九州大学学術情報リポジトリ
Kyushu University Institutional Repository

SASプログラミング

田中, 省作
九州大学情報基盤センター外国語情報メディア研究部門

https://doi.org/10.15017/1470473

出版情報：九州大学情報基盤センター広報 : 全国共同利用版. 3 (1), pp.29-54, 2003-03. 九州大学情報基盤センター
バージョン：
権利関係：

KYUSHU UNIVERSITY

SASプログラミング

田中省作 *

1 はじめに

SASは "Statistical Analysis System" の略で, 1960年代に大型汎用計算機用に開発された汎用統計解析システム[1]です. SASはさまざまな統計解析をお手軽に実行することができます. そこで本稿では, SASの事始めとして, (1変数の)データ要約を目標に, その使用法を紹介します.

2 準備

九州大学情報基盤センター (以後, 本センターと記します) では, 汎用UNIXサーバ (富士通GP7000Fモデル900; kyu-cc.cc.kyushu-u.ac.jp) にSASが導入されています[2]. SASはいくつかのオプション・システムからなり, 本センターで使用可能なものは, 次の5つです.

1. Base SAS: SASの基本システムで, データ入力・管理・記述統計処理などを行います.
2. SAS/STAT: 多変量解析など, 高度な統計処理を行います.
3. SAS/GRAPH: データや解析結果を可視化します.
4. SAS/ETS: 計量経済分析や時系列分析などを行います.
5. SAS/IML: 対話的に行列演算を行います.

本センターのSASの利用については,

http://www.cc.kyushu-u.ac.jp/scp/system/library/SAS/SAS.html

もご参照下さい. 本稿ではSASのインタフェースとして, 基本的にX-Window上で対話的にSASを使用するディスプレイマネージャ・モードを考えます.

2.1 SASの起動と終了

SASを起動するには, まず本センターの汎用UNIXサーバに接続します. 接続方法については,

http://www.cc.kyushu-u.ac.jp/scp/system/general/GP7000F/01_login.html

をご参照下さい. 接続後, 次のように起動コマンドを与えます.

```
kyu-cc% sas &
```

SASが正しく起動すると, "OUTPUT", "PROGRAM EDITOR", "LOG", "TOOLBOX" という4つのウィンドウが現れます.

- PROGRAM EDITORウィンドウ: SASプログラムを編集するウィンドウです (図1).

*情報基盤センター 外国語情報メディア研究部門 E-mail: sho@cc.kyushu-u.ac.jp

[1]SAS Institute Japan Ltd., http://www.sas.com/japan

[2]2003年1月時点でのバージョンは6.12です.

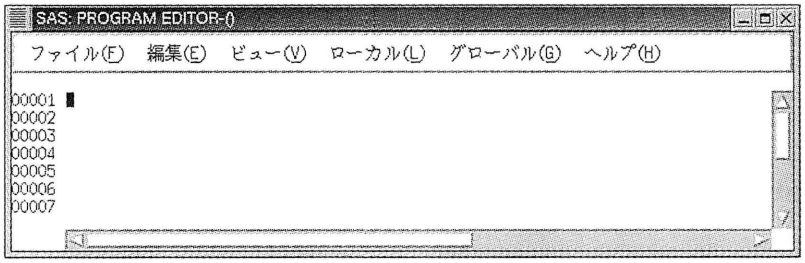


図 1: PROGRAM EDITOR ウィンドウ

- LOG ウィンドウ: SASプログラムの実行時の各種メッセージ (プログラムのエラー, リソースに関する情報など) が表示されるウィンドウです (図 2). ログはデバッギング (プログラムのエラーの修正作業) などでも非常に有用となります.

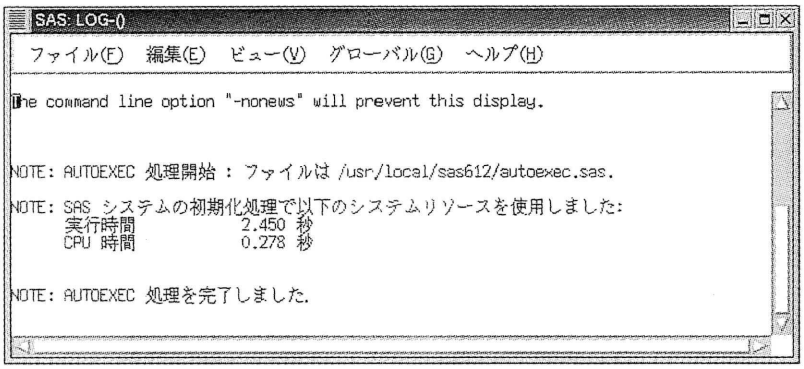


図 2: LOG ウィンドウ

- OUTPUT ウィンドウ: SASプログラムの実行結果が表示されます (図 3).
- TOOLBOX ウィンドウ: 頻繁に使う機能 (「サブミット (プログラムの実行)」やファイルのオープン」など) がアイコンで集められています (図 4).

その他, SAS/GRAPH でグラフを描画すると, グラフ・ウィンドウが現れます (4.4 節).
SAS を終了するには, PROGRAM EDITOR または LOG ウィンドウで,

[ファイル] → [SAS の終了...]

を選択します. ただし, [α] は “ウィンドウのメニューバーのメニュー α” を表すことにします.

2.2 PROGRAM EDITOR の基本操作

SASプログラムを編集するための PROGRAM EDITOR の基本操作を紹介します.

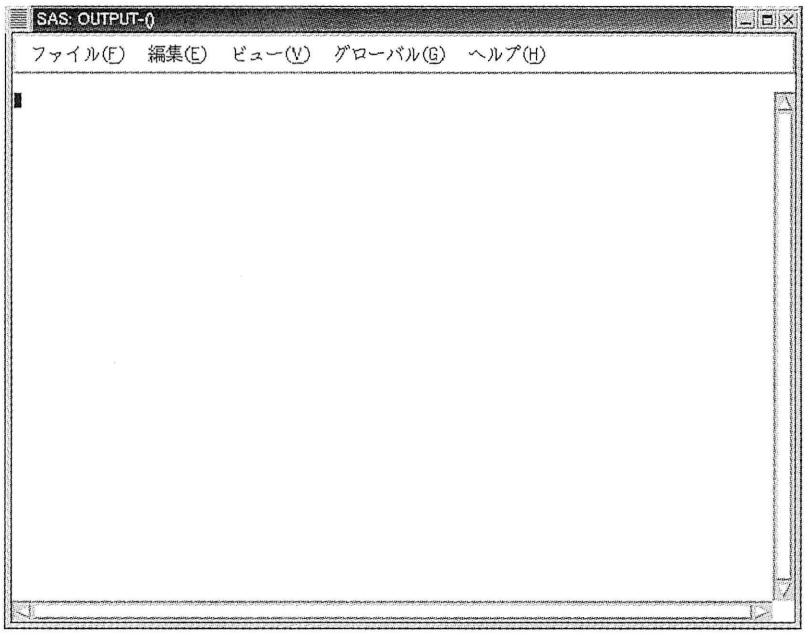


図 3: OUTPUT ウィンドウ

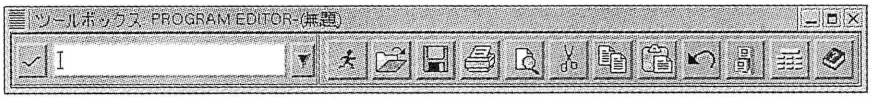


図 4: TOOL BOX ウィンドウ

挿入モードと上書きモードの切り替え

起動時は上書きモード (カーソルが黒のボックス; ■) で, 挿入モード (カーソルがアンダーライン; _) には,

[Insert]

で切り替えることができます. ただし, [α] は "α というキー" を表すことにします.

1 字削除

カーソルの現在の位置の文字を消すには,

[Delete]

を押します. また, 1 つ前の位置の文字を消すには,

[BackSpace]

を押します.

n 行削除

行単位で削除することもできます. PROGRAM EDITOR の行番号部分にカーソルを移動し,

[d] → n → [Enter]

と入力すると, カーソルの行以降 n 行が一括して削除されます[3].

1 行のみ削除するには, 行番号部分で,

[d] → [Enter]

でも可能です.

領域単位の削除

n 行削除では, 削除する領域の行数を予め数えておかなくてはならないので繁雑です. そこで, 領域を明示的に指定してばっさりと削除する方法も紹介しておきます.

まず, 削除したい領域の始めの行番号部分で,

[d] → [d]

と入力します (このとき, 行番号部分に赤字で "DD" と表示されます). 次に, 削除したい領域の終わりの行番号部分で再び,

[d] → [d]

と入力すると, 指定した領域が削除されます.

[3] n の部分は, 例えば 10 行削除の際は, [1] → [0] と入力します.

n行挿入

空行を挿入することができます. n行の空行を挿入するには, 行番号部分で,

$$\boxed{\text{i}} \rightarrow n \rightarrow \boxed{\text{Enter}}$$

とします.

1行のみ挿入するには, 行番号部分で,

$$\boxed{\text{i}} \rightarrow \boxed{\text{Enter}}$$

でも可能です.

SASプログラムは必ずしもこのPROGRAM EDITORで作成する必要はありません. 自分の使い慣れたテキスト・エディタでSASプログラムを記述し, それをPROGRAM EDITORにロードして実行しても構いません. また, ここで挙げているように対話的に実行するのではなく, 非対話的に実行することも可能です (付録B).

2.3 プログラムのロードとセーブ

ファイルに保存しているSASプログラムをPROGRAM EDITORにロードするには, まず,

[ファイル] → [オープン]

を選択します. すると, ファイル指定のダイアログ・ウィンドウ (図5) が開くので, そこでロードしたいファイルを指定します.

現在編集しているSASプログラムを保存するには, 次のような操作を行います. まだファイル名を付けていない場合や別のファイル名で保存したい場合は,

[ファイル] → [新規保存]

を選択すると, ファイル指定のダイアログ・ウィンドウが開くので, そこで保存したいファイル名を指定します.

また, 既にファイル名が付いている場合は,

[ファイル] → [保存]

を選択すれば上書き保存されます.

2.4 SASプログラムの実行

SASプログラムを実行するには, まず, PROGRAM EDITORでプログラムを作成, またはプログラムをロードした後,

[ローカル] → [サブミット]

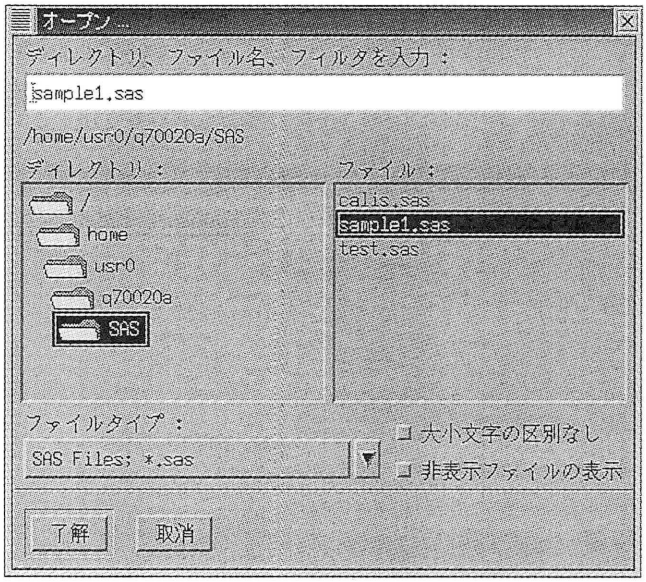


図 5: SASプログラム・ファイルの指定

を選択します．プログラムの実行結果は，OUTPUT ウィンドウに表示されます．また，LOG ウィンドウには，実行時のデータやリソースに関する情報，プログラムに誤りがある場合は，そのエラーや警告といった情報が表示されます．

プログラム実行後，プログラムは PROGRAM EDITOR から消去されます．実行したプログラムを再度 PROGRAM EDITOR に表示するには，

[ローカル] → [テキスト呼び出し]

を選択します[4]．

2.5　サンプル・プログラム

簡単なサンプル・プログラムを例に，SAS で実際に実行する様子を見てみましょう．まず，次のような身長・体重のデータを考えましょう．

```
1 M 175 69
2 M 167 53
3 F 156 42
4 F 170 50
5 M 190 80
```

データは`sample.dat`というファイルに保存されていて，1 行に 1 人分のデータが書かれているとします．1 列目が識別番号，2 列目が性別 (`M`が男性，`F`が女性)，3 列目が身長 (cm)，4 列目が体重 (kg) です．

[4]直前に実行したプログラムを再度実行する場合でも，この操作で PROGRAM EDITOR に呼び出す必要があります．

やりたいことは, 体重と身長の平均・標準偏差といった要約統計量の算出としましょう. そのためのSASプログラムの1つが次のようなものです.

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
PROC MEANS DATA=sample1;
RUN;
```

プログラムの内容は次節で説明することにして, ここでは実際にSASプログラムが実行されるまでを追ってみることにします. まず, PROGRAM EDITORで上のプログラムを入力します. SASプログラムの各文は必ず ";"(セミコロン) で終わります. 入力後, 打ち間違いなどがないことを確認して, 実行です. 実行は,

[ローカル] → [サブミット]

でした. うまく実行されると, 図6のように実行結果がOUTPUTウィンドウに表示されるはずです[5]. OUTPUTウィンドウでは, 左端にID,HEIGHT,WEIGHTと識別番号・身長・体重のラベルが表示され, それぞれ1行ごとに, そのデータの個数 (N), 平均 (Mean), 標準偏差 (Std Dev), 最小値 (Minimum), 最大値 (Maximum) が表示されています. この例では, 身長は平均171.6cm, 標準偏差が約12.4cm, 最小値が156cm, 最大値が190cmということが分かります. 簡単に計算することができました.

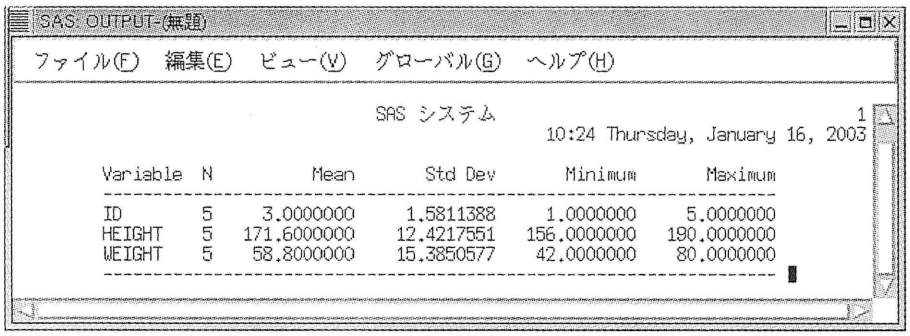


図 6: サンプル・プログラムの実行結果 (OUTPUTウィンドウ)

では, ここからはこのサンプル・プログラムを例に, SASプログラムの概要を解説していきます.

3 SASプログラミング

SASプログラムは, **DATA**ステップと**PROC**ステップの繰り返しで構成されます. DATAステップとは, SAS内部にデータセットとよばれる統計解析を施す対象となるデータ群を作成

[5]OUTPUTウィンドウになにも表示されない場合は, プログラムに誤りがあった可能性があります. LOGウィンドウを見てみましょう.

する段階です. PROC ステップは, 特定のデータセットに対してプロシジャとよばれる統計解析を適用する段階です.

本稿では, SAS 言語の予約語については大文字で表記し, プロシジャ名, データセット名や変数名については小文字で表記することにします[6].

3.1 DATA ステップ

DATA ステップは, SAS 内部にデータセットとよばれるデータ群を作成します. まず, サンプル・プログラムの単純な DATA ステップを見てみましょう. その後, データセットを加工・編集するような少し複雑な DATA ステップも紹介します.

3.1.1 サンプル・プログラムにおける DATA ステップ

SAS では, 観測 (入力) されたデータをオブザベーションとよびます. i 番目に入力されたオブザベーションには i という値 (OBS) がふられます. オブザベーションは, さらに各変数に対する値で構成されています. データセットはオブザベーションの集合です. データセットにはデータセット名という名前がつけられます. データセットは次のような構造になります[7].

データセット名: *dataset*

OBS	変数 1	変数 2	···	変数 p
1				
2				
⋮				
n				

サンプル・プログラムの DATA ステップ部分を見てみましょう.

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
```

まず, 最初の DATA 文により, 「ここから, sample1 という名前のデータセットを作ります」ということを宣言しています. これで, 新たな DATA 文や PROC 文に出会うまで sample1 というデータセットを作成することになります.

2 行目の INFILE 文でデータを読み込むファイルを指定します (ファイル名は, "'"(シングルクォート) でくくります). ここでは, sample.dat というファイルからデータセット sample1 を作成するためのデータを読み込みます.

3 行目の INPUT 文で, sample.dat からデータを読み込む際のフォーマットを指定しています. この場合, 「1 列目のデータに id, 2,3,4 列目のデータをそれぞれ sex, height, weight という変

[6] 実際には, プロシジャ名, データセット名, 変数名などに大文字/ 小文字の区別はありません. ファイル名については SAS の稼働する OS に依存します. 本センターの場合, サーバの OS は UNIX なのでファイル名については大文字/小文字が区別されることになります.

[7] 正確には, 上記した部分はデータセットのデータ部です. 実際には, データ部に加え, ディスクリプタ部が存在します. ディスクリプタ部には, 変数や作成に関する情報が記録されており, CONTENTS プロシジャにより参照できます. 本稿では, ディスクリプタ部の詳細については割愛します.

数に入れる」ということになります. SASの変数には, 数値変数と文字列変数があります. sexの後には "$"(ダラー)が付いているので, 2列目のデータは文字列データとして扱われます.

DATAステップでは, ファイルから1行づつデータを読み込み, オブザベーションとして順次, 変数に値を割り当てていきます. SAS内部では最終的に次のようなデータセットが構成されます.

データセット名: sample1

OBS	id	sex	height	weight
1	1	M	175	69
2	2	M	167	53
3	3	F	156	42
4	4	F	170	50
5	5	M	190	80

このサンプル・プログラムではデータセットを構成するのに, 解析対象のデータを外部ファイルから読み込みました. それとは別に, これらのデータをプログラム中に直接記述しておくこともできます. その場合, DATAステップでは次のようにCARDS文を使います.

```
DATA sample1;
INPUT id sex $ height weight;
CARDS;
 1 M 175 69
 2 M 167 53
 3 F 156 42
 4 F 170 50
 5 M 190 80
;
```

サンプル・プログラムのINFILE文の代わりにCARDS文を書きます. その後, sample.datの中身同様にデータを直接記述しておけば良いことになります. 今回のように解析対象のデータが小さい場合には, プログラム中に直接記述しても良いでしょう.

3.1.2 複数のデータセットの作成

SASプログラムでは, 複数のデータセットを構成することも可能です. 例えば, sample2.datに別のグループの識別番号・性別・身長・体重のデータが入力されているとします.

```
 6 F 153 43
 7 M 179 77
 8 M 171 54
 9 F 148 39
10 M 154 45
11 F 166 45
12 M 171 59
```

DATA ステップで,

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
DATA sample2;
INFILE 'sample2.dat';
INPUT id sex $ height weight;
```

とすると, SAS には 2 つのデータセット sample1,sample2 が作成されることになります.

データセット名: sample1

OBS	id	sex	height	weight
1	1	M	175	69
2	2	M	167	53
3	3	F	156	42
4	4	F	170	50
5	5	M	190	80

データセット名: sample2

OBS	id	sex	height	weight
1	6	F	153	43
2	7	M	179	77
3	8	M	171	54
4	9	F	148	39
5	10	M	154	45
6	11	M	166	45
7	12	M	171	59

後述するプロシジャを適用する際, 複数のデータセットが存在する場合には, 対象を適宜選択することになります.

3.1.3 変数の付加 (代入文)

データをファイルから読み込む際に, 新たな変数を付加することができます. サンプル・プログラムを例に, 識別番号・性別・身長・体重に加え, BMI 値[8]を加えることにしましょう.

```
DATA sample3;
INFILE 'sample.dat';
INPUT id sex $ height weight;
bmi = weight/(height/100)**2;
```

bmi = weight/(height/100)**2; は代入文とよばれ, オブザベーションごとに weight/(height/100)**2; を計算し, bmi の値とします. その結果, 変数bmi が付加され, 次のようにデータセットが作成されます.

[8]BMI(Body Mass Index) とは, 身長と体重から計算される値で, 肥満度の指標として広く用いられています. 次式で計算されます.

$$\text{BMI 値} = \frac{\text{体重 (kg)}}{\{\text{身長 (m)}\}^2}$$

大雑把には, BMI 値 20 ～ 24 が普通, 20 未満がやせぎみ, 24 より大きいと太りぎみ, となります.

データセット名: sample3

OBS	id	sex	height	weight	bmi
1	1	M	175	69	22.5···
2	2	M	167	53	19.0···
3	3	F	156	42	17.2···
4	4	F	170	50	17.3···
5	5	M	190	80	22.1···

SASの算術演算子を表1にあげておきます.

演算子	意味
+	加算
-	減算
*	乗算
/	除算
**	べき乗

表 1: 算術演算子

3.1.4 条件によるデータの選択 (IF文)

IF文を使えば, 条件付きで実行を分岐させることができます. 例えば, BMI値に応じて, 次のような評価を付加することを考えましょう. 評価はcという変数で付加することにします.

$$\begin{cases} \text{BMI} < 20 \Rightarrow \text{痩せぎみ} \\ \text{BMI} > 24 \Rightarrow \text{太りぎみ} \\ \text{それ以外} \Rightarrow \text{ふつう} \end{cases}$$

SASでは“’”(シングル・クォーテーション)で囲めば, 日本語も記述することができます (文字コードはEUC日本語コード). IF文を使って次のように書きます[9].

```
DATA sample4;
INFILE 'sample.dat';
INPUT id sex $ height weight;
bmi = weight/(height/100)**2;
IF bmi<20 THEN c='痩せぎみ';
  ELSE IF bmt>24 THEN c='太りぎみ';
    ELSE c='ふつう';
```

IF bmi<20 THEN c='痩せぎみ'; ELSE ...; でbmi<20の条件が成り立つ (真) とき, c='痩せぎみ'が実行されます. 一方, 条件が成り立たない (偽) ときは, ELSE文が実行されますが, ELSE文以下が無い場合は, なにも実行されません. このプログラムでは, 次のようなデータセットが作成されます.

[9]この例のプログラムでは, 読み易いようにインデントを入れています. SASでインデントはプログラムの動作そのものには影響しません. 本稿では, プログラムの読み易さのために, 適宜インデントを入れています.

データセット名: sample4

OBS	id	sex	height	weight	bmi	c
1	1	M	175	69	22.5···	ふつう
2	2	M	167	53	19.0···	痩せぎみ
3	3	F	156	42	17.2···	痩せぎみ
4	4	F	170	50	17.3···	痩せぎみ
5	5	M	190	80	22.1···	ふつう

SASの比較演算子, 論理演算子を表2,3にあげておきます.

比較演算子	意味
=またはEQ	等しい
^=またはNE	等しくない
<またはLT	より小さい
<=またはLE	以下
>またはGT	より大きい
>=またはGE	以上

表 2: 比較演算子

論理演算子	意味
&またはAND	かつ
|またはOR	または
^またはNOT	ではない

表 3: 論理演算子

3.1.5　オブザベーションの除去 (DELETE 文)

DATA ステップでは, データセットを作成するわけですが, ある特定のオブザベーションはデータセットの作成対象から除くことができます. 例えば, サンプル・プログラムで「男性のみ」のデータセットを作成することを考えましょう. IF 文とDELETE 文を使って次のように書きます.

```
DATA sample5;
INFILE 'sample.dat';
INPUT id sex $ height weight;
IF sex='F' THEN DELETE;
```

IF sex='F' THEN DELETE; で条件sex='F' が真の場合, つまり「女性の」オブザベーションの場合は, THEN 以下のDELETE 文が実行されます. DELETE 文は, 現在処理しているオブザベーションを除去します. その結果, 次のようなデータセットが作成されます.

データセット名: sample5

OBS	id	sex	height	weight
1	1	M	175	69
2	2	M	167	53
3	5	M	190	80

3.2 PROCステップ

PROCステップは, DATAステップで作成したデータセットに対して統計解析を施す段階です. SASには予めプロシジャという解析ルーチンが準備されています. PROCステップでは, これらのプロシジャと解析対象のデータセットを指定します.

3.2.1 サンプルプログラムにおけるPROCステップ

サンプル・プログラムでは, 以下の部分がPROCステップになります.

```
PROC MEANS DATA=sample1;
```

sample1というデータセットに対して, MEANSというプロシジャが呼び出されています. MEANSは, 各変数の平均・標準偏差・最小値・最大値を求めるプロシジャです. ただし, サンプル・プログラムでsexは文字列変数なので, 自動的に解析対象から除かれます.

DATA=sample1で, MEANSプロシジャをデータセットsample1に対して適用することを明示しています. もしsample2というデータセットが存在し, MEANSプロシジャ呼び出しでDATA=sample2とすれば, sample2の各変数に対して平均・標準偏差・最小値・最大値を求めることになります. プロシジャ呼び出しで, 特に最後に作成したデータセットを対象とする場合に限り, DATA=部分は省略可能です.

3.2.2 対象とする変数の選択 (VAR文)

サンプル・プログラムでは, 全ての変数をMEANSの対象としましたが, 実際には, 識別番号の平均や標準偏差を求めても意味はありません. そこで, プロシジャ呼び出しの際に, VAR文で解析対象の変数を明示的に指定することができます. 逆に, (他のプロシジャでも)VAR文で解析対象の変数を指定しなければ, 基本的にデータセットの全変数が対象となります.

PROCステップを次のように,

```
PROC MEANS DATA=sample1;
  VAR weight height;
```

とすると, 図7のように身長・体重に対する解析結果のみが出力されます. また, VARに書いたweight,heightという順番は, 出力順序にも反映されます.

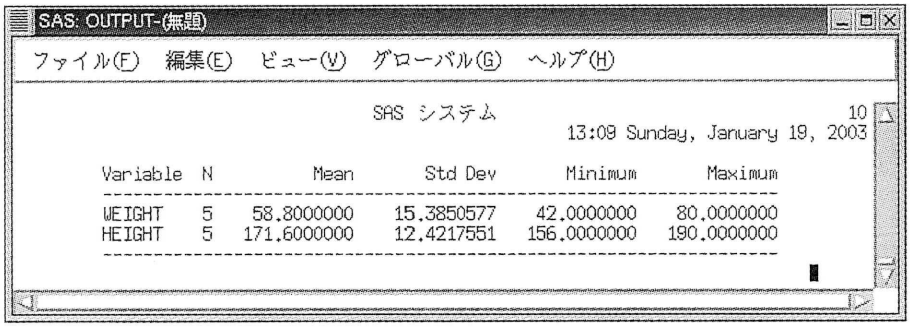


図 7: height,weight のみを解析対象とした結果

3.2.3　条件によるオブザベーションの選択 (WHERE 文)

サンプル・データでは, 全てのオブザベーションを解析対象としましたが, 例えば, 「男性」かつ「身長 170cm 以上」のオブザベーションを対象としたい場合があります. このようなとき, PROC ステップのプロシジャ呼び出しに, WHERE 文を加えることで解析対象となるオブザベーションを限定することができます. ここでは, 「男性」かつ「170cm 以上」のオブザベーションを解析対象としたい場合を考えてみましょう.

```
PROC MEANS DATA=sample1;
  WHERE sex='M' AND height>=170;
  VAR weight height;
```

WHERE sex='M' AND height>=170; で, MEANS の解析対象は, sex='M' と height>=170 という条件が共に真となるオブザベーションのみになります. 結果, 図 8 のように, 2 つのオブザベーションのみが MEANS の解析対象となりました[10].

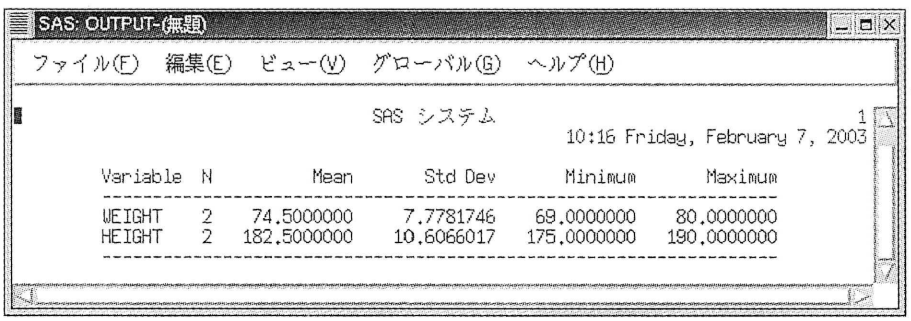


図 8: WHERE で解析対象を限定した結果

[10]DATA ステップで IF 文を使って, 別に「男性」かつ「身長 170cm 以上」のデータセットを作成し, そのデータセットにプロシジャを適用する, という方法もあります.

3.2.4 変数の値ごとに処理 (BY文)

特定の変数の値でオブザベーションをグループ分けして, それぞれに同じ解析を行いたいという場合があります. 例えば, 変数sexの値, 男性・女性で別々にMEANSを適用したいといった場合です. このようなときは, PROCステップのプロシジャ呼び出し時に, BY文で変数を指定することで, その変数の値ごとに繰り返しプロシジャが適用されます. ここでは, BY文で性別ごとに身長・体重を解析対象としたMEANSプロシジャを適用してみましょう. サンプル・プログラムのPROCステップを,

```
PROC SORT;
  BY sex;
PROC MEANS DATA=sample1;
  BY sex;
```

とします. 注意しなくてはならないのは, MEANSプロシジャ呼び出しの前にSORTプロシジャを呼び出していることです. BY文で変数の値ごとにプロシジャを繰り返し適用するには, あらかじめデータセット内のオブザベーションをその変数 (ここではsex) でソートしておかなければなりません[11]. SORTプロシジャでは, つづくBY sexによって, 変数sexをソート・キーに設定しています. その結果, オブザベーションはsexの値でソートされます (4.2節). その後, MEANSプロシジャを適用しています. MEANSプロシジャ呼び出しにつづくBY sexによって, 変数sexの値ごとにプロシジャが繰り返し適用されます. 図9のように結果が表示されます[12].

3.3 DATA/PROCステップ以外の文

SASプログラムは以上のDATAステップ, PROCステップに関連するものがほとんどです. ここではそれ以外のRUN文, コメント文を紹介します.

3.3.1 RUN文

サンプル・プログラムで最後に,

```
RUN;
```

とRUN文を記述しています. これは, それまでのPROCステップで指定したプロシジャを実行します. RUN文が無いと, SASプログラムを実行しても, データセットが作成されるだけでプロシジャは適用されません.

[11]もし変数sex でソートせずにBY 文付きのMEANS プロシジャを呼び出すと, 『ERROR: データセットWORK.SAMPLE1 は 昇順 ソートされていません. 現在のBYグループはSEX = Mで 次のBYグループはSEX = Fです. NOTE: エラーのため, このステップの処理を中止しました.』というメッセージがLOGウィンドウに表示され, プログラムの実行が中途で終了します.

[12]DATAステップでIF文を使って, 「男性」のみのデータセット, 「女性」のみのデータセット, と複数作成しておいて, 各々プロシジャを適用する, という方法もあります.

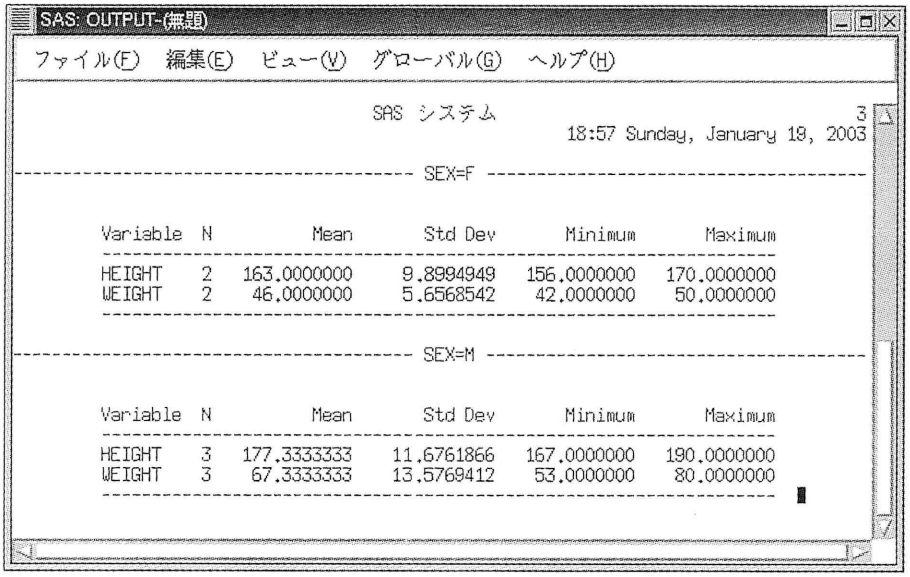


図 9: BY 文による処理の繰り返し

3.3.2　コメント文

多くのプログラミング言語でそうであるように, SASプログラムでもコメントを与えることができます. "COMMENT" または "*"(アスタリスク)で始まって ";" で終わる箇所はコメントとなり, プログラムの動作そのものには全く影響しません. また, C 言語同様 "/*" で始まって "*/" で終わる箇所もコメントとなります. 例えば,

```
* 識別番号・性別・身長・体重データの読み込み;
DATA sample1;
INFILE 'sample.dat';
```

といった具合にコメントを与えることができます.

4　SASプログラムの例

ここからは, データ要約の際に便利なプロシジャを紹介します.

4.1　データセットの表示 (PRINT プロシジャ)

データセットを表示するのが, PRINTプロシジャです. さきのサンプル・プログラムで,

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
PROC print DATA=sample1;
RUN;
```

とすると, 図10のようにデータセットが表示されます.

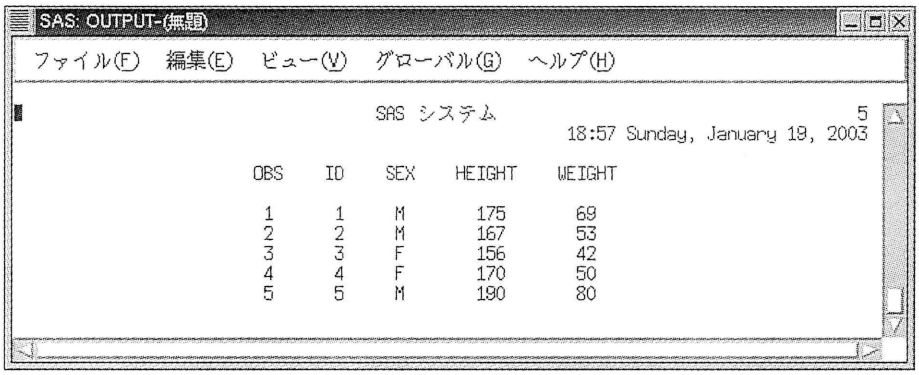


図 10: PRINT 文によるデータセットの表示

4.2 オブザベーションのソート (SORTプロシジャ)

BY文(3.2.4節)でも紹介しましたが, SORTプロシジャを使えば, ある変数をもとにデータセット内のオブザベーションをソートすることができます. ソート・キーの変数の指定は, SORTプロシジャ呼び出しにつづくBY文で行います. サンプル・プログラムを次のように,

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
PROC SORT DATA=sample1;
  BY height weight;
RUN;
```

とすると, ソートの最優先キーの変数はheight, 次に優先されるキーがweightということになります. 以上を実行すると, sample1データセットがソートされ, 次のように置き換えられます.

データセット名: sample1

OBS	id	sex	height	weight
1	1	M	175	69
2	2	M	167	53
3	3	F	156	42
4	4	F	170	50
5	5	M	190	80

$\Longrightarrow$

データセット名: sample1

OBS	id	sex	height	weight
1	3	F	156	42
2	2	M	167	53
3	4	F	170	50
4	1	M	175	69
5	5	M	190	80

また, 上のようにsample1 自身が変更されるのが不都合な場合は, SORT 呼び出しの部分で, 次のようにOUT=でソートしたデータセットを別のデータセットに出力できます.

```
PROC SORT DATA=sample1 OUT=sorted_sample1;
```

こうすると, 次のように複数データセットが作成されることになります.

データセット名: sample1

OBS	id	sex	height	weight
1	1	M	175	69
2	2	M	167	53
3	3	F	156	42
4	4	F	170	50
5	5	M	190	80

データセット名: sorted_sample1

OBS	id	sex	height	weight
1	3	F	156	42
2	2	M	167	53
3	4	F	170	50
4	1	M	175	69
5	5	M	190	80

4.3　度数分布表の作成 (FREQ プロシジャ)

SAS では度数分布表が容易に作成できます. 度数分布表の作成には, FREQプロシジャを使います. どの変数で度数をとるかは, FREQプロシジャ呼び出しにつづくTABLES 文で指定します.

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
PROC FREQ DATA=sample1;
  TABLES height;
RUN;
```

このようにすると, height の変数の値で度数をとり, 図 11 のような結果 (度数・相対度数 [%]・累積度数・累積相対度数 [%]) が得られます.

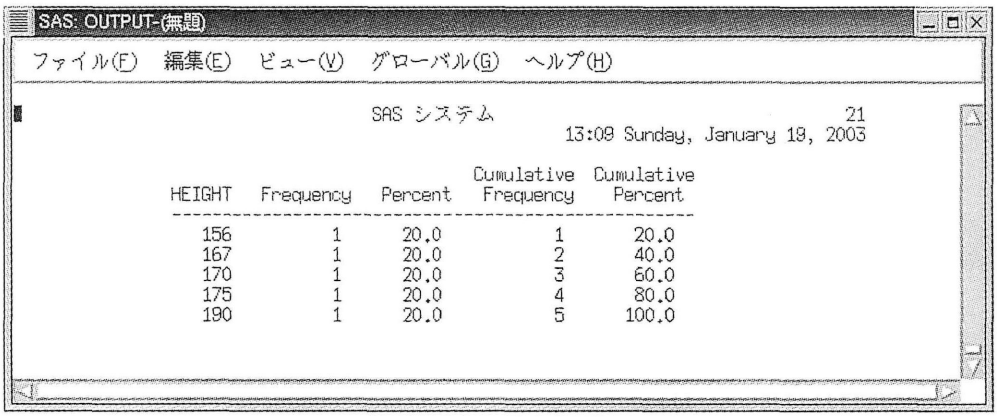


図 11: height による度数分布表

この例では, height で度数をとっても, 値が細かくてあまり有意ではありません. そのような場合は, 適当にグルーピングを行った後, 度数をとると見通しがよくなります. ここでは, 身長を 10cm 刻みで "170cm 以上-180cm 未満" といったグルーピングをした後, 度数をとるように変更しましょう. 以下がそのプログラム例です.

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
  IF height<150 THEN hc='   -150';
    ELSE IF height<160 THEN hc='150-160';
      ELSE IF height<170 THEN hc='160-170';
        ELSE IF height<180 THEN hc='170-180';
          ELSE hc='180-   ';
PROC FREQ DATA=sample1;
  TABLES hc;
RUN;
```

このプログラムでは, DATA ステップでheightをもとにして, 各オブザベーションにhcという変数を付加しています. 次のようなデータセットが作成されます.

データセット名: sample1

OBS	id	sex	height	weight	hc
1	1	M	175	69	170-180
2	2	M	167	53	160-170
3	3	F	156	42	150-160
4	4	F	170	50	170-180
5	5	M	190	80	180-

hc でFREQプロシジャを呼び出すと, 図 12 のような度数分布表が得られます.

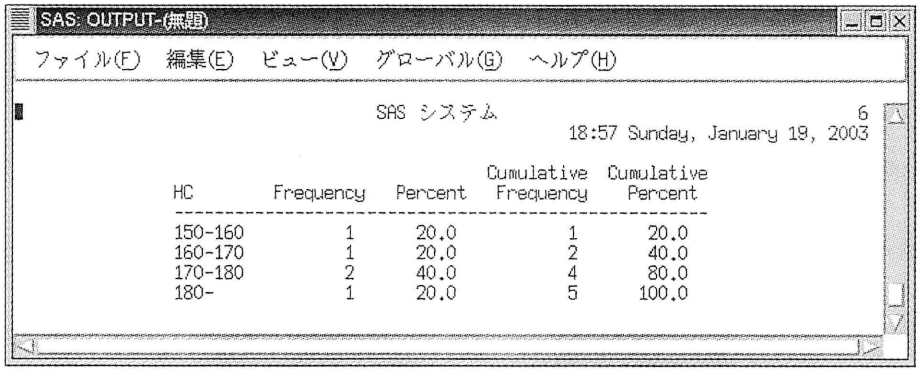


図 12: hc による度数分布表

FREQプロシジャでは, クロス集計表も容易に作成できます. TABLES 文で, クロス集計にとりたい変数を "*"(アスタリスク) で区切って列挙します. 例えば, 変数sex とhc でクロス集計をとるには, PROC ステップのFREQプロシジャ呼び出しを,

```
PROC FREQ DATA=sample1;
  TABLES sex*hc;
```

とします. その結果, 図 13 のようなクロス集計表がえられます.

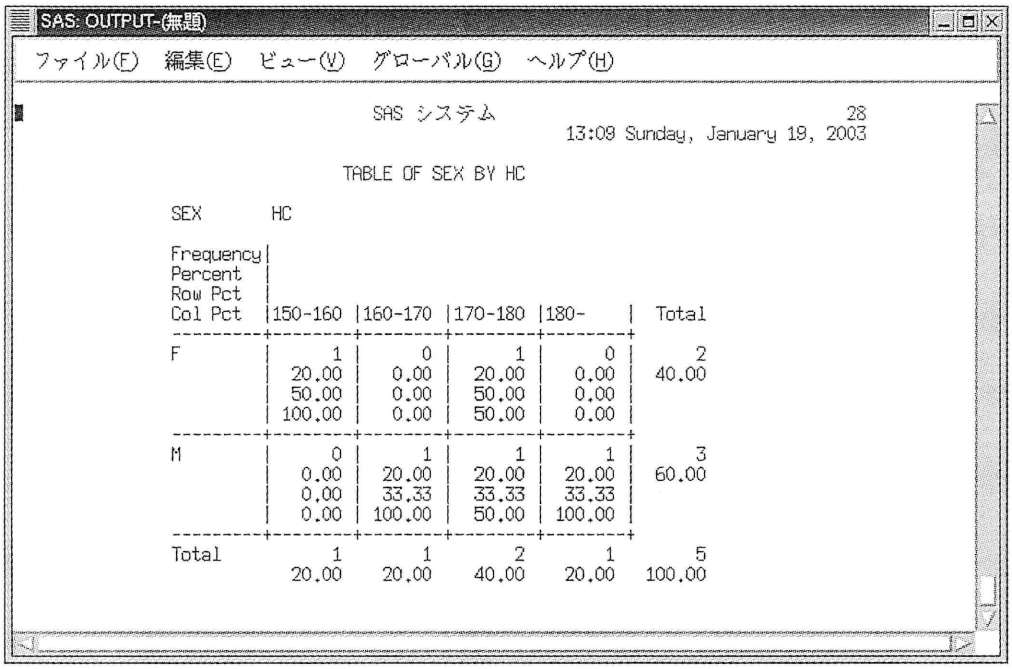


図 13: sex,hc によるクロス集計表

4.4　グラフの作成 (CHART/GCHART プロシジャ)

SAS では, 統計処理だけでなくグラフ描画も容易に行うことができます. グラフ描画のプロシジャは, CHART とGCHART があります. CHART はテキストベースのグラフを描画, GCHART はSAS/GRAPH オプションで利用可能なものでグラフィカルなグラフを描画します. 本センターではSAS/GRAPH が利用できますので, GCHARTプロシジャを使って解説します. CHART の使用法もGCHART と基本的に同じです. CHART 使用時は, 適宜GCHART をCHART に置き換えて下さい.

SAS で描画できるグラフには, 縦棒グラフ・横棒グラフ・ブロックチャート・円グラフなどがあります. まずは, 前節の度数分布表をグラフ化することを考えてみましょう. hc の縦棒グラフを描画するには, GCHARTプロシジャ呼び出しにつづいて, VBAR 文で変数hc を指定します. プログラムは,

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
  IF height<150 THEN hc='   -150';
    ELSE IF height<160 THEN hc='150-160';
      ELSE IF height<170 THEN hc='160-170';
        ELSE IF height<180 THEN hc='170-180';
          ELSE hc='180-   ';
PROC GCHART DATA=sample1;
  VBAR hc;
RUN;
```

となります. これを実行すると, 図 14 のような縦棒グラフがグラフ・ウィンドウに表示されます.

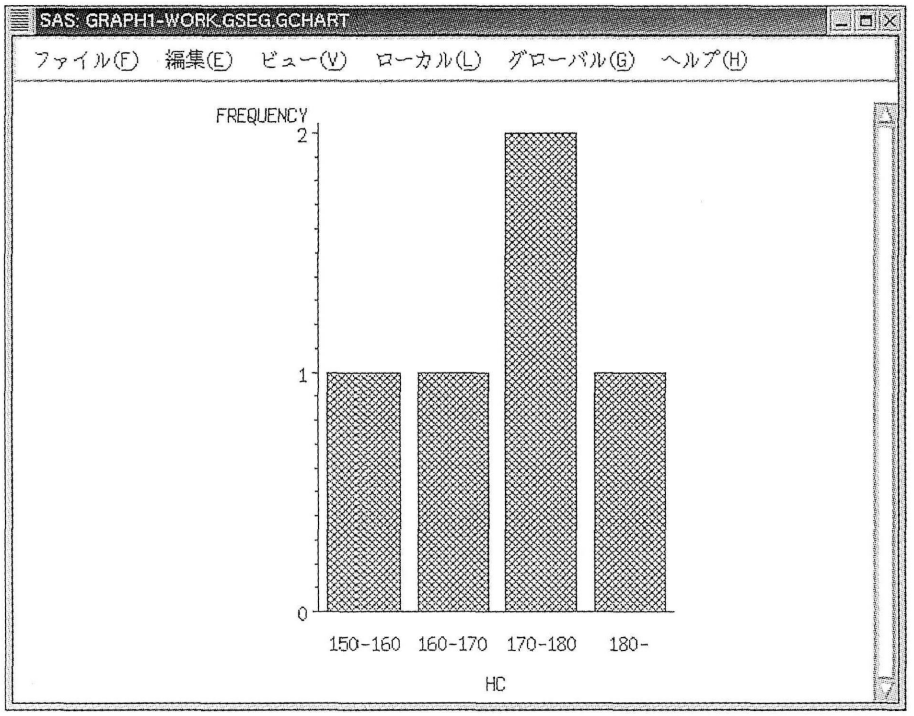


図 14: hc の縦棒グラフ

グラフの種類は, VBAR をHBAR にすると横棒グラフ, BLOCK にするとブロックチャートに, と変数を指定する文に応じて変わります. 表 4 にその対応をあげておきます.

また, VBAR やHBAR のあとに/SUBGROUP=と変数を加えると, 指定した変数の値に応じて棒やブロックを色分けします. 例えば, 次のようにGCHART プロシジャを呼び出すと,

変数を指定する文	グラフの種類
VBAR	縦棒グラフ
HBAR	横棒グラフ
BLOCK	ブロックチャート
PIE	円グラフ
STAR	スターグラフ

表 4: グラフの種類の指定

```
PROC GCHART DATA=sample1;
  VBAR hc/SUBGROUP=sex;
```

図 15 のようなグラフが得られます.

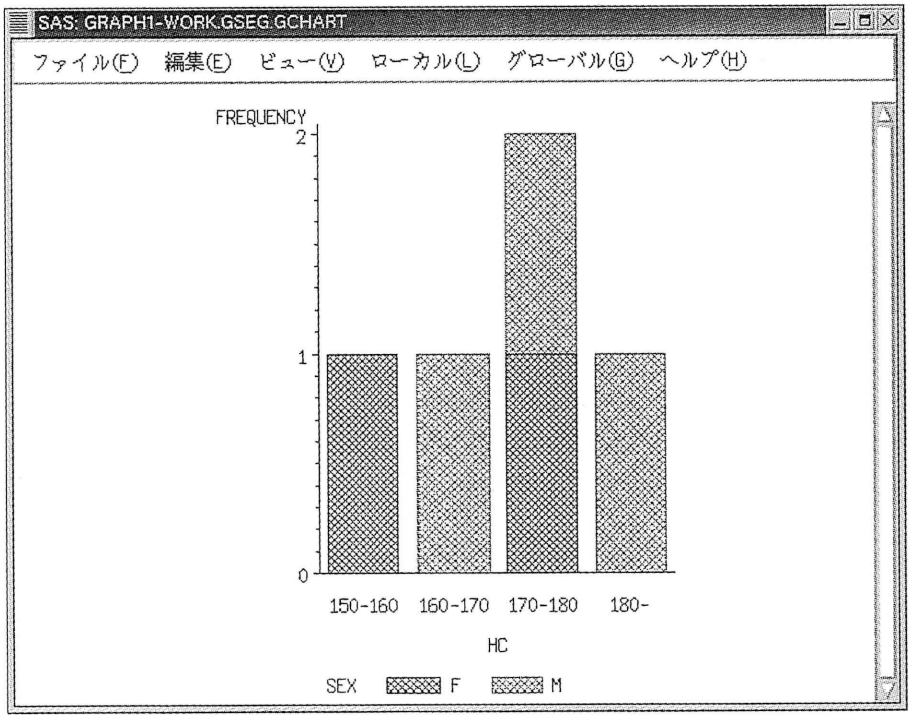


図 15: 性別情報付きhc の縦棒グラフ

4.5 散布図の作成 (PLOT/GPLOT プロシジャ)

散布図を描くには, PLOT またはGPLOTプロシジャを使います. PLOT とGPLOT の違いは, CHART とGCHART と同じで前者がテキストベースの散布図を, 後者はグラフィカルな散布図を描画します. 本稿ではGPLOT を使うことにします.

身長と体重で散布図を描くには, GPLOTプロシジャの呼び出しにつづいて, PLOT文で "*"(アスタリスク)で区切ってheight*weightと記述します. *の前に書いた変数は縦軸, 後に書いた変数は横軸に対応付けられます.

```
DATA sample1;
INFILE 'sample.dat';
INPUT id sex $ height weight;
PROC GPLOT DATA=sample1;
  PLOT height*weight;
RUN;
```

を実行すると図16のような散布図が得られます.

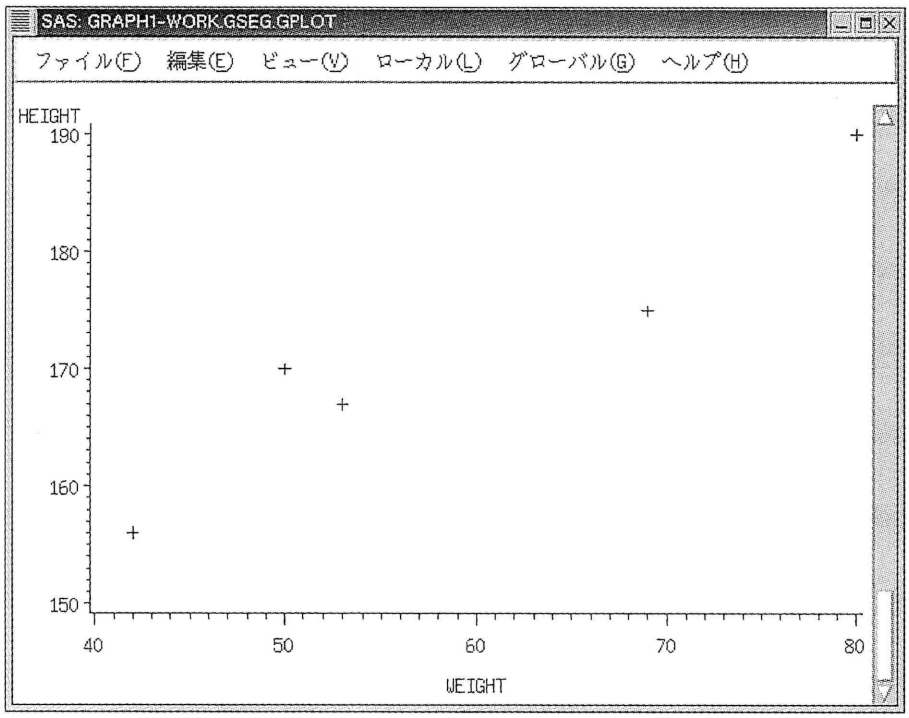


図 16: height,weightの散布図

5 デバッギング

プログラムが実行されない, 意図したような結果が得られない, という場合にはプログラム中にバグ (エラー) がある可能性があります. デバッギング (エラーの除去・修正の作業) では, まずLOGウィンドウのログを参照することをお薦めします. LOGウィンドウでは, エラー箇所が赤字で表示されます. まずその中で一番最初のエラー箇所から修正しましょう. というの

も, 一番最初のエラーが原因で, 本来正しく記述されている部分までエラーと認識されてしまう場合があるからです.

また, LOG ウィンドウには以前実行したプログラムの情報も残されるので, 以前のプログラムのログと混同するかもしれません. LOG ウィンドウで,

[編集] → [テキスト消去]

で一度, 全体を消してからログをとってみると良いでしょう.

6　おわりに

本稿では, データ要約を念頭に SAS の使用法を概観しました. SAS は, その他の GUI ベースの統計解析システムに比べると, プログラミング色が強いといえます. 逆に言えば, プログラムという形で処理の流れが形式的に記述されますので, 慣れてしまえば何が行われているのか明確となり, (主観ですけど) すっきりします. ぜひお試し下さい.

謝辞

筆者が大学院生時に SAS を勉強する機会を与えて下さった本学経済学研究院 古川哲也助教授, そして当時の西南学院大学情報処理センターのスタッフの皆様に深く感謝致します.

参考文献

[1] UNIX 版 SAS システム 使用の手引き, Version 6, First Edition, SAS 出版局

A　値と変数に関する補足

ここでは, 変数の省略記法と欠損値に関して補足しておきます.

A.1　変数の省略記法

サンプル・データでは, 識別番号 (id)・性別 (sex)・身長 (height)・体重 (weight) と, 扱う変数がたかだか 4 つでした. このような場合, INPUT 文に変数名をそれぞれ列挙しても大した労力とはなりません. しかし, 変数が数十を超えるような場合には, それぞれに変数名を付ける/参照するのは, 大変な作業となります. そのような場合には, x1,x2,··· というように変数に番号を付けた形で取り扱うと省略記法が使えて便利です (その代わり変数の意味が一様になってしまいますけど…). 例えば, 識別番号の後に 100 個の数値データが並んでいるとき, INPUT 文で次のように記述すればよいことになります.

```
INPUT id $ x1-x100;
```

x1,x2,···,x100 で, それぞれ 2 列目, 3 列目, ···101 列目のデータが対応します. また例えば, x1 から x100 の総和をとって変数 y に入れる, という場合も,

```
INPUT id $ x1-x100;
y = SUM(OF x1-x100);
```

と省略記法が使えます. ただし, SUM(OF ...) は OF 下に列挙した変数の値の総和を計算します.

A.2　欠損値

本稿の例では, 全サンプル・データで性別・身長・体重の値が観測されています. しかし実際, 統計解析を行う場合, 各データで全変数の値が観測されているとは限りません. 例えば, 身長の値がないデータや体重の値がないデータなど, 一部の変数の値が欠けているデータが解析対象に含まれることがあります. そのような箇所には "値がないことを表す値"(欠損値とよびます) を割り当てることになります. SAS では欠損値を "."(ピリオド) で表します.

また, アンケート調査などでは欠損値の種類を細分化したい場合があります. そんな場合には, MISSING 文を使って, 新たに欠損値を定義することが可能です. 例えば,

```
MISSING A,B
```

とすれば, 2 種類の欠損値が A,B で扱えるようになります.

B　非対話モードでの SAS プログラムの実行

ディスプレイマネージャではなく, 直接コマンドラインから SAS プログラムを実行する方法を説明しておきます. sample.sas という SAS プログラムを実行するには,

```
kyu-cc% sas sample.sas
```

とします. 拡張子が “sas” の場合は, 拡張子は省略しても構いません. 実行すると, カレントディレクトリにsample.lstとsample.logという2つのファイルが生成されます. それぞれプログラムの実行結果とログが出力されています.

また, プログラムの実行結果とログを出力するファイルを明示的に指定することもできます. -print,-logオプションをsasコマンドに指定します.

```
kyu-cc% sas sample.sas -print foo -log hoge
```

このようにすると, 実行結果はfoo, ログはhogeに出力されます.

C　図表の保存方法

GCHARTやGPLOTプロシジャで作成したグラフを保存するには, グラフが表示されるグラフウィンドウで,

[ファイル] → [書出し]

を選びます. すると, ダイアログが開くので, そこで, “ファイルタイプ” を適宜変更して保存します.