1*<…;*command-n*;> **END**;

command
デバッガコマンドを指定します。複数コマンドはセミコロンで区切ります。

DO グループは2行以上になったり、IF-THEN/ELSE ステートメントを含んだりする場合があります。

IF *expression* **THEN** *command;* <ELSE *command;*>

IF *expression* **THEN DO** *group;* <ELSE DO *group;*>

IF は式を評価します。条件が真の場合、デバッガコマンドまたは THEN 句の DO グループが実行されます。条件が真ではない場合、オプションの ELSE コマンドが代替アクションを指示します。IF には次の引数を使用できます。

expression
デバッガ式を指定します。評価結果が非ゼロおよび非欠損の場合、この式は真になります。評価結果がゼロまたは欠損の場合、この式は偽になります。

command
デバッガコマンドを 1 つだけ指定します。

DO *group*
DO グループを指定します。

詳細

BREAK コマンドは指定のステートメントで DATA ステップの実行を一時停止します。BREAK コマンドの実行は*ブレークポイントの設定*とも言われます。

デバッガはブレークポイントを検知すると、次のことをします。

- AFTER *count* の値をチェックして、存在する場合、*count* がブレークポイントアクティベーションに到達しているならば、実行を一時停止
- WHEN 式を評価して、存在する場合は評価した条件が真ならば実行を一時停止
- AFTER も WHEN 句も存在しない場合は、実行を一時停止
- 実行が停止している場所の行番号を表示
- DO グループに存在するコマンドを実行
- >プロンプトでコントロールをユーザーに戻す

ブレークポイントが1つ以上のステートメントを含むソース行に設定されている場合、ブレークポイントはソース行の各ステートメントに適用となります。ブレークポイントがマクロ起動を含む行に設定されている場合、デバッガはマクロで生成される各ステートメントで一時停止します。

例

- 現在のプログラムの 5 行目にブレークポイントを設定

```
b 5
```

- ステートメントラベルが **eoflabel** のステートメントの後ろにブレークポイントを設定

```
b eoflabel
```

- 45 行目が 3 回実行される毎に有効となるブレークポイントを 45 行目に設定

```
b 45 after 3
```

- 45 行目が 3 回実行されて、かつ DIVISOR と DIVIDEND の値が 0 の場合に有効となるブレークポイントを 45 行目に設定

```
b 45 after 3
when (divisor=0 and dividend=0)
```

- プログラムの 45 行目にブレークポイントを設定して、変数 NAME と変数 AGE の値を評価

```
b 45 do; ex name age; end;
```

- プログラムの 15 行目にブレークポイントを設定 DIVISOR の値が 3 より大きい場合、STEP を実行 そうでない場合は DIVIDEND の値を表示

```
b 15 do; if divisor>3 then st;
else ex dividend; end;
```

関連項目:

コマンド:

- “DELETE” (804 ページ)
- “WATCH” (814 ページ)

CALCULATE

デバッガ式を評価して、結果を表示

カテゴリ: DATA ステップ変数の操作

構文

CALC *expression*

必須引数

expression
デバッガ式を指定します。

制限事項 デバッガ式に関数を含むことはできません。

詳細

CALCULATE コマンドはデバッガ式を評価して結果を表示します。結果は数値になります。

例

- 1.1 と 1.2 と 3.4 を足して合計に 0.5 をかける

```
calc (1.1+1.2+3.4)*0.5
```

- STARTAGE と DURATION を足す

```
calc startage+duration
```

- 変数 SALE の値から変数 DOWNPAY の値を引いて、その値に変数 RATE の値をかける その値を 12 で割って、50 を足す

```
calc (((sale-downpay)*rate)/12)+50
```

関連項目:

“式の処理” (405 ページ)

DELETE

ブレークポイントを削除または DATA ステップにおいて変数のステータスを監視

カテゴリ: デバッグリクエストの操作

別名: D

構文

DELETE BREAK *location*

DELETE WATCH *variable(s)* | _ALL_

必須引数

BREAK
ブレークポイントを削除

別名 B

location
削除したいブレークポイントの場所を指定します。*location* には、次の値を指定できます。

ALL
DATA ステップにあるすべてのブレークポイント

label
ステートメントラベルの後のステートメント

line-number
プログラムの行番号

*
現在の行のブレークポイント

WATCH
ウォッチ対象変数のステータスを削除します

別名 W

variable(s)
ステータスが削除されるウォッチ対象変数の名前です

ALL
すべてのウォッチ対象変数のステータスを削除すると指定します

例

- ステートメントラベルでブレークポイントを削除します

```
eoflabel
:
d b eoflabel
```

- 現在の DATA ステップの変数 ABC のウォッチステータスを削除します

```
d w abc
```

関連項目:

コマンド:

- “BREAK” (801 ページ)
- “WATCH” (814 ページ)

DESCRIBE

変数の属性を表示します。

カテゴリ: DATA ステップ変数の操作
別名: DESC

構文

DESCRIBE *variable(s)* | _ALL_

必須引数

variable(s)
DATA ステップの変数を示します

ALL
DATA ステップに定義されているすべての変数を示します

詳細

DESCRIBE コマンドは指定した変数の属性を表示します（複数可）。

DESCRIBE は名前、タイプ、変数の長さ、そしてある場合は入力形式と出力形式または変数ラベルを表示します。

例

- 変数 ADDRESS の属性を表示します

```
desc address
```

- 配列エレメント ARR{*i* + *j*}の属性を表示します

```
desc arr{i+j}
```

ENTER

ENTER キーにデバッガコマンドを割り当てます。

カテゴリ: デバッガのカスタマイズ

構文

ENTER *command-1* <; *command-2*; …>

必須引数

command
デバッガコマンドを指定します。

デフォルト STEP 1

詳細

ENTER コマンドは ENTER キーにデバッガコマンドを割り当てます（複数可）。新しいコマンドを ENTER キーに割り当てると、既存のコマンド割り当ては置き換えられます。

複数のコマンドを割り当てる場合は、コマンドをセミコロンで区切ります。

例

- ENTER キーにコマンド STEP 5 を割り当てます

```
enter st 5
```

- ENTER キーに変数 CITY に対してコマンド EXAMINE とコマンド DESCRIBE をに割り当てます

```
enter ex city; desc city
```

EXAMINE

変数の値を表示します。

カテゴリ: DATA ステップ変数の操作

別名: E

構文

EXAMINE *variable-1* <*format-1*> <*variable-2* <*format-2* …> >

EXAMINE _ALL_ <*format*>

必須引数

variable
DATA ステップ変数を示します。

ALL
現在の DATA ステップに定義されているすべての変数を示します。

オプション引数

format
SAS 形式かユーザーが作成した形式かを示します。

詳細

EXAMINE コマンドは指定した変数の値を表示します(複数可)。デバッガは変数に現在関連付けられている出力形式で値を表示します。

例

- 変数 N と変数 STR の値を表示します

  ```
  ex n str
  ```

- 配列 TESTARR のエレメント *i* を表示します

  ```
  ex testarr{i}
  ```

- 配列 CRR のエレメント *i*+1, *j**2、および *k*-3 を表示します

  ```
  ex crr{i+1}; ex crr{j*2}; ex crr{k-3}
  ```

- SAS 日付変数 T_DATE を DATE7.出力形式で表示します

  ```
  ex t_date date7.
  ```

- 配列 NEWARR のすべてのエレメントの値を表示します

  ```
  ex newarr{*}
  ```

関連項目:

コマンド:

- “DESCRIBE” (805 ページ)

GO

DATA ステップの実行を開始または再開します。

カテゴリ: プログラム実行の制御

別名: G

構文

GO <*line-number* | *label*>

引数なし

引数を省略した場合、GO は DATA ステップの実行を再開し、ブレークポイントに到達するまで、またはウォッチ対象変数の値が変化するまで、または DATA ステップが実行を完了するまでステートメントを継続して実行します。

オプション引数

line-number
次に実行を一時停止させるプログラム行の番号を指定します

label
ステートメントラベルです。実行は次のステートメントラベルの後で一時停止します。

詳細

GO コマンドは実行を開始または再開します。すべてのオブザベーションを読み込む、または GO コマンドに指定されたブレークポイントに到達する、または BREAK コマンドで以前に設定されたブレークポイントに到達するまで、実行は継続されます。

例

- プログラムの実行を再開して、ステートメントを継続して実行します。

```
g
```

- プログラムの実行を再開して、104 行でステートメントの実行を一時停止します

```
g 104
```

関連項目:

コマンド:

- “JUMP” (809 ページ)
- “STEP” (812 ページ)

HELP

デバッガコマンドの情報を表示します。

カテゴリ: ウィンドウの管理

構文

HELP

引数なし

HELP コマンドはデバッガコマンド要覧を表示します。コマンド名を選択すると、そのコマンドの構文や使用法の情報が表示されます HELP コマンドはウィンドウのコマンド行またはメニューまたはファンクションキーで入力しなければなりません。

JUMP

一時停止したプログラムの実行を再開します。

カテゴリ: プログラム実行の制御

別名: J

構文

JUMP *line-number* | *label*

必須引数

line-number
一時停止しているプログラムが再開するプログラム行の番号を示します。

label
ステートメントラベルです。実行はラベルの後のステートメントで再開されます。

詳細

JUMP コマンドは、途中のステートメントを実行することなく、特定の場所にジャンプしてプログラムを実行します。JUMP 実行後は、GO または STEP で実行を再開しなくてはなりません。DATA ステップのどの実行ステートメントにもジャンプすることができます。

注意:
JUMP コマンドを使用して DO ループ内のステートメントまたは LINK-RETURN グループの対象となっているラベルのステートメントにジャンプしないでください。 そのような場合、ループの最初や LINK ステートメントに設定されているコントロールを飛ばしてしまうことになり、予期しない結果が生じる原因となります。

JUMP は次の 2 つのケースで有益です

- 他の場所に集中して問題を起こしているコードの箇所を飛ばしたい場合このケースでは、JUMP コマンドを使って、DATA ステップで問題のある箇所の後ろのポイ

ントに移動します。このケースでは、JUMP コマンドを使って、DATA ステップで問題のある箇所の後ろのポイントに移動します。

- 問題を引き起こした一連のステートメントを再実行したい場合 このケースでは、JUMP を使って DATA ステップで問題あるステートメントの前の箇所に移動し、そして SET コマンドを使って関係のある変数の値を当時の値にリセットします。そしてそれらのステートメントを STEP または GO で再実行します。

例

- 5 行目にジャンプ

```
j 5
```

関連項目:

コマンド:

- “GO” (808 ページ)
- “STEP” (812 ページ)

LIST

引数にある項目のすべての出現個所を表示します。

カテゴリ: デバッグリクエストの操作

別名: L

構文

LIST <_ALL_ | BREAK | DATASETS | FILES | INFILES | WATCH>

必須引数

ALL
すべての項目の値を表示します。

BREAK
ブレークポイントの表示

別名 B

DATASETS
現在の DATA ステップで使用されている SAS データセットをすべて表示します。

FILES
現在の DATA ステップが書き込むすべての外部ファイルを表示します。

INFILES
現在の DATA ステップが読み込むすべての外部ファイルを表示します。

WATCH
ウォッチ対象の変数を表示します。

別名 W

例

- 全てのブレークポイント、SAS データセット、外部ファイル、そしてウォッチ対象の変数をリスト表示します。

```
l _all_
```

- 現在の DATA ステップのすべてのブレークポイントを表示します。

```
l b
```

関連項目:

コマンド:

- “BREAK” (801 ページ)
- “DELETE” (804 ページ)
- “WATCH” (814 ページ)

QUIT

デバッガセッションを終了します。

カテゴリ: デバッガの終了

別名: Q

構文

QUIT

引数なし

QUIT コマンドはデバッガセッションを終了させ、SAS セッションに戻ります。

詳細

SAS はデバッグしている DATA ステップによってビルドされたデータセットを作成します。しかし、デバッガを終了するのに QUIT を使用した場合、SAS は現在のオブザベーションを現在のデータセットに追加しません。

デバッガセッション中はいつでも QUIT コマンドを使用できます。デバッガセッションの終了後は、新しいデバッグセッションを開始するには DEBUG オプションを付けて DATA ステップを再サブミットしなければなりません。終了後にセッションを再開することはできません。

SET

特定の変数に新たな値を割り当てます。

カテゴリ: DATA ステップ変数の操作

別名: なし

構文

SET *variable=expression*

必須引数

variable
DATA ステップ変数名または配列参照名を指定します。

expression
デバッガ式を示します。

ヒント *expression* には、等号の左側で使用する変数名を指定します。1 つの変数が等号の両側に表示される場合、右側のオリジナル値を使用して式を評価し、等号の左側にある変数に結果が格納されます

詳細

SET コマンドは指定した変数に値を割り当てます。プログラム実行時にエラーを検知した場合、このコマンドを使って変数に新しい値を割り当てられます。これにより、デバッグセッションを継続できます。

例

- 変数 A に値 3 を設定します:

```
set a=3
```

- 変数 B に値 **12345** を割り当て、それを以前の B の値に連結させます:

```
set b='12345' || b
```

- 配列エレメント ARR{1}を式 a+3 の結果に設定します:

```
set arr{1}=a+3
```

- 配列エレメント CRR{1,2,3}を式 crr{1,1,2} + crr{1,1,3}の結果に設定します:

```
set crr{1,2,3} = crr{1,1,2} + crr{1,1,3}
```

- 変数 A を式 a+c*3 の結果に設定します:

```
set a=a+c*3
```

STEP

アクティブなプログラムでステートメントを1つずつ実行します。

カテゴリ: プログラム実行の制御

別名: ST

構文

STEP <*n*>

引数なし

STEP は 1 つのステートメントを実行します。

オプション引数

n
実行するステートメントの数を指定します。

詳細

STEP コマンドは DATA ステップのステートメントを、実行が一時停止していたステートメントから実行します。

STEP コマンドを発行すると、デバッガーは:

- 指定した数のステートメントを実行します。
- 行番号を表示します。
- >プロンプトを表示してコントロールをユーザーに戻します。

注: デフォルトで、ENTER キーを押すと STEP コマンドが実行できます。

関連項目:

コマンド:

- “GO” (808 ページ)
- “JUMP” (809 ページ)

SWAP

SOURCE ウィンドウと **LOG** ウィンドウを切り替えます。

カテゴリ: ウィンドウの管理

別名: なし

構文

SWAP

引数なし

SWAP コマンドは、デバッガの起動中に **LOG** ウィンドウと **SOURCE** ウィンドウを切り替えます。デバッグセッションを開始すると、デフォルトでは **LOG** ウィンドウがアクティブになります。DATA ステップの実行中に SWAP コマンドを使用して **SOURCE** ウィンドウと **LOG** ウィンドウの切り替えができ、プログラムの文字列をスクロールして表示したり、またプログラムの実行をモニタリングしたりできます。SWAP コマンドはウィンドウのコマンドライン、またはメニュー、またはファンクションキーから入力します。

TRACE

デバッガが DATA ステップ実行の連続レコードを表示するかどうかを制御します。

カテゴリ: デバッグリクエストの操作

別名: T

デフォルト: OFF

構文

TRACE <ON | OFF>

引数なし

引数なしで TRACE コマンドを使って、トレーシングがオンかオフかが確認できます。

オプション引数

ON
デバッガーが DATA ステップ実行時の連続レコードを表示するための準備を開始します。次の DATA ステップの実行を再開するステートメント(たとえば GO)で **DEBUGGER LOG** ウィンドウに DATA ステップ実行中のすべてのアクションを記録します。

OFF
表示を停止します。

比較

TRACE は TRACE コマンドの現在の状況を表示します。

例

- TRACE が ON か OFF か判別します:

```
trace
```

- デバッガ実行のレコードを表示する準備をします:

```
trace on
```

WATCH

指定した変数の値が変わると実行を一時停止します。

カテゴリ: デバッグリクエストの操作

別名: W

構文

WATCH *variable(s)*

必須引数

variable(s)
DATA ステップ変数を指定します(複数可)。

詳細

WATCH コマンドは指定した変数をモニターして、その値が変化したときにプログラムの実行を一時停止します。

ウォッチしている変数の値が変化する度に、デバッガは次のことをします。

- 実行を一時停止します
- 実行が一時停止している行番号を表示します。
- 変数の古い値を表示します。
- 変数の新しい値を表示します。
- >プロンプトを表示してコントロールをユーザーに戻します。

例

- 変数 DIVISOR の値の変化をモニターします。

```
w divisor
```

用語集

1 対 1 のマージ
MERGE ステートメント(BY ステートメントなし)を使用し、データセットにおけるオブザベーションの位置に基づいて、2 つ以上のデータセットのオブザベーションを結合する処理。

1 対 1 のマッチング
複数の SET ステートメントを使用して各データセットからオブザベーションを読み込むことによって、複数のデータセットにあるオブザベーションを結合し、1 つのオブザベーションにする処理。

ACROSS 変数
REPORT プロシジャで、フォーマット値のそれぞれがレポートの列を形成する変数。変数に出力形式が適用されていない場合、各未フォーマット値により列が形成されます。

autoexec ファイル
SAS の起動時に自動で実行される SAS ステートメントを含むファイル。autoexec ファイルを使用すると、一部の SAS システムオプションを指定したり、頻繁に使用されるデータソースにライブラリ参照名およびファイル参照名を割り当てたりできます。

Base SAS
SAS Foundation を構成するコア製品で、SAS ソフトウェアの配置ごとにインストールされます。Base SAS では、データのアクセス、管理、分析、提示のための情報配信システムが提供されます。

BREAK 変数
REPORT プロシジャでは、区切り行の場所を指定するために選択する GROUP 変数または ORDER 変数を意味します。BREAK 変数の値が変わるたびに、BREAK に指定した処理が REPORT プロシジャによって実行されます。

BY 値
BY 変数の値。

BY グループ
BY ステートメントで指定された変数に対して、同じ値を有するオブザベーションまたは行のグループ。BY ステートメントで複数の変数が指定されている場合、BY グループはそれらの変数の値の固有な組み合わせを持つオブザベーションの集まりになります。

BY グループ処理
オブザベーションを 1 つ以上の変数の値に従って並べ替え、グループ化またはインデックスを付けるために BY ステートメントを使用する処理。BY グループ処理は多数の SAS プロシジャと DATA ステップによってサポートされています。たとえば、PRINT プロシジャで BY グループ処理を使用すると、1 つの SAS データセット内の異なるオブザベーショングループに対して別々のレポートを印刷できます。

BY グループ変数
BY 変数を参照。

BY 変数
BY ステートメントで指定された変数。処理されるオブザベーションのグループがその値により定義されます。

CALL ルーチン
変数の値を変更したり、他のシステム操作を実行したりする SAS プログラミング言語のコンポーネント。割り当てステートメントでは CALL ルーチンを使用できないという点を除けば、CALL ルーチンは関数に類似しています。すべての SAS CALL ルーチンは、CALL ステートメントで呼び出されます。すなわち、CALL ステートメントでキーワード CALL の後にルーチンの名前を指定する必要があります。

DATA ステップ
SAS プログラムで、DATA ステートメントで始まり、RUN ステートメント、別の DATA ステートメント、PROC ステートメント、ジョブ終了のいずれかで終わるステートメントのグループ。DATA ステップでは、生データまたはその他 SAS データセットの読み取り、SAS データセットの作成ができます。

DO グループ
単純な DO ステートメントで始まり、対応する END ステートメントで終わる一連のステートメント。

DO ループ
反復 DO、DO WHILE または DO UNTIL ステートメントで始まり、対応する END ステートメントで終わる一連のステートメント。ステートメントは、DO ステートメントで指定された指示に従って(通常は繰り返し)実行されます。

FIRST.変数
各 BY グループの最初のオブザベーションを識別するために作成される一時変数。この変数は、SAS データセットに追加されません。

format
SAS 出力形式を参照してください。

LAST.変数
各 BY グループの最後のオブザベーションを識別するために作成される一時変数。この変数は、SAS データセットに追加されません。

ODS
Output Delivery System を参照。

ODS 出力先
ODS (Output Delivery System)が特定の種類の出力を生成するために使用する出力先。ODS 出力先の種類には、HTML、XML、リスト、PostScript、RTF、SAS データセットが含まれますが、これに限定されるわけではありません。

Output Delivery System
マークアップ言語(HTML、XML)、PDF、リスト、RTF、PostScript、SAS データセットなどのさまざまな出力形式で出力を生成できる SAS のコンポーネント。略称: ODS。

PDV
プログラムデータベクトルを参照。

Profile カタログ
Sasuser.Profile カタログを参照。

Sasuser.Profile カタログ
特定のユーザーまたはサイトに対する SAS ウィンドウ環境の属性についての情報が格納される SAS カタログ。これには、ファンクションキー定義、グラフィックアプリケーションのフォント、ウィンドウ属性、対話型 SAS プロシジャで使用されるその他の情報が含まれます。

Sasuser ライブラリ
最初の SAS セッションの開始時に作成されるデフォルトの永久 SAS ライブラリ。Sasuser ライブラリには、SAS 用に指定したカスタマイズ済み機能または設定を格納する PROFILE カタログが含まれます。

SAS カタログ
さまざまな種類の情報を、カタログエントリと呼ばれるより小さな単位で格納する SAS ファイル。単一の SAS カタログに、異なる種類のカタログエントリを含めることができます。

SAS カタログエントリ
SAS カタログ内での個々の格納単位。各エントリには使用目的に応じて、さまざまなエントリタイプが割り当てられています。

SAS 日付値
SAS で日付を表す整数。この整数は、1960 年 1 月 1 日から指定の日付までの日数を表します。たとえば、SAS 日付値 366 はカレンダ日付 1961 年 1 月 1 日を表します。

SAS 日付定数
SAS ステートメントで日付を表す'ddMMMyy'd または'ddMMMyyyy'd 形式の文字列。文字列は引用符で囲み、その後に文字 d を続けます('6JUL01'd、'06JUL01'd、'6 JUL2001'd、'06JUL2001'd など)。

SAS 関数
式または割り当てステートメントで使用してゼロ以上の引数を処理したり値を返したりできる SAS 言語要素の一種。SAS 関数の例としては、MEAN や SUM があります。略称: 関数。

SAS キーワード
SAS 言語の主要部分となるリテラル。たとえば、SAS キーワードには、DATA、PROC、RUN、SAS 言語要素名、SAS ステートメントオプション名、システム変数などが含まれます。

SAS 言語
データ分析とレポート作成のためのプロシジャ、SAS ファイル管理とデータ操作のためのステートメントと関数、SAS 環境を定義するオプション、マクロ機能、ヘルプメニュー、テキスト編集とファイル管理のためのウィンドウ環境を含むプログラミング言語。

SAS コマンド
SAS を起動するコマンド。このコマンドは、動作環境とサイトによって異なる場合があります。

SAS コンパイル
SAS 言語の各種ステートメントを、ユーザーが入力した形式から SAS システムで利用可能な形式に変換する処理。

SAS 時間値
SAS で時間を表す整数。この整数は、現在日付の午前 0 時から指定の時間値までの秒数を表します。たとえば、午前 9 時 30 分の SAS 時間値は 34200 です。

SAS 時間定数
SAS ステートメントで時間を表す'hh:mm:ss't 形式の文字列。文字列は引用符で囲み、その後に文字 t を続けます('09:53:22't など)。

SAS 式
数値、文字値またはブール値を生成するために評価される命令セットを形成する、一連のオペランドと算術演算子から成るマクロ式の一種。オペランドの例としては、定数やシステム関数があります。SAS はプログラムステートメントで算術演算式を使用して、変数の作成、値の割り当て、新しい値の計算、変数の変換、条件付き処理を行います。

SAS システムオプション
指定された時点から変更されるまで、SAS プログラム全体の処理や対話型 SAS セッションの処理に影響を与えるオプション。SAS システムオプションによって制御される項目の例としては、SAS 出力の外観、SAS によって使用される一部のファイルの処理、システム変数の使用、SAS データセットのオブザベーションの処理、SAS 初期化の機能、SAS とホスト動作環境との対話方法などがあります。

SAS 出力形式
データ値を出力として表示したり書き込んだりするために、パターンを適用したり命令を実行したりする SAS 言語要素の一種。出力形式の種類は、データの種類(数値、文字、日付、時間、タイムスタンプ)に対応しています。ユーザー定義の出力形式を作成することもできます。SAS 出力形式の例としては、BINARY や DATE があります。略称: 出力形式。

SAS 初期化
SAS セッションを開始するために有効にする必要があるグローバル特性の設定処理。初期化オプションとよばれる特定の SAS システムオプションを設定して、初期化を行います。SAS 初期化は、SAS の起動時に自動的に行われます。

SAS ステートメント
SAS に操作の実行を指示するかまたは情報を提供する SAS キーワード、SAS 名、および特殊文字と演算子の文字列。SAS ステートメントの最後には、必ずセミコロン(;)を追加します。

SAS セッション
特定の SAS ソフトウェア製品の起動から終了までの間のアクティビティ。

SAS データセット
ネイティブ SAS ファイル形式のコンテンツを含むファイル。SAS データセットには、SAS データファイルと SAS データビューの 2 種類があります。

SAS データセットオプション
SAS データセット名の後にかっこに囲んで記述されるオプション。データセットオプションにより、その SAS データセットの処理にのみ適用されるアクションが指定されます。

SAS データビュー
他のファイルからデータ値を取得する SAS データセットの一種。SAS データビューに含まれているのは、変数(列)のデータ型やデータ長などのディスクリプタ情報のみです。これに加えて、他の SAS データセットや、他のソフトウェアベンダのファイル形式で格納されているファイルからデータ値を取得するのに必要となる他の情報などがあります。略称: データビュー。

SAS データファイル
SAS データセットの一種。データ値と、データに関連付けられたディスクリプタ情報を含みます。ディスクリプタ情報には、変数のデータ型や長さ、データの作成に使用されたエンジンの名前などが含まれています。

SAS 名
SAS 変数や SAS データセットなどの項目に割り当てられた名前。ほとんどの SAS 名では、最初の文字はアルファベットかアンダースコアにする必要があります。以降の文字には、アルファベット、数字またはアンダースコアを使用できます。ブランクと特殊文字は使用できません(アンダースコアを除く)。ただし、SAS 変数名に適用する規則は、VALIDVARNAME=システムオプションで確定されます。SAS 名の最大長は、その SAS 名が割り当てられる言語要素によって異なります。

SAS 日時値
SAS で日付と時間を表す整数。この整数は、1960 年 1 月 1 日午前 0 時から指定の日時までの秒数を表します。たとえば、2000 年 6 月 5 日午前 9 時 30 分の SAS 日時値は 1275816600 です。

SAS 日時定数
SAS で日付と時間を表す'ddMMMyy:hh:mm:ss'dt または 'ddMMMyyyy:hh:mm:ss'dt 形式の文字列。文字列は引用符で囲み、その後に文字 dt を続けます('06JUL2001:09:53:22'dt など)。

SAS 入力形式
データ値を入力として読み込むためにパターンを適用したり命令を実行したりする SAS 言語要素の一種。入力形式の種類は、データの種類(数値、文字、日付、時間、タイムスタンプ)に対応します。ユーザー定義の入力形式を作成する機能もサポートされています。SAS 入力形式の例としては、BINARY や DATE があります。略称: 入力形式。

SAS ファイル
SAS が作成、編成、管理する特殊構造化ファイル。SAS ファイルには、SAS データセット、カタログ、ストアドプログラム、アクセスディスクリプタ、ユーティリティファイル、多次元データベースファイル、財務データベースファイル、データマイニングデータベースファイル、アイテムストアファイルなどがあります。

SAS プログラム
入力データの読み込みや変換、出力の生成を行うため、1 つまたは一連のプロセスで SAS に指示する SAS ステートメントのグループ。DATA ステップとプロシジャステップ(単独かまたは組み合わせて使用)が SAS プログラムの基礎となります。

SAS プロシジャ
特定の機能を提供し、PROC ステートメントを使用してアクセスされるプログラム。たとえば、SAS プロシジャを使用すると、レポートの作成、ファイルの管理、または

データの分析が行えます。SAS ソフトウェアには、多くのプロシジャが含まれています。

SAS 変数
SAS データセット内または SAS データビュー内の列。各変数のデータ値は、すべてのオブザベーション(行)の単一の特性を表します。

SAS ログ
入力した SAS ステートメントの記録、およびプログラムの実行についてのメッセージを含むファイル。

WHERE 式
オブザベーションの選択基準を定義します。

WHERE 処理
WHERE 式の実行時に使用され、処理するオブザベーションを条件に従って選択する方法。

Work ライブラリ
各 SAS セッションまたは SAS ジョブの最初に、SAS が自動的に定義する一時 SAS ライブラリ。User ライブラリが指定されていない場合、新たに作成される 1 レベル名の SAS ファイルは、デフォルトでは Work ライブラリに置かれ、現在の SAS セッションまたはジョブの最後に削除されます。

値へのブランクの埋め込み
SAS ソフトウェアでは、変数の長さより短い文字値の末尾にブランクが追加される処理を意味します。

一時 SAS データセット
現在のプログラムまたは対話型 SAS セッションの間のみ存在するデータセット。一時 SAS データセットは、それ以降の SAS セッションでは使用できません。

一時 SAS ファイル
SAS セッションまたはジョブの終了時に削除される、SAS ライブラリ(通常は Work ライブラリ)内の SAS ファイル。

一時 SAS ライブラリ
現在の SAS セッションまたはジョブに対してのみ存在するライブラリ。最も一般的な一時ライブラリは、Work ライブラリです。

印刷ファイル
キャリッジコントロール(プリンタコントロール)情報を含む外部ファイル。

インセットテーブル
グラフィックモードで作成されたプロットの中や横に配置される統計値の表。工程能力分析では、統計量に、能力指数、仕様限界、適合度統計量、曲線パラメータ、記述統計量、分位点を含められます。

インタリーブ
2 つ以上の並べ替え済み SAS データセットを、BY 変数の値に基づいて、1 つの並べ替え済み SAS データセットに結合するプロセス。

インデックス
SAS データセットのオブザベーションに SAS が迅速かつ効率的にアクセスできるようにする SAS データセットのコンポーネント。SAS インデックスの目的は、WHERE 句処理を最適化し、BY グループ処理を促進することです。

生データ
統計分析において、標準化などの特定の操作が実行されていないデータ(SAS データセットのデータを含む)。

生データファイル
レコードのフィールドにデータ値が含まれている外部ファイル。DATA ステップでは、INFILE ステートメントおよび INPUT ステートメントを使用して生データファイルを読み込みます。

永久 SAS データセット
現在のプログラムまたは対話型 SAS セッションが終了した後も削除されない SAS データセット。永久 SAS データセットは、現在以降の SAS セッションで使用可能です。

永久 SAS ファイル
SAS セッションまたはジョブの終了時に削除されない SAS ライブラリ内のファイル。

永久 SAS ライブラリ
SAS セッションの終了時に削除されないため、それ以降の SAS セッションで使用可能な SAS ライブラリ。

英数字
英文字、数字、特殊文字やブランクのいずれかの種類の文字。大部分のコンピュータシステムでは、数値データを、英数字データやテキストデータとは厳密に異なる方法で格納します。

エラーメッセージ
プログラムの処理を続行できなかったことを示す SAS ログまたはメッセージウィンドウのメッセージ。

エントリタイプ
カタログエントリの構造と属性を SAS で特定する SAS カタログエントリの特性。SAS カタログエントリを作成すると、自動的にエントリの種類が名前の一部に割り当てられます。

オブザベーション
SAS データセットの行。オブザベーション内のデータ値はすべて、顧客や状態などの単一のエンティティに関連付けられます。各オブザベーションには、各変数に対する 1 つのデータ値か欠損値インジケータのどちらかが含まれます。

オブザベーション番号
データセット全体を順番に読み込むときの、オブザベーションの SAS データセットでの相対的な位置を示す番号です。この番号は内部に格納されません。

外部ファイル
ホストオペレーティングシステムまたは別のベンダのソフトウェアアプリケーションによって作成および管理されるファイル。外部ファイルは、データと格納された SAS ステートメントの両方が読み取り可能です。

カタログ
SAS カタログを参照。

カタログエントリ
SAS カタログエントリを参照。

カタログディレクトリ
カタログの各メンバの名前、種類、説明、および更新ステータスに関する情報を格納して保持する、SAS カタログの一部。

日付値
SAS 日付値を参照。

日付定数
SAS 日付定数を参照。

日付と時間出力形式
数値を日付、時間および日時として書き出す方法を SAS に指示する命令。

日付と時間入力形式
日付、時間および日時として表された数値を読み取る方法を SAS に指示する命令。

カテゴリ
TABULATE プロシジャでは、分類変数の一意の値の組み合わせを意味します。TABULATE プロシジャにより、データセットのオブザベーションに存在する値の一意の組み合わせそれぞれについて、個別のカテゴリが作成されます。PROC TABULATE で作成される各カテゴリは、カテゴリを記述するページ、行、および列が交差する、テーブルの 1 つ以上のセルによって表されます。

カラム入力
DATA ステップにおける入力スタイルで、INPUT ステートメントにカラム番号を含めて、各変数の値がどのカラムに含まれているかを SAS に指示します。各変数の値がすべてのレコードで同じ場所にある場合、この入力スタイルが役立ちます。

関数
SAS 関数を参照。

キーワード
SAS キーワードを参照。

キャリッジコントロール文字
用紙送りの行数や、改ページ、行のスキップ、重ね打ちのための現在行保持の時点をプリンタに指示する記号。

区切り
REPORT プロシジャでは、レポート要素を視覚的に分割、統計量と計算変数を要約、テキストを表示、レポートの行セットの計算値を表示、DATA ステップステートメントを実行するレポートのセクションを意味します。選択した変数の値の変更時、またはレポートの先頭や末尾に区切りを作成できます。

区切り行
REPORT プロシジャでは、レポートの各部分を視覚的に分割する文字、統計量と計算変数の要約(要約行と呼ばれる)、テキスト、レポートの行セットの計算値を含むレポートの行を意味します。

区切り文字
テキスト文字列の要素を区切る境界として機能する文字。

グローバルマクロ変数
同じ名前のローカルマクロ変数が存在する場合を除いて、SAS プログラムでグローバルスコープかローカルスコープのどちらかで参照できるマクロ変数。グローバルマクロ変数は、当該セッションまたはプログラムの終了まで存在します。

計算変数
値が計算される変数。たとえば、REPORT プロシジャでは、COMPUTE ウィンドウで入力されるステートメントから値が計算されます。

欠損値
特定の行または列にデータが含まれていない変数値の種類。デフォルトでは、数値変数の欠損値は 1 つのピリオドに、文字変数の欠損値はブランクに置き換えて書き込まれます。

欠損値のプロパゲーション
欠損値の使用結果。算術式の欠損値の存在により式の結果が欠損値として設定されます。その結果を別の式で使用すると、次の結果も欠損値となります。

合計ステートメント
プラス記号の右側にある式の結果を、プラス記号の左側にあるアキュムレータ変数に追加する DATA ステップステートメント。合計ステートメントの形式は、variable + expression;です。

交差
TABULATE プロシジャでは、2 つ以上の要素の効果を結合する処理を意味します。

更新
マスタデータセットの変数値をトランザクションデータセットのオブザベーションの値で置換するプロセス。

構成オプション
SAS コマンドまたは構成ファイルで指定される SAS システムオプション。構成オプションにより、コンピュータハードウェアとオペレーティングシステムとの対話方法が決定されます。

構成ファイル
SAS の実行環境を定義する SAS システムオプションを含む外部ファイル。これらのシステムオプションは、SAS 起動時に毎回有効になります。

後置アットマーク
入出力行を保持し、以降の INPUT または PUT ステートメントで SAS が読み取りや書き込みできるようにするため使用される特殊記号@。

構文エラー
SAS ステートメントのスペルまたは文法のエラー。構文エラーは、実行前の各 SAS ステップのコンパイル時に検出されます。

構文のチェック
使用法が適切か、スペルが正しいか、SAS 命名規則に従っているかなど、各 SAS ステートメントが SAS によりチェックされる処理。

コマンド
オペレーティングシステムで特定タスクを実行するためのディレクティブ。

コメント
コメントステートメントを参照。

コメントステートメント
SAS プログラムに埋め込まれ、説明文として役に立つ情報。処理時にコメントは無視されますが、SAS ログに書き込まれます。コメント構文にはいくつか形式があります。たとえば、* message ;のように、アスタリスクで開始しセミコロンで終了するステートメントとして表すことができます。

コンパイル
プログラムコンパイルを参照。

サイト番号
SAS ソフトウェアのライセンス先の企業または組織の識別に使用される番号。サイト番号は、すべての SAS セッションでログの上部付近に表示されます。

算術演算式
SAS 式を参照。

時間値
SAS 時間値を参照。

時間定数
SAS 時間定数を参照。

次元
テーブル次元を参照。

次元式
TABULATE プロシジャの TABLE ステートメントにおいてテーブルの行、列、またはページのコンテンツおよび配置を定義する部分。

指数
数式で、基本数または基本式を何乗するかを示す数字または式。たとえば、.1234 * 10 の 4 乗という式では、指数は 4 になります。

システムオプション
SAS システムオプションを参照。

実行可能ステートメント
DATA ステップにおいて、DATA ステップのコンパイル時ではなく、DATA ステップの実行時になんらかのアクションを起こさせる SAS ステートメント。

自動マクロ変数
ユーザーではなく SAS によって定義されるマクロ変数で、SAS セッションについての情報を提供します。たとえば、SYSPROCESSID 自動マクロ変数には、現在の SAS プロセスのプロセス ID が含まれます。

自動呼び出し機能
マクロを定義するソースステートメントを格納し、必要に応じてそのマクロを呼び出せる SAS の機能。プログラムに定義を含める必要はありません。

修飾リスト入力
INPUT ステートメントで入力形式およびフォーマット修飾子と呼ばれる特殊命令を使用する入力スタイル。修飾リスト入力では、入力レコードをスキャンして、少なくと

も 1 つのブランク(またはその他の区切り文字)か、場合によっては複数のブランクによって区切られたデータ値を探します。

出力オブジェクト
DATA ステップまたは PROC ステップで生成されたデータが含まれるプログラミングオブジェクト。そのデータのフォーマット方法についての情報を提供するテーブル定義も含むことが可能です。

出力先
ODS 出力先を参照。

出力バッファ
DATA ステップにおいて、PUT ステートメントが指定したファイルや出力デバイスに書き出す前の書き出し先メモリ領域。

条件
SAS プログラムにおける 1 つ以上の数値式または文字式で、この結果値に基づいてなんらかの決定が下されます。

詳細行
REPORT プロシジャでは、データセットの単一オブザベーションからの情報を含むレポートの行か、またはすべてのグループ変数の値の組み合わせが同じグループのオブザベーションの情報をまとめたレポートの行を意味します。

数値
通常は数字のみを含む値。指数表記および 16 進表記の数字が含まれます。数値には、小数点(.)、正符号(+)または負符号(-)が付く場合があります。数値は数値変数に格納されます。

数値出力形式
特定のパターンを使用して数値変数の値を書き出すよう SAS に指示する命令セット。

数値定数
SAS 式で使用される数字。

数値入力形式
特定のパターンを使用して数値データ値を読み取るよう SAS に指示する命令セット。

数値変数
数値と関連記号(小数点、プラス記号、マイナス記号など)のみを含む変数。

ステートメント
SAS ステートメントを参照。

ステートメントオプション
特定の SAS ステートメントで指定する語であり、そのステートメントが実行する処理にのみ影響を与えます。

ステートメントラベル
後ろにコロンが付く SAS 名。DATA ステップであるステートメントの前に置くと、必要に応じて、別のステートメントがそのステートメントの実行を直接指示して、ステップ内の他のステートメントをスキップできます。

ステップ境界
SAS プログラムにおいて DATA ステップまたは PROC ステップの完了を SAS が認識する箇所。

宣言ステートメント
SAS に情報を提供するステートメント。プログラムステートメントのコンパイル時に有効になります。

属性
変数の属性を参照。

タイトル
SAS 出力または SAS ログの各ページの上部に出力される見出し。

対話型ラインモード
SAS プログラムを SAS セッションプロンプトで 1 行ずつ入力する、SAS プログラム実行方法。Enter キーまたは Return キーを押すと即時に各行が処理されます。プロシジャ出力と情報メッセージはディスプレイデバイスに直接返されます。

単一インデックス
1 つの変数のみの値を使用してオブザベーションを検索するインデックス。

単純式
演算子を 1 つだけ使用する SAS 式。

チャート
棒や円のセクションなどの要素でデータのビューが表されるグラフ。

チャート統計量
度数、累積度数、パーセント、累積パーセント、合計、平均など、チャート変数用に計算された統計値。

チャート変数
値が、棒、ブロック、スライス、スパインによって表されるデータのカテゴリである入力データセットの変数。

データ値
データレコードに 1 項目として格納される、文字、数値または英数字の情報の単位。

データエラー
SAS プログラムが無効な値を含むデータを分析したときに発生する実行エラーの一種。たとえば、文字データのための INPUT ステートメントで数値変数を指定した場合、データエラーが発生します。このエラーは SAS ログにレポートされますが、プログラムの実行は続行されます。

データ行
未処理(生)データの行。

データセット
SAS データセットを参照してください。

データセットオプション
特定の SAS データセットに対して適用するアクションを指定する SAS 言語要素。たとえば、データセットオプションでは、変数の名前を変更したり、処理対象のオブ

ザベーションのみを選択したり、変数を処理から除外したり、パスワードを指定したりできます。

データセットラベル
SAS データセットでは、SAS データセットの付記に使用される最大 200 文字のユーザー定義の属性を意味します。

データビュー
SAS データビューを参照。

テーブル次元
ページ、列、行などのテーブル基本要素の 1 つ。

テーブル定義
Output Delivery System (ODS)での出力のフォーマット方法を記述する命令セット。

定数
SAS ソフトウェアにおいて、固定された値を表す数字または文字列。

定数テキスト
マクロの一部またはマクロ変数の値としてオープンコードで格納される文字列。これを使用して、マクロプロセッサは、SAS ステートメント、ディスプレイマネージャコマンド、または他のマクロプログラムステートメントとして使用されるテキストを生成します。定数テキストはモデルテキストとも呼ばれます。

ディスクリプタ情報
SAS データセットのコンテンツおよび属性についての情報。たとえば、ディスクリプタ情報には、変数のデータ型や長さ、データの作成に使用されたエンジンなどが含まれます。SAS は、ディスクリプタ情報の作成および管理をすべての SAS データセット内で行います。

テキスト編集コマンド
特定のテキストエディタで使用されるコマンド。

テキスト文字列
文字列(character string)を参照。

デフォルトディレクトリ
ユーザーが作業を行っているディレクトリ。ログイン時のデフォルトディレクトリは通常、ホームディレクトリになります。

トランザクションデータセット
更新操作において、マスタデータセットを更新するために必要とされる情報を含むデータセット。

ニーモニック演算子
記号ではなく文字で構成される(例:=ではなく EQ)算術演算子または論理(ブール型)演算子。

二重後置アットマーク
DATA ステップの反復を複数回実行する間に、一行のデータを入力バッファに保持するために使用する特殊記号@@。

日時値
SAS 日時値を参照。

日時定数
SAS 日時定数を参照。

入力形式
SAS 入力形式を参照。

入力バッファ
INPUT ステートメントの実行時に、データの各レコードが読み込まれる先のメモリの一時領域。

ヌル値
情報が欠落していることを示す特殊な値。ヌル値は SAS の欠損値に類似する概念です。

ヌルステートメント
1 つのセミコロンかまたは 4 つのセミコロンから成るステートメント。ヌルステートメントは、DATA ステップで入力ストリームデータの最後を指定するために最もよく使用されます。

ネーム出力
PUT ステートメントで等号を使用し、variable=data-value の形式で変数値を書き出すスタイル。

ネーム入力
INPUT ステートメントで等号を使用し、variable=data-value の形式でデータ値を読み取るスタイル。

配列
SAS プログラミング言語において、データの種類が同じで、特定の順序で配置され、配列名によって識別される、SAS 変数の一時グループ。配列は、現在の DATA ステップが処理されている間だけ存在します。

配列参照
配列で処理される要素の参照。

配列名
変数または一時データ要素のグループを識別するために選択された名前。同じ DATA ステップまたは SCL (SAS コンポーネント言語)プログラム内の変数名とは異なる、有効な SAS 名にする必要があります。

バックグラウンド処理
コンピュータとの対話ができない処理。バックグラウンドセッションは、プロセッサ時間が使用可能になったときに実行されるため、フォアグラウンドセッションより実行に多少時間がかかる場合があります。

バッチジョブ
バッチ処理のためにオペレーティングシステムにサブミットされる作業単位。たとえば、UNIX ではバッチジョブはバックグラウンド処理、Windows ではバッチジョブはタスク、z/OS ではバッチジョブは JCL ステートメントセットです。

バッチモード
SAS プログラムの非対話型実行方式。これにより、ファイル(必要なオペレーティングシステムコマンドすべてに加えて SAS ステートメントを含む)が、実行のために動作環境のバッチキューにサブミットされます。

比較演算子
プログラミング言語において、2 つの値またはテキスト文字列間の特定の関係をテストするために式で使用される記号またはニーモニックコード。たとえば、一方の値が他方より小さいかどうかを決定するには、記号<および対応するニーモニック LT が使用されます。

非対話型処理
非対話型モードを参照。

非対話型モード
ユーザーが SAS ステートメントのファイルを準備して、オペレーティングシステムにプログラムをサブミットする、SAS プログラムの実行方法。プログラムはすぐに実行され、現在のセッションを構成します。

標準データ
数字 1 桁または 1 文字が 1 バイトのストレージを占有するデータ。

ファイル参照名(file reference)
ファイル参照名(fileref)を参照。

ファイル参照名(fileref)
外部ファイルまたはディレクトリやフォルダなどの集合保存場所に対して一次的に割り当てられる名前のこと。SAS システムでは、ファイル参照名を使用してファイルや保存場所を識別します。

ファイル指定
外部ファイルの名前です。この名前で、ホスト動作環境によりファイルが認識されます。ディレクトリベースのシステムでは、ファイル指定は、完全なパス名または現在の作業ディレクトリからの相対パス名です。

ブール演算子
論理演算子の別称。

フォーマット修飾子
INPUT ステートメントと PUT ステートメントで使用される特殊記号。これにより、入力データの読み取りと出力データの書き出しを制御できます。

フォーマット出力
PUT ステートメントで SAS 出力形式を使用して、変数値を書き出す方法を指定する出力スタイル。

フォーマット入力
INPUT ステートメントで入力形式と呼ばれる特殊命令を使用して、データフィールドに入力された値の解釈方法を決定する入力スタイル。

複合インデックス
2 つ以上のキー変数の値を検証することによって、SAS データセットのオブザベーションの場所を特定するインデックス。

複合式
演算子が 2 つ以上含まれる式。

複数パネルレポート
変数の値を表示するために、ページで列のセットを使用する出力。たとえば、電話帳は通常、単一ページに名前、住所、および電話番号の複数パネルが配置されています。

物理ファイル名
オペレーティングシステムがファイルを識別するために使用する名前。

プログラミングエラー
SAS プログラムの失敗やプログラマの意図とは異なる動作の原因となる SAS プログラムの論理の欠陥。

プログラムコンパイル
構文をチェックし、プログラム部分をコンピュータで実行可能な形式に変換する処理。

プログラムデータベクトル
コンピュータメモリの一時領域。ここに SAS データセットのオブザベーションが 1 つずつ作成されます。プログラムデータベクトルは論理的概念であり、必ずしも一続きのメモリ領域に相当するわけではありません。略称: PDV。

プロシジャ
SAS プロシジャを参照。

分割文字
一部の SAS プロシジャでは、複数の行にわたるヘッダーを分割する文字を意味します。列ヘッダー分割文字を含めた場合、プロシジャでは分割文字に達するとヘッダーが改行され、次の行にヘッダーが続けられます。分割文字自体は列ヘッダーの一部ではありません。

分析変数
統計量の計算や値の表示に使用される数値変数。通常、分析変数には定量値または連続値が含まれますが、必須ではありません。

分類変数(class variable)
分類変数(classification variable)を参照。

分類変数(classification variable)
この変数の値を使用して、データセット内のオブザベーションを、分析に有効な異なるグループに分類します。分類変数には、文字値か数値のどちらかを設定できます。分類変数には、グループ、サブグループ、カテゴリ、BY 変数が含まれます。

ヘッダールーチン
印刷ファイルのページヘッダーを生成する DATA ステップステートメントのグループ。ヘッダールーチンは、ステートメントラベルで始まり、RETURN ステートメントで終わります。FILE ステートメントの HEADER=オプションとは同じとみなせます。

変数
SAS 変数を参照。

変数属性
名前、ラベル、出力形式、入力形式、データ型、長さなど、特定の変数に関連付けられた特性。

変数の種類
数値か文字のどちらかの変数分類。種類は SAS 変数の属性です。

ポインタ
DATA ステップで、入出力バッファ内の位置を追跡するために SAS が使用するプログラミングツール。

ポインタコントロール
データを読み書きする前にポインタの移動を SAS に指示する処理。

ホスト
ホスト動作環境を参照。

ホスト動作環境
IP アドレスまたはドメイン名によって識別され、ソフトウェアアプリケーションに対する集中制御機能を提供する動作環境(コンピュータ、オペレーティングシステムおよびそれ以外のソフトウェアとハードウェア)。

マージ
複数の SAS データセットからオブザベーションを読み込んで結合し、新しい SAS データセットの単一のオブザベーションにする処理。

マクロ起動
マクロ呼び出しの別称。

マクロ機能
SAS プログラムの拡張やカスタマイズに使用できる Base SAS ソフトウェアのコンポーネント。マクロ機能を使用すると、一般的なタスクを実行するのに入力する必要のあるテキスト量を削減できます。マクロ機能は、マクロプロセッサとマクロプログラミング言語で構成されています。

マクロ言語
マクロプロセッサとの通信に使用されるプログラミング言語。

マクロ変数
SAS マクロプログラミング言語に含まれる変数。マクロ変数の値は、ユーザーが変更しない限り一定のままの文字列です。マクロ変数はシンボリック変数とも呼ばれます。

マクロ変数参照
アンパサンドが前に付くマクロ変数名(&name)。マクロプロセッサによって、マクロ変数参照が指定マクロ変数の値に置換されます。

マスタデータセット
更新操作において、更新対象の情報を含むデータセット。

マッチマージ
BY 変数の値に従って 2 つ以上の SAS データセットのオブザベーションを結合するプロセス。

明示添字配列
DATA ステップにおいて、有効な SAS 名、変数または一時データ要素の数の参照、およびオプションの配列要素リストで構成される配列。明示添字配列の要素を参照する場合は、配列要素の添字を指定する必要があります。

明示添字配列参照
明示添字配列で処理される要素の記述です。

メンバタイプ
SAS ファイルに格納された情報の種類を識別する SAS 名。メンバタイプには、ACCESS、AUDIT、DMBD、DATA、CATALOG、FDB、INDEX、ITEMSTOR、MDDB、PROGRAM、UTILITY、VIEW があります。

文字値
アルファベット、0 から 9 までの数字、およびその他の特殊文字を含む値。

文字関数
関数の一種。文字列の操作、比較、評価、分析のいずれかが行えます。

文字出力形式
特定のパターンを使用して文字データ値を書き出すよう SAS に指示する命令セット。

文字定数
SAS ステートメントにおいて引用符で囲まれた文字列で、変数の名前ではなく固定値を示します。許容される最大文字数は 32,767 です。文字定数は文字リテラルと呼ばれることもあります。

文字入力形式
特定のパターンを使用して文字データ値を文字変数に読み込むよう SAS に指示する命令セット。

文字変数
値がアルファベットおよび特殊文字ならびに数字から成る変数。

文字リテラル
文字定数の別称。

文字列
文字列(character string)を参照。

文字列(character string)
1 つ以上の連続した英数字またはその他キーボード文字、あるいはその両方。

ユーザー定義の出力形式
FORMAT プロシジャを使用するか、SAS/TOOLKIT ソフトウェアで C、PL/I、FORTRAN、IBM 370 の各アセンブラ言語を使用して定義する出力形式。

ユーザー定義の入力形式
FORMAT プロシジャを使用するか、SAS/TOOLKIT ソフトウェアで C、PL/I、FORTRAN、IBM 370 の各アセンブラ言語を使用して定義する入力形式。

要約テーブル
データセットの情報の簡明な概要を示す出力。

ライブラリ参照名
ライブラリ参照名(libref)を参照。

ライブラリ参照名(libref)
SAS ライブラリの場所に関連付けられる SAS 名。たとえば、MYLIB.MYFILE という名前では、MYLIB がライブラリ参照名で、MYFILE が SAS ライブラリ内のファイルです。

ラインホールド指定子
INPUT ステートメントと PUT ステートメントで使用される特殊記号。これにより、以降の処理のために入力バッファまたは出力バッファにレコードを保持できます。ラインホールド指定子には、後置アットマーク(@)および二重後置アットマーク(@@)が含まれます。

ラインモード
対話型ラインモードを参照。

ラベル
変数に関連付けられた説明テキスト。デフォルトでは、このテキストは、LABEL=オプションで事前に割り当てられた変数またはラベルの名前です。

リスト出力
PUT ステートメントで文字列または変数を、文字列や値の配置される場所の明示的な指示なしに、指定する出力スタイル。

リスト入力
INPUT ステートメントでカラム位置ではなく変数名を指定する入力スタイル。リスト入力では、入力レコードをスキャンして、少なくとも 1 つのブランクかまたはその他の区切り文字で区切られたデータ値を探します。

リテラル
固定値を示す数字または文字列。

連結
2 つ以上の要素のコンテンツの端と端を結合して、別の要素を形成すること。要素の例としては、文字値、テーブル、外部ファイル、SAS データセット、SAS ライブラリがあります。

ログ
SAS ログを参照。

論理演算子
複数の比較演算を結び付けるために式で使用される演算子。論理演算子は AND、OR および NOT です。

割り当てステートメント
式を評価して結果を変数に格納する DATA ステップステートメント。

キーワード

_

_ (アンダースコア), SAS 名 6
_DSEMTR コード 337
_DSENMR コード 337
_ERROR_変数 398
_IORC_自動変数 337
_N_変数 398
_SOK コード 337

,

, (カンマ), 入力データ 61

;

; (セミコロン)
  ステートメント 5
  データの終わりインジケータ 45

:

: (コロン)
  フォーマット修飾子 64
  文字の比較 162
:(コロン)
  フォーマット修飾子 64

!

!! (感嘆符), 連結演算子 141

/

/ (スラッシュ), カラムポインタコントロール
  次の行へのポインタの強制移動 79
  説明 69, 88
/ (スラッシュ), 列ヘッダーの分割 503

.

. (ピリオド)
  欠損値 30, 136
  入力形式名 61
  入力データ 60

'

' (引用符)
  変数インジケータ 133
  リテラル文字 133

[

[ ] (大かっこ), STYLE=オプション 480

{

{ } (中かっこ), STYLE=オプション 481

@

@ (後置@)
  生データレコードの読み込み 72
  固定された出力行の解除 588
  出力行の書き込み 587, 597
  説明 89
@@ (後置@@)
  DATA ステップの実行 74
  説明 89
  定義 74
@n, カラムポインタコントロール 69, 88
@n, ポインタコントロール 591
  関連項目: カラムポインタコントロール

$

$ (ドル記号)
  入力データ 60
  変数名 133
  文字変数の定義 43

数字

#n, カラムポインタコントロール 69, 88
#n, ラインポインタコントロール
  DATA ステップの実行 82
  入力変数のスキップ 81

%

%INCLUDE ステートメント
  説明 14
  対話型ラインモード 12
%LET ステートメント 459
%LIST ステートメント 12
%PUT マクロステートメント 380

+

+n, カラムポインタコントロール 69, 88

=

= (等号)
  線を引く 509
  要約テーブルラベルの定義 479

|

|| (縦棒), 連結演算子 141

1

1969 to 2010 オプション
  目盛値 526
1 次元要約テーブル 468
1 レベル名 671, 672

2

2 次元要約テーブル 469
2 レベル名 671, 673

3

3 次元グラフ 561
3 次元要約テーブル 470

A

A (後)コマンド 754
ACCESS メンバタイプ 671
ACROSS 変数 491, 505, 507
ANALYSIS 変数 491, 498
APPEND プロシジャ 276
  SAS データセットの連結 271
  SET ステートメントとの比較 275
  説明 276
ARRAY ステートメント
  説明 220
  配列の定義 217
ASCII 照合順序 193
AUTOEXPAND コマンド 769
  関連項目: TREE コマンド
AUTOSYNC コマンド 769

B

B (前)コマンド 744, 754
BACKWARD コマンド 741, 754
BFIND コマンド 105
BLANKLINE=オプション
  PROC PRINT ステートメント 457
BLANKLINE オプション, PROC PRINT 450
BLOCK ステートメント, CHART プロシジャ 575
  3 次元グラフ 561
  ブロックチャート 548
BODY=オプション 660
BOTTOM コマンド 741, 753
BOX オプション
  PLOT ステートメント 536
  プロット周囲の外枠罫線 529
BREAK コマンド
  DATA ステップデバッガ 801
BREAK ステートメント, REPORT プロシジャ 511
  区切り行 508
BYE コマンド
  SAS セッションの終了 724
  ウィンドウの管理 741
  説明 728
BY 値 280
BY グループ
  合計 202
  定義 280
BY ステートメント
  FIRST.変数と LAST.変数 194
  PRINT プロシジャ 458
  SAS データセットのインタリーブ 283
  SAS データセットの更新 331, 336
  SAS データセットの変更 345
  SAS データセットのマージ 314
  SAS データセットのマッチマージ 294
  SAS ログへの書き込み 596
  SORT プロシジャ 459, 661
  UNIVARIATE プロシジャ 662
  オブザベーションのグループ化 183, 444
  グループごとに値を印刷 593
  グループの小計の計算 437, 441
  グループの小計の識別 439
  最初のオブザベーションまたは最後のオブザベーションの検索 188
  出力ファイルへの書き込み 596
  詳細レポート 437
BY 変数
  SAS データセット更新のための選択 316
  重複 340
  定義 280

C

C (コピー)コマンド 754
CALCULATE コマンド
  DATA ステップデバッガ 803
CAPS コマンド 746, 754
CATALOG プロシジャ 676
CATALOG メンバタイプ 671
CCL (小文字変換)コマンド 747
CCU (大文字変換)コマンド 747
CENTER オプション 624
  DEFINE ステートメント 512
  列の配置 502
CFILL=オプション 573
CFRAME=オプション 573
CFREQ オプション
  HBAR ステートメント 576
  横棒グラフ 546
CGRID=オプション
  HISTOGRAM ステートメント 577
  ヒストグラム 565
CHANGE コマンド 743, 754
CHANGE ステートメント 702
CHART プログラム
  HBAR ステートメント, 横棒グラフ 546
CHART プロシジャ 540
  BLOCK ステートメント, 3 次元グラフ 561
  BLOCK ステートメント, 説明 575
  BLOCK ステートメント, ブロックチャート 548
  HBAR ステートメント, 説明 575
  PIE ステートメント, 円グラフ 549
  PIE ステートメント, 説明 575
  PROC CHART ステートメント 574
  VBAR ステートメント, 説明 575
  VBAR ステートメント, 縦棒グラフ 544
  度数グラフの作成 543
CHILD コマンド
  コンテンツペインのオン/オフの切り替え 762
  説明 769
CITY データセット 92, 792
CL (小文字変換)コマンド 747
CLASSLEV ステートメント
  TABULATE プロシジャ 661
CLASS ステートメント
  TABULATE プロシジャ 485, 661
  UNIVARIATE プロシジャ 576, 662
  比較ヒストグラム 572
  要約テーブル分類変数の指定 465
CLEAR コマンド
  ウィンドウのクリア 387, 741, 754
  説明 769
CLIMATE.HIGHTEMP データセット 694, 706, 798
CLIMATE.LOWTEMP データセット 694, 706, 798
COLOR コマンド 742
COLS コマンド 746
COLUMN ステートメント
  ODS 出力のカスタマイズ 651
  REPORT プロシジャ 512
  TEMPLATE プロシジャ 651, 662
  レポートのレイアウト 69, 495
COLWIDTH=オプション
  説明 510
  列幅 502
COMPUTED 変数 492
CONTENTS=オプション 660
CONTENTS ステートメント 690
  SAS データセットコンテンツの説明 92
  SAS データセットのリスト 685
  説明 104, 690
CONTENTS プロシジャ 676
COPY ステートメント
  SAS データセットの移動 711
  SAS データセットのコピー 706
  説明 715
COPY プロシジャ 676
CPERCENT オプション
  HBAR ステートメント 576
  横棒グラフ 546
CU (大文字変換)コマンド 747
CURSOR コマンド 742
CUT コマンド 744

D

D (削除)コマンド 746, 754
DATA_NULL ステートメント
  DATA ステップからのレポートの書き込み 584
  説明 597
DATA=オプション 14
  PROC CHART ステートメント 575
  PROC PLOT ステートメント 536
  PROC PRINT ステートメント 457
  PROC REPORT ステートメント 511
  PROC TABULATE ステートメント 485
  PROC UNIVARIATE ステートメント 576
  説明 14
  要約テーブルの作成 465
DATALINES ステートメント
  SAS データセット 45
  説明 491
  対話型ラインモードでの SAS プログラムの実行 725
DATASETS プロシジャ 678
  CHANGE ステートメント 702

CONTENTS ステートメント, SAS データセットのリスト 685
CONTENTS ステートメント, 説明 690
COPY ステートメント, SAS データセットの移動 711
COPY ステートメント, SAS データセットのコピー 706
COPY ステートメント, 説明 715
DELETE ステートメント, SAS データセットの削除 713
DELETE ステートメント, 説明 715
EXCLUDE ステートメント, SAS データセットの移動 712
EXCLUDE ステートメント, SAS データセットのコピー 710
EXCLUDE ステートメント, 説明 715
FORMAT ステートメント, SAS データセット変数属性の再フォーマット 696
FORMAT ステートメント, 説明 703
LABEL ステートメント, SAS データセットラベルの削除 699
LABEL ステートメント, SAS データセットラベルの変更 699
LABEL ステートメント, SAS データセットラベルの割り当て 699
LABEL ステートメント, 説明 703
MODIFY ステートメント, SAS データセット変数属性の再フォーマット 696
MODIFY ステートメント, SAS データセット変数属性の名前の変更 696
MODIFY ステートメント, SAS データセット変数属性の変更 695
MODIFY ステートメント, SAS データセットラベルの削除 699
MODIFY ステートメント, SAS データセットラベルの変更 699
MODIFY ステートメント, SAS データセットラベルの割り当て 699
MODIFY ステートメント, 説明 702
PROC DATASETS ステートメント, KILL オプション 715
PROC DATASETS ステートメント, SAS ライブラリの管理 677
PROC DATASETS ステートメント, 説明 104, 678, 690
PROC DATASETS ステートメント, ディレクトリリスト 682
RENAME ステートメント, SAS データセットの名前の変更 694
RENAME ステートメント, SAS データセット変数属性の名前の変更 696
RENAME ステートメント, 説明 703
SAS データセットのリスト 684
SAS ライブラリの管理 676
SAVE ステートメント, SAS データセットの削除 714
SAVE ステートメント, 説明 715
SELECT ステートメント, SAS データセットの移動 712
SELECT ステートメント, SAS データセットのコピー 710
SELECT ステートメント, 説明 715
定義 676
DATA ステートメント 14
SET ステートメントとの比較 102
出力 601
説明 14, 49
変数の削除/保持 104
DATA ステップ 4
オブザベーション, グローバル変更 111
オブザベーション, 選択的変更 112
コンパイル済みプログラムファイル 671
コンパイルフェーズ 35
実行フェーズ 36
重複した BY 変数 340
定義 4
ディスクリプタ情報 36
入力バッファ 36
プログラムデータベクトル 36
プロセス概要 38
変数, 記憶域 116
変数, 効率的な使用法 114
変数, 作成 111
変数, 長さの定義 116
変数, 変更 113
例 37
レポートの生成 584
割り当てステートメント 111
DATA ステップデバッガ 403
DATA ステップ実行の開始 808
DATA ステップ実行の再開 808
DATA ステップ実行の連続レコード 813
DO ループのデバッグ 417
新たな変数値を割り当てる 811
一度に 1 つのステートメントを実行する 812
ウィンドウ 405
ウィンドウ制御の切り替え 813
ウォッチステータスの削除 804
項目のリスト 810
コマンドの入力 405
コマンドのヘルプ 809
コマンドを Enter キーに割り当てる 806
コマンドをファンクションキーに割り当てる 406
式 405
式の評価 803
実行の中断 801, 814
終了 811

出力形式 412
中断したプログラムの再開 809
デバッガセッション 404
デバッグツールとしてのマクロ 406
フォーマット指定された変数値 418
ブレークポイントの削除 804
プログラム行へのジャンプ 809
変数値の表示 807
変数属性の表示 805
マクロ機能 406
マクロを用いた DATA ステップの生成 406
マクロを用いたコマンドのカスタマイズ 406
例 407
DATA メンバタイプ 670
DATE7.入力形式
説明 226, 240
年の長さ 228
DATE9.入力形式
説明 226, 240
年の長さ 228
DATE オプション 624
DBMS ファイル, SAS データセットの作成 46
DEBUGGER LOG ウィンドウ 405
DEBUGGER SOURCE ウィンドウ 405
DEFINE ステートメント
GROUP 変数の定義 498
ODS 出力のカスタマイズ 651
REPORT プロシジャ 512
TEMPLATE プロシジャ 651, 662
列幅とスペース 502
レポート項目のフォーマット 504
レポートのレイアウト 69, 496, 499
DELETESELS コマンド 769
DELETE コマンド
DATA ステップデバッガ 804
DELETE ステートメント 117
関連項目: オブザベーション, サブセット化
IF ステートメントとの比較 171
SAS データセットの削除 713
オブザベーションの削除 117, 169
説明 117, 178, 715
DESCENDING オプション
説明 513
レポートレイアウト 499
DESCRIBE コマンド
DATA ステップデバッガ 805
DESELECT_ALL コマンド 769
DETAILS コマンド
エクスプローラウィンドウのカスタマイズ 782
説明 769
DISCRETE オプション
BLOCK ステートメント 575
HBAR ステートメント 575
PIE ステートメント 575
VBAR ステートメント 575
離散値と連続値 555
DISPLAY 変数 492
DLGFONT コマンド
説明 785
フォントウィンドウを開く 784, 786
DLGPREF コマンド
SAS セッションのカスタマイズ 780
出力形式の設定 760
説明 785
プリファレンスウィンドウを開く 786
DLM=オプション 491
DMFILEASSIGN コマンド
説明 769
ファイルショートカットの変更 752
DMOPTLOAD コマンド
システムオプションの取得 780
説明 769
DMOPTSAVE コマンド
システムオプションの保存 780
説明 770
DMSEXP オプション 724
DMS オプション 728
DOL オプション
RBREAK ステートメント 514
等号(=), 線を引く 509
DOUBLE オプション, PROC PRINT
ダブルスペースの LISTING レポート 450
DO グループ 215
関連項目: 配列処理
反復 DO ループ 218
DO ステートメント
DO グループ 215
説明 221
反復 DO ループ 218
DO ループ 218
関連項目: 配列処理
デバッグ 417
DROP=オプション 103
DATA ステートメントと SET ステートメントの比較 102
KEEP=オプションとの比較 99
効率性 102
説明 103
選択した変数の削除 98
DROP ステートメント 104
説明 104
変数の削除 98
DSD オプション
リスト入力 55

E

EBCDIC 照合順序 194
EDOPT コマンド 786
ELSE ステートメント
  オブザベーションの選択 151
  説明 165
END=オプション
  最後のオブザベーションの決定 201, 357
  説明 210, 360
ENDSAS コマンド
  SAS セッションの終了 724
  説明 728
  対話型ラインモードの終了 726
ENDSAS ステートメント
  SAS セッションの終了 724
  説明 728
END コマンド 741
END ステートメント
  DO グループ 215
  TEMPLATE プロシジャ 651, 662
  説明 221
  反復 DO ループ 218
ENTER コマンド
  DATA ステップデバッガ 806
ERRORS=オプション
  エラーメッセージを非表示にする 377
  説明 380
EXAMINE コマンド
  DATA ステップデバッガ 807
EXCLUDE ステートメント
  SAS データセットの移動 712
  SAS データセットのコピー 710
  説明 715
EXOPTS コマンド 786
EXPFIND コマンド
  説明 769
  ファイルの検索 749
EXPLORER コマンド
  エクスプローラウィンドウを開く 733
  説明 728

F

FILENAME ステートメント 31
  外部ファイルのファイル参照名 47
  説明 49
FILE コマンド 388
  アウトプットウィンドウの格納 388
  プログラムエディタの保存 758
  ログウィンドウの格納 388
FILE ステートメント
  SAS 出力ファイルへのレポートの書き込み 589
  説明 597
FIND コマンド
  説明 770
  テキストの検索と変更 743, 754
FIRST.変数
  最初のオブザベーションの検索 188
  説明 194
FIRSTOBS=オプション 104
  最初のオブザベーションを指す 95
  説明 104
FLOWOVER オプション 89
  説明 89
  予想外のレコードの終わり 86
FOOTNOTES オプション
  FILE ステートメント 597
  SAS 出力ファイルへのレポート書き込み 590, 595
FOOTNOTE ステートメント
  PRINT プロシジャ 458
  説明 623
  プロシジャ出力のフットノート 606, 609
  レポートのフットノート 446
FORMAT=オプション
  DEFINE ステートメント 513
  INSET ステートメント 578
  PROC TABULATE ステートメント 485
  ヒストグラム 570
  要約テーブルのフォーマット 465
  レポート項目のフォーマット 504
FORMAT ステートメント
  PRINT プロシジャ 458
  SAS データセット変数属性の再フォーマット 696
  永久日付出力形式 232
  詳細レポートのフォーマット 435
  チャートのフォーマット 561, 572, 578
  日付のフォーマット 240
  変数属性の再フォーマット 696
  変数のフォーマット 703
  レポート項目のフォーマット 504
FORMAT プロシジャ 622
FORWARD コマンド 741, 754
FRAME=オプション 660
FREQ オプション
  HBAR ステートメント 576
  横棒グラフ 546

G

GOPTIONS ステートメント
  説明 578
  ヒストグラム 563
GO コマンド
  DATA ステップデバッガ 808
GRADES データセット 541, 796
GRID オプション
  HISTOGRAM ステートメント 577
  ヒストグラム 565

GROUP=オプション
BLOCK ステートメント 575
HBAR ステートメント 575
VBAR ステートメント 575
GROUP 変数 492, 498

H

HAXIS=オプション
PLOT ステートメント 536
目盛値 526
HBAR ステートメント
CHART プロシジャ 575
横棒グラフ 546
HEADER=オプション 578
FILE ステートメント 623
INSET ステートメント 578
特定のカラムのヘッダー 616
ヒストグラム 570
HEADER ステートメント, TEMPLATE プロシジャ 651, 662
ODS 出力のカスタマイズ 651
HEADLINE オプション 511
PROC REPORT ステートメント 511
列ヘッダー 503
HEADSKIP オプション 511
PROC REPORT ステートメント 511
列ヘッダー 503
HELP コマンド 736
DATA ステップデバッガ 809
HIGHLOW データセット 520, 795
HISTOGRAM ステートメント, UNIVARIATE プロシジャ 577
ヒストグラム 562
HOFFSET=オプション
HISTOGRAM ステートメント 577
ヒストグラム 568
HPCT=オプション
PROC PLOT ステートメント 536
同一ページの複数のプロット 532
HPERCENT=オプション
PROC PLOT ステートメント 536
同一ページの複数のプロット 532
HSCROLL コマンド 741
HTML 出力 630

I

I (挿入)コマンド 754
ID ステートメント
PRINT プロシジャ 458
キー変数の強調 427
詳細レポート 439
IF-THEN/ELSE ステートメント
オブザベーションの選択的変更 112
説明 117, 359
IF-THEN ステートメント
オブザベーションの選択 149
説明 165
IF ステートメント 179
関連項目: オブザベーション, サブセット化
DELETE ステートメントとの比較 171
オブザベーションの受け入れ 171
オブザベーションの結合 356
オブザベーションの削除 169
説明 179, 359
IN=オプション 359
COPY ステートメント 715
SAS データセットとライブラリの移動 711
説明 359
複数の SAS データセットからのオブザベーション 351
IN=データセットオプション
データのマージ 305
INCLUDE コマンド 759
INDEX 関数
説明 166
文字列の検索 164
INFILE DATALINES ステートメント 491
INFILE ステートメント
SAS データセットの作成 45
説明 49, 89
予想外のレコードの終わり 86
INPUT ステートメント
1 オブザベーションにつき複数レコード 78
新しいレコードの強制読み込み 79
カラム入力 42, 57
説明 49
データ行のスキップ 81
日付変数の読み取り 226, 240
フォーマット入力 42, 60
複数のステートメント 78
複数の入力スタイルの使用 65
変数の定義 43
リスト入力 42, 53
レコードの 2 回読み込み 72
レコードの保持 74
INSET ステートメント, UNIVARIATE プロシジャ 577
ヒストグラムの要約統計量 570
item-store ステートメント 768

J

JC (中央揃え)コマンド 745
JJC (中央揃え)コマンド 745
JJL (左揃え)コマンド 745
JJR (右揃え)コマンド 745
JL (左揃え)コマンド 745

JR (右揃え)コマンド 745
JUMP コマンド
  DATA ステップデバッガ 809

K
KEEP=オプション 104
  DATA ステートメントと SET ステートメントの比較 102
  DROP=オプションとの比較 99
  効率性 102
  説明 104
  選択した変数の保持 97
KEEP ステートメント 104
  説明 104
  変数の保持 97
KEYS ウィンドウ 738
KEYS コマンド 738
KEYWORD ステートメント
  TABULATE プロシジャ 661
KILL オプション
  PROC DATASETS ステートメント 715
  SAS データライブラリメンバの削除 714

L
LABEL オプション
  PROC PRINT ステートメント 457
  レポートの列ラベル 448
LABEL ステートメント 458
  PLOT プロシジャ 536
  PRINT プロシジャ 458
  SAS データセットラベルの削除 699
  SAS データセットラベルの変更 699, 703
  SAS データセットラベルの割り当て 699
  詳細レポートの列ヘッダー 458
  プロシジャ出力の変数ラベル 608, 609, 623
  プロット軸ラベル 525, 536
  レポートの列ヘッダー 448
LARGEVIEW コマンド
  アイコンサイズの設定 782
  説明 770
LAST.変数
  最後のオブザベーションの検索 188
  説明 194
LEFT オプション 502
LEFT 関数
  説明 145
  文字値の位置調整 139
LEFT コマンド 741, 754
LENGTH ステートメント
  SAS データセットの連結 268
  数値変数の長さ 116
  精度の損失 128
  説明 129
  配置 135
  変数の長さの定義 116, 118, 145
  文字変数の長さ 135
LEVELS=オプション
  BLOCK ステートメント 575
  HBAR ステートメント 575
  PIE ステートメント 575
  VBAR ステートメント 575
  中間点の数 554
LGRID=オプション
  HISTOGRAM ステートメント 577
  ヒストグラム 565
LIBNAME ステートメント 31
  SAS ライブラリへのライブラリ参照名の割り当て 668
  説明 49
LIBRARY=オプション
  構文 690
  ディレクトリリスト 682
LINESIZE=オプション
  出力行サイズ 611, 720
  説明 624, 728
LINESLEFT=オプション
  FILE ステートメント 623
  改ページ 619
LIST コマンド
  DATA ステップデバッガ 810
LIST ステートメント 372, 379
LOG=オプション
  SAS ログの出力先の指定 386
  説明 389
LOG ウィンドウ
  DATA ステップデバッガ 813
LOG コマンド 769
LPI=オプション
  PROC CHART ステートメント 575
  円グラフ 549

M
M (移動)コマンド 754
MARK コマンド 744
MAX コマンド 742
MEMTYPE=オプション
  PROC DATASETS ステートメント 690
  ディレクトリリスト, メンバタイプ 684
MERGE ステートメント
  MODIFY ステートメントと UPDATE ステートメントとの比較 250
  SAS データセットの作成 45
  SAS データセットのマージ 288
  UPDATE ステートメントとの比較 324
  欠損値 326

説明 314
複数のオブザベーションを含む BY グループ 327
MIDPOINTS=オプション
BLOCK ステートメント 575
HBAR ステートメント 575
HISTOGRAM ステートメント 577
PIE ステートメント 575
VBAR ステートメント 575
数値変数の中間点 551
ヒストグラムの中間点 568
文字変数の中間点 557
MISSING=オプション
出力レポートの欠損値 621
説明 624
MISSING=システムオプション 621
MISSING オプション
CLASS ステートメント 485
PROC TABULATE ステートメント 485
要約テーブルでの欠損値 466
MISSOVER オプション 89
説明 89
予想外のレコードの終わり 86, 87
MM (移動)コマンド 744
MMDDYY10.入力形式
説明 240
年の長さ 228
MMDDYY8.入力形式
説明 240
年の長さ 228
MODIFY ステートメント 346
MERGE ステートメントと UPDATE ステートメントとの比較 250
SAS データセットの作成 45
SAS データセット変数属性の再フォーマット 696
SAS データセット変数属性の名前の変更 696
SAS データセット変数属性の変更 695
SAS データセットラベルの削除 699
SAS データセットラベルの変更 699
SAS データセットラベルの割り当て 699
欠損値 327, 343
説明 346, 702
MOVE オプション
COPY ステートメント 715
SAS データセットとライブラリの移動 711

N

N=オプション
PROC PRINT ステートメント 457
NEW オプション
SAS ログの出力先の指定 386
説明 389
NEXT コマンド 741
NOCENTER オプション
出力の中央揃え 611
説明 624
NODATE オプション
説明 624
日付値 612
NODMS オプション
SAS プログラムの実行 725
説明 728
NODS オプション
CONTENTS ステートメント 690
ディレクトリリスト 687
NOFRAME オプション
INSET ステートメント 578
インセットテーブルのフレームの非表示 573
NOLEGEND オプション
PROC PLOT ステートメント 536
プロット凡例の削除 529
NONOTES オプション
システム情報を非表示にする 376, 377
説明 380
NONUMBER オプション
説明 624
ページ番号付け 611
NOOBS オプション
PROC PRINT ステートメント 458
オブザベーションの列の非表示 426
NOPRINT オプション
PROC UNIVARIATE ステートメント 576
統計量テーブルの非表示 563
NOSOURCE オプション
SAS ステートメントを非表示にする 376, 377
説明 380
NOSTAT オプション
HBAR ステートメント 576
横棒グラフ 546
NOTEPAD ウィンドウ 769
説明 769
開く 757
NOTEPAD エディタ 757
NOTEPAD コマンド
NOTEPAD ウィンドウを開く 757
説明 769
NOTESUBMIT コマンド 757
NOTES オプション
システム情報を非表示にする 376
説明 380
NOTES コマンド
NOTEPAD ウィンドウを開く 757
説明 769
NOTITLES オプション

FILE ステートメント 597
SAS 出力ファイルへのレポート書き込み 590
NOVERBOSE オプション 786
NOWINDOWS オプション
REPORT ウィンドウのバイパス 493
説明 511
NROWS=オプション 573
NUMBERS コマンド 738, 746
NUMBER オプション 624

O

OBS=オプション 104
オブザベーションの列のラベル付け 425
最後のオブザベーションを指す 96
説明 104
OBS オプション
PROC PRINT ステートメント 458
ODS 626
機能 602
データ, 定義 9, 628
テーブルテンプレート 628
ODS _ALL_ CLOSE ステートメント 660
ODS EXCLUDE ステートメント
ODS 出力オブジェクトの除外 643
説明 659
ODS HTML CLOSE ステートメント 660
ODS HTML ステートメント 660
説明 624
ODS LISTING CLOSE ステートメント 660
ODS LISTING ステートメント 660
説明 623
ODS OUTPUT CLOSE ステートメント 660
ODS OUTPUT ステートメント
SAS データセットの作成 645
説明 660
ODS PATH ステートメント 768
ODS PDF 660
ODS PDF ステートメント 660
ODS POWERPOINT 660
ODS POWERPOINT ステートメント 660
ODS PRINTER CLOSE ステートメント 660
ODS RTF 660
ODS RTF CLOSE ステートメント 660
ODS RTF ステートメント 660
ODS SELECT ステートメント
ODS 出力オブジェクトの選択 643
説明 660
ODS TRACE ステートメント
ODS 出力オブジェクトの識別 642
説明 661
ODSRESULTS コマンド
結果ウィンドウを開く 762
説明 769
ODSTEMPLATE コマンド 766
ODS 出力 629
HTML, Web ブラウザ 630
PostScript 出力, 高解像度プリンタ 638
PowerPoint 出力, Microsoft PowerPoint 用 639
RTF 出力, Microsoft Word 用 639
SAS データセット, 作成 645
形式, リスト 8
定義 10, 629
フォーマット, 選択 657
リンクの格納 629
ODS 出力, カスタマイズ 647
関連項目: 出力, カスタマイズ
SAS ジョブレベル 647
スタイルテンプレート 647
テーブルテンプレートの使用 650
ODS 出力オブジェクト 628
識別 642
除外 643
選択 643
定義 9, 628
ODS 出力先
定義 9, 628
開く 629
ODS 出力先, 閉じる 629
ODS _ALL_ CLOSE ステートメント 660
ODS HTML CLOSE ステートメント 660
ODS LISTING CLOSE ステートメント 660
ODS OUTPUT CLOSE ステートメント 660
ODS PRINTER CLOSE ステートメント 660
ODS RTF CLOSE ステートメント 660
ODS テーブル定義
定義 9
ODS テーブルテンプレート
ODS 出力のカスタマイズ 650
ODS へのリンク, 格納 629
OL オプション 512
OPTIONS コマンド
SAS オプションウィンドウを開く 774
SAS システムオプションウィンドウを開く 786
説明 785
定義 727
OPTIONS ステートメント
SAS セッションのカスタマイズ 774
出力のカスタマイズ 610
説明 624, 728

OPTIONS プロシジャ 785
SAS システムオプションのリスト 727
システムオプションのリスト 774
説明 785
ORDER=オプション
CLASS ステートメント 485, 576
DEFINE ステートメント 513
PROC TABULATE ステートメント 485
分類変数の並べ替え 483, 573
レポートレイアウト 496, 499
ORDER 変数 492
OUT.ERROR1 データセット 394
OUT.ERROR2 データセット 394
OUT.ERROR3 データセット 394
OUT.SAT_SCORES3 データセット 384
OUT.SAT_SCORES4 データセット 384
OUT.SAT_SCORES5 データセット 384
OUT=オプション
COPY ステートメント 715
SAS データセットとライブラリの移動 711
Output Delivery System
参照項目: ODS
OUTPUT コマンド 769
OUTPUT ステートメント 173
関連項目: SAS データセット, オブザベーションの書き込み
MODIFY ステートメント 346
説明 173, 179
配置 175
OVERLAY オプション
PLOT ステートメント 537
同じ軸の複数の組み合わせの変数 534

P

PAGE=オプション 660
PAGEBY ステートメント
PRINT プロシジャ 458
詳細レポートのオブザベーションのグループ化 444
PAGENO=オプション
説明 624
ページ番号付け 611
PAGESIZE=オプション
説明 624
ページサイズ 611
PAGE ステートメント
説明 380
PASTE コマンド 744
PERCENT オプション
HBAR ステートメント 576
横棒グラフ 546
PGM コマンド 769
PIE ステートメント, CHART プロシジャ 575
円グラフ 549
PLOT ステートメント, PLOT プロシジャ
1 つの組み合わせの変数のプロット 536
説明 536
複数の組み合わせの変数のプロット 530
プロット記号 528
PLOT プロシジャ 519
関連項目: プロット
LABEL ステートメント, 説明 536
LABEL ステートメント, プロット軸ラベル 525
PLOT ステートメント, 1 つの組み合わせの変数のプロット 536
PLOT ステートメント, 説明 536
PLOT ステートメント, 複数の組み合わせの変数のプロット 530
PLOT ステートメント, プロット記号 528
PROC PLOT ステートメント, 説明 536
PROC PLOT ステートメント, 同一ページの複数のプロット 532
TITLE ステートメント 537
PMENU コマンド
説明 728, 770
メニューの表示 720, 737
POSITION=オプション 573
INSET ステートメント 578
ヒストグラム 570
PostScript 出力 638
PowerPoint 出力 639
PRECIP.RAIN データセット 694, 706, 799
PRECIP.SNOW データセット 694, 706, 799
PREVWIND コマンド 741
PRINT=オプション
SAS ログの出力先の指定 388
説明 389
PRINTTO プロシジャ 389
SAS ログの出力先の指定 386
説明 389
プロシジャ出力先の指定 385
PRINT コマンド 387, 770
PRINT プロシジャ 457
関連項目: レポート
PROC CHART ステートメント 574
PROC DATASETS ステートメント
KILL オプション 715
SAS ライブラリの管理 677
説明 104, 678, 690
ディレクトリリスト 682
PROC PLOT ステートメント
説明 536

同一ページの複数のプロット 532
PROC PRINT ステートメント 457
PROC REPORT ステートメント
説明 510
列幅とスペース 502
PROC SORT ステートメント 661
SORT プロシジャ 661
詳細レポートの並べ替え 428
説明 194, 459
PROC TABULATE ステートメント 485
ODS 出力 661
PROC TEMPLATE ステートメント 662
PROC UNIVARIATE ステートメント
ODS 出力 662
説明 576
PROGRAM メンバタイプ 671
PUT ステートメント 380
SAS 出力ファイルへのレポート 584
説明 380, 597

Q

QUIT コマンド
DATA ステップデバッガ 811
QUIT ステートメント 678

R

RBREAK ステートメント, REPORT プロシジャ 513
区切り行 508
RCHANGE コマンド 105
RECALL コマンド 741, 754
REFRESH コマンド 770
REGEDIT コマンド 769
関連項目: SAS レジストリ, 編集
関連項目: SAS レジストリエディタ
SAS レジストリエディタウィンドウを開く 786
SAS レジストリの編集 735
出力形式の設定 761
説明 769
REGEDIT ステートメント 785
REGISTRY プロシジャ 786
SAS レジストリの編集 776
説明 786
RENAME=オプション 306
RENAMESELS コマンド 770
RENAME ステートメント
SAS データセットの名前の変更 694
SAS データセット変数属性の名前の変更 696
説明 703
REPLACE ステートメント 346
REPORT プロシジャ 510
BREAK ステートメント 511
COLUMN ステートメント 512
DEFINE ステートメント 512
PROC REPORT ステートメント 510
RBREAK ステートメント 513
RESET コマンド 746
RETAIN ステートメント
値の保持 207
説明 210
RFIND コマンド 105
RIGHT オプション
DEFINE ステートメント 513
列の配置 502
RIGHT コマンド 741, 754
ROUND 関数
数字を丸める 124
説明 128
RTF 出力 639
RUN ステートメント
説明 14, 728
対話型ラインモード 12

S

SAS Supervisor
参照項目: デバッグ, SAS Supervisor の使用
SAS System
基本ソフトウェアコンポーネント 4
データ管理機能 4
データ分析ユーティリティ 6
定義 3
ラインモードでの起動 725
SAS/ASSIST 11
SASColor ウィンドウ
説明 786
開く 786
SASCOLOR コマンド
SASColor ウィンドウを開く 786
ウィンドウのカスタマイズ 742
SASCOLOR ステートメント 785
SAS ウィンドウ 739
カスタマイズ 742
管理 741
スクロール 753
開く 739
SAS ウィンドウ環境 10
関連項目: SAS セッション
関連項目: テンプレートウィンドウ
1 つ上のレベルへ移動 770
SAS ウィンドウ, カスタマイズ 742
SAS ウィンドウ, 管理 741
SAS ウィンドウ, スクロール 741
SAS ウィンドウ, 開く 739
アイコン, 大きい 770
アイコン, 小さい 770
起動 724

行コマンド 738
行番号, オン/オフの切り替え 738
項目の詳細, オン/オフの切り替え 769
項目の選択 770
項目の選択解除 769
コマンド行のコマンド 736
コマンドの種類 736
コンテンツの更新 770
システムオプション設定, 保存 770
システムオプション設定, ロード 769
出力ポインタ, 名前変更 770
ツリービュー, オン/オフの切り替え 722, 770
ツリービュー, 展開 769
定義 10
ファンクションキー, コマンドの割り当て 738
プルダウンメニュー 737
プログラムの実行 722
マウス操作に相当するキーボード操作 732
ライブラリの割り当てに関する問題 735
リスト出力, 削除 769
リストの印刷 770
ログメッセージの出力 387
SAS ウィンドウ環境, SAS セッション 724
開始 724
終了 724
中断 724
ホストコマンドの発行 724
例 12
SAS ウィンドウ環境, SAS レジストリの編集
参照項目: SAS レジストリエディタ
SAS ウィンドウ環境, ウィンドウ 739
関連項目: SAS レジストリエディタ
関連項目: アウトプットウィンドウ
関連項目: エクスプローラウィンドウ
関連項目: フォントウィンドウ
関連項目: プリファレンスウィンドウ
関連項目: プログラムエディタ
関連項目: ログウィンドウ
関連項目: 結果ウィンドウ
KEYS ウィンドウ, キーボードの割り当て 738
NOTEPAD ウィンドウ, 説明 769
NOTEPAD ウィンドウ, 開く 757
SASColor ウィンドウ, 説明 786
SASColor ウィンドウ, 開く 786
SAS オプションウィンドウ, システムオプションのカスタマイズ 774
SAS オプションウィンドウ, 開く 774
SAS システムオプションウィンドウ, 説明 786
SAS システムオプションウィンドウ, 開く 786
ウィンドウの呼び出し 739
エクスプローラオプションウィンドウ, 説明 786
エディタウィンドウ 753
エディタオプションウィンドウ, 説明 786
キーウィンドウ, キーボードの割り当て 738
検索ウィンドウ, 説明 769
検索ウィンドウ, ファイルの検索 749
コマンドウィンドウ, コマンドの実行 736
コンテンツペイン, オン/オフの切り替え 769
使用可能な最初の出力に移動 769
テンプレートウィンドウ, 説明 769
ファイルショートカットの割り当てウィンドウ 751, 768
プログラミングウィンドウ 723
SAS ウィンドウ環境, カスタマイズ 781
関連項目: エクスプローラウィンドウ, カスタマイズ
エディタ 784
フォント 784
プリファレンスの設定 785
SAS ウィンドウ環境, 出力 759
関連項目: テンプレートウィンドウ
関連項目: ログウィンドウ
関連項目: 結果ウィンドウ
概要 759
出力形式, プリファレンスウィンドウを使用した設定 760
出力形式, レジストリエディタを使用した設定 761
デフォルトのビューア, 割り当て 761
SAS ウィンドウ環境, テキストの編集
参照項目: SAS テキストエディタ
SAS ウィンドウ環境, ファイル管理 748
ファイル固有のコマンド, 発行 750
ファイルショートカット, 変更 752
ファイルショートカット, 割り当て 751
ファイルの印刷 752
ファイルの検索 748
ファイルを開く 750
SAS ウィンドウ環境, プログラムの編集
参照項目: プログラムエディタ
SAS ウィンドウ環境, ヘルプ 736
ウィンドウヘルプ 736
オンラインヘルプシステム 736
SAS エクスプローラ 676
SAS オプションウィンドウ
システムオプションのカスタマイズ 774
システムオプションの検索 775
システムオプションの設定 775
開く 774, 775
SAS カタログ

SAS/ACCESS ファイル 671
定義 671
SAS 関数
参照項目: 関数
SAS 言語 5
大文字と小文字の区別 5
命名規則 6
要素 5
SAS コマンド
SAS セッションの開始 720
非対話型モードの開始 726
SAS システムオプション
欠損値出力のカスタマイズ 621
検索 775
取得 780
設定 774, 775
表示 774, 786
保存 780
リスト 774
ログ 786
SAS システムオプションウィンドウ
説明 786
開く 786
SAS 出力ファイル
参照項目: レポート, SAS 出力ファイル
SAS ステートメント
参照項目: ステートメント
SAS セッション 720
関連項目: SAS ウィンドウ環境, SAS セッション
中断, ラインモード 726
ホスト動作環境で開始 720
SAS セッション, カスタマイズ 772
OPTIONS ステートメント 774
SAS ステートメントを自動的に実行する 773
起動時 773
起動に関するオプション 773
システムオプションの使用 774
システムオプションの設定 774
システムオプションの表示 774
プリファレンスウィンドウの使用 780
SAS セッション, セッション間のカスタマイズ 776
SAS レジストリエディタの使用 776
システムオプションの保存/取得 780
SAS データセット 91
関連項目: SAS ライブラリ
関連項目: オブザベーション
関連項目: 生データ
関連項目: 変数
SAS ライブラリでの参照 671
SAS ライブラリに格納 670
一時 32, 672
生データ, 定義 28
永久 32, 673
機能 28
構造 29
データ値 4
定義 4, 670
名前の変更 694
バイパス 584
SAS データセット, APPEND プロシジャを使用した連結
APPEND プロシジャ, SET ステートメントとの比較 275
APPEND プロシジャ, 説明 271, 276
変数が異なる 272
変数と属性が同じ 271
変数の属性が異なる 274
SAS データセット, SET ステートメントを使用した連結
SET ステートメント, APPEND プロシジャとの比較 275
SET ステートメント, 説明 254, 276
変数が同じ 254
変数が異なる 257
変数の出力形式が異なる 262
変数の種類, 変更 260
変数の種類が異なる 259
変数の属性が異なる 258
変数の長さが異なる 268
変数の入力形式が異なる 262
変数のラベルが異なる 262
SAS データセット, 移動 711
選択したデータセット 712
ライブラリ全体 711
SAS データセット, インタリーブ 279
関連項目: SAS データセット, マージ
関連項目: SAS データセット, 更新
関連項目: SAS データセット, 変更
関連項目: SAS データセット, 連結
BY グループ処理 280
BY ステートメント 283
SET ステートメント 283
データの並べ替え 280
定義 246
プロセス概要 282
SAS データセット, オブザベーションの書き込み
参照項目: オブザベーション, SAS データセットへの書き込み
SAS データセット, 結合
参照項目: SAS データセット, インタリーブ
参照項目: SAS データセット, マージ
参照項目: SAS データセット, 更新
参照項目: SAS データセット, 変更
参照項目: SAS データセット, 連結
SAS データセット, 更新 315
関連項目: SAS データセット, インタリーブ

関連項目: SAS データセット, マージ
関連項目: SAS データセット, 変更
関連項目: SAS データセット, 連結
BY 変数の選択 316
UPDATE ステートメント, 説明 316
欠損値 248, 326, 327
増分値 322
定義 247
トランザクションデータセット 316
変更との比較 250
マージとの比較 250, 324
マスタデータセット 316
例 317
SAS データセット, このドキュメントで使用
CITY 92, 792
CLIMATE.HIGHTEMP 694, 706, 798
CLIMATE.LOWTEMP 694, 706, 798
GRADES 541, 796
HIGHLOW 520, 795
OUT.ERROR1 394
OUT.ERROR2 394
OUT.ERROR3 394
OUT.SAT_SCORES3 384
OUT.SAT_SCORES4 384
OUT.SAT_SCORES5 384
PRECIP.RAIN 694, 706, 799
PRECIP.SNOW 694, 706, 799
SAT_SCORES 364, 793
STORM.TORNADO 694, 706, 799
USCLIM.BASETEMP 682, 694, 706, 798
USCLIM.HIGHTEMP 682, 694, 706, 797
USCLIM.HURRICANE 682, 694, 706, 797
USCLIM.LOWTEMP 682, 694, 706, 797
USCLIM.REPORT 682, 694, 706, 798
USCLIM.TEMPCHNG 682, 694, 706, 798
YEAR_SALES 422, 467, 492, 794
SAS データセット, コピー 706
重複した名前 706
その他のライブラリ 708
データセットの選択 710
プロシジャ入力ライブラリ 706
SAS データセット, コンテンツ情報
1 つのデータセットのリスト 685
DATASETS プロシジャ 684
コンテンツリストのフォーマット 688
すべてのデータセットのリスト 687
SAS データセット, 削除 713
削除の確認 713
特定ファイル 713
ライブラリ全体 714
SAS データセット, 作成
DBMS ファイル 46
ODS の使用 645
外部ファイル 45, 46
カラム入力 42
ジョブストリームの生データ 45
他の SAS データセット 45
データの場所 45
年値, 2 桁と 4 桁 43
入力スタイル 42
フォーマット入力 42
変数, 定義 43
リスト入力 42
SAS データセット, サブセット化
参照項目: オブザベーション, サブセット化
SAS データセット, 出力
関連項目: ODS
従来の出力 7
SAS データセット, 入力の指定
参照項目: DATA=オプション
SAS データセット, 変更 333
関連項目: MODIFY ステートメント
関連項目: SAS データセット, インタリーブ
関連項目: SAS データセット, マージ
関連項目: SAS データセット, 更新
関連項目: SAS データセット, 連結
欠損値 343
更新とマージとの比較 250
重複した BY 変数 340
定義 249
プログラムエラーのチェック 337
プログラム例 337, 341
マスタデータセット, 更新エラー 340
マスタデータセット, トランザクションデータセット 336
マスタデータセット, ネットワークオブザベーションあり 337
SAS データセット, 変数属性
再フォーマット 696
削除 696
名前の変更 696
変更 695
割り当て 696
SAS データセット, マージ 288
関連項目: SAS データセット, インタリーブ
関連項目: SAS データセット, 更新
関連項目: SAS データセット, 変更
関連項目: SAS データセット, 連結
MERGE ステートメント 288
更新と変更との比較 250
定義 247
SAS データセット, マージ(1 対 1) 247
同じ数のオブザベーション 288
同じ変数 291

異なる数のオブザベーション 288
異なる変数 288
使用例 310
定義 247
プログラム例 290
マッチマージとの比較 309
SAS データセット, マージ(マッチマージ) 247
1 対 1 のマージとの比較 309
BY ステートメント 294
共通変数あり 306
共通変数なし 307
削除変数あり 304
使用例 312
定義 247
複数のオブザベーションを含む BY グループ 298
プログラム例 292
SAS データセット, ラベル 699
削除 699
変更 699
割り当て 699
SAS データセット, 連結 253
関連項目: SAS データセット, インタリーブ
関連項目: SAS データセット, マージ
関連項目: SAS データセット, 更新
関連項目: SAS データセット, 変更
定義 246
SAS データセットからのコピー
参照項目: SAS データセット, コピー
SAS データセット行
参照項目: オブザベーション
SAS データセット名
参照項目: SAS 名
SAS データセットの 1 対 1 のマージ
参照項目: SAS データセット, マージ
参照項目: SAS データセット, マージ(1 対 1)
SAS データセットのインタリーブ
参照項目: SAS データセット, インタリーブ
SAS データセットの結合
参照項目: SAS データセット, インタリーブ
参照項目: SAS データセット, マージ
参照項目: SAS データセット, 更新
参照項目: SAS データセット, 変更
参照項目: SAS データセット, 連結
SAS データセットの更新 247
関連項目: SAS データセット, 更新
SAS データセットのマージ
参照項目: SAS データセット, マージ
SAS データセットのマッチマージ
参照項目: SAS データセット, マージ(マッチマージ)
SAS データセットの連結
参照項目: SAS データセット, 連結
SAS データセット列
参照項目: 変数
SAS データビュー 670
SAS データファイル 670
関連項目: SAS ファイル
定義 670
例 670
SAS データライブラリ
関連項目: SAS データセット
SAS ウィンドウ環境での探索 733
WORK 32
コンテンツリストのフォーマット 688
ディレクトリリスト, すべてのファイル 682
ディレクトリリスト, 定義 682
ディレクトリリスト, メンバタイプ 684
表現の検索 769
ファイルコンテンツのリスト, 1 つのデータセット 685
ファイルコンテンツリスト, すべてのデータセット 687
ライブラリの割り当てに関する問題 735
SAS データライブラリ, 移動 711
選択したデータセット 712
ライブラリ全体 711
SAS データライブラリ, ライブラリ参照名の割り当て
SAS ウィンドウ環境 734
SAS テキストエディタ 744
大文字/小文字, デフォルトの設定 746
大文字/小文字, 変更 747
カラム番号, 表示 746
行番号, 表示 746
切り取り、貼り付け、コピー 744
テキストの移動 744
テキストの結合 748
テキストの検索と変更 743
テキストの再配置 744
テキストの配置 745
テキストの分割 748
SAS 名 6
アンダースコア(_) 6
ブランク 6
命名規則 6
SAS の起動
ラインモード 725
SAS 日付値
参照項目: 日付関数
参照項目: 日付値
SAS 日付定数
参照項目: 日付関数
参照項目: 日付値
SAS ファイル 670

SAS データファイル 670
SAS ライブラリ 670
定義 670
SAS ファイル, 出力
参照項目: ODS 出力
参照項目: SAS ウィンドウ環境, 出力
参照項目: 出力
SAS プログラム, 実行 10
関連項目: SAS ウィンドウ環境
関連項目: プログラムエディタ
NOTEPAD 757
SAS/ASSIST 11
SAS ウィンドウ環境 722
一度に 1 行ずつ 12
対話型ラインモード 12, 725
バックグラウンド処理 722
バッチモード 11, 726
非対話型モード 11, 727
フォアグラウンド処理 721
方法の選択 10, 721
SAS プロシジャ
参照項目: プロシジャ
SAS マクロ機能 453
定義 453
マクロ変数, 参照 455
マクロ変数, 自動 453
マクロ変数, ユーザー定義 454
SAS マクロ言語 459
SAS ライブラリ 668
SAS エクスプローラ 676
SAS データセットの格納 670
SAS データセットの参照 671
アクセス 668
カタログ管理 676
管理 675
定義 668
場所 668
ファイル管理 676
ファイルの格納 670
ファイルまたはメンバのコピー 676, 677
ライブラリコンテンツ, リスト 676, 677
ライブラリ情報, リスト 676, 677
SAS ライブラリ, ライブラリ参照名の割り当て
LIBNAME ステートメント 668
SAS レジストリ, 編集 776
関連項目: REGEDIT コマンド
関連項目: SAS レジストリエディタ
概要 776
キー, 削除 777
キー, 設定 777
キー, 定義 776
キー値, 設定 778
キー値, 編集 778
サブキー 776
レジストリファイル, アンインストール 779
レジストリファイル, インポート 778
レジストリファイル, エクスポート 778
SAS レジストリエディタ 776
関連項目: REGEDIT コマンド
関連項目: SAS レジストリ, 編集
SAS レジストリの編集 735
カスタマイズ 779
情報の検索 777
説明 769, 786
開く 777, 786
SAS ログ 364
関連項目: デバッグ
関連項目: デバッグ, SAS Supervisor の使用
SAS ウィンドウ環境からの出力 387
エラーの解決 365
行サイズ 720
参照 727
出力 387
場所 366
役割 364
SAS ログ, 書き込み 583
関連項目: ODS
関連項目: SAS ログ, 出力先の指定
LIST ステートメント 372
PUT ステートメント 371, 584
SAS データセットのバイパス 584
従来の出力, 定義 8
従来の出力, 例 8
SAS ログ, 出力先の指定 386
関連項目: SAS ログ, 書き込み
LOG=オプション 386
LOG=システムオプション 388
NEW オプション 386
PRINT=システムオプション 388
PRINTTO プロシジャ 386
SAS ウィンドウ環境 387
構成ファイル 389
代替の場所 386
デフォルト場所, SAS ウィンドウ環境 387
デフォルト場所, バッチ環境 388
デフォルト場所, 復元 387
デフォルト場所, 変更 388
SAS ログ, 非表示
ERRORS=オプション 377
NONOTES オプション 376, 377
NOSOURCE オプション 376, 377
NOTES オプション 376
SAS システムオプション 376
SAS ステートメント 376, 377
SOURCE オプション 376
エラーメッセージ 377
システム情報 376, 377

プログラム例 377
SAS ログへの書き込み
参照項目: PUT ステートメント
参照項目: SAS ログ, 書き込み
SAT_SCORES データセット 364, 793
SAVE ステートメント 715
SCAN 関数
記憶域の節約 141
説明 145
文字列のスキャン 139
SELECT_ALL コマンド 770
SELECT ステートメント
SAS データセットの移動 712
SAS データセットのコピー 710
説明 715
SET コマンド
DATA ステップデバッガ 811
SET ステートメント 210
APPEND プロシジャとの比較 275
DATA ステートメントとの比較 102
SAS データセットのインタリーブ 283
SAS データセットの作成 45
SAS データセットの連結 254
オブザベーションの結合 356
オブザベーションのソースの決定 351
最後のオブザベーションの決定 201, 357
説明 210, 254, 359
選択した変数の保持 97
SHORT オプション
CONTENTS ステートメント 690
コンテンツリストのフォーマット 688
SKIP オプション
BREAK ステートメント 512
RBREAK ステートメント 514
区切り行, 挿入 509
SKIP ステートメント
説明 380
SMALLVIEW コマンド
アイコンサイズの設定 782
説明 770
SORT プロシジャ 194
BY ステートメント 661
オブザベーションのグループ化 184
オブザベーションの並べ替え 190
詳細レポートの並べ替え 428
説明 194, 459
SOURCE ウィンドウ
DATA ステップデバッガ 813
SOURCE オプション
SAS ステートメントを非表示にする 376
説明 380
SPACING=オプション
DEFINE ステートメント 513
PROC REPORT ステートメント 511
列スペース 502
SPLIT=オプション
PROC PRINT ステートメント 458
PROC REPORT ステートメント 511
複数行の列ラベル 449
STEP コマンド
DATA ステップデバッガ 812
STOPOVER オプション 89
説明 89
予想外のレコードの終わり 86
STORE コマンド 744
STORM.TORNADO データセット 694, 706, 799
STYLE=オプション
ODS PRINTER ステートメント 647
次元式 481
STYLE オプション
PROC PRINT ステートメント 458
SUBGROUP=オプション
BLOCK ステートメント 575
HBAR ステートメント 575
VBAR ステートメント 575
範囲内のサブグループ 558
SUBMIT コマンド 754
SUMBY ステートメント
PRINT プロシジャ 459
グループの合計の計算 443
SUMLABEL オプション
PROC PRINT ステートメント 458
SUMMARIZE オプション
BREAK ステートメント 512
RBREAK ステートメント 514
要約行 509
SUMVAR=オプション
BLOCK ステートメント 575
HBAR ステートメント 575
PIE ステートメント 575
VBAR ステートメント 575
平均のチャート作成 560
SUM 関数
数字の合計 124
説明 128
SUM ステートメント
PRINT プロシジャ 459
詳細レポートの合計 436
SUPPRESS オプション 512
SWAP コマンド
DATA ステップデバッガ 813
SYNCOLOR コマンド 742
SYNCONFIG コマンド 742
SYNCONFIG ステートメント 785
SYSDATE9 自動マクロ変数
詳細レポートの日付 453
説明 460

T

TABLE ステートメント
  TABULATE プロシジャ 661
TABLE ステートメント, TABULATE プロシジャ 486
  制限 466
  要約テーブル構造の定義 465
TABULATE プロシジャ 485
  CLASS ステートメント 465, 485
  CLASS ステートメント, 説明 661
  KEYLABEL ステートメント 661
  KEYWORD ステートメント 661
  KEYWORD ステートメント, 説明 661
  PROC TABULATE ステートメント 485, 661
  TABLE ステートメント 465, 466, 486
  VAR ステートメント 465, 486, 661
  必須ステートメント 464
TC (テキスト接続)コマンド 748
TEMPLATE プロシジャ 662
  COLUMN ステートメント, ODS 出力のカスタマイズ 651
  COLUMN ステートメント, 説明 662
  DEFINE ステートメント, ODS 出力のカスタマイズ 651
  DEFINE ステートメント, 説明 662
  DELETE ステートメント 662
  END ステートメント, ODS 出力のカスタマイズ 651
  END ステートメント, 説明 662
  HEADER ステートメント, ODS 出力のカスタマイズ 651
  HEADER ステートメント, 説明 662
  PROC TEMPLATE ステートメント 662
  ソースコード, 参照 767
  ソースコード, 編集 767
  テンプレート情報の設定 770
TITLE ステートメント
  PRINT プロシジャ 459
  REPORT プロシジャ 514
  グローバルステートメント 484
  出力タイトル 604, 609
  出力タイトルの中央揃え 614
  説明 624
  プロットタイトル 537
  プロットのタイトル付け 537
  レポートタイトル 446, 453
TODAY()関数
  説明 241
TOP コマンド 741, 753
TRACE コマンド
  DATA ステップデバッガ 813
TREE コマンド 770
  関連項目: AUTOEXPAND コマンド
  説明 770
  ツリービュー, オン/オフの切り替え 722
TRIM 関数
  説明 145
  文字の追加 142
TRUNCOVER オプション 89
  説明 89
  予想外のレコードの終わり 86, 88
TS (テキスト分割)コマンド 748
TURNVLABELS オプション 573
TYPE=オプション
  BLOCK ステートメント 575
  HBAR ステートメント 575
  PIE ステートメント 575
  VBAR ステートメント 575
  平均のチャート作成 560

U

UNDO コマンド 754
UNIVARIATE プロシジャ 540
  BY ステートメント 662
  CLASS ステートメント, 説明 576, 662
  CLASS ステートメント, 比較ヒストグラム 572
  HISTOGRAM ステートメント, 説明 577
  HISTOGRAM ステートメント, ヒストグラム 562
  INSET ステートメント, 説明 577
  INSET ステートメント, ヒストグラムの要約統計量 570
  PowerPoint 出力, Microsoft PowerPoint 用 639
  PROC UNIVARIATE ステートメント 576, 662
  RTF 出力, Microsoft Word 用 639
  VAR ステートメント 662
UPCASE 関数
  説明 166
  文字を大文字に変換 161
UPDATEMODE=オプション
  SAS データセットの更新 327
  SAS データセットの変更 343
  説明 346
UPDATE ステートメント
  MERGE ステートメントと MODIFY ステートメントとの比較 250
  MERGE ステートメントとの比較 324
  SAS データセットの作成 45
  欠損値 326, 327
  説明 316, 331
  複数のオブザベーションを含む BY グループ 327
UPLEVEL コマンド
  エクスプローラウィンドウのナビゲート 749
  結果ウィンドウでの移動 764

説明 770
テンプレートウィンドウでの移動 767
USCLIM.BASETEMP データセット 682, 694, 706, 798
USCLIM.HIGHTEMP データセット 682, 694, 706, 797
USCLIM.HURRICANE データセット 682, 694, 706, 797
USCLIM.LOWTEMP データセット 682, 694, 706, 797
USCLIM.REPORT データセット 682, 694, 706, 798
USCLIM.TEMPCHNG データセット 682, 694, 706, 798

V

VARNUM オプション
CONTENTS ステートメント 690
コンテンツリストのフォーマット 688
VAR ステートメント
PRINT プロシジャ 459
SORT プロシジャ 661
TABULATE プロシジャ 486, 661
UNIVARIATE プロシジャ 662
選択した変数のレポート作成 430
要約テーブル分析変数の指定 465
VAXIS=オプション
HISTOGRAM ステートメント 577
PLOT ステートメント 537
ヒストグラム 567
目盛値 526
VAXISLABEL=オプション 568
VBAR ステートメント, CHART プロシジャ 575
縦棒グラフ 544
VERBOSE オプション
SAS セッションのカスタマイズ 773
説明 728, 786
VIEW メンバタイプ 670
VMINOR=オプション
HISTOGRAM ステートメント 577
ヒストグラム 565
VPCT=オプション
PROC PLOT ステートメント 536
同一ページの複数のプロット 532
VPERCENT=オプション
PROC PLOT ステートメント 536
同一ページの複数のプロット 532
VSCALE=オプション
HISTOGRAM ステートメント 577
ヒストグラム 567
VSCROLL コマンド 741

W

WATCH コマンド
DATA ステップデバッガ 814
WEEKDATE29.出力形式
説明 240
日付の表示 231
WEEKDAY 関数
説明 241
曜日を返す 237
WHERE ステートメント
PRINT プロシジャ 459
REPORT プロシジャ 514
大文字と小文字の区別 432
レポートデータの選択 493
レポートの印刷 432
WIDTH=オプション
DEFINE ステートメント 513
PROC PRINT ステートメント 458
列幅 502
WINDOWS オプション 511
WORDDATE18.出力形式
説明 240
日付の表示 231
WORK ライブラリ 32

X

X コマンド
SAS セッションの中断 724
説明 728
対話型ラインモードの中断 726
ホスト環境からのコマンドの発行 724
X ステートメント
説明 728
対話型ラインモードの中断 726

Y

YEAR_SALES データセット 422
using 492
作成 794
使用 467
YEARCUTOFF=システムオプション 241
世紀の決定 43, 225
説明 241

Z

ZOOM コマンド 741, 754

あ

アウトプットウィンドウ 723
関連項目: SAS ウィンドウ環境, ウィンドウ
クリア 755

コンテンツの保存 388
参照 727
説明 769
定義 723
プロシジャ出力 387
例 13
値の位置調整 139
アプリケーション, カスタマイズ 776
アポストロフィ
参照項目: 引用符(')
アンダースコア, SAS 名 6
アンパサンド
フォーマット修飾子 64
マクロ変数名 455
一時 SAS データセット 672
色
SASCOLOR コマンド 742
SASCOLOR ステートメント 785
SYNCOLOR コマンド 742
ウィンドウ 742
印刷 752
関連項目: PRINT プロシジャ
関連項目: REPORT プロシジャ
関連項目: レポート
関連項目: 出力
SAS ウィンドウ環境 752, 768
出力 768
引用符(') 133
変数インジケータ 133
リテラル文字 133
ウィンドウ, SAS
参照項目: SAS ウィンドウ
ウィンドウ, SAS ウィンドウ環境
参照項目: SAS ウィンドウ環境, ウィンドウ
ウィンドウのスクロール 753
BACKWARD コマンド 741, 754
BOTTOM コマンド 741, 753
CURSOR コマンド 742
FORWARD コマンド 741, 754
HSCROLL コマンド 741
LEFT コマンド 741, 754
MAX コマンド 742
RIGHT コマンド 741, 754
TOP コマンド 741, 753
VSCROLL コマンド 741
ウィンドウの呼び出し 739
ウィンドウヘルプ 736
中かっこ, STYLE=オプション 481
生データ 28
関連項目: SAS データセット
SAS データセットの作成 45
定義 28
フィールド 29
レコード 29
生データ, 整列
参照項目: カラム入力
生データ, 整列されていない
参照項目: リスト入力
生データ, 読み込み
参照項目: 生データレコードの読み込み
生データレコードの読み込み 71
関連項目: カラム入力
関連項目: フォーマット入力
関連項目: リスト入力
2 回読み込み 72
可変長レコード 85
欠損値 85
後置@ (@) 72
後置@@ (@@) 74
条件のテスト 72
予期しないレコードの終わり 85
読み込み後に保持 72
ラインホールド指定子 72, 74, 89
埋め込み特殊文字, 読み込み
参照項目: 入力形式
埋め込みブランク 58
カラム入力 58
リスト入力 64
永久 SAS データセット 32, 673
エクスプローラウィンドウ 722
関連項目: SAS ウィンドウ環境, ウィンドウ
定義 722
開く 733
ファイルの検索 749
エクスプローラウィンドウ, カスタマイズ 781
アイコンサイズ 782
コンテンツのみのビューとエクスプローラビュー 781
コンテンツビュー 782
ファイルタイプ, 非表示 784
ファイルタイプ, 表示の有効化 783
フォルダ, 追加と削除 783
フォント 784
編集オプション 784
ポップアップメニューアクション, 追加 783
エクスプローラオプションウィンドウ
説明 786
開く 786
エディタ
関連項目: NOTEPAD ウィンドウ
関連項目: SAS テキストエディタ
関連項目: プログラムエディタ
カスタマイズ 784
エディタウィンドウ
参照項目: プログラムエディタ
エディタオプションウィンドウ
説明 786

開く 786
エラー
種類 393
処理 392
診断 394
診断と回避 391
エラー処理 392
関連項目: デバッグ
関連項目: デバッグ, SAS Supervisor の使用
エラー診断
参照項目: デバッグ
参照項目: デバッグ, SAS Supervisor の使用
エラーの種類 393
エラーの診断
参照項目: デバッグ
参照項目: デバッグ, SAS Supervisor の使用
エラーの診断と回避 391
エラーメッセージ, ログの非表示 377
円グラフ 549
大文字/小文字, デフォルトの設定 746
大文字/小文字, 変更 746
CAPS コマンド 746, 754
CCL (小文字変換)コマンド 747
CCU (大文字変換)コマンド 747
CL (小文字変換)コマンド 747
CU (大文字変換)コマンド 747
UPCASE 関数 166
大文字小文字, 変更
UPCASE 関数 161
大文字小文字の区別
文字変数 133
大文字と小文字の区別
SAS 言語 5
オブザベーションの並べ替え 193
ステートメント 5
変数名 6
文字の比較 161
文字を大文字に変換 161
大文字変換
参照項目: 大文字/小文字, 変更
オブザベーション 29
関連項目: SAS データセット
関連項目: 変数
グローバル変更 111
重複の削除 191
条件付き削除 117, 169
条件付き処理 347
選択的変更 112
定義 29
変数, 記憶域 116
変数, 効率的な使用法 114
変数, 作成 111
変数, 変更 113
割り当てステートメント 111
オブザベーション, SAS データセットへの書き込み 173
関連項目: OUTPUT ステートメント
1 つ以上のデータセットに複数回の書き込み 177
異なるデータセット 204, 205
複数データセット, よくある間違い 174
複数データセット, 例 173
オブザベーション, グループ化 183
関連項目: オブザベーション, サブセット化
関連項目: オブザベーション, 並べ替え
BY ステートメント, 基本グループ 183
BY ステートメント, 最初のオブザベーションまたは最後のオブザベーションの検索 188
BY ステートメント, 説明 194
FIRST.変数, 最初のオブザベーションの検索 188
FIRST.変数, 説明 194
LAST.変数, 最後のオブザベーションの検索 188
LAST.変数, 説明 194
SORT プロシジャ, オブザベーションのグループ化 184
SORT プロシジャ, 説明 194
降順 186
最初のオブザベーションまたは最後のオブザベーションの検索 187
複数の変数 185
オブザベーション, 計算 199
END=オプション, 最後のオブザベーションの決定 201
END=オプション, 説明 210
RETAIN ステートメント, 値の保持 207
RETAIN ステートメント, 説明 210
後のオブザベーションのために値を保持 207
各 BY グループの合計 202
オブザベーションを異なるデータセットに書き込む 204
合計ステートメント, 中間結果 199
合計のみの出力 201
合計を異なるデータセットに書き込む 205
中間結果 199
オブザベーション, 作成
1 つのレコードから複数 74
生データレコードのテスト 72
単一 DATA ステップから複数処理 100
複数のレコードから 1 つ 78
オブザベーション, サブセット化 167, 183
関連項目: DATA ステートメント
関連項目: DELETE ステートメント
関連項目: DROP=オプション

関連項目: DROP ステートメント
関連項目: FIRSTOBS=オプション
関連項目: IF ステートメント
関連項目: KEEP=オプション
関連項目: KEEP ステートメント
関連項目: OBS=オプション
関連項目: SET ステートメント
関連項目: オブザベーション, 並べ替え
1 条件が真(OR) 156
ELSE ステートメント 151
IF-THEN ステートメント 149
SAS データセット 173
オプションとステートメントの比較 98
効率性 102
最後のレコードを指す 96, 104
最初のレコードを指す 95, 104
条件付き削除 117, 169
条件の作成 153
数値の比較, 省略 159
全条件が真(AND) 155
相互排他的条件 152
代替処理 151
単純条件 150, 154, 171
比較演算子 153
否定演算子 157
複雑な比較 158
複数の比較 155
文字グループ, 選択 162, 163, 164
文字の比較, 大文字と小文字の区別 161
文字の比較, タイプ 161
オブザベーション, 選択
参照項目: オブザベーション, サブセット化
オブザベーション, 並べ替え 190
関連項目: オブザベーション, グループ化
NODUPRECS オプション, 重複レコードの削除 191
NODUPRECS オプション, 説明 194
SORT プロシジャ, オブザベーションの並べ替え 190
SORT プロシジャ, 説明 194
大文字と小文字の区別 193
重複の削除 191
照合順序, ASCII 193
照合順序, EBCDIC 194
照合順序, 文字の大きさ 163
例 190
オブザベーション, 複数の SAS データセット
関連項目: IN=オプション
最後のオブザベーションの計算 357
選択したオブザベーションの結合 356
ソースデータセットの決定 351
プログラム例 352, 358
オブザベーションのグループ化
参照項目: オブザベーション, グループ化
オブザベーションのサブセット化
参照項目: オブザベーション, サブセット化
オブザベーションの選択
参照項目: オブザベーション, サブセット化
オブザベーションの並べ替え
参照項目: オブザベーション, 並べ替え
オンラインヘルプ, SAS ウィンドウ環境
参照項目: SAS ウィンドウ環境, ヘルプ

か

階層テーブル 473
階層ビュー
参照項目: ツリービュー
外部ファイル 47
SAS データセットの作成 45, 46
入力として指定 46, 47
ファイル参照名の割り当て 47
外部ファイル, 出力
関連項目: ODS
従来の出力 8
改ページ
出力レポート 619
レポート 444
書き込み
参照項目: ODS
参照項目: 出力
数, レポートのフォーマット 504
カスタマイズ
関連項目: ODS 出力, カスタマイズ
関連項目: SAS セッション, カスタマイズ
関連項目: SAS セッション, セッション間のカスタマイズ
関連項目: エクスプローラウィンドウ, カスタマイズ
関連項目: プロット, カスタマイズ
関連項目: 出力, カスタマイズ
SAS ウィンドウ 742
SAS ウィンドウ環境 781
SAS レジストリエディタ 779
アプリケーション 776
結果ウィンドウ 762
欠損値, プロシジャ使用 622
欠損値出力, システムオプション 621
詳細レポート 453
テンプレートウィンドウ 766
度数グラフ 551
レポート 446
レポートの列ヘッダー 503
カタログ管理 676

可変長レコード, 読み込み 85
カラム入力 57
  関連項目: フォーマット入力
  関連項目: リスト入力
  関連項目: 生データレコードの読み込み
  SAS データセットの作成 42
  埋め込みブランク 58
  規則 60
  サンプルプログラム 57
  定義 57
  入力ポインタ 67
  フィールドのスキップ 59, 81
  複数の入力スタイルの使用 65
  リスト入力との比較 58
カラムポインタコントロール 62
  関連項目: ポインタコントロール
  関連項目: ラインポインタコントロール
  / (スラッシュ), 次の行へのポインタの強制移動 79
  @n 69, 88
  #n, 説明 69, 88
  +n 69, 88
  スラッシュ (/), 説明 69, 88
  絶対指定 63
  説明 69, 88
  相対指定 63
  定義 62
  フォーマット入力 62, 69
カレンダ日付
  関連項目: 日付関数
  関連項目: 日付値
  SAS 日付値との比較 224
  SAS 日付値への変換 230, 240
関数 125
  関連項目: 日付関数
  関連項目: 日付値
  INDEX 166
  LEFT 145
  ROUND 128
  SCAN 145
  SUM 128
  TRIM 145
  UPCASE 166
  組み合わせ 125
感嘆符(!!), 連結演算子 141
カンマ, 入力データ 61
キー, SAS レジストリ
  値, 設定 778
  値, 編集 778
  削除 777
  設定 777
  定義 776
記憶域, 節約
  SCAN 関数 141
  数字を変数として処理 143
記憶域, 変数の定義 116
記述統計量, 要約テーブルのための計算 475
行, SAS データセット
  参照項目: オブザベーション
行, 生データ 29
行, レポート
  並べ替え 496, 499
  レイアウト 496, 499
行コマンド 754
行サイズ, 出力レポート 611
切り捨て
  関連項目: FLOWOVER オプション
  関連項目: MISSOVER オプション
  関連項目: STOPOVER オプション
  関連項目: TRUNCOVER オプション
  制御 86
  文字変数 134
グラフ
  参照項目: チャート
  参照項目: プロット
グリッド線, ヒストグラム 565
クロス集計表 462, 473, 507
結果ウィンドウ 769
  ウィンドウプロパティ, 表示 765
  エクスプローラビュー 764
  カスタマイズ 762
  結果ポインタ, 削除 764
  結果ポインタ, 名前変更 764
  コンテンツのみのビュー 763
  出力の移動 763
  出力の操作 762
  出力ポインタ項目, 表示 765
  説明 769
  ツリービュー 763
  定義 723
  開く 762
  別の形式での保存 764
欠損値
  MERGE ステートメント 326
  MODIFY ステートメント 327
  SAS データセット 248
  SAS データセットの更新 326, 327
  UPDATE ステートメント 326, 327
  生レコードの読み込み 85
  カスタマイズ, システムオプション 621
  カスタマイズ, プロシジャ使用 622
  出力レポート 621
  数値変数 123, 124
  要約テーブル 466
欠損値, 文字変数
  確認 137
  設定 138
  ピリオド 136
  ブランク 136
検索ウィンドウ

説明 769
ファイルの検索 749
検索と置換 754
減算 210
合計ステートメント
説明 210
中間結果 199
後置@
生データレコードの読み込み 72
固定された出力行の解除 588
出力行の書き込み 587, 597
説明 89
後置@@ (@@)
DATA ステップの実行 74
説明 89
定義 74
構文エラー
診断 394
構文のチェック 392
コマンド
SAS ウィンドウ環境 736
行コマンド 754
コマンド行のコマンド 753
動作環境, SAS セッションから発行 724, 726
ファイル固有 750
コマンドウィンドウ 736
コマンド行のコマンド 753
小文字変換
参照項目: 大文字/小文字, 変更
コロン(:)
フォーマット修飾子 64
文字の比較 162
コンテンツペイン, オン/オフの切り替え 769

さ

サブキー, SAS レジストリ 776
算術演算
参照項目: オブザベーション, 計算
参照項目: 数値変数, 計算
時間値, レポート 453, 612
式
DATA ステップデバッガ 405
次元式 465
システム情報, ログの非表示 376, 377
実行時エラー
定義 393
出力 7
関連項目: ODS
関連項目: SAS ウィンドウ環境, 出力
関連項目: SAS ログ, 出力先の指定
関連項目: SAS ログ, 書き込み
関連項目: レポート
DATA ステップ 601
印刷 768
参照 727
出力形式の設定 760
出力形式の設定, プリファレンスウィンドウの使用 760
出力形式の設定, レジストリエディタの使用 761
ポインタでの移動 763
出力, SAS 出力ファイル
参照項目: レポート, SAS 出力ファイル
出力, カスタマイズ 604
関連項目: ODS 出力, カスタマイズ
SAS システムオプション 610, 612, 624
改ページ 619
行サイズ 611
欠損値 621
欠損値, システムオプション 621
欠損値, プロシジャ使用 622
時間値 612
出力の中央揃え 611, 614
タイトル, 中央揃え 614
タイトル, 追加 604
タイトル, 特定のカラム 616
日付値 612
フットノート 606, 609
ページサイズ 611
ページ番号付け 611, 617
変数ラベル 608, 609
列ヘッダー, 中央揃え 614
列ヘッダー, 特定のカラム 616
レポートヘッダー, シンボリック値 617
出力, 従来
SAS データセット 7
SAS ファイル 8
SAS ログ, 定義 8
SAS ログ, 例 8
外部ファイル 8
データベースエントリ 8
レポート 8
出力オブジェクト
参照項目: ODS 出力オブジェクト
出力形式
DATA ステップデバッガ 412
出力形式, 日付値
WEEKDATE29. 240
WORDDATE18. 240
出力形式属性 258
出力先の指定, SAS ログ
参照項目: SAS ログ, 出力先の指定
出力先の指定, プロシジャ 600
PRINT=オプション 385
PRINTTO プロシジャ 385
SAS ウィンドウ環境 387
SAS カタログエントリ 385
永久ファイル 385
概要 600

出力の非表示 385
ダミーファイル 385
デフォルトの場所 603
デフォルト場所, SAS ウィンドウ環境 387
出力先の指定, 要約テーブル 480
出力テンプレート
参照項目: テンプレートウィンドウ
出力ファイルへの書き込み
参照項目: DATA ステップ
参照項目: PUT ステートメント
参照項目: レポート, SAS 出力ファイル
種類属性 258
ショートカット
参照項目: ファイルショートカット
小計 438
照合順序 193
ASCII 193
EBCDIC 194
文字の大きさ 163
詳細レポート 489
関連項目: レポート
関連項目: 印刷
オブザベーション, 選択 432
オブザベーション, 選択(1 つの比較) 432
オブザベーション, 変数値ごとのグループ化 437
拡張レポートの作成 434
カスタマイズ 453
キー変数, 強調 427
グループの小計, 識別 439
時間, 自動的に含める 453
数値変数の合計 436
タイトル 453
単純なレポートの作成 424
定義 491
並べ替えられていないキー変数 428
並べ替えられているキー変数 428
日付, 自動的に含める 453
フォーマット 435
マクロ機能 453
小数, 精度の損失 128
数字の加算
参照項目: オブザベーション, 計算
参照項目: 数字の合計
参照項目: 数値変数, 計算
数字の合計 124, 128
関連項目: オブザベーション, 計算
関連項目: 数値変数, 計算
数字を丸める 124, 128
数値の比較, 省略 159
数値変数 119
埋め込み特殊文字 63
効率的格納 127
コンテンツ 43
小数, 精度の損失 128
定義 120
短くする 127
数値変数, 計算 121
関連項目: 関数
欠損値 123, 124
変数の比較 125
論理演算子 125
割り当てステートメント, 算術演算子 121
割り当てステートメント, 数値式 123
数値変数の合計 436
スタイル, 要約テーブル 480
スタイルテンプレート 647
ステートメント 5
大文字と小文字の区別 5
記述規則 5
起動時に自動的に実行する 773
継続行 6
セミコロン(;) 5
ログの非表示 376, 377
ステートメント, サブミット
参照項目: SAS プログラム, 実行
スラッシュ (/), カラムポインタコントロール
説明 69, 88
スラッシュ(/), カラムポインタコントロール
次の行へのポインタの強制移動 79
スラッシュ(/), 列ヘッダーの分割 503
世紀カットオフ
参照項目: YEARCUTOFF=システムオプション
整列されていない生データ
参照項目: リスト入力
整列した生データ
参照項目: カラム入力
絶対指定のカラムポインタコントロール 63
セマンティックエラー 394
セミコロン(;)
ステートメント 5
データの終わりインジケータ 45
セル, レポート 505
総計 438
相対指定のカラムポインタコントロール 63
属性, 変数 258

た

タイトル, プロット 537
タイトル, レポート 446
関連項目: ヘッダー, レポート
概要 446
作成 447, 604, 609
シンボリック値 453
中央揃え 614

特定のカラム 616
対話型ラインモード 12, 725
関連項目: ラインモード
SAS セッションの中断 726
縦棒, 連結演算子 141
縦棒グラフ 544
作成 544
中間値 551
中間点の数 554
ダブルクリック, 相当するキーボード操作 732
チャート 540
関連項目: CHART プロシジャ
関連項目: PLOT プロシジャ
関連項目: UNIVARIATE プロシジャ
関連項目: ヒストグラム
関連項目: プロット
関連項目: 縦棒グラフ
関連項目: 度数グラフ
3 次元 561
円グラフ 549
すべての値のチャートの作成 555
ツール 540
統計量のテーブル, 非表示 546, 563
平均のチャート作成 560
範囲内のサブグループ 558
ブロックチャート 548
横棒グラフ 546
離散値と連続値 555
チャート, 中間点
数値変数, 値 551
数値変数, 数 554
ヒストグラム 568
文字変数, 値 557
中間点
数値変数, 値 551
数値変数, 数 554
ヒストグラム 568
文字変数, 値 557
ツリービュー 722, 763
オン/オフの切り替え 722, 770
展開 769
データ, ODS 9, 628
データ, 生
参照項目: 生データ
データ値 4
データエラー
診断 398
定義 394
データ管理機能 4
データセット
参照項目: SAS データセット
データセット名
参照項目: SAS 名
データのマージ
IN=データセットオプション 305
データ分析ユーティリティ 6
データベースエントリ, 出力
関連項目: ODS
従来の出力 8
テーブル
参照項目: 要約テーブル
テーブル定義(ODS)
関連項目: テンプレートウィンドウ
定義 9
テーブルテンプレート(ODS) 650
ODS 出力のカスタマイズ 650
定義 628
ディレクトリリスト
コンテンツリストのフォーマット 688
すべてのファイル 682
定義 682
メンバタイプ 684
テキスト 744
SAS テキストエディタ 744
移動と再配置 744
大文字/小文字の変更 746
結合と分割 748
検索と置換 754
テキストの編集
参照項目: NOTEPAD エディタ
参照項目: SAS テキストエディタ
デバッグ 391
関連項目: SAS ログ
関連項目: ログウィンドウ
DATA ステップデバッガ 403
品質管理チェックリスト 401
プログラム, プログラムエディタ 758
プログラム, ログウィンドウ 756
ライブラリの割り当てに関する問題 735
デバッグ, SAS Supervisor の使用 394
_ERROR_変数 398
_N_変数 398
SAS エラー処理 392
エラーの種類 393
構文エラー, 診断 394
構文のチェック 392
実行時エラー, 定義 393
セマンティックエラー 394
データエラー, 診断 398
データエラー, 定義 394
テンプレートウィンドウ 765
関連項目: SAS ウィンドウ環境, ウィンドウ
関連項目: テーブルテンプレート(ODS)
ウィンドウプロパティ, 表示 768
エクスプローラビュー 766
概要 765
カスタマイズ 766
コンテンツのみのビュー 767
コンテンツペイン 766

出力の印刷 768
説明 769
ソースコード, 参照 767
ソースコード, 編集 767
ツリービュー 766, 767
開く 765
等号(=)
線を引く 509
要約テーブルラベルの定義 479
動作環境
マウスに相当するキーボード操作 772
特殊文字, 読み込み
参照項目: 入力形式
年値, 2 桁と 4 桁 43, 225, 228
関連項目: 日付関数
関連項目: 日付値
度数カウント 506
度数グラフ 543
カスタマイズ 551
作成 543
数値変数 543
数値変数の中間点 552
文字変数 557
トラブルシューティング
参照項目: デバッグ
参照項目: デバッグ, SAS Supervisor の使用
トランザクションデータセット 316
ドル記号($)
入力データ 60
変数名 133
文字変数の定義 43

な

長さ属性 258
名前, データセット
参照項目: SAS 名
名前属性 258
名前の変更
MODIFY ステートメント 696
RENAME ステートメント 694, 696
SAS データセット 694
SAS データセット変数属性 696
名前変更
RENAME=オプション 306
結果ポインタ 764
出力ポインタ 770
並べ替え順序
参照項目: 照合順序
平均, チャート作成 560
入力形式 60
アンパサンドフォーマット修飾子 64
コロン(:)フォーマット修飾子 64
特殊文字の読み込み 60
長い文字変数の作成 64
命名規則 61
リスト入力での埋め込みブランクの読み込み 64
入力形式, 日付値
DATE7. 226, 240
DATE9. 226, 228, 240
MMDDYY10. 226, 240
MMDDYY8. 226, 240
入力形式属性 258
入力スタイル
関連項目: カラム入力
関連項目: フォーマット入力
関連項目: リスト入力
関連項目: 生データレコードの読み込み
複数使用 65
ラインポインタへの影響 66
入力バッファ, DATA ステップ 36
入力変数のスキップ 59, 81
入力ポインタ 62, 67, 69

は

配列, 定義 217
配列処理 217
関連項目: DO グループ
現在の変数の選択 218
配列の定義 217
反復 DO ループ 218
バックグラウンド処理 722
バッチモード 11, 726
反復 DO ループ 218
関連項目: DO グループ
関連項目: 配列処理
比較演算子 153
ヒストグラム 562
HISTOGRAM ステートメント 562
SAS/GRAPH ソフトウェア 563
グリッド線 565
軸の変更 565
単純なヒストグラム 563
中間点 568
比較ヒストグラム 572
目盛 565
要約統計量 570
非対話型モード 11, 726
左シフトコマンド 745
日付値 224
2 桁と 4 桁の年 43, 225, 228
DATE7.入力形式, 説明 226, 240
DATE7.入力形式, 年の長さ 228
DATE9.入力形式, 説明 226, 240
DATE9.入力形式, 年の長さ 228
D 接尾辞 230
FORMAT ステートメント, 永久日付出力形式 232

FORMAT ステートメント, 説明 240
MMDDYY10.入力形式, 説明 226, 240
MMDDYY10.入力形式, 年の長さ 228
MMDDYY8.入力形式, 説明 226, 240
MMDDYY8.入力形式, 年の長さ 228
SAS 格納形式 224
WEEKDATE29.出力形式, 説明 240
WEEKDATE29.出力形式, 日付の表示 231
WORDDATE18.出力形式, 説明 240
WORDDATE18.出力形式, 日付の表示 231
YEARCUTOFF=システムオプション, 世紀の決定 43, 225
YEARCUTOFF=システムオプション, 説明 241
カレンダ日付, SAS 日付値との比較 224
カレンダ日付, SAS 日付値への変換 230, 240
計算 235
作成 236
出力形式 231
世紀カットオフ, 決定 43, 225
定数 230
並べ替え 235
入力 226
入力形式 226
入力データ 60
表示 231
プログラミング実践 228
読み取り 227, 228
レポート 453, 612
日付値, 計算
期間の比較 239
曜日, 検索 237
日付値, フォーマット
一時的 235
永久的 233
出力用 231
入力用 226
日付値の D 接尾辞 230
日付値の並べ替え 235
日付関数 237
関連項目: 日付値
TODAY(), 説明 241
WEEKDAY, 説明 241
WEEKDAY, 曜日を返す 237
否定演算子 157
ビュー 670
ピリオド(.)
欠損値 30, 136
入力形式名 61
入力データ 60
品質管理チェックリスト 401
ファイル
関連項目: SAS ウィンドウ環境, ファイル管理
関連項目: SAS ファイル
関連項目: 外部ファイル
SAS データファイル 670
印刷 752
上書き 388
検索 748
検索, エクスプローラを使用 749
検索, 検索ウィンドウを使用 749
コピー 676, 677
操作 748
開く 750
ファイル固有のコマンドの発行 750
ファイル, 書き込み
参照項目: SAS ログ, 出力先の指定
参照項目: SAS ログ, 書き込み
参照項目: レポート, SAS 出力ファイル
参照項目: 出力先の指定, プロシジャ
ファイル管理
参照項目: SAS ウィンドウ環境, ファイル管理
ファイルコンテンツ, リスト 687
関連項目: CONTENTS ステートメント
1 つのファイル 685
CONTENTS プロシジャ 676
コンテンツリストのフォーマット 688
ライブラリのすべてのファイル 687
ファイル参照名, 外部ファイル 47
ファイルショートカット 751
変更 752
割り当て 751, 759
ファイルショートカットの割り当てウィンドウ
説明 768
ファイルショートカットの割り当て 751
ファイルのエイリアス 47
ファイルまたはメンバのコピー 676, 677
フィールド(生データ) 29
フォアグラウンド処理 721
フォーマット入力 60
関連項目: カラム入力
関連項目: リスト入力
関連項目: 生データレコードの読み込み
SAS データセットの作成 42
カラムポインタコントロール 62, 69
規則 63
サンプルプログラム 61
絶対指定のカラムポインタコントロール 63
相対指定のカラムポインタコントロール 63
定義 60
入力ポインタ 62, 67, 69

複数の入力スタイルの使用 65
ポインタ配置 62, 69
フォント, SAS ウィンドウ環境 784
フォントウィンドウ 786
説明 786
開く 784, 786
フォントのカスタマイズ 784
フットノート
SAS 出力ファイルのレポート 590, 595
プロシジャ出力 606, 609
レポート 446, 606, 609
ブランク
SAS 名 6
欠損値 30, 136
先頭, 削除 140
リスト入力区切り文字 53
ブランク, 埋め込み
参照項目: 埋め込みブランク
プリファレンスウィンドウ 786
SAS セッションのカスタマイズ 780
エクスプローラウィンドウのカスタマイズ 785
出力形式の設定 760
説明 786
開く 786
プログラミングウィンドウ 723
プログラミング言語
参照項目: SAS 言語
プログラム, 実行
参照項目: SAS プログラム, 実行
参照項目: プログラムエディタ
プログラムエディタ 753
関連項目: SAS ウィンドウ環境, ウィンドウ
概要 753
行コマンド 754
説明 769
定義 723
ファイルショートカット, 割り当て 759
プログラムの作成 757
プログラムのサブミット 757
プログラムのデバッグ 758
プログラムの編集 759
プログラムの保存 758
プログラムを開く 758
例 13
プログラムデータベクトル 36
プロジェクトエディタ
コマンド行のコマンド 753
プロシジャ 6
欠損値出力のカスタマイズ 622
プロシジャ, 説明と使用法
APPEND 276
CATALOG 676
CHART 540
CONTENTS 676
COPY 676
DATASETS 678, 690
FORMAT 622
OPTIONS 785
PLOT 519
PRINT 457
PRINTTO 389
REGISTRY 786
REPORT 510
SORT 194, 661
TABULATE 480
TEMPLATE 662
UNIVARIATE 540
ブロックチャート 548
プロット 519
関連項目: PLOT プロシジャ
関連項目: チャート
プロット, 1 つの組み合わせの変数
2 次元プロット 525
PLOT ステートメント 524
例 525
プロット, カスタマイズ 525
軸ラベル, 指定 525
外枠罫線 525
タイトル付け 537
凡例, 削除 529
プロット記号 528
目盛値 526
プロット, 複数の組み合わせの変数
同じ軸の複数の組み合わせ 534
それぞれ異なるページの複数のプロット 530
同一ページの複数のプロット 532
分析変数, 指定 465
分類変数
欠損値 466
指定 465
並べ替え 483
ページサイズ, 出力レポート 611
ページ番号付け
NONUMBER オプション 611
NUMBER オプション 611
PAGENO=オプション 611
出力レポート 611, 617
ヘッダー, レポート 503
関連項目: タイトル, レポート
SAS 出力ファイル 595
カスタマイズ 503
中央揃え 614
特定のカラム 616
変数 505
ヘルプ, SAS ウィンドウ環境
参照項目: SAS ウィンドウ環境, ヘルプ
変数 29
関連項目: SAS データセット
関連項目: オブザベーション

関連項目: 数値変数
関連項目: 文字変数
記憶域 116
効率的な使用法 114
作成 111
属性 258
定義 29, 43
長さの定義 116
比較 125
変更 113
命名規則 6
変数属性 258
変数の合計
参照項目: オブザベーション, 計算
参照項目: 数字の合計
変数の削除
参照項目: DATA ステートメント
参照項目: DROP=オプション
参照項目: DROP ステートメント
参照項目: SET ステートメント
変数の保持
参照項目: DATA ステートメント
参照項目: KEEP=オプション
参照項目: KEEP ステートメント
参照項目: SET ステートメント
ポインタ
結果ポインタの削除 764
結果ポインタの名前変更 764
出力の移動 763
ポインタコントロール 591
関連項目: カラムポインタコントロール
関連項目: ラインポインタコントロール
@n 591
ポインタ配置 62, 69
棒グラフ
縦, 作成 544
縦, 中間値 551
縦, 中間点の数 554
横 546

ま

マウス, 相当するキーボード操作 732
マクロ
生成された DATA ステップのデバッグ 406
デバッグコマンドのカスタマイズ 406
デバッグツール 406
マクロ機能
関連項目: SAS マクロ機能
DATA ステップデバッガ 406
マクロ変数
アンパサンド, 名前 455
参照 455
自動 453
詳細レポートのカスタマイズ 453
ユーザー定義 454
マスタデータセット
更新 316
更新エラー 340
定義 316
変更, オブザベーションの追加 337
変更, トランザクションデータセット 336
右クリック, 相当するキーボード操作 732
右シフトコマンド 745
命名規則
SAS 言語 6
SAS 名 6
入力形式 61
変数 6
メニュー, 表示 720
目盛, ヒストグラム 565
メンバ
コピー 676, 677
削除 714
メンバ, コンテンツのリスト
参照項目: CONTENTS ステートメント
参照項目: CONTENTS プロシジャ
参照項目: ファイルコンテンツ, リスト
文字グループ, 選択 162, 163, 164
文字の比較
大文字と小文字の区別 161
タイプ 161
文字変数 131
8 バイトよりも長い 64
値の位置調整 139
一部抽出 139
引用符('), 変数インジケータ 133
引用符('), リテラル文字 133
大文字小文字の区別 133
切り捨て 134
欠損値, 確認 137
欠損値, 設定 138
欠損値, ピリオド 136
欠損値, ブランク 136
コンテンツ 43
作成 43, 139
識別 133
定義 132
ドル記号($), 変数名 133
長さ, 決定 134
長さ, 最大 135
長さ, 設定 135
長さ, デフォルト 43
長さ, 表示 136
ブランク, 先頭削除 140
文字としての数字 143
文字列のスキャン 139
文字変数, 結合
参照項目: 文字変数, 連結
文字変数, 連結 141
簡単な連結 142

感嘆符(!!), 連結演算子 141
縦棒(||), 連結演算子 141
文字の追加 142
文字変数の連結
参照項目: 文字変数, 連結
文字列, スキャン 139

や

曜日, 検索 237
要約テーブル 462
記述統計量, 計算 475
クロス集計表 462, 473
欠損値 466
コードの削減 477
交差要素 462, 473
構造の定義 465
サブグループのレポート 473
次元式 465
出力先 480
出力のフォーマッティング 475
スタイル 480
すべての変数の要約 478
定義 462
入力データセット, 指定 465
分析変数, 指定 465
分類変数, 欠損値 466
分類変数, 指定 465
分類変数, 並べ替え 483
要素の組み合わせ 473, 476
要素の連結 476
ラベル, 定義 479
ラベル, 複数要素の単一化 477
要約テーブル, 作成
1 次元 468
2 次元 469
3 次元 470
PROC TABULATE ステップでの複数テーブル 471
階層テーブル 473
要約テーブル要素の組み合わせ 473, 476
要約テーブル要素の交差 462, 473
要約テーブル要素の連結 476
要約レポート 489
関連項目: レポート
作成 498, 509
定義 491
横棒グラフ 546
統計量 546

ら

ライブラリコンテンツ, リスト 676, 677
ライブラリ参照名 668
LIBNAME ステートメントを使用して割り当てる 668
SAS ウィンドウ環境での割り当て 734
USER, 予約名 672
ライブラリ情報, リスト 676, 677
ライブラリのエイリアス
参照項目: ライブラリ参照名
ラインホールド指定子
生データの読み込み 72, 74, 89
行の固定 587, 597
出力行の書き込み 587, 597
ラインポインタコントロール 81
関連項目: カラムポインタコントロール
関連項目: ポインタコントロール
#n, DATA ステップの実行 82
#n, 入力変数のスキップ 81
ラインモード 725
ラベル
小計と総計 438
ラベル, SAS データセット
参照項目: SAS データセット, ラベル
ラベル, 要約テーブル
定義 479
複数要素の単一化 477
ラベル属性 258
リスト
参照項目: レポート
リスト入力 53
関連項目: カラム入力
関連項目: フォーマット入力
関連項目: 生データレコードの読み込み
SAS データセットの作成 42
アンパサンドフォーマット修飾子 64
埋め込み特殊文字 63
埋め込みブランク 63
カラム入力との比較 58
規則 56
区切り文字 491
コロン(:)フォーマット修飾子 64
修飾リスト入力 63
定義 53
長い文字変数の作成 63
入力ポインタ 68
複数の入力スタイルの使用 65
ブランク区切り文字 53
文字区切り文字 54
ループ
参照項目: DO グループ
参照項目: 配列処理
参照項目: 反復 DO ループ
レコード, SAS データセット
参照項目: オブザベーション
レコード, 生データ 29
レジストリ, 編集
参照項目: SAS レジストリ, 編集

レジストリファイル
アンインストール 779
インポート 778
エクスポート 778
列, SAS データセット
参照項目: 変数
列, レポート
スペース 502
並べ替え 495
幅 502
レイアウト 495
列(生データ) 29
列ヘッダー, レポート
カスタマイズ 503
中央揃え 614
特定のカラム 616
変数 505
レポート 489
関連項目: ODS
関連項目: PRINT プロシジャ
関連項目: REPORT プロシジャ
関連項目: 出力
関連項目: 出力, カスタマイズ
関連項目: 要約レポート
ACROSS 変数 491, 505, 507
ANALYSIS 変数 491, 498
COMPUTED 変数 492
DATA ステップ 584
DISPLAY 変数 492
GROUP 変数 492, 498
ORDER 変数 492
オブザベーション, 選択(複数の比較) 433
オブザベーション, ページによるグループ化 444
オブザベーション, まとめる 498
オブザベーション, 要約 508
オブザベーションの列, 非表示 426
改ページ 444
数, フォーマット 504
カスタマイズ 446
行, 並べ替え 496, 499
行レイアウト 496, 499
区切り行 508
グループの合計, 計算 443
グループの小計, 単一の変数の計算 437
グループの小計, 複数の変数の計算 441
グループ要約 509
クロス集計表 507
作成 490
従来の出力 8
種類 490
すべての変数の表示 424, 493
セル 505
選択した変数のレポート作成 430
タイトル 446, 447
度数カウント 506
複数行の列ラベル 449
フットノート 446, 447
ブランク行の追加 450
ヘッダー, シンボリック値 617
マクロ変数の使用 454
レイアウト, 作成 491
レイアウト, 調整 502
列, 並べ替え 495
列スペース 502
列の配置 502
列幅 502
列ヘッダー, カスタマイズ 503
列ヘッダー, 変数 505
列ラベル, 定義 425, 448
列レイアウト 495
レポート項目のフォーマット 504
ログメッセージ, 出力 387
レポート, SAS 出力ファイル 584
PUT ステートメント 584
行サイズ 720
行の出力先の指定 589
グループごとに値を印刷 593
合計の計算 594
固定された行の解除 588
数値データ値, 出力形式 592
データ値 591
同一行の再書き込み 587
フットノート 595
ヘッダー 595
変数値 586
文字列 585
レポートのレイアウトの設計 590
レポート項目のフォーマット 504
レポート作成ツール 490
レポートの区切り行 508
レポートの作成
参照項目: PRINT プロシジャ
参照項目: REPORT プロシジャ
参照項目: レポート
連結演算子
感嘆符(!!) 141
縦棒(||) 141
ログ
参照項目: SAS ログ
ログウィンドウ 769
関連項目: SAS ウィンドウ環境, ウィンドウ
SAS ログ出力 387
クリア 387
コンテンツの保存 388
参照 727
説明 769
定義 723

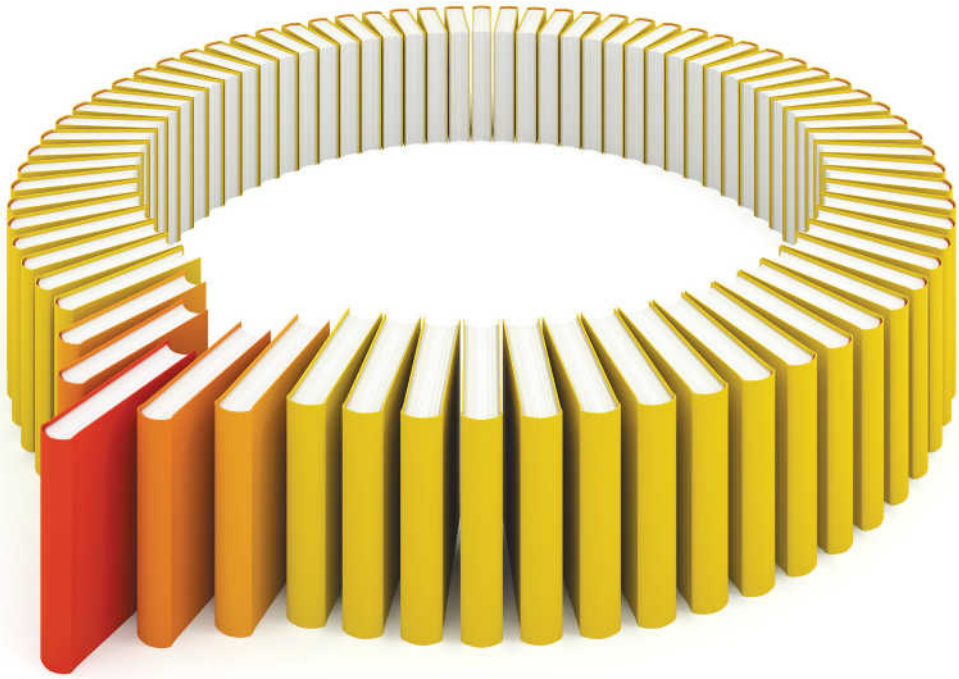
プログラムのデバッグ 756
論理演算子 125
数値式 123
説明 118, 129

わ

割り当てステートメント
DATA ステップ 111
概要 111
算術演算子 121

大

大かっこ, STYLE=オプション 481
大なり記号, DATALINES ステートメント
での使用 725



Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.