2.6 行列 文字と変数ラベル　　ver. 0.1

少し細かいことに立ち入るようですが、行列変数（または変数）に数値ではなくて、文字を入れる方法と、その操作の仕方、そして出力の際の結果データ行列に変数ラベルをつけることを説明します。ワークシート形式の統計計量のパッケージソフトとは違って、変数のタイプに気をつけたり、ファイルを作ったり開いて読んだり付け加えたりと低レベル言語なみの非常に込み入った理解を必要とします。困難をともなうと思われますが、１つ１つの小さなプログラム例を動かしながら、徐々に体得していってください。

まずは、文字の扱いには、数値と区別された GAUSS 独特の作法があります。大きなルールは３つ。１つ目は数値と同じ列には直接には文字列は入れることはできないこと。そして、２つ目は文字列の入った変数の演算や変形操作には、＄のマークプラス演算子を使うことです。最後の３つ目は、数字はそのまま入力できますが、それと区別するために文字は引用符〝”に包んで入力することです。

ワークシートを使用する統計計量ソフトでは、１つの列に文字や数字がごっちゃになったり、ラベルのところと数値のところがあまり意識されなかったりするかもしれません。しかしながら、GAUSS はこの点、非常に厳密です。ワークシートで数値を入れていく方式でないためと、あわせて、もともと行列の計算変換に重きを置くソフトであることから、数値は数値、文字列は文字列、変数ラベルはラベルで、厳密に分けてプログラム上管理されます。つまり、原則数値と文字列は一緒には共存できないのです。たとえば、次のようなプログラムでは、

プログラム

```
new;  cls;
a={x y z,
   2 2 2,
   3 3 3};
print a;
```

バージョン3.2では、syntax errorとなります。その一方で、バージョン3.6では画面表示はなされますが、１行目の文字の入ったところは+DENとなって、十進法入力denaryを促されます。（**厳密には、DENはunderflow denormal、NaNはnot a number、UNNはunnormal of number close to zero、そして無限大のときには＋方向にindefiniteという意味で+INF、無限小のときには - INFとなります。これらをもとに何かの演算をすると計算不能という意味でドット.を返すこともあります。**）ちなみに、マイナスの計算の場合には－DENとなります。ですから、ラベルと計算に使う行列の数値はGAUSSでは別に扱わなくてはなりません。つまり、同じ列には数字と文字は原則共存できないのです。ただし、すでにラベルがついた生の統計ファイルをGAUSS上に表示してみることは次のようにして可能です。ｄドライブにdatafile1.txtがあるとします。

プログラム

```
new;  cls;
print getf("d:datafile1.txt",0);
```

上のようにすると１行目にｘとｙがついているデータが覗けるはずです。組込み関数getfは、その第１要素にファイル名（ドライブ名、ディレクトリ名を含む）を引用符に包んだものが「文字」として入ります。第２要素には、０または１の数字がきます。数字が０の場合には、制御文字で制御されたASCIIモードで、１の場合には制御されないBINARYモードで、ファイルの内容を文字列として返します。y=getf(ファイル名，0)であれば、ファイルの制御つき内容が、文字列変数ｙのなかに（行列ではなくて）一続きの文字列として入ることになります。上のプログラムは、このｙとおいてそのｙをprintする作業を１つにまとめて、getfそのものをprintしているのです。なお、通常は、第２要素の数字が０でも１でも同じ結果が得られます。制御文字、例えば＾Zというファイルの終わりを表す制御文字を最後に入れたからといって（または消したからといって）その制御文字が表示されるわけではありません。制御文字で制御された結果どおりにASCIIモードの０では見えるだけであって、制御文字は通常は見えませんので誤解をしないでください。ファイルによっては途中に不要な制御文字が入っていてファイルの入力に支障をきたすことが少なからずあるので、このような確認の方法があるだけです。ちなみに、getfの返す値は一続きの文字列なので、中規模以上のデータファイルをgetfですべて読み取ることはできません。GAUSSが作り出す形式のファイルは(print fileは除く)読み取ることはできませんので用途は限られています。生であるデータを読みとってみる場合に使えて、その場合、最初や最後のところにラベルや説明があっても、文字列として読み取ることができます。

ポイント　getf(ファイル名,０)　制御文字で制御されたファイルの内容を文字列で返す
getf(ファイル名,１)　制御されないそのままのファイル内容を文字列で返す

次に、文字はその演算および操作に＄のマークプラス演算子を使うことに注意してください。＄のマークを使うことによって、変数の中に文字列が入っているものという前提のもとで、加法、水平方向のマージ、垂直方向のマージなどができます。

プログラム

```
new; cls;
x1= "abc"; x2="def"; x3="ghi"; x4="jkl";
x=x1$+x2$+" "$+x3$+x4;
print x;
y=x1$~x2$~" "$~x3$~x4;
print y;
```

```
z=x1$|x2$|x3$|x4;
print z;
w=x1$~x2$|x3$~x4;
print w;
```

画面表示

```
abcdef ghijkl
        abc             def                     ghi             jkl

        abc
        def
        ghi
        jkl

        abc             def
        ghi             jkl
```

まず、1番目の文字列の加法では、＄＋の演算記号でつなぎ合わされた文字列を間隔なしにひと続きの１個の変数としてまとめる働きをします。スペースを入れたい場合には、〝　”を＄＋の演算記号で足し合わせればスペースの間隔だけ開きます。それに対して、２番目の水平方向のマージでは、＄～の演算記号でつなぎ合わされた文字列をそれぞれ１つの行列の要素とみたてて水平方向にマージした行列（横ベクトル）を作成します。上の1番目の画面表示と2番目のものを見比べてもわかるように、加法では１変数として、水平方向のマージでは行列として16文字スペースごとにそれぞれの行列の要素として入っています。なお、２番目の水平方向のマージの場合には空スペースをつなぎ合わせるという意味で、スペースなしの””をつなぎあわせても結果は同じになります。＄｜の演算記号を用いた３番目の垂直方向のマージでは、文字列がそれぞれ１つの行列の要素とみたてて水平方向に並んでいることがわかると思います。最後に、水平方向と垂直方向のマージを組み合わせて、２×２の文字行列を作りました。注意すべきことは、＄の記号がそれぞれの変数の後ろについているのではなくて、そのあとの演算記号と１つの組になっていることです。＄＋、＄～、そして＄｜を１つの文字列演算記号と見てください。

ポイント　文字を代入する場合には引用符” ”に包んで数値と区別する
ポイント　文字行列の演算には、演算記号の前に＄をつける

さて、実際に文字というのはGAUSSの上の行列データとしてどのようにして扱われるか

を示していきましょう。
プログラム

```
new;  cls;
a={"saru" "nomi",
   "dani" "karasu"};
print $a;
print $a';
b=a[.,2]$~a[.,1];
print $b;
```

画面表示

```
    saru          nomi
    dani        karasu

    saru          dani
    nomi        karasu

    nomi          saru
  karasu          dani
```

上のように、数値ではない文字をGAUSS上の行列変数（または変数）データとして扱うには、引用符“　”にくるんだ形で、数値とは区別して代入していきます。{　}の中にカンマまでが１列目、そのあとが２列目…という代入の仕方は数値と同じです。画面表示させるために、print文で出力するためには、これも数値と区別するために、＄のマークを文字の入っている行列変数の前につけます。また基本的に、文字行列の計算では、行列の要素が違う値に変換されるような計算でないかぎり、通常数値演算や行列操作と同じことができます。ダッシュのマークを行列変数の後ろにつけることにより転置もできます。第2番目の出力は、ｃ＝ａ';としてprint $c;とするかわりに、ひとつにまとめてprint $a';として文字行列ａの転置行列に相当するものを画面表示させています。最後の行列の出力は、文字行列ａの２列目をa[.,2]としてワイルドカードのドットを数値行列と同じように記述したものに１列目のa[.,1]を、$~で水平方向につなぎあわせてｂとおき、それを画面出力させるために＄のマークをつけてprint文に＄付き行列変数名をおいています。このように、自由に数値と同じように切ったり貼ったりできますが、計算の途中の演算記号には必ず＄のマークをつけるのですが、計算途中の行列変数に＄のマークをつけてはいけません。あくまで、画面表示させる際にprint文の中でのみ最終的に＄のマークを行列変数につけてください。計算途中に変数名の前に＄のマークをつけることはエラーを返す結果になります。

ポイント　print文では、文字行列には変数の前に＄をつけて数値行列と区別する

実際に文字行列が生のデータとなっている場合には、もとのデータに引用符の”“のマークに文字がくるまれている必要はありません。データの取り込みは、数値が入ったデータと同じようにloadを使います。いま、ｄドライブにdatafile3.txtという文字のみでできたテキストファイルがあるとします。データは５×４の文字行列になっています。

プログラム

```
new;  cls;
load x[5,4]=d:datafile3.txt;
print $x;
```

画面表示

ITO	C	C	A
ETO	A	A	A
OTA	B	A	A
KAN	C	C	B
KIDA	A	A	B

上のように数値と同じ方法でloadを使って、行列変数のディメンションを決定した上で、その変数に場所付のファイル名を代入してデータは取り込まれます。直前の例と同じように画面表示する際にはprint文の中で変数の前に＄のマークを、数値と区別して文字という意味を出すために、つける必要があります。ここで気をつけないといけないことは、引用符のマーク” “によってくるまれていない文字は、GAUSSのプログラミングで大文字小文字の区別がないのと同様に、区別はされません。すべて大文字のアルファベットと認識されてしまいます。

次に、文字と数値が混じっている行列データはどうでしょうか。これは、通常のprint文では対応できません。なぜかと言えば、GAUSSでは文字と数値が厳密に区別されていて、タイプが違うからです。タイプが違うということを指定してやってから、特殊なprint関数で出力する必要があります。ここからは少しというかだいぶん厄介な話になってきます。プログラムで厄介だというのは、逆に言えば、細かい設定までできるということを意味して、プラグラムが厄介なことと細かい設定が出来ることとは、いわばトレードオフの関係にあります。

プログラム

```
new; cls;
data={"Ito" "M" 90 79 100,
      "Ueda" "F" 85 82 95,
```

```
        "Eda" "M" 91 75 83,
        "Ota" "F" 89 70 79,
        "Kida" "M" 80 83 70};
    print data;
print $data;
mask={0 0 1 1 1};
z=printfmt(data,mask);
```

画面表示(version3.6)

```
        +DEN        +DEN        90.000000        79.000000        100.00000
        +DEN        +DEN        85.000000        82.000000        95.000000
        +DEN        +DEN        91.000000        75.000000        83.000000
        +DEN        +DEN        89.000000        70.000000        79.000000
        +DEN        +DEN        80.000000        83.000000        70.000000

         Ito          M
        Ueda          F
         Eda          M
         Ota          F
        Kida          M
 Ito      M          90          79          100
Ueda      F          85          82           95
Eda       M          91          75           83
Ota       F          89          70           79
Kida      M          80          83           70
```

１番目が、print文で数値変数としてdataという変数を出力した結果。２番目が、print文で$dataとして文字変数としてdataという変数を出力した結果。そして最後の３番目が、変数のタイプ（０なら文字、１なら数値）をマスク変数とおいて、printfmt関数として出力した結果です。なお、version3.2では１番目の数値変数としての出力はしてはいけないことになっていて、無理やりに出力するとその文字が入っているところがすべて10の300何乗という数が出力されてしまいます。最新のversion3.6では、数値以外のものを出力しようとすると+DENとなるように統一のうえ修正されています。print文で＄付きの行列変数を出力しようとすると、２番目の出力結果のように、文字が入った列のみ出力されます。数値の入った列も、文字の入った列も、両方ともに出力しようとする場合には、フォーマットされたprint関数であるprintfmまたはprintfmtを用いなくてはなりません。printfmでは各行の出力形式も決められた複雑な記号で表記することが必須であるので、比較的扱いやすいprintfmt

を上のプログラムでは用いています。組込み関数printfmtは、その第１要素に変数名（ここでは＄のマークはつけてはいけない)、そして第２要素に変数のタイプの０か１の入った１行のベクトル変数を別に設定した上で代入します。そしてその全体としての出力結果とは別に、出力に成功すると１を返します。もし、z=printfmt(...)としないで、printfmt(...)とだけしたならば、出力結果とは別に1.0000000を余計に最後に出力してしまいます。これを回避する方法としては、上のように任意の変数＝という形で止めてやる方法または、call printfmt(...)としてやる方法の２つがあります。なぜ、callで呼び出す形になるかは、くわしくは次節のプログラムのところのprocedureのところで扱いますが、ここではこのprintfmtがproc系統の作り方をされているとだけ理解しておいてください。このことは、下のプログラムのように、この関数がグローバル変数をとることでも明らかです。

ポイント　call printfmt(**行列変数,マスク行ベクトル**);
文字と数値の列の混じった行列変数の出力。マスクの内容は
０の時、文字。１の時、数字。その横方向の行ベクトル。

上の最後の出力方法では、文字の列のところが極端につまってしまっています。文字の列も数値の列も等幅にするには、さらに文字値をフォーマットするという略のグローバル変数__fmtcvの値を変えなくてはいけません。{ "*.*s " 8 8 }がそのデフォルトのところを下のようにして変更します。

プログラム

```
new; cls;
data={"Ito" "M" 90 79 100,
      "Ueda" "F" 85 82 95,
      "Eda" "M" 91 75 83,
      "Ota" "F" 89 70 79,
      "Kida" "M" 80 83 70};
mask={0 0 1 1 1};
__fmtcv={"*.*s " 16 8};
call printfmt(data,mask);
```

画面表示

Ito	M	90	79	100
Ueda	F	85	82	95
Eda	M	91	75	83
Ota	F	89	70	79
Kida	M	80	83	70

上のように、グローバル変数__fmtcvの中に入る値を１×３の行列値として代入します。引用符にくるまれたところがformat文の右寄せ文字列に相当する部分で、特殊なコーディングによって文字として代入します。その後の２つの数値は、format文の/re 16 8などの16と８に相当する部分にあたります。この場合、数値のほうが16桁のフォーマットがデフォルトで、文字の方が８桁のフォーマットになっていますので、はじめの８のところを16にかえてやればよいことになります。（２番目の８は任意の値でかまいません。）なお、グローバル変数の前についている下線は２つ分がくっついた形になっていますので、１つの下線と見誤らないようにしてください。グラフィックのグローバル変数と区別するために２つの下線となっています。数値についても、__fmtnvをかえることによって、書き込むスペースや桁数、それに左寄せをかえることができますが特殊な場合だけであろうと思われますので省略します。デフォルトは、__fmtnv={ "*.*lg " 16 8 }になっています。引用符を閉じる前の空白スペースは意味をもっていて、それをなくしてしまうとそれぞれの文字スペースと文字スペースの間の１文字スペース分の間はなくなります。

ポイント　printfmtのフォーマットを変更するには、
文字の場合、デフォルト__fmtcv={"*.*s " 8 8};を変更する
数値の場合、デフォルト__fmtnv={"*.*lg " 16 8}を変更する
__は２つの下線、Sとgの後のスペースは文字スペース間の１文字間隔を意味

上のように、文字タイプごとのフォーマットはprintfmtでは任意であって、そのままにしておくならグローバル変数を指定する必要はありません。しかしながら、printfmというｆm関数では、各列ごとのフォーマットまで１つ１つ指定する必要があります。文字タイプのマスクは以前の例と同じです。文字スペース幅を16ではなく12に狭めるプログラムは下のようになります。

プログラム

```
new; cls;
data={"Ito" "M" 90 79 100,
      "Ueda" "F" 85 82 95,
     "Eda" "M" 91 75 83,
     "Ota" "F" 89 70 79,
     "Kida" "M" 80 83 70};
mask={0 0 1 1 1};
fmt={"*.*s "   12 8,
     "*.*s "   12 8,
     "*.*lg " 12 8,
     "*.*lg " 12 8,
```

```
        "*.*lg " 12 8};
    call printfm(data,mask,fmt);
```

画面表示

Ito	M	90	79	100
Ueda	F	85	82	95
Eda	M	91	75	83
Ota	F	89	70	79
Kida	M	80	83	70

上のように、printfm関数では、文字タイプの入った行ベクトルであるマスクの変数にほかに、フォーマット変数を列数×３だけのディメンションで指定してやらなければなりません。これはｆｍの方を用いた場合、任意ではなく、必須です。ｆｍｔの場合には、文字は文字、数値は数値で一括変更できたのですが、ｆｍの場合には１列ごとに指定します。１行目がデータ行列の１列目のフォーマット、２行目はデータ行列の２列目のフォーマットというふうに、データ行列の列数だけfmt行列変数の行数が必要です。なお、maskもfmtも変数名は任意ですが、必ず設定する必要があります。おのおのの行の始めの引用符に包まれた部分の文字が意味を持たない記号であるためにかなり複雑な作業ですが、教育およびヒューマンリソースのパッケージなどの文字と数値が混在したパッケージをGAUSS上で構築しようという特殊な用途のためには重要な機能となります。

ポイント　printfm(行列変数,マスクベクトル,フォーマット行列)
でprintfmtと同じ動作をする。一列ごとのフォーマットが可能。

次に、各列ごとにラベルをつけようとすれば、どのようにしたらよいのでしょうか？１番簡単な方法は、print文で引用符” ”の中に適当に間隔を伸ばして（１６文字プラス１文字間隔ずつに文字は入るのがデフォルト)、変数名を指定することが一番てっとりばやい方法です。これは、procedureや外部パッケージのプログラムにもよく用いられる方法です。しかしながら、ワークシート系の統計計量ソフトのように、行列の各行にラベルがついているわけではなくて、あくまで「見かけ上」print文で作成できているだけです。特に、文字数が多くなった時には頭の中で管理することはほぼ不可能です。そこで、GAUSSでは独自のファイルフォーマットにラベルはラベル、行列は行列でデータを作成したり読んだり付け加えたりすることが可能です。まずは、組込み関数createという命令を用いて、ラベルつきのGAUSSデータフォーマット(.dat)のファイルを作ってみましょう。

プログラム

```
new; cls;
data={"Ito" "M" 90 79 100,
```

```
                "Ueda" "F" 85 82 95,
                "Eda" "M" 91 75 83,

                "Ota" "F" 89 70 79,
            "Kida" "M" 80 83 70};
    dataset="d:datafile0";
    vname={"name" "sex" "listen" "write" "read"};
    vtype={   0        0        1          1          1   };
    create fh=^dataset with ^vname, 5, 8, vtype;
    call writer(fh,data);
    print rowsf(fh) colsf(fh);
    fh=close(fh);
```

画面表示

```
      5.0000000          5.0000000
```

上のように、dataという行列変数に文字と数値を行列として代入するところまでは同じです。文字には引用符”　”を用いるのに対して、数値はそのまま代入します。カンマごとに１列をなしていて、同じ列に文字と数値は混在はできません。文字の列であれば、すべてが文字である必要がありますし、数値の列であれば、すべてが数値である必要があります。さて、GAUSSデータ形式のファイルをつくる(createする)のには、いろいろな手続きや概念が必要になります。いよいよプログラムらしくなってきます。２つの新しい重要なプログラム上の概念が登場します。それは、「ファイルのhandleハンドル(名)」と「文字変数のsubstitution置換」の概念です。まず、ファイルをcreateしたりopenしたりしている最中には、ファイルは実際のファイル名ではなくてハンドル名で呼ばれます。ちょうど、チャットや掲示板で実際のアクセスIPではなくて自分で勝手にペンネームをつけられることと似ています。ここではfile handleの英語を略してfhとしていますが、実際には何であってもかまいません。ただし、慣習上、fからはじまる短い名前、ｆｈとかｆ１、ｆ２、ｆ３...などがよく使われます。作ったり開いたりするファイルが１つであるのなら、そのファイルハンドルの中には数字の１が割り当てられます。それ以上のファイル数を操作する場合には、順次番号が割り当てられていきます。（version3.6では１から番号は割り当てられますが他のバージョンはそのかぎりではありませんので、番号を見込んでのプログラムはしないようにしなければなりません。あくまでGAUSSが任意にファイルハンドルに番号を割り当てると考えてください。）GAUSSはこの番号をもとにファイルをプログラム上で管理する一方、我々は、この割り当てられる番号ではなく、ハンドル名の方を使うことになります。ということで、create fhというようにしてハンドル名fhのファイルをcreateしています。それから、場所ディレクトリつきのファイル名などの文字列string、それから文字が入った文

字行列変数は、記号キャデット ^ を使うことによって、変数名の置き換えが可能です。置き換えるべき変数の前にキャデット ^ の記号をつけることにより、ちょうど何かを「挿入すべき」というような形になります。まず、データファイルが入るべきファイル名を場所つきの名前で指定して、ひと続きの文字列として、引用符に包んでdatafileという名前をつけます。直接このステップをとばして、ファイルハンドル名＝の後に直接場所つきのファイル名を書きこんでもかまいません。次に、各変数のラベル名をこれも引用符に包んで文字行列としてvnameという名前をつけます。datafileという名前もvnameという名前も任意です。何であってもかまいません。変数のラベルについては、それをカンマをつけて列ベクトルとして代入しても、上のようにカンマなしで見た目どおり行ベクトルとして代入しても、結果は同じになります。（次のところでやるように、ファイルからラベルを取り出す場合には、列ベクトルとして縦になってでてきます。）それから、それぞれの縦にならんでいる変数のタイプを設定します。文字の場合には０、数字の場合には１とします。上の場合、最初の２つの名前と性別は文字行列なので０を２つ、その後は数字なので１が３つとなります。見た目どおり、行ベクトルとして横にvtypeとします。このvtypeという名前は任意です。そして、createという命令で新規にGAUSSデータ形式のファイルを作成します。ファイルハンドル名fhに対して、datafileという名前で置換された場所つきファイル名を指定するという意味で、datafileの前に置換を表す ^ のマークをつけています。校正記号のようにちょうどその変数に何か入るという感じです。なお、べき乗の意味では、ここではないのでご注意を。withは「以下の変数を使って」という意味になります。変数名もvnameという名前の変数に置換されていますから、 ^ のマークをそのvnameの前につけて、withの後に置いています。そのあとの数字は、最初の５というのは変数名が５個あるので、その５個分の列を確保するということ。その次の８というのは、８バイトを使った倍数精度という意味で、８以外に、単精度の４、整数の２があります。通常は、GAUSSの計算と同じ精度の８をよく使います。そして、最後にデータタイプの入ったベクトル変数を置きます。これは、数字ばかりの入ったベクトル変数なので、置換のマーク ^ はつけません。置換のマークは、文字列または文字行列にのみ使います。これだけでは、変数dataにあるデータの内容はまだファイルハンドルfhには書きこまれていません。まだ、変数ラベルのみが書きこまれているだけの状態です。そこでwriterという命令を使って、その第１要素fhに第２要素dataを書き込むという命令をします。call writerと呼び出すことによって、writer自身の値を返さないようにしています。これはprintfmtなどの場合と同様です。それから次に実際にデータが５行５列で入っているかを調べるためにファイルハンドルfhの行数と列数を計算してその値を画面表示させています。変数の行数列数を計算する場合には、rowsfおよびcolsfを使います。rowsやcolsではないことに注意してください。作成しているあるいは開いているファイルの行数列数ということでfという文字がついています。最後にcreate命令で開いているファイルを閉じます。closeという命令を使います。この命令もそれ自身

に成功か成功でないかで数字が返されるので、それをとめるためにfhに代入してとめていま

す。closeallを使って、closeall ハンドル名;としても同じことをします。ハンドル名のところにはファイルのハンドルが複数ある場合には列挙します。これらが、createでファイルを作成してデータをファイルに書きこむ一連のプログラムの標準的な手順になります。

ポイント　文字列または文字行列の変数置換には＾のマークを変数の前につける
ポイント　ファイルをcreateしたりopenしたりする時にはファイルハンドル名で呼ぶ

ポイント　create fh =**位置名置換変数**　with **ラベル置換変数 ラベル数 精度　マスク**;
ファイル位置にGAUSSデータファイルを作って、それを操作上fhと命名する
fh: **任意の変数**
精度:　２(整数)，４(単精度)、または８(倍精度)
マスク:　０(文字)か１(数字)でできた行ベクトル

ポイント　writer(**ハンドル,行列変数**)　**ハンドルの指定するファイルに行列変数のデータを書きこむ。それ自身の返す値は、書きこまれたデータの行数。**　call writer()**として使う。**
ポイント　ファイルをクローズするには、
fh=close(fh); **または** closeall fh; **とする。必須。**

今度は、上のプログラムで書きこんだ変数ラベルとデータをそれぞれ読み出してみましょう。ラベルとデータはGAUSSデータファイルの中で別々に管理されています。それらを別々にある一定の手続きにしたがって読み出すことになります。
プログラム

```
    new; cls;
dataset="d:datafile0.dat";
open fh=^dataset;
print $getname(dataset)';
mask={ 0 0 1 1 1};
__fmtcv={ "*.*s " 16 16 };
call printfmt(readr(fh,5),mask);
closeall fh;
```

画面表示

name	sex	listen	write	read
Ito	M	90	79	100

Ueda	F	85	82	95
Eda	M	91	75	83
Ota	F	89	70	79
Kida	M	80	83	70

以前のファイルをcreateするプログラムでは、datafile0とだけｄドライブのトップにファイルを作るように命令しましたが、できあがっているファイル名は、datafile0.datというふうに、.datの拡張子が自動的についています。まず、この拡張子.datがついたファイル名を置換の方法によって、datasetという任意の名前をつけます。そして、その置換変数をopenという命令で開くことになります。GAUSSではファイルをcreateしたりopenしたりする最中には、ファイルはハンドル名で呼ばれますから、そのハンドルfhをdatasetという置換変数と定義代入します。ですから、置換の＾のマークがついていることになります。もちろんそのままfh=のあとに場所つきファイル名を直接置いてもかまいません。getnameという組込み関数はデータファイルについているラベルの名前を抽出して、縦方向の列ベクトルとして返す関数です。この関数はファイルハンドルではなくて、実際にデータの入っているファイル名を指す置換変数そのものを引数にとります。引数の中では＾のマークは必要ありません。この列ベクトルは、内容はすべて文字ですから、print文を使うには＄のマークを前につけます。そして、文字行列も転置可能ですからダッシュをつけて列ベクトルから横方向の行ベクトルに返還ののち画面表示しています。なにか別の変数にgetname( )のところをおいて、それを転置記号のダッシュで転置させて、その結果をprint文で変数の前に＄をつけて出力してもかまいません。これらのステップをすべてprint文の中で計算させています。わかりずらければ、１ステップずつプログラムを組んでもよいでしょう。まず、ラベルの部分が出力できました。次に、データ行列の部分を出力します。これには、readrという命令を使います。writerのｒとreadrのｒはともにrow行という意味の略であると思われます。組込み関数readrはその第１要素に読み取るファイルハンドルをとり、その第２要素に読み取るべき行数を数字で入れます。この場合、開いているファイルのハンドル名はfhですから、第１要素にfhを入れて、（ラベルを除いて）５行分のデータがありますから５を第２要素に入れます。その読み出された行列をprintfmtで０か１のmaskで文字か数字の属性を指定してやって画面出力しています。以前説明しましたように、文字の列は８文字スペースがデフォルトなので、グローバル変数__fmtcvで16文字スペースに変更して、上のラベルの表示の間隔と同じに設定しています。最後に、closeallでファイルハンドルfhを閉じます。なお、fh=close(fh);としても同じことになります。

ポイント　open fh =**位置名置換変数**;
　　ファイル位置のGAUSSデータファイルを開いて、それを操作上fhと命名する
ポイント　readr(**ハンドル,行数**)　**ハンドルで指定されたファイルの行数だけ読み出す**

ポイント　getname(ファイル名)　ファイル名の位置にあるデータのラベルを読み出して列ベクトルで返す

データの各列の属性であるmaskを自分で０と１を入れて設定する方法もありますが、バージョン3.2よりも大きいバージョンでは、vartypefという組込み関数が新しく設けられていて、この例でいくと、mask=vartypef(fh)'とファイルハンドルの変数タイプをしらべて列ベクトルで返される０と１のベクトルを転置させて行ベクトルになおしたものをmaskに指定できるようになりました。便利な機能ですが、まだまだアメリカの大学、研究機関では3.2程度のバージョンが走っていますから、新しい関数を多用すると研究者間での互換性がなくなります。ここでは、すべてのバージョンで動くようなプログラムを解説していくように努めます。以上が、ラベルつきデータを管理するGAUSSデータファイル作成し、それを読み出すという基本的な手続きを解説しました。