## 2.4 行列 組込み関数 ver. 0.1

ここでは、GAUSS でプログラムを書くときに基礎となる各種の行列関連の組込み関数を紹介しましょう。2.1 では、基本的な演算についてその方法についてざっと見ましたが、ここではより統計計量で使われる方法や組込み関数について踏み込んで示します。

まず、統計計量のプログラムを書くうえで重要なものは、transpose と inverse です。transpose については、standard normal の分布から $\chi^2$ 分布を作る際に用いたように、ダッシュを用いてあらわします。このとき、ダッシュのあとには、それがたとえかける場合であっても、＊の印は必要ありません。したがって、モーメントは x'x として＊のマークなしで直接表すことができます。また、inverse はその略の inv が組込み関数として使われます。この inv(x)という形で、引数に正方行列を代入します。OLS estimator を求める原始的な方法の中で、その使い方を見ていきましょう。まず、y とｘの列ベクトルデータがそれぞれ与えられているとします。（スペースの関係で、{ }の中はカンマをつけて横に書いていますが、おのおの１列の縦ベクトルです。）そうすると、次の行列式での OLS の公式

$$B = (X'X)^{-1}X'Y$$

を用いて OLS estimator を求めるプログラムは次のようになります。

プログラム

```
new;  cls;
y={2,4,5,6,8,9,11};
x={1,2,3,3,5,6,7};
x=ones(7,1)~x;
b= inv(x'x)*x'y;
print b;
```

画面結果

```
1.0247525
1.4009901
```

上のプログラムの４行目では、切片のある OLS を行なうので、データ行列の行数と同じ行数の縦１列の内容がすべて１の行列を、もともとのｘのデータ行列にくっつけて、それをあらためてｘとしています。次の行では、OLS の公式をそのまま使って、GAUSS の転置の演算記号ダッシュのマークと逆行列を求める組込み関数 inv を用いて、書きなおしたものです。転置のダッシュがついたあとで別の行列をかけ合わせる場合には＊のマークは省略できるルールが GAUSS にはあります。ただし、inv(x'x)と x'y の間には、そのルールはあてはまりませんから、行列のかけ合わせに＊のマークが必要です。そして最後に、OLS の estimator ｂを画面表示しています。画面結果の上の数字は、１の縦一列のベクトルに対する係数（すなわちｙ切片の値）です。下の数字はｘに対する係数です。

**ポイント　転置 transpose の演算記号は、ダッシュ '**
**ただし、ダッシュの後の行列のかけ合わせの＊は省略できる**
**ポイント　逆行列の組込み関数は、　　inv(正方行列)**

次に、データについて平均、標準偏差、分散共分散を求めるプログラムを見ましょう。

プログラム

```
new;  cls;
y={2,4,5,6,8,9,11};
x={1,2,3,3,5,6,7};
m=meanc(y~x);
s=stdc(y~x);
s2=s^2;
cv=vcx(y~x);
cvsq=sqrt(cv);
print "mean" m; print "s.d." s; print "var" s2;
print "var cov" cv; print "square of var cov" cvsq;
```

画面結果

```
mean
        6.4285714
        3.8571429
s.d.
        3.1014590
        2.1930627
var
        9.6190476
        4.8095238
var cov
        9.6190476       6.7380952
        6.7380952       4.8095238
square of var cov
        3.1014590       2.5957841
        2.5957841       2.1930627
```

前と同じｘとｙの列ベクトルが与えられています。それを～の印で水平方向に横につなぎ合わせたものを行列として分析します。まず meanc はそれぞれの column についての mean を計算します。画面表示のように、結果の戻り値は、横ベクトルではなく、縦ベクトルで

すので注意が必要です。stdc も同様にそれぞれの column についての standard deviation を計算するものです。これも結果の戻り値は、横ベクトルではなく、縦ベクトルで出てきます。以下、たいていの行列のそれぞれの column の値を分析したり操作したりするものの戻り値は、縦ベクトルで出てきます。variance は standard deviation を２乗したものですから、行列として ^ の演算記号で ^ ２として２乗しています。その次は、組込み関数 vcx によって、データ行列からそれぞれの列の間の variance covariance matirx を計算しています。当然のことながら、その戻り値の対角行列は、上で計算した variance に等しいはずです。この vcx は引数に１列の縦ベクトルのデータが入る場合には、variance だけを求める関数の代用になることは前節で紹介しました。最後に、 ^ ２の２乗の逆である、行列の平方根 square をとる組込み関数として sqrt があります。variance covariance matirx の平方根をとれば、その結果の対角行列は stdc で求めた標準偏差に等しくなるはずです。

**ポイント　meanc(行列変数)　各列の平均を計算して列行列で返す**
**ポイント　stdc(行列変数)　　各列の標準偏差を計算して列行列で返す**
**ポイント　vcx(行列変数)　　各列と列の間の分散共分散を計算する**
**ポイント　sqrt(行列変数)　　行列の要素の平方根を計算する**
**その逆の２乗は　行列変数^2**

同様に、中位値 median を見つけるには組込み関数 median が用意されています。
プログラム

```
new;  cls;
y={2,4,5,6,8,9,11};
x={1,2,3,3,5,6,7};
print median(y~x);
```

画面表示

```
6.0000000
3.0000000
```

**ポイント　median(行列変数)　各列の中位値を計算して列行列で返す**

このほかに組込み関数として、その行の最大値や最小値をみつけるものがあります。
プログラム

```
new;  cls;
y={2,4,5,6,8,9,11};
x={1,2,3,3,5,6,7};
max=maxc(y~x);
```

```
min=minc(y~x);
maxmax=maxc(maxc(y~x));
minmin=minc(minc(y~x));
print max; print min;
print;   print maxmax;
print;   print minmin;
```

画面表示

```
11.000000
7.0000000

2.0000000
1.0000000

11.000000

1.0000000
```

maxcおよびmincは、それぞれcolumnの最大値、最小値を見つけて列行列で結果を出すものです。なお、結果は列行列で出されますから、２重にしてmaxc(maxc(x))やminc(minc(x))としてやれば、その行列全体の最大値および最小値が計算できます。GAUSSでは、出力が行列（または縦ベクトル）であると、前の結果との間に１行間隔があくのですが、スケラーの場合には間隔があかないので、print;の命令によって、１行間隔を入れています。

**ポイント　maxc(行列変数)　各行の最大値を見つけて列行列で返す**
**ポイント　minc(行列変数)　各行の最小値を見つけて列行列で返す**

統計関係のプログラム簡便化に効果を発揮する組込み関数には、それぞれの列の要素を足し合わせて合計をとったり、かけ合わせてその積全体を計算したりするものがあり、ループ構造のプログラムを使ってわざわざ計算しなくてもよいようになっています。

プログラム

```
new;   cls;
y={2,4,5,6,8,9,11};
x={1,2,3,3,5,6,7};
z=y~x;
print rows(z) cols(z);
```

```
meanz=sumc(z)./rows(z);
print "arithmetic mean" meanz;
gmeanz=prodc(z)^(1/rows(z));
print "geometric mean" gmeanz;
```

画面表示

```
        7.0000000        2.0000000
arithmetic mean
        6.4285714
        3.8571429
geometric mean
        5.6771945
        3.2439209
```

うえのプログラムは、まずｙとｘのデータ列を水平方向につないで行列としたものをｚとしています。そしてその行数と列数を組込み関数 rows と cols によって計算しています。それぞれの関数にはｓの字をお忘れなく。その値をもとに、meanc を使わずに、各行の合計を計算する組込み関数 sumc を使って、その値を rows で計算された行数によって割ることによって、算術平均を求めています。最後に各行の要素をかけ会わせる組込み関数 prodc を使って、その値を行数の逆数のべき乗、すなわち行数の乗数根をとって、幾何平均を計算しています。prodc は product と column の略です。計算結果から、幾何平均の値の方が算術平均の値よりも小さくなっていることが示されています。このように、平方根以外の乗数根は逆数をべき乗して^の演算記号を使って計算できます。

**ポイント　rows(行列変数) cols(行列変数)　それぞれ行列の行数、列数を計算する**
**ポイント　sumc(行列変数)　　各行の合計を求めて列行列で返す**
**ポイント　prodc(行列変数)　　各行の要素をかけ合わせた値を求めて列行列で返す**

　このほかに、合計とかけ合わせた数を見る組込み関数のたぐいには、累積和と累積積を求める組込み関数も GAUSS 用意されています。

プログラム

```
new;  cls;
data={0.2 180,
      0.2 280,
      0.2 350,
      0.2 450,
      0.2 750};
cum=cumsumc(data);
```

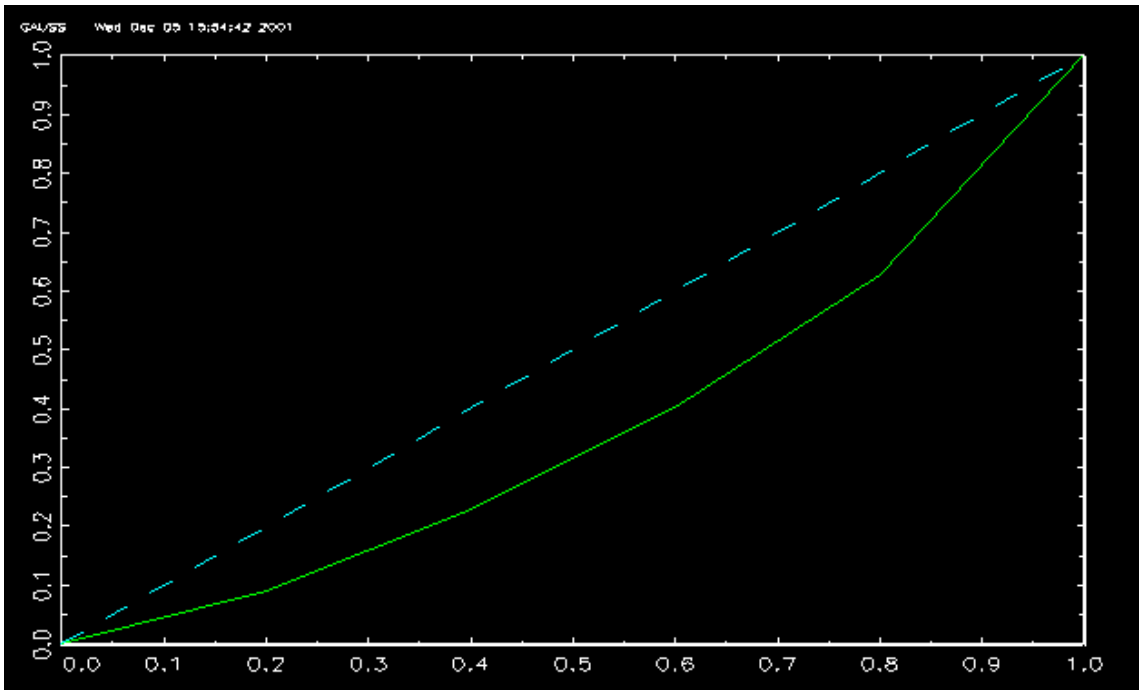
```
cumrate=cum[.,2]./sumc(data[.,2]);
print cum; print cumrate;
library pgraph;
graphset;
x=0|cum[.,1];
cumrate=0|cumrate;
y=cumrate~x;
xy(x,y);
```

画面表示

```
0.20000000        180.00000
0.40000000        460.00000
0.60000000        810.00000
0.80000000        1260.0000
 1.0000000        2010.0000


0.089552239
 0.22885572
 0.40298507
 0.62686567
  1.0000000
```

グラフ表示



このプログラムは、組込み関数 cumsumc を利用して累積和をもとめて、それを合計値で割ることによって累積値比率を出しています。横軸（ｘ軸）に累積相対度数を、縦軸（ｙ軸）に累積値比率をとって、ローレンツ曲線を描いています。なお、破線は完全平等線です。まず変数 data に５行２列のデータを入力しています。１行目は度数、２行目は仮に所得としましょう。組込み関数 cumsumc を使って、変数 data の累積相対度数と累積所得からなる行列変数 cum を求めています。そのうち２列目の値、すなわち累積所得を、所得の列の合計で割ることによって、累積所得比率を求め、それを変数 cumrate としています。そして、行列変数 cum と行列変数 cumrate を画面表示しています。さらに、cum の１列目からなる列ベクトルの最初に０をつけるために０と変数 cum[.,1]を縦方向にマージしてｘとします。そして、同様に cumrate 列の最初に０をつけるため、０と行列変数 cumrate を縦方向にマージします。そしてそのマージした行列とｘとを横方向にマージさせて、ｙとします。このマージしたｘは完全平等線を描くのにｘと全く対称ナ座標を得るためにつなぎ合わせています。それをグラフ機能のライブラリーを用いて、ｘｙ平面上に描いて、ｙの１列目がローレンツ曲線、ｙの２列目が完全平等線となるグラフをそれぞれ同一平面上に描いています。組込み関数 cumsumc(行列変数)は、最初の画面表示のところに示されているように、各行ごとに累積の１列目からその行までの累積和を計算するものです。cumulative な sum を column について行なうという略です。Residual Sum of Squares などを求めるときにもよく使う組込み関数です。なお、あまり使われませんが、cumprodc という各行の累積積を求める組込み関数も用意されています。使い方は、cumsumc と同じです。

**ポイント　cumsumc(行列変数)　　各行の累積和を計算して列行列で返す**
**ポイント　cumprodc(行列変数)　　各行の累積積を計算して列行列で返す**

そのほか、プログラムに必須な組込み関数として、round という数値を四捨五入して整数にするものがあります。ここでは階級の個数を決定する Sturges の公式をプログラムしてみましょう。

プログラム

```
new;  cls;
n=seqm(1,10,20);
sturges=round(1+log(n)/log(2));
format /rd 20,0; print n~sturges;
```

画面表示

```
                   1                    1
                  10                    4
                 100                    8
                1000                   11
```

| | |
|---|---|
| 10000 | 14 |
| 100000 | 18 |
| 1000000 | 21 |
| 10000000 | 24 |
| 100000000 | 28 |
| 1000000000 | 31 |
| 10000000000 | 34 |
| 100000000000 | 38 |
| 1000000000000 | 41 |
| 10000000000000 | 44 |
| 100000000000000 | 48 |
| 1000000000000000 | 51 |
| 10000000000000000 | 54 |
| 100000000000000000 | 57 |
| 1000000000000000000 | 61 |
| 10000000000000000000 | 64 |

上で使われているように組込み関数 round は引数が小数点を含んだ数であるときに、小数点以下を四捨五入するものです。データサイズがｎのときの階級の個数ｋはスタージェスの公式　ｋ　１＋(logｎ/log２)　にしたがいますから、例えば、小数点で出てきた数を四捨五入する際にこの round を使うのにもってこいです。ここでは、seqm で１から 10 倍ごとになっていく数を２０個つくる意味で、seqm(1,10,20)でｎのデータ列をつくり、それに対する階級の個数ｋの列を作っています。その際に公式にしたがい、その答えを round で小数点以下を四捨五入してｋとしています。最後、ｎとｋを水平方向にマージしてフォーマットを 20 文字スペース分とって、小数点以下０になる右寄せ十進法表示に設定して画面表示させています。なお、ｎのところを seqm(1,2,20)にするとｋは１, ２, ３,...とならびますが、これは公式を見ても理由は明らかです。この場合は round は必要なくなります。四捨五入は round を使いますが、小数点以下切り捨て truncate の場合には trunc を代わりに使います。例えば、数値が 1.99 であれば、round では２になりますが、trunc では小数点以下が切り捨てになって、答えは１になります。

**ポイント　round(行列変数)　　行列の要素の数値を小数点以下四捨五入して整数にする**
**ポイント　trunc(行列変数)　　行列の要素の数値を小数点以下切り捨てで整数にする**

　このほかに行列の計算で欠かせない行列式 determinant をとったり、固有値 eigenvalue や固有ベクトル eigenvector を計算したりすることができます。

プログラム

```
new;  cls;
x={1 -2,
   3  4};
print "eig" eig(x);
print "eigv" eigv(x);
print;  print "det" det(x);
print;  print "rank" rank(x);
```

画面表示

```
eig
        2.5000000 +          1.9364917i
        2.5000000 -          1.9364917i
eigv
        2.5000000 +          1.9364917i
        2.5000000 -          1.9364917i

      -0.64549722 -        0.50000000i       -0.64549722 +        0.50000000i
       0.00000000 +        1.0000000i         0.00000000 -        1.0000000i

det         10.000000

rank         2.0000000
```

簡略化のため２×２の行列についての場合について示していますが、実際にはそれ以上のディメンションの場合に効果を発揮します。組込み関数eigは固有値eibenvalueを求めるもの。組込み関数eigvは固有値eigenvalueと固有ベクトルeigenvectorをもとめるものです。組込み関数detはデターミナントを求めるものです。理論上、このdetの値は固有値をかけ合わせた数に等しくなっているはずです。チェックしてみてください。組込み関数rankは行列のランクを計算するものです。この場合、あきらかにどの行も列もそれぞれの倍数になっていませんからランクは２のはずです。なお、固有値や固有ベクトルを計算する関数には、もとの行列の型によっていろいろと変形バージョンが用意されていますので詳しくは必要に応じて使い分けてください。eigとeigvで普通は事足りると思われます。一部には、eigrgやeigrg2といった行列がreal generalなケースにのみ用いられる関数を使っているプログラムも見かけるかもしれませんが、結果は上のeigとeigvにぞれぞれ同じになります。

**ポイント　det(正方行列)　　行列のデターミナントを計算する**

**ポイント　eig(正方行列)　　行列の固有値を計算する**
**ポイント　eigv(正方行列)　行列の固有値と固有ベクトルを計算する**
**ポイント　rank(行列変数)　 行列のランクを求める**

最後にラグをとる組込み関数lag1とlagnについて説明します。
プログラム

```
new;  cls;
x=seqa(1,1,10);
y=lag1(x);
z=lagn(x,2);
print x~y~z;
a=x~y~z;
print a[3:10,.];
u=lagn(x,-1);
v=lagn(x,-2);
print x~u~v;
```

画面表示

```
1.0000000              .              .
2.0000000      1.0000000              .
3.0000000      2.0000000      1.0000000
4.0000000      3.0000000      2.0000000
5.0000000      4.0000000      3.0000000
6.0000000      5.0000000      4.0000000
7.0000000      6.0000000      5.0000000
8.0000000      7.0000000      6.0000000
9.0000000      8.0000000      7.0000000
10.000000      9.0000000      8.0000000

3.0000000      2.0000000      1.0000000
4.0000000      3.0000000      2.0000000
5.0000000      4.0000000      3.0000000
6.0000000      5.0000000      4.0000000
7.0000000      6.0000000      5.0000000
8.0000000      7.0000000      6.0000000
9.0000000      8.0000000      7.0000000
10.000000      9.0000000      8.0000000
```

| | | |
|---|---|---|
| 1.0000000 | 2.0000000 | 3.0000000 |
| 2.0000000 | 3.0000000 | 4.0000000 |
| 3.0000000 | 4.0000000 | 5.0000000 |
| 4.0000000 | 5.0000000 | 6.0000000 |
| 5.0000000 | 6.0000000 | 7.0000000 |
| 6.0000000 | 7.0000000 | 8.0000000 |
| 7.0000000 | 8.0000000 | 9.0000000 |
| 8.0000000 | 9.0000000 | 10.000000 |
| 9.0000000 | 10.000000 | . |
| 10.000000 | . | . |

上のように容易に行列関数を時系列データを見たててラグをとることができます。単に１期のラグをとるには、組込み関数lag1を使います。引数に行列変数だけをとって、１期ずらした変数を作ります。なおその戻り値の最初の行にはミッシングバリューを表すドット．が入ります。この際に、変数に時系列を表す列などはGAUSSの場合必要ありません。ただ単に行列変数を時系列と仮定してラグをとってくれます。２期以上のラグを作る場合にはlagnという組込み関数を使います。lagn(行列変数,ラグ数)というようにラグ数をその２番目の引数に設定するとラグ数だけずらした変数が作られます。なお、ラグ数だけの数の行数その行の最初にはミッシングバリューを表すドット．が入ります。これらをもとの列ベクトルｘとつなぎ合わせると、行列として分析するには最初の２行が邪魔ですから、その行列をａとおいて、行列のディメンションを操作してa[3:10,.]というように３行目から１０行目までで列はワイルドカードと指定すると、分析する準備が整います。lagnにも他の言語に必要な時系列を指定するための列ベクトルなどは必要ありません。行列を勝手に時系列のデータが入った行列とみなしてくれます。プログラム後半にあるように、lagnの第２番目の引数にマイナスの数を入れると、今度はラグではなくて、１期進めてくれます。画面表示にあるように、この場合には、最後のほうの行に負のラグ数だけのミッシングバリューのドットが入ることになります。この場合には、上と同様のディメンション指定の手法により使える部分の行列を切り取ることが容易にできます。

**ポイント**　lag1(**行列変数**)　**行列変数を時系列データと見たてて１期ラグをとる**
**ポイント**　lagn(**行列変数,ｋ**)　**行列変数を時系列データと見たててｋ期ラグをとる**
**ポイント**　**ｎがマイナスの数の場合には、変数をｎ期進ませる役割を果たす**