0.3 ジュニア版　　関数について

関数とは何だろう？GAUSS のプログラミングで中心となる関数についてその基本的な概念と GAUSS での扱い方を解説します。なぜ私たちは関数を使わなければならないのかが端的にわかるでしょう。

関数とは何か

誰でも中高生の時に y＝f(x)というのを習ったことがあるのではないでしょうか。その頃は単なる「約束事」として、ｆというある決められた計算をするブラックボックスの中にｘを入れるとｙが出てくるとして、いくつかの計算例に取り組んだことでしょう。なんて無意味のことをしているのだろう、こんなの同じパターンを解けばいいのだから考える必要もない、など当時は考えたかもしれません。あえてもう一度、そこから始めましょう。

	代入する数値		ブラックボックス		計算結果
関数：	x	→	f(x)	→	y

上の図の関係のように関数とは、簡単に言ってしまえば、足したり引いたりかけたり割ったりするあらかじめ決められた約束事にしたがって計算されるブラックボックス f(x)があって、その中にｘという数値を入れてやると計算がされてｙという数値が出てくるということです。その例として、f(x)が今仮に

ブラックボックス　$f(x)=5x^2+4x+1$

であるとしましょう。このブラックボックスにいろいろな数値を与えてみましょう。

ｘ＝１のとき、f(1)＝５×１²＋４×１＋１＝10 となって、計算結果はｙ＝10
ｘ＝２のとき、f(2)＝５×２²＋４×２＋１＝29 となって、計算結果はｙ＝29
　　．　．　．　．　．　．　．　．　．　．　．　．　．　．　．
ｘ＝10 のとき、f(10)＝５×10²＋４×10＋１＝541 となり、計算結果はｙ＝541

などという計算ができました。ここでブラックボックス f(x)を単に２倍の値を求めるような f(x)=2x などと変更してしまえば、ブラックボックスに入れる数値が同じでも、

ｘ＝１のときｙ＝２
ｘ＝２のときｙ＝４
．　．　．　．　．

ｘ＝10 のときｙ＝20

というふうに、明らかに計算結果は異なるものになります。また、上ではブラックボックスのことを f(x)と呼んでいますが、何か違うものを区別してやりたい場合には、

g(x)＝3x

f1(x)=5x^2+4x+100

f2(x)=5x^2+4x+sin(x)

black(x)=0

などというふうにｇであるとかｆに添え字をつけて f1 とか f2 とかにすることもできるでしょうし、何か単語の名前をとって black などとすることさえできます。つまり、ｆだとかｇだとかというブラックボックスの名前は、複数のブラックボックスを考える際に、他のブラックボックスとの混乱をさけるために、その名前は何であってもよいのです。それではなぜこんなまどろっこしいことをするのでしょうか？答えはプログラム学習にあります。

以下は、上でやっていることをプログラムにしてみます。

プログラム

```
new;
cls;
x={1,2,3,4,5,6,7,8,9,10};          @ 10 x 1 column vector @
fn f(x)=5*x^2+4*x+1;
y=f(x);
print y;
```

画面表示

```
10.000000
29.000000
58.000000
97.000000
146.00000
205.00000
274.00000
353.00000
442.00000
541.00000
```

プログラム各行の説明

１行目　メモリを初期化する

２行目　スクリーン（画面）をクリアする

３行目　ｘに与える数値を列ベクトル（縦ベクトル）で与える
４行目　関数定義 fn という命令を使って f(x)=5*x^2+4*x+1 と定義する
５行目　その f(x)を変数ｙに代入する
６行目　変数ｙの内容を画面表示する

上のプログラムでは、あらかじめ f(x)を定義してやって、そこにｘを列ベクトルで一括して与えると瞬時に計算結果ｙのあつまり、つまり、列ベクトルが得られることがわかると思います。繰り返しになりますが、プログラムの冒頭には、new;は必須で、できれば cls;もつけてください。各変数定義や命令の後には必ずセミコロン「；」を忘れないでください。GAUSS では、行の概念がないのでセミコロン「；」までが１つの定義または命令です。ベクトルや行列を表現するときに、カンマ「,」までが１つの行です。プログラムでは、上の x=1 や y=f(x)のように後ろのものを前に代入または定義します。決して 1=x や f(x)=y とはなりませんので注意が必要です。もっと上の関数の意義が体感できる例を示しましょう。

プログラム

```
new;
cls;
x=seqa(1,1,1000);      @ {1,2,3,...,10000} From 1, step 1, 1000 rows. @
fn f(x)=5*x^2+4*x+1;
fn f1(x)=5*x^2+4*x+100;
fn f2(x)=5*x^2+4*x+sin(x);
y=f(x);
y1=f1(x);
y2=f2(x);
print y~y1~y2;
```

画面表示

```
      10.000000       109.00000       9.8414710
      29.000000       128.00000       28.909297
      58.000000       157.00000       57.141120
         .    .    .    .    .    .    .    .    .    .
      4994002.0       4994101.0       4994001.0
      5004001.0       5004100.0       5004000.8
```

プログラム各行の説明

１行目　メモリを初期化する
２行目　スクリーン（画面）をクリアする
３行目　ｘに与える数値を列ベクトルの１から１ステップで 1000 個分（すなわち１から 1000 まで）のシークエンスを与える seqa(1,1,1000)を用いて設定する。

４行目　関数定義 fn という命令を使って f(x)=5*x^2+4*x+1 と定義する
５行目　関数定義 fn という命令を使って f1(x)=5*x^2+4*x+100 と定義する
６行目　関数定義 fn という命令を使って f2(x)=5*x^2+4*x+sin(x)と定義する
７行目　その f(x)を変数ｙに代入する
８行目　その f1(x)を変数ｙ1 に代入する
９行目　その f2(x)を変数ｙ2 に代入する
10 行目　変数ｙ、ｙ1、ｙ2 の内容（列ベクトル）を水平方向のマージの記号「～」を使って束ねた上で、まとめて画面表示する

プログラムは３行目のシークエンス（続き番号）の設定と 10 行目の「～」による水平方向のマージのところが新しいだけで基本的には前のプログラムと同じです。今度は、ｘに１から 1000 まで、しかも、関数も f(x)だけでなく f1(x)と f2(x)も設定しておいて、まとめて計算してしまおうというものです。あっという間に計算は出来てしまいます。GAUSS をはじめとする数値計算のソフトウエアの１つの醍醐味は、「あらかじめ計算手順を決めてしまっていれば」どんなに入れる数値が多かろうと一瞬にして計算してしまうということなのです。それを明らかな形で表現したものが、「関数」ということになります。数値演算のソフトウエア言語の中でも、特に GAUSS は実際の経済や金融のデータと同じように「列ごとに」データを一括して計算するように設計されています。

注意事項

☆関数定義は fn の後に **fn f(x)=代数式;**の形で書くことができる
☆列ベクトルを水平方向にマージするには「～」を使う（なお、垂直方向は「｜」）
☆続き番号からなる列ベクトルを作成するには **seqa(スタート値,ステップ,個数)**
☆個数のところは、単一の列ベクトル計算の場合１万要素程度、そうでない場合には系列でその数を割った数程度までが、Light 版で計算できる限度

組込み関数について

それでは、実際の経済データをそのまま使って、関数を用いて「列ごとに」計算するのがいかに便利なのかを GAUSS の組込み関数（標準装備されている関数）を用いて以下に示そう。やることは極めて簡単で、年度も含めた 12 の系列からなる 7 行 12 列からなる実際のデータをカットアンドペーストで GAUSS の作業エディター上にもってきて、そのもの自身の画面表示をした後に、各系列の平均と分散を計算したものをまとめて表示させてみよう。

プログラム（緑の部分をカット＆ペーストで貼りつける）

```
new;
cls;
let data[7,12]=
1995 2.5 2.0 -6.5 3.6 665.9 3.9 7.8 -20.2 -34.3 4.6 14.7
1996 3.4 2.5 13.6 8.5 20.6 2.5 -1.9 -20.9 -9.3 7.9 10.5
1997 0.2 -1.2 -20.9 8.9 1.4 1.2 -6.3 -11.1 93.6 8.8 -1.6
1998 -0.8 1.1 -10.4 -5.1 -119.9 2.3 1.9 -101.0 8.6 -3.6 -6.6
1999 1.9 2.1 5.2 -0.3 -194.4 5.1 -0.7 3709.9 2.7 5.4 6.2
2000 1.7 -0.1 -1.5 9.3 -2.5 4.4 -7.4 67.9 8.7 9.4 9.6
2001 -1.8 1.3 -8.0 -4.7 -521.4 2.6 -6.7 -101.6 -20.1 -8.0 -4.7
;
print/rz data;
print;
print meanc(data)';
print (stdc(data)^2)';
```

画面表示　　(中略)

1995	2.5	-34.3	4.6	14.7
1996	3.4	-9.3	7.9	10.5
1997	0.2	93.6	8.8	-1.6
1998	-0.8	8.6	-3.6	-6.6
1999	1.9	2.7	5.4	6.2
2000	1.7	8.7	9.4	9.6
2001	-1.8	-20.1	-8	-4.7

1998.0000	1.0142857	7.1285714	3.5000000	4.0142857
4.6666667	3.5047619	1704.8624	44.956667	68.724762

プログラム各行の意味

1行目　メモリを初期化する

2行目　スクリーン（画面）をクリアする

3～11行目　dataという名前の変数に7×14のディメンションに強制的におさまるように取り込んで入れる

12行目　変数dataを「/rz」オプション（通常どおり「右寄せ（r）」「小数点以下のゼロ省略（z）」で画面表示する

13行目　改行（print;命令は変数もメッセージもない場合には改行として機能する）

14行目　列ごとの平均を求める組込み関数meancを用いて、dataを計算し1列として出てくる答えを「'」をつけて転置して画面表示する

15行目　列ごとの標準偏差を求める組込み関数stdcを用いて、dataを計算しそれを2乗したものが1列として出てくるのを「'」をつけて転置して画面表示する

今まで行列にデータを入れるのには、{　}を用いてきました。もちろんそれでも同じことができます。すなわち、

```
data={
1995 2.5 2.0 -6.5 3.6 665.9 3.9 7.8 -20.2 -34.3 4.6 14.7,
1996 3.4 2.5 13.6 8.5 20.6 2.5 -1.9 -20.9 -9.3 7.9 10.5,
1997 0.2 -1.2 -20.9 8.9 1.4 1.2 -6.3 -11.1 93.6 8.8 -1.6,
1998 -0.8 1.1 -10.4 -5.1 -119.9 2.3 1.9 -101.0 8.6 -3.6 -6.6,
1999 1.9 2.1 5.2 -0.3 -194.4 5.1 -0.7 3709.9 2.7 5.4 6.2,
2000 1.7 -0.1 -1.5 9.3 -2.5 4.4 -7.4 67.9 8.7 9.4 9.6,
2001 -1.8 1.3 -8.0 -4.7 -521.4 2.6 -6.7 -101.6 -20.1 -8.0 -4.7
};
```

上のように緑のところをカットアンドペーストで取り込んで、それぞれの行の後にカンマ「,」をつけてそれらが各行であることを示し、それらを{　}で囲むわけです。GAUSSには行の概念がないのでどこに{と}をつけようとかまわないわけで、最後に1つだけセミコロン「;」があれば、そこまでが1つの設定であると見なされます。これは従来行なってきた方法です。これをプログラム中のようにカンマや{　}をまったくつけない形で一続きのデータとして読みこみ、それをletで強制的に7×12のディメンションにしたものを変数dataに入れているわけです。

読みこんだ変数dataは「c(olumn)列」という略号がついている平均と標準偏差を求める組込み関数meancおよびstdcによって、「列ごとに」1つの系列として計算され、結果もまた「1列で」出てきます。そのため、「'」によって転置させて1行のベクトルに直してそれぞれ表示させています。なお、標準偏差の2乗は分散になります。小数点以下のゼロを表示させない方法としてprint命令の直後に「/rz」というオプションをつけてその変数

を画面表示させる方法があります。なお、ｒは右寄せ、ｚはゼロをなくすという意味です。このように、大きなディメンションのデータを「一括して」「列ごとに」「行列データの形のままで」計算してしまうのが、GAUSS の組込み関数の特色です。なお、画面表示では、その１、２列目の後、３列目から９列目まで省略してありますが実際にはもっと横に長くなります。それぞれの系列は、「年 度」「経済成長率」「民間最終消費支出」「民間住宅」「民間企業設備」「民間在庫品増加」「政府最終消費支出」「公的固定資本形成」「公的在庫品増加」「純輸出」「輸出」「輸入」の順に 14 個の系列からなっています。

それでは、さらに上のプログラムに、各列の「合計」「最小値」「最大値」を求めるプログラムの末尾に追加してみましょう。

```
print/rz sumc(data)';
print/rz minc(data)';
print/rz maxc(data)';
```

それには、同じく Column（列）ごとにと言う意味のｃの文字がついた sumc,minc,maxc でそれぞれ計算します。それを表示するには、これまでと同じように結果は１列として出てきますから、それを１行のベクトルに直すために「'」で転置させます。そして、小数点以下のゼロを消すために「/rz」オプションをつけて画面表示させます。

これだけではなくて、列と列のそれぞれの関係を調べる組込み関数として「分散共分散行列」や「相関係数」を求める組込み関数もあります。

```
print/rz vcx(data);
print/rz corrx(data);
```

「分散共分散行列」には vcx、「相関係数」には corrx という組込み関数を用います。なお、計算の結果は列と行の長さが等しい正方の対称な行列として出てきますから、これらの場合には転置させる必要はありません。これらの組込み関数もｃの代わりにｘで終わっていてデータを列ごとに扱うということを明示しています。おそらく、ｃをつけたのでは covariance のｃや correlation のｃと紛らわしいので区別するために、また結果が列としてではなく「行列データとして」出てくるという意味でｘが略語として採用されたのであると思われます。ここで、前者の vcx で出てくる答えの対角成分（ダイアゴナル）は上のプログラムで最初に求めた標準偏差の２乗の値、すなわち分散の値そのものです。それ以外の成分（オフダイアゴナル）は１行２列目であれば第１番目の系列と第 2 番目の系列の共分散、５行９列目であれば第５番目の系列と第９番目の系列の共分散です。なお、行と列をひっくり返したところの値は意味から言っても同じになります。それを対角成分（ダイアゴナル）を１にするように計算したものが相関係数となります。これらも一瞬の計算によって出てきます。

今度は、これまでの列ごとの計算ではなくて行列のそれぞれの要素を計算する組込み関数を使ってみましょう。「絶対値」「絶対値の自然対数」「絶対値の平方根」「指数関数ｅの

値」それに「100 分の 1 にした値」をそれぞれ計算してみましょう。

```
print/rz abs(data);
print/rz ln(abs(data));
print/rz sqrt(abs(data));
print/rz exp(data);
print/rz data/100;
```

絶対値には組込み関数 abs を使います。英語のアブソリュートバリューの略です。自然対数には数学の記号と同じように ln を用います（log は底が 10 のもので別物です）。それらを繰り返し入れ子構造にして使っても何ら支障はありません。平方根はスクエアの略である sqrt を使います。指数関数はエクスポネンシャルの略の exp を使います。パーセント表示を小数表示にするのに 100 分の 1 にするには、関数はなくて、単に変数 data をまとめて 100 で割ってやれば、それですみます。無論これらの関数は通常の数値も計算できます。

これまでとは違ったデータを使って、上でやった各種統計量の扱い方を確かめておきます。ここでは、乱数データを変数に入れて、それを利用する方法を行ないましょう。

プログラム

```
new;
cls;
data=rndn(10000,1);
print "       N=" rows(data);
print "    mean=" meanc(data);
print "     var=" stdc(data)^2;
print "     max=" maxc(data);
print "     min=" minc(data);
print "median=" median(data);
```

画面表示

```
       N=         10000.000
    mean=     0.0028847203
     var=       0.99771527
     max=        4.2579063
     min=       -4.0839463
  median=    0.00089989261
```

プログラム各行の説明

1 行目　メモリを初期化する

2 行目　スクリーンをクリアする

3 行目　標準正規分布にしたがう乱数を 1 万行 1 列生成させて、変数 data に入れる

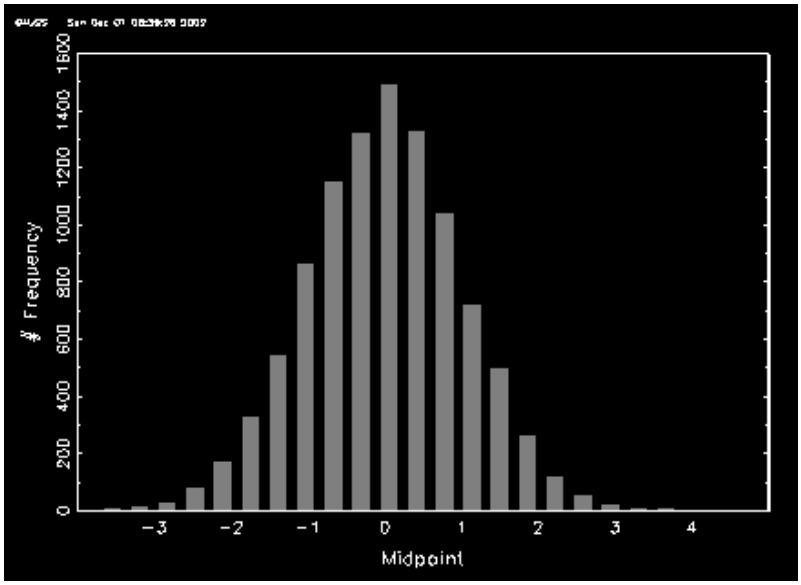
４行目　変数の列数を求める組込み関数 rows に変数 data を入れて、その結果をメッセージ部分とともに直接画面表示する(以下、すべて引用符” “の内側はメッセージ）

５行目　平均を求める組込み関数 meanc に変数 data を入れて計算させた結果を直接画面表示する（ここでは、１列のデータしか扱わないので、結果は１個だけなので結果の転置は不要）

６行目　標準偏差を求める組込み関数 stdc に変数 data を入れて計算させた結果を２乗して分散とした結果を直接画面表示する

７～９行目　最大値、最小値、メディアンを求める組込み関数 maxc、minc、median に変数 data を入れて計算された結果を直接画面表示する

上のプログラムは実際のデータを与えるのではなくて、平均０分散１のN(0,1)すなわち標準正規分布にしたがう乱数を１万個与えてそれを変数dataとしています。正規分布とはベルカーブ状にその頻度が分布するものの総称です。その中でも、その平均の位置が０、ベルカーブの裾野の広がりの大きさを示す分散が１であるものが標準正規分布です。そのように平均０分散１の比べやすいように「標準化」しようというものが、この「標準」正規分布です。GAUSSではこの標準正規分布の乱数をrndnという組込み関数で手軽に扱えるようになっています。乱数ですから、特に設定しなければ、Run実行するだびに若干ですが乱数の出具合は異なります。また、平均０分散１に近い値にはなりますが、完全には一致はしません。これが乱数です。上の結果では、mean=0.0028847203 およびvar=0.99771527となっていて、ほぼ０と１であることがわかると思います。最大値と最小値はその分布の広がりの限界を示しています。±４を少し超えるくらいまでこの分布は広がっています。最後に、メディアンはその分布の中位の値、ここではデータの個数は偶数なので、真ん中は真ん中の２つの値を合計して２分の１した値になります。



具体的なイメージは上のようなベル型のかたちをしたヒストグラムで表させる分布です。（なお、このグラフは後の章で取り上げることにします。）

　列ごとに計算する組込み関数が他にもたくさんあるのですが、もちろん前の章で扱ったような行列式detや逆行列invなどのように行の長さと列の長さが同じ正方行列を扱うものもたくさんあります。また、さらに行列の対角要素をはさんで全く対称になるような対象行列であることを必要とする組込み関数もたくさんあります。それらは、計量の理論計算に使われます。よく使う正方行列を前提とする組込み関数をいくつか挙げておきます。

プログラム

```
new;
cls;
A={ -2   4   -3,
     4   -2    3,
     6    0   3};
print rank(A);

B={8 2 1,
   2 4 2,
   1 2 3};
print chol(B);
print chol(B)'chol(B);
```

画面表示

```
   2.0000000

   2.8284271   0.70710678   0.35355339
  0.00000000    1.8708287   0.93541435
  0.00000000   0.00000000    1.4142136

   8.0000000   2.0000000   1.0000000
   2.0000000   4.0000000   2.0000000
   1.0000000   2.0000000   3.0000000
```

上のプログラムは、前の章で扱った行列に関する組込み関数の続きです。最初にＡという行列を与えてやって、組込み関数 rank によってその階数（ランク）を求めています。実際に、上のプログラムに前回の最後に計算した行列式の組込み関数 det を用いて

```
print det(A);
```

という 1 行を入れてみてください。答えは０で、|Ａ|＝０ということがわかります。そのＡの小行列（例えば{-2 4，4 -2}の行列式は明らかに(-2)×(-2)-4×４＝-12 で０ではあり

ませんから、そのような１つ次数が小さい（この場合もとの行列の次数が３で、考えている小行列の次数は２）小行列の行列式が１つでもゼロではないものがあるので、「階数」ランはそのときの次数に同じ２になります。少し概念的にはややこしいですが、小さい次数（＝行数＝列数）の行列の場合、計算方法はいたって簡単です。次に、Ｂという対象行列（上三角行列と下三角行列がひっくり返せば等しくなる）を与えてやって、その Cholesky 分解をしています。これはもとの行列の「行列のルート」のようなものであって、その前提として対称行列という条件に加えて、行列の正の値に相当する positive　definite であることがインプットする行列には求められます。そうでない場合にはエラーが生じます。それから最後に、実際に chol(B)'chol(B)が行列Ｂそのものに等しいことを画面表示させています。掛け算の場合、転置の「'」がついていれば、その後の掛け算の記号は省略できます。

このように、GAUSS では行列も扱える組込み関数がほとんどで、それらは、各種統計量を列ごとに計算する組込み関数、（普通の数値やベクトルも含めて）行列のすべての要素の値を変換する組込み関数、極めて計量の理論計算用の数学的な正方行列を前提とする組込み関数の３つぐらいに分類できます。これらの組み合わせて、より高度な統計計量的な計算を進めていくことができます。