FAQ(2)　プログラミングの基本

【質問１】
例題の短いプログラムを打ちこんだのですが、どうも動きません。タイプミスは見つからないのですが。

【解答】
まず、GAUSSはプログラムを保存しないとRunできません。また、バージョン４以上のGAUSSでは、Run Active Fileをしないと前に実行したプログラムが実行されます。それがわかった上でも、なおプログラムが動かないというのであれば、それはおそらく各命令ごとの最後につくセミコロン；のマークをどこかわすれているのか、あるいはそれをコロン：とタイプミスしているものと思われます。

【対処法】
プログラムを終えたならば、まず、各行の行末にセミコロン；が全部ついているのか必ず目視で確認します。その次に、１つの行に２つ以上の定数設定などをしている行だけに注目して、その定数設定ごとのそれぞれの後にセミコロン；がついているかを確認します。/*　*/や@　@自身にはセミコロンはつけませんが、その注釈部分の前後の実行行の最後のセミコロン；は忘れがちになります。最後に、if 条件文；や else；などの通常の他の言語ではつけない部分にも GAUSS ではセミコロンをつけることに注意します。１つでもセミコロンをぬかしてしまうと、プログラムはエラーを発して止まってしまいます。また、エラーの行ナンバーは、そのようなタイプミスの場合、通常示されるエラーの行番号よりも前、あるいは直前で起きているエラーを見つけ出しています。プログラムの上級になっても、エラーの大半はこのセミコロンもれにあります。

【質問２】
今度は、Web などに掲載されているプログラムをカット＆ペーストしたものを実行しようとしているのですが、エラーがなさそうな行でエラーを発してしまって動きません。

【解答】
断言はできませんが、カット＆ペーストのプログラムの原因不明のエラーの場合、それが日本語文章からのものの場合、見えない制御コードが GAUSS のプログラム上に入りこんでしまって、うまく実行されていないことがあります。

【対処方法】
エラーが見つかった行の前後の行下げに相当するスペースの部分を GAUSS のエディター上で、その行の先頭にまで一度スペースを BackSpace で全部なくしてみます。そして、またスペースを入れます。それで Run してみてください。エラー行が変わるようであれば、同じような行下げのスペースを一度 BackSpace で消して、スペースを入れなおす作業を繰り返してください。その部分に制御コードが入っている、もしくは半角分だけズレが生じています。もう１つ、注意すべきは、GAUSS では日本語の注釈を入れてはいけません。その部分を消したとしても、日本語入力の際にスペース自体が全角スペースになってしまっていて、ここでも半角ズレが生じてしまうことがあります。GAUSS は好むと好まざるとを言わず、TSP のように日本語の注釈をいれては絶対にいけません。原因不明のエラーの温床になります。

【質問３】
単純なプログラムなのに、Too many arguments or misplaced assignment operator などの原因不明のエラーになります。タイプミスやプログラムに構造上のエラーはないものと確信していますが。

【解答】
必ずしもこのエラーログになるとは限りませんが、変数名または関数名に GAUSS 上の予約語またはロードされているライブラリや関数と同じ名前が使われている可能性があります。非英語圏の人々に多い誤りです。また、当然のことながら、変数名に全角文字が誤って入っていないか確認してください。

【対処方法】
使われている変数名の中で、単純な英語はありませんか？その変数名自体を略語に変更するか変数に１や２をつけて新たな変数名にしてプログラム全体を書きなおしてみてください。プログラムが動くようであれば、その変数名は予約語、あるいはロードされている関数名、グローバル変数の類です。

【質問４】
Print 文のあたりの行で Operator missing とエラーになります。例題のとおりに打ちこんだのですが、どうしても原因がわかりません。

【解答】
これは Print 文前後のスペースの欠如によるものです。スペースが入っていないためにGAUSS のシステムがどこからどこまでが print 命令で、どこからどこまでが引用符で包まれたコメント文であるかの区別がつかないためこうしたことが発生します。

【対処方法】
コメント文付きの変数を画面表示するプログラムは、例えば、次のようになります。

```
new; cls;
a=1;
print "a=" a;
```

上のプログラムでは、特に、print 命令の後の引用符に包まれた後の ” のマークと a;の間のスペースは必須です。紙面上はつめて書かれているように見えても、引用符の前後には区別をはっきりさせるためスペースが入っていますので注意してください。

【質問５】
内積計算がうまくいきません。

【解答】
GAUSS では、すべての変数は行列として扱われます。また、カンマごとに各行を表していて、通常の数学の{1,2,3,4,5}という表記の仕方を GAUSS は５×１の列ベクトルと解釈します。計量のデータのように列としてのデータを処理することに特化したシステムです。

【対処方法】
例えば、ｐとｘのベクトルの内積を計算するものとしましょう。プログラムは、

```
new; cls;
p={1,2,3};
x={1,2,5};
print p'x;
```

というふうになります。つまり、３×１のｐベクトルと、同じく３×１のｘベクトルを考えて、その内積を計算するのには、１×１＋２×２＋３×５＝20 がほしいのですから、これを行列の計算と考えて、前者のｐベクトルを転置させて、１×３のベクトルにしておいてから、これと３×１のｘベクトルをかけることで、（１×３）×（３×１）で１×１の行列、すなわち数値として内積が計算できます。上のｐとｘのベクトルはプログラム上では横に並んでいますが、他言語のリストや通常の数学の表記とは違って、実際には、

```
p={1,          x={1,
   2,             2
   3};            5};
```

という３×１のベクトルの意味です。カンマごとに改行がなされます。

【質問６】
ループを用いて単純なプログラムをしたのですが、Variable not initialized のエラーが出てしまいます。タイプミスもプログラム上の構造的なミスもないものと思われますが。

【解答】
これは中上級の人で、自分でプログラムを書けるようになってから一番よく出くわすエラーです。GAUSS では、変数を宣言する必要はありませんが、はじめて使う変数には何も入っていません。ゼロやミッシングバリューのドットすら入っていないのです。何も入っていない変数をプログラム上に置くとエラーになります。

【対処方法】
通常は、ゼロ（または零行列）で変数を０に設定してから、その変数の中にループによって何か計算されたものを代入していくことになります。何も入っていない変数に代入することは GAUSS 上では不可能です。したがって、下のようなプログラムでは

```
new; cls;
i=1;
do while i<=10;
    sum=sum+i;
    i=i+1;
endo;
print sum;
```

sum という変数が初期化されていませんからエラーになります。ここでは、sum=0;という１文を do ループよりも前の場所に置くことが必要です。上の例は sum が数値のケースですが、行列の場合には、例えば行列変数 m=zeros(m,n);という具合にｍとｎの部分に計算で使われるディメンションの数を入れて、あらかじめそのディメンションの領域を確保しておく必要があります。その場合は＝０ではいけません。上の sum=0;は sum という行列変数に０という数値を入れると同時に１×１と領域確保している役目を果たしています。

【質問７】
単純な行列計算をしているのですが、Rows don't match や Syntax error や Matrices are not conformable が出て本当に疲れてしまいます。どうすればよいのでしょうか？

【解答】
Excel などの表計算や、自動形式の計量ソフトウエアでは、データが欠如していても列の長さが一致してなくても、一致しているところだけを「自動的に選別して」計算を行なってくれます。しかしながら、GAUSS は行列を「列として」扱う言語体系です。たいていの場合、プログラムをするユーザー自身の手で、列の長さをそろえたりデータの欠如の処理をしてやらなくてはなりません。これからは、表計算の自動選別のことは忘れれてしまってください。

【対処方法】
特に let 命令で変数の配列を指定するのでなければ、通常 GAUSS では、カンマのマークまでが１行です。例えば、５つの数字がカンマで１つずつ区切られていれば、それは５×１の列ベクトルを表します。列ベクトルや行列を足し合わせたり、～の記号で水平方向にマージする場合には、常に全ての変数の縦の長さ（行数）が一致しているか念入りに精査確認してください。行列またはベクトルの掛け算の場合には、例えば、ｘを 100×２、ｙを 100×１の行列またはベクトルとすれば、inv(x'x)*x'y の計算であれば、

[（２×100）×（100×２）]×（２×100）×（100×１）＝２×１

とういふうに、「かけ合わせる列数と隣の行数が一致」するように必ず目視で確認してください。また、上の場合「一番最初の行数と一番最後の列数が計算結果の配列」になります。GAUSS の計算結果も、この行列計算に忠実に、２×１の列ベクトルで出てきます。足し合わせる場合や水平マージをする場合には行数の確認を、また、行列をかけ合わせる場合には隣の行列との列数と行数の一致条件である conformablity を確認してください。自動的に転置してくれるようなことは GAUSS では全く行なわれません。

【質問８】
画面表示の時の桁数がおかしくなってしまいました。どうやったらもとに戻せますか？

【解答】
GPE2 などのサードパーティーのアドオンを使用した後や、他の人の書いた procedure を使用した後などに、桁数が変わってしまっていることがあります。これは、そのアドオンや procedure が Format 文を使用しているためです。ご自分で他の人も使う procedure をプログラムする場合には、少なくとも、procedure 内には Format 文を使わないようにすることが肝要です。もっとも、これを修正するのは簡単にできます。

【対処方法】
原始的な方法は、一度 GAUSS を終了させて、もう一度起動することです。これによって、桁数はデフォルトの設定に戻ります。終了させずに、もとの桁数に戻すのには、

format /mb1 /ros 16,8　または単に　format /ros 16,8

とすれば、もとの桁数に戻ります（ただし、そのパッケージを使った計算は戻りません)。上の Format 設定がデフォルト設定と言えます。詳しくは、1.7 の数値のフォーマットの章をご覧ください。この桁設定の Format 文は内部計算には影響を与えません。あくまで、見た目の結果表示にだけ関係するだけです。私は、procedure の内部でこの Format 文を使うことを推奨しません。外部で使うか、使うのであればその他の文字列のフォーマット設定付き表示文を使うべきです。わずらわしいのなら、procedure 内で Format 文を使ったなら内部の最後のところに上のデフォルト設定の文を 1 行挿入してもとにもどしてやる作業をすべきでしょう。

【質問９】
論文に付属されていたり Web に掲載されていたりするプログラムを自分で動かしてみたのですが、全く動きません。あるいは途中まで計算がなされるのですが、それ以上進みませんが、どうすればよいでしょうか？

【解答】
冒頭付近にある library 文のところに、pgraph 以外のライブラリが指定されているならば、そのプログラムは標準の GAUSS では動きません。別途購入のライブラリが必要です。また、プログラムの途中で書き込みをするディレクトリが使用の環境では存在しないことも考えられます。その場合には、書き込みをするディレクトリの階層のフォルダーを自分であらかじめ Windows 上から作成しておいてください。自動作成はなされません。

【対処方法】
現在のところ、世の中に公開されている GAUSS のコードのうちで、全体の９割５分は GAUSS と maxlik,cml,co,optmum の４つのライブラリのどれかを使ったものです。残りのほとんどは、optimum（これは Optimization のこと）の上で動作する tsm という金融工学及び時系列のライブラリのコードです。それらを購入していれば、たいていのことはできることになります。ただし、これらのほとんどのライブラリは、ただ単に最適値の収束計算をしているだけなので、自作の最適値計算の procedure や公開されているもの、あるいは GPE2 で代用ができます。目的関数の procedure をデータ付きのものからデータ付きでないものに変えたり、データと係数の順番を交換して、グローバル設定を変えてやるだけで、条件付の込み入ったものでなければ、第４節を読み終わった読者であれば、他の代替的な procedure やパッケージによって計算可能です。また、上の４つのライブラリ間でも、そのかなりの部分で代替的にプログラムの手直しが可能です。

【質問 10】
行列を行列で割り算したいのですが、計算はできるようなのですが、思っていた計算結果とだいぶん違います。おかしいのではないですか？

【解答】
GAUSS では、１×１の数値で割ったりべき乗にする場合にはそのまま計算できるのですが、「行列と行列の割り算」の場合には、要素対要素ということでドットを演算子の前につけなくてはいけません。

【対処方法】
例えば、下のような２×２のｘとｙの行列の要素ごとの割り算をする場合、

```
new; cls;
x={1 2,3 4}; y={2 3, 1 3};
print y/x;
```

としてしまっては結果がおかしいことになります。要素対要素の場合にはドットをつけて、

```
print y./x
```

としなくてはいけません。ドットは演算子の前で「要素対要素」ということを表します。
それでは、上の最初の y/x の答えは何なのでしょうか？それは、最小 2 乗推定量もどきの

```
print inv(x'x)*x'y;
```

の結果を表しています。この場合のｙは２×２の行列です。最小２乗法の計算は、y/x とする方が Matlab では速いという記述もありますが、GAUSS では同じなので、行列を行列のまま計算するという立場から、inv(x'x)*x'y を、長いですが、計算式のまま用いるのがよいと思います。

【質問 11】
論理式で、大小関係を使って結果を出したいのですが、もともとの変数が列ベクトルなのに、結果は１×１の数値でててきてしまいます。なにかおかしいです。

【解答】
これも「要素対要素」のドットの観念が抜け落ちてしまっています。ドットがなければ、たとえ比べるものが１×１の数値であったとしても、全体を比べて真偽の判定を１か０でしてしまうことになります。

【対処方法】
例えば、０よりも大きいものを判定するには、

```
new; cls;
x={1,2,3,4};
print x>0;
```

としてしまっては、たとえ結果が１で真になっても誤りです。この場合には、ドットを

```
print x.>0;
```

というふうにつけてやると、

```
1.0000000
1.0000000
1.0000000
1.0000000
```

という結果になって、４×１の列ベクトルのそれぞれが１でｘ＞１に対して真であることがわかると思います。同様にして、他の数に対しても判別ができます。逆に、最適値計算の時の Tolerance のように、すべての値がそれ以下の場合になるのを判別する場合は、ドットはつけません。

【質問 12】
条件分岐の if 文を使っているのですが、そのあたりでうまくいきません。

【解答】
if 文はそれだけでは機能しません。必ず if 文と対になる endif;文が必要です。つまり、if 文が階層的にあろうが、並列に並んでいようが、常に endif;文が if の数だけ必要になるのです。また、else や endif の使い方を間違えている可能性があります。

【対処方法】
（1）if と endif の数が一致しているか確認する。
（2）elseif 文の elseif は一語である。else if とスペースを入れてないか確認する。
（3）if 文の条件式の後に（elseif 文もあれば）セミコロンがついているか確認する。
（4）endif の直後に（else があればその直後も）セミコロンがついているか確認する。

特に、他のプログラム言語に精通した人々は、endif や elseif の直後のセミコロンをつけるのを忘れがちになります。GAUSS ではどのような文でもセミコロン；で切ることが必須です。これは、else;や enif;それに Do ループ関連の締めに相当する endo;や endfor;にもセミコロンは必須です。

【質問 13】
画面表示させると一部の結果は表示されますが、画面が乱れます。

【解答】
これは、冒頭に new;文が足りないものと思われます。また、cls;文がないと、run ファイル名の部分が表示され、前の計算結果を消さずに画面表示がその後に表示されます。

【対処方法】
new;　メモリの初期化
cls;　　画面のクリア
通常の GAUSS のプログラムであれば、この 2 文を冒頭に置くことを習慣づけてください。ただし、前の計算結果も残したい場合には、cls;文は省略も可能です。標準の GAUSS 環境では、少なくとも new;文は必須です。長いプログラムになると、new;文がなければ、前回の残像がメモリに残ってしまって、画面表示は乱れます。ただし、この原則は、アドオンパッケージの GPE2 を用いた計算には、あてはまりません。すなわち、use gpe2;から始まっているプログラムでは、反対に new;文を絶対に使ってはいけません。GPE2 を呼び出す段階で、new;文も兼ねています。この new;文の原則は、GAUSS 本体と GPE2 を使った計算とでは全く逆になりますから、どちらか一方から始めて他方でもプログラムを試みる場合には、単純ですがこの原則は重要になってきます。

【質問 14】
データを load することはできるのですが、結果を保存することができません。

【解答】
おそらく、どこかに保存はされているのだがその場所がわからないのであるか、あるいはディレクトリや階層付きの保存の場合には、そのディレクトリや階層そのものが Windows 上に存在しないためだと思われます。

【対処方法】
ディレクトリも階層も付けないでファイル名だけを指定して保存した場合には、保存されたファイルは、Working Directory に保存されているはずです。GAUSS のメニューの中の File のところにある Change Working Directory…のところを見れば、現在の保存されるフォルダの階層がわかるはずです。その中に保存されているはずです。その場所が都合悪ければ、そのメニューのところでクリックして Working Directory のフォルダの指定を変更してみてください。一方、ディレクトリや階層付きでファイル名を保存した場合に、ファイルが保存されていない場合には、それはフォルダの名前自体を間違えているか、あるいは、フォルダそのものがまだ存在していないのであると思われます。フォルダは、ご自分で Windows 上から予め作成しておいてください。自動的には生成されません。

【質問 15】
GPE2 やその他のアドオンを使っているのですが、プログラムの変数受け渡しがうまくいかないらしく、動作しません。

【解答】
GPE2 や一部のライブラリやアドオンは２つの下線から始まる変数をその特殊関数の特定のアウトプットに使っています。グローバル変数自体である場合もあります。

【対処方法】
下線がある場合、２つ分の下線であるケースも考えられますから、下線が１つなのか２つなのか、通常は続いて見えますが、入念に確認してください。GPE2 ではグローバル変数は１つの下線、estimate で生成されたアウトプットは２つの下線というように統一されているようです。その他のプログラムでは１つの下線や２つの下線を変数そのものと使用しているケースもあります。２つ分の下線は１続きに見えますが、全角の下線ではないことにも注意してください。GAUSS では日本語などの全角は使ってはいけません。

【質問 16】
変数をワイルドカードのドットを使って利用したり表示したりしようとしているのですが、Indexing a matrix as a vector というエラーがでてしまいます。

【解答】
これは中級以上の人がよく遭遇するエラーです。列ベクトルまたは行ベクトルの場合には、その行または列の数は１ですから、例えばｘというベクトルのｋ番目には x[k]という表現が可能です。しかしながら、ベクトルではない行列の場合には、そういう表現方法はできません。

【対処方法】
例えば、行列ｘのｊ列目を表すには、
　x[j]　⇒　x[.,j]
同様に、行列ｘのｋ列目を表すには、
　x[k]　⇒　x[k,.]
というふうに、ワイルドカードにすることを忘れないでください。なお、GAUSS ではすべての変数は行列として扱われますので、逆にベクトルを上のように表記しても全く支障はありません。上のようなエラーメッセージの場合、たいていの場合、ワイルドカードのドットを忘れているものと考えてよいでしょう。

【質問 17】
変数を表示した際に、数値は表示されるのですが文字の入っている所が表示されません。

【解答】
GAUSS では、厳格に数値と文字は区別されます。表計算ソフトのように、文字と数値が混在することは、通常の方法ではできません。混在させる方法もないわけではありませんが、通常のプログラミングでは、print 文のコメントとして引用符” “に包んで表示させるので十分でしょう。変数内の文字だけを表示させる、あるいは数値だけを表示させることのいずれかは可能ですが、その両方はできないと考えてください。

【対処方法】
文字だけは x={"A","B","C"};というふうにそれぞれを文字列として引用符に包んで代入して print $x;とすることで表示できます。$ のマークは文字列を扱う際につける付属演算子です。これにより、変数の中の文字列が表示されます。通常の推定結果にラベルをつけるには、

```
print “          b1          b2          s2”;
```

あるいは、余白の部分を気にしない方法が必要であれば、

```
label={"b1" "b1" "s2"};
print $label;
```

というふうに、文字列が入っている文字列を $ 付きの変数の形で print 表示すればよいでしょう。上の場合、１×３のベクトルですが、括弧中をカンマで区切ってしまえば、３×１のベクトルになるので、変数は ’ をつけて転置する必要があります。要するに、文字は文字で別に扱った方が簡単であるということです。

【質問 18】
論理式や条件分岐のところで、変数が何かに等しいと置きたいのですが、うまくいきません。どうすればよいですか？

【解答】
GAUSS では、通常の数値設定や計算結果の変数への代入にはイコール＝を使います。しかしながら、これは論理や条件分岐の論理式では使うことができません。

【対処方法】
論理や条件分岐の論理式での目的の場合、ダブルの＝＝を使います。特に条件文中の＝の符号のみはほとんどの場合エラーを起こします。ダブルの＝＝はこの符号の左右のものを比べるという意味です。比べて真であれば１を、偽であれば０を返します。要素対要素で比べるべきところには、さらにドットがついて.＝＝となります。一方、イコール＝だけの場合の意味は、後ろの数値または行列を前の変数に代入格納するという意味です。GAUSS ではこのところの意味の違いははっきりとしていますから間違わないようにしてください。

【質問 19】
組み込み関数で行列の平均やメディアン、標準偏差、最小、最大などを求めて、結果を変数に受け渡すプログラムを書いたのですが、行数または列数が一致しないようです。どうしてですか？

【解答】
GAUSS では、特に断わりがなければ、ほとんどの場合、行列を組み込み関数で処理すると、その「列ごとの計算結果」が「列として出力」されます。

【対処方法】
まず、もとの計算される行列が１列のベクトルでない限り、計算結果は列ベクトルとして出力されますから、それに見合った変数を用意して受け渡す必要があります。行列全体をまとめたものの統計値や最大最小が１×１の数値としては出てこないということです。列ごとの結果が、こともあろうに「列として」出てきます。これには、理由があります。計量のデータはその多くが列として並んでいます。その計算に特化した GAUSS は、列ごとに計算するのです。また、その結果も、次の段階のデータとして見るので、GAUSS 上ではその結果もすべて「列として」出てくることになります。極めて計量に特化した論理的な仕様と言えます。数学的な仕様ではありませんので注意が必要です。

【質問 20】
複雑な計算をする組み込み関数を使うと、Bad expression or missing arguments というエラーが出てしまいます。どこもおかしくないと思うのですが。

【解答】
固有値を求めたり行列の複雑な計算をする組み込み関数の中には、一部２つ以上のリターンを返す関数があります。そういう関数の場合、イコール＝でもって１変数と結びつけることはできません。

【対処方法】
そういう関数の場合には、2 リターンの場合｛a,b｝=関数　とします。なお、a,b は任意の変数名でかまいません。要するに、{　}を用いて、そのリターンの数だけ変数をカンマで区切って置いてやる必要があるということです。例えば、

```
new; cls;
a={1 2,3 4};
r=eigrs2(a);
print r;
```

という場合には、組み込み関数 eigrs2 は２つのリターンをとる関数ですから、ｒ＝などというふうにすることはできなくて、それを受け渡すには、

```
{a,b}=eigrs2(a);
print a b;
```

｛　｝の中に２変数をカンマで区切って置くことが必要になります。

【質問 21】
ガンマ関数がおかしいとかいうエラーが出てしまうのですが、さっぱり原因がわかりません。ガンマ関数などは使っていません。

【解答】
分布の PDF や CDF、それに乱数発生に関する関数を使用していませんか？これらの組み込み関数は基本的にガンマ関数を内部に利用してプログラムされているため、エラーメッセージは、ガンマ関数に関するものになってしまうことがあります。

【対処方法】
２つほど原因が考えられます。１つには、GAUSS の組み込み関数はそのほとんどが複数の列の行列を一括して列として計算できます。しかしながら、その一部、特にこのガンマ関数を使っている関数は、複数列の一括計算ができません。その結果、エラーが生じます。その場合、今使っている関数が複数の列をインプットに受けつける関数であるのかをもう一度確認してしてください。また、それでも計算がやりたい場合には１列１列その関数に数値を代入して、ループで複数列の計算ができるようにプログラムを改造するのも１つの手です。もう１つの原因としては、ガンマ関数のパラメータの正値条件が、なんらかのプログラムミスによって、満たされていない場合が考えられます。自作や他人が作った分布関連の procedure を使用している場合には、もう一度、入力パラメータの条件を確認してください。

【質問 22】
組み込み関数の計算のところで、'{' : Syntax error のエラーが出てしまいます。何もプログラムミスもしていないように思われますが。

【解答】
組み込み関数の丸括弧の中では、論理式や計算式を直接書くことは許されていますが、直接数値が入った{　}括弧を代入することはできません。

【対処方法】
組み込み関数の丸括弧の中では、数値の入った{　}括弧は代入できない絶対的なルールがあるわけですから、{　}括弧で数値を代入した際には、必ず何か変数で一度受けておいて、それからその変数をその組み込み関数に代入することをやりましょう。GAUSS では、丸括弧には何でもリターンの中に計算式を書いたり、組み込み関数の中で論理判定をしたり、ほとんど何でもできますが、この{　}括弧は唯一の例外です。

【質問 23】
行列を列単位にソートしたいのですが、うまくいきません。１列の場合にはうまくいくのですが、複数列になるとうまく行かなくなりますが？

【解答】
組み込み関数 sortc(変数,数字)は、変数の数字の番目の列に注目して全体を「行ごとに」一括してソートするものです。したがって、ループで数字の番目をまわすだけでは、すべてての列で、小さいものから大きいものへ昇順にソートされている結果は得られません。最後の列だけがソートされたものになってしまいます。

【対処方法】
複数の列の行列を sortc でソートするのには工夫が必要です。いろいろな方法がありますが、

```
new; cls;
m={1 3 4,
   2 4 1,
   5 1 2,
   3 9 3};
i=1;
do while i<=cols(m);
   m[.,i]=sortc(m[.,i],1);
   i=i+1;
endo;
print m;
```

というふうに最初から m[.,i]として１列だけを抜き出して、それをこれもまた１列の m[.,i]に代入することを繰り返せばすべての列がソートされます。１列ごと処理して、受け渡す相手の変数も１列であることが大切です。

【質問 24】
論理の記号が条件分岐のところにあるのですが、１なのか l（エル）なのか区別がつかない上に、なかなかなんなのか覚えられません。

【解答】
論理の記号はドット部分を除いて、通常２文字の略号からなっていますが、英語圏以外のものにとって、これはマニュアルを毎回見なければならないほどわずらわしいものです。しかしながら、これはすべて通常の＞＜＝の記号とその組み合わせで数学と同じように代用できます。無理をして LT とか EQ とかその小文字の lt や eq などを使う必要はありません。

【対処方法】
以下のアルファベットの略号はすべて数学符号と同等です。

LT あるいは lt	<
LE あるいは le	<=
GT あるいは gt	>
GE あるいは ge	>=
EQ あるいは eq	= =
NE あるいは ne	/=

上の場合で、さらに要素対要素の比較の場合には略号または符号の前にドットがつきます。

【質問 25】
組み込み関数 gradp や hessp を用いたときに、Wrong number of parameters というエラーが出てしまいます。何がおかしいのでしょうか？

【解答】
関数の gradient を求める gradp や関数の Hessian を求める hessp という関数は、GAUSS の標準関数で、１値のインプットの関数をもとに、gradp(&f,b)などというふうに、ｂという数値または数値ベクトルをもとに数値的に関数の gradient や Hessian を求めるもので、別に proc または fn 定義でｆという関数を別途作成しておく必要がありますが、そのインプット引数は１値だけの関数になります。すなわち、例えば、ｆという名前であれば、f(x)というふうに１つのインプットである必要があります。ｘとｂは同じ名前である必要はありませんが、インプットが１個であることが大切です。

【対処方法】
もし、１値ではなくて、データ付きの２値関数の gradient や Hessian を求めたいのであれば、その関数 f(data,b)あるいはと全く f(b,data)とまったく同じ構造の１値の関数をデータ変数部分の処理を procedure 外から串刺しにする形で設定してやって、あくまで１値の関数を別に作る必要があります。なお、GPE2 を呼び出す場合には、２値関数を受けつけるような gradp2 という GPE2 の内部関数も利用できるようになりますが、GAUSS 本体でそれと同じことをやろうとすれば、目的関数あるいは Log-Likelihood 関数も１値関数に強引に書きかえる必要があります。

【質問 26】
行列の配列で、０がくるとエラーになるのですが？

【質問】
０の配列がとれないのは極めて GAUSS 的な仕様の問題です。数学では仮想の配列も考えられますが、計量のデータには０行３列などという概念は存在しないので、将来的には変更があるかもしれませんが、現在の仕様では、配列のインデックスに０は使えません。

【対処方法】
配列のインデックスが０となるケースのみを条件分岐させて別途処理するか、あるいは、行列の配列を＋１だけずらして計算して、最後に－１だけ戻して、修正を加えてみるのも１つの方法です。