2.3 行列 データの生成 ver. 0.1

前回は生のデータを加工する手法を説明しましたが、今度は自分でデータを作り出したり代入したりする方法を紹介しましょう。GAUSS は他の統計計量ソフトよりもこの面で格段に充実したところを見せてくれます。まずは繰り返しになりますが、GAUSS の真骨頂である乱数作成と順列数作成から始めましょう。

プログラム

```
new;  cls;
   rndseed 10000;
   xn=seqa(1,1,10)~rndn(10,1);
   xu=seqa(1,1,10)~rndu(10,1);
   print xn; print xu;
```

画面結果

```
 1.0000000      -1.2355640
 2.0000000      0.45194399
 3.0000000       1.1973594
 4.0000000     -0.33016131
 5.0000000      -1.9308480
 6.0000000      0.43909975
 7.0000000      -1.4560906
 8.0000000     -0.49176805
 9.0000000     -0.92582422
 10.000000     -0.20847823

 1.0000000      0.83007734
 2.0000000      0.72723276
 3.0000000      0.34134168
 4.0000000      0.98962896
 5.0000000      0.37630376
 6.0000000      0.25351444
 7.0000000      0.36401240
 8.0000000      0.98341029
 9.0000000      0.25557991
 10.000000      0.38073625
```

このプログラムは、乱数のシードを 10000 と設定して、いつ何時にプログラムを実行しても同じ乱数が作成されるようにした上で、seqa 関数で１から１ステップで 10 個のシリーズの１列行列を作成して、10 行１列の standard normal にふるまう乱数を、～の演算子を用いて、横に水平方向にくっつけあわせたものを xn と定義しています。また、seqa 関数で１から１ステップで 10 個のシリーズの１列行列を作成して、10 行１列の uniform にふるまう乱数を横に水平方向にくっつけあわせたものを xu と定義して、それぞれを画面表示させています。seqa は以前取り上げたように、年次や 1,2,3,4,….というような数列を１列の縦ベクトルとして生成する組込み関数です。seqa(**最初の数、ステップ幅、個数**)という書き方で、ステップ幅ごとに数が増していく数列を作成します。sequence と additive の略で seq+a となっています。同様な行列を生成する組込み関数には seqm があります。seqm(**最初の数、倍数、個数**)という書き方で、今度は、倍数のところの数、例えば、２であると２倍ずつ最初の数が掛け合わされていく数列を１列に生成します。こちらは sequence と multiplicative の略で seq+m となっています。rndn は random+normal の略で、rndn(**行数,列数**)によって、行数列数のディメンションの standard normal にふるまう乱数を生成します。rndu は random+uniform の略で、rndu(**行数,列数**)によって、行数列数のディメンションの normal にふるまう乱数を生成します。seqa でできた 1,2,3,…..,10 の 10 行１列の行列と乱数の 10 行１列の行列をそれぞれ横に水平方向につなぎあわせています。１から始まる必要はなくて、例えば 1980 年から１年ずつの場合には seqa(1980,1,20)などとすることができます。ただし、変数の個数についてはフェンスポスト問題を考えて設定してください。1980 年から 2000 年までの年は 21 個あります。さて、ここで rndseed **数値**という文がなければ、乱数は起動時からのコンピュータ時に依存して生成されます。プログラム実行ごとに乱数は変化します。ただし、乱数のシードは rndn も rndu も共通です。なお、以前説明したように、rndns(**行数,列数**,seed)や rndu(**行数,列数**,seed)というふうに、シード seed の意味のｓの文字を加えて、外部で seed=10000 などとすることもできます。次のプログラムは上のプログラムとまったく同じ結果を生成します。

プログラム

```
new;  cls;
seed=10000;
xn=seqa(1,1,10)~rndns(10,1,seed);
xu=seqa(1,1,10)~rndus(10,1,seed);
print xn; print xu;
```

シードseedのところは、任意の変数も可能です。例えば、ａとかｂでもかまいません。ただし、rndnsとrndusの第３要素に直接数値を代入することはできません。常に間接的に変数を入れます。１列の乱数ではなくて、２列以上の行列として乱数を生成するには、次のように、第２要素を１以外の数に変更します。

プログラム

```
new;  cls;
seed=10000;
xn=seqa(1,1,10)~rndns(10,3,seed);
xu=seqa(1,1,10)~rndus(10,4,seed);
print xn; print xu;
```

ポイント　rndseed 数字　　乱数関数用のシードの任意の数値設定
ポイント　rndn(行数,列数)　　Standard Normal分布を生成する乱数関数
ポイント　rndu(行数,列数)　　Uniform分布を生成する乱数関数
ポイント　seqa(最初の数,ステップ幅,個数)　　１列の数列を生成する関数
rndnとrnduにのみ"s"をつけてseedを引数として指定する方法はver.4.0から廃止され、ほかの乱数の設定と同じになりました。なおrndseed 10000; rndn(10,3); rndu(10,4);とします。

Standard Normal分布やUniform分布以外にも、GAUSSでは標準の組込み関数として、Poisson分布、Negative Binomial分布、Beta分布、Gamma分布などを生成する乱数生成関数が備えつけられています。ここで、1000行１列の乱数をそれぞれの分布について発生させてみましょう。同じデータを再現するためにrndseed 10000の文を使ってみます。それぞれの乱数関数の最初の２要素は、それぞれ乱数の行数と列数で、ここでは1000行１列の乱数を発生しますから1000,１となっています。それ以降、第３番目以降の要素は、それぞれの分布が必要とするパラメーターとなっています。I/Oのところでやったように、分布のときはヒストグラムを描くのが常套手段ですから、pgraphというグラフを描くライブラリを呼び出して、graphsetでその中のグローバル変数をリセットしたのち、histでヒストグラムを描いています。hist(x,10)のｘはデータ行列、ここでは１列の縦ベクトル。10はヒストグラムの区切りの数です。meancは列columnの平均を得る組込み関数、vcxは行列のvariance covariance matrixを得る組込み関数で、１列だけのデータの場合には１変数ですからvarianceだけが計算されます。なおそれぞれの列のstandard deviationを求めるにはstdcという組込み関数を普通は使います。２乗してプログラムが長くなるので、vcxでここは代用します。

プログラム(Poisson分布)

```
new;  cls;
rndseed 10000;
x=rndp(1000,1,3);
library pgraph;
graphset;
hist(x,10);
```

```
print meanc(x) vcx(x);
```

まず、上の最初のプログラムは、poisson分布のグラフと平均分散を求めるものです。rndpの第３番目の要素である３は下のdensity functionでλ＝３という意味になります。

$$P(X = x) = \frac{\lambda^x}{x!} e^{-\lambda}$$

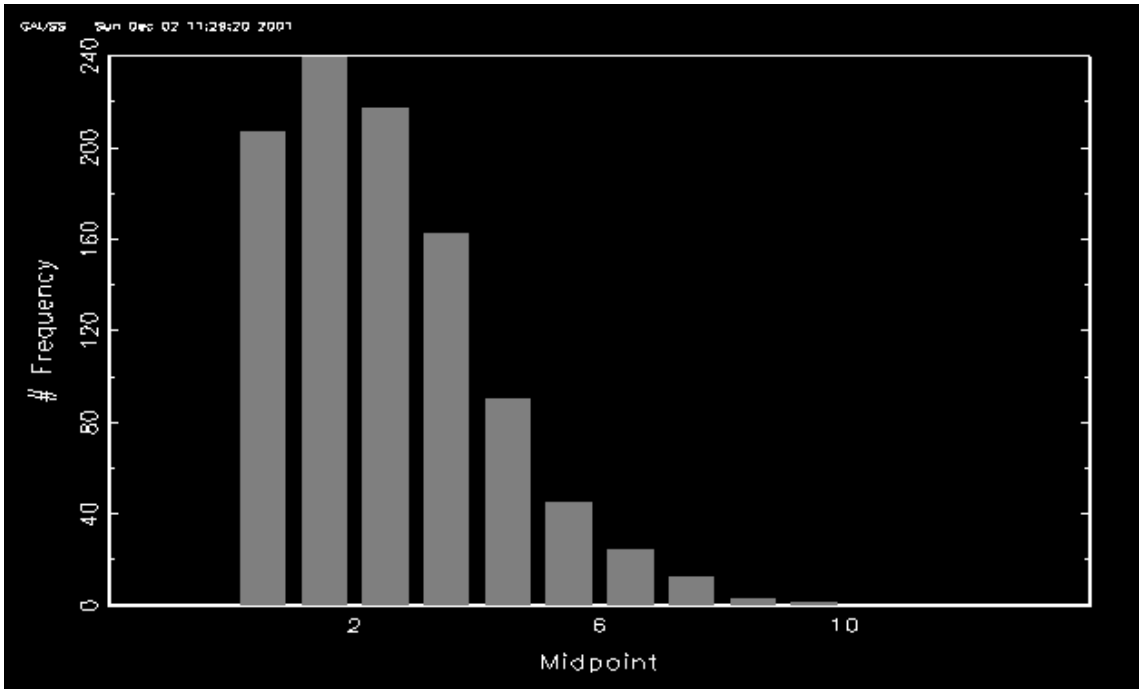
$x = 0,1,2,...,n$

$\lambda > 0$

このpoisson分布の場合の平均分散のパラメーターλとの関係は次のようになるはずです。

E(x)＝λ，Var(x)＝λ

グラフ表示



画面結果

```
        2.9560000           3.0811451
```

なお、パラメータを変更した場合のグラフは、ヒストグラムでは複数行は重ねて描けませんのでI/Oのセクションで説明したパネル画面を分割する方法をとらなくてはなりません。平均、分散ともに、だいたいのところパラメーターλに近い値となっています。シードを変更したり、乱数の数（行数）を増やしたりすることによってλの値に平均分散は近づくはずです。

ポイント　rndp(行数,列数,λ)　　パラメーターλのPoisson分布を生成する乱数関数

さて、次にNegative Binomial分布の行列を作って、ヒストグラムと平均分散を求めてみましょう。簡単なプログラムは、下のようになります。

プログラム(Negative Binomial分布)

```
new;  cls;
rndseed 10000;
x=rndnb(1000,1,3,0.3);
library pgraph;
graphset;
hist(x,10);
print meanc(x) vcx(x);
```

ここで、rndnb の nb は Negative Binomial の略です。rndnb の第３要素はパラメーターｋで第４要素は確率ｐとなっていて、次のような density function の形になっています。

$P(x = X)=_{k+x-1}C_x p^x (a-p)^k$

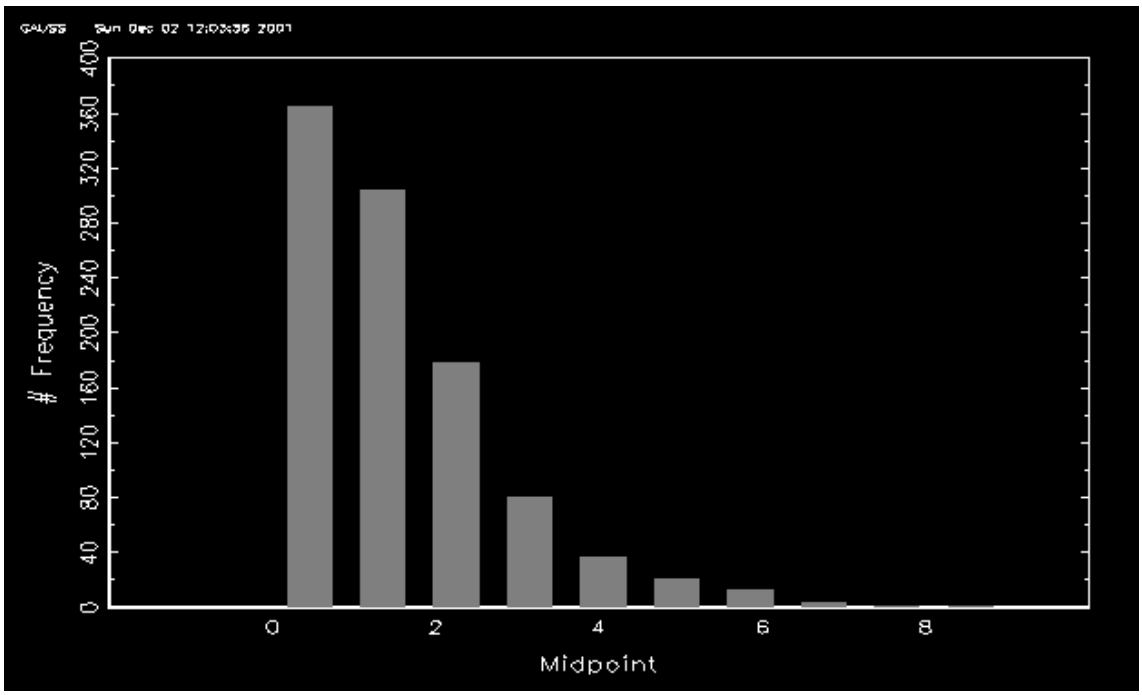
$x = 0,1,2,....k$

$o < p < 1$

このときのパラメーターｋとｐと平均分散の関係は次のようになるはずです。

$$E(x) = \frac{p}{1-p}k, Var(x) = \frac{p}{(1-p)^2}k$$

グラフ表示



画面表示

1.2500000　　1.9274274

ポイント　rndnb(**行数,列数,ｋ,ｐ**)　　**パラメーターｋ、ｐの** Negative Binomial **分布を生成する乱数関数**

次に Beta 分布を示しておきます。

プログラム(Beta 分布)

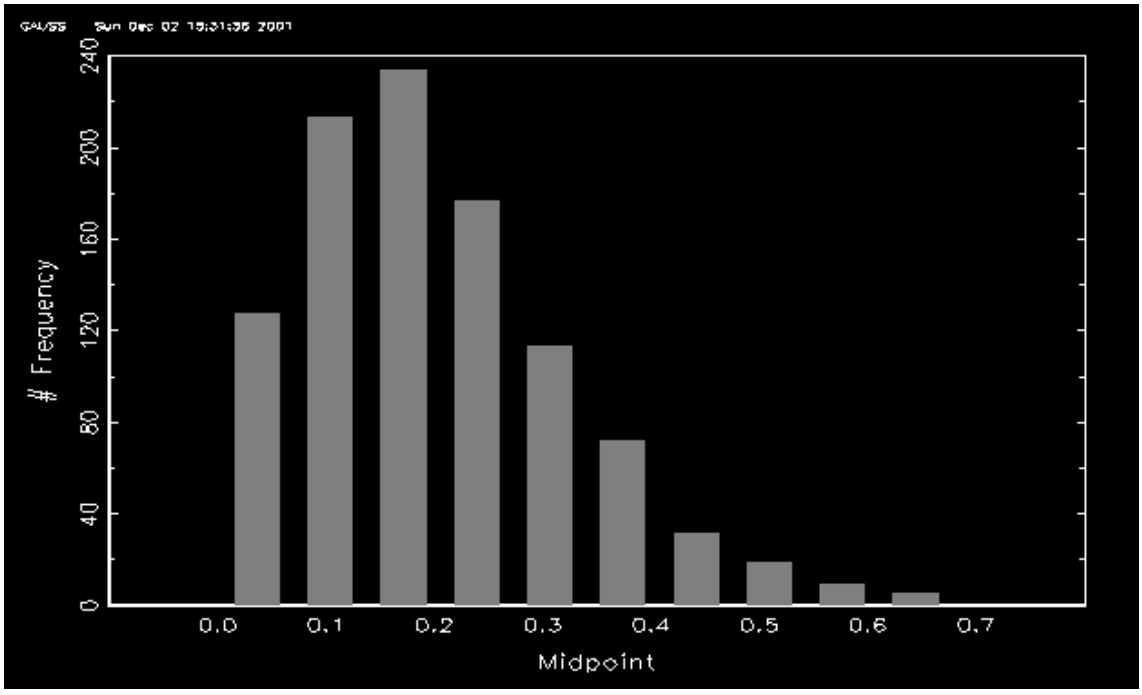
```
new;　cls;
rndseed 10000;
x=rndbeta(1000,1,2,8);
library pgraph;
graphset;
hist(x,10);
print meanc(x) vcx(x);
```

この場合のdensity functionは０＜ｘ＜１について、パラメータａ＞０，ｂ＞０に対して

$$P(X=x)=\frac{1}{B(a,b)}x^{a}(1-x)^{b-1}$$

となります。この場合、第３要素がａ、第４要素がｂとなっています。

グラフ表示



画面表示

0.20260811　　0.014814629

この場合の平均と分散とパラメーターとの関係は次のようになっているはずです。

$$E(x)=\frac{a}{a+b}$$

$$Var(x)=\frac{ab}{(a+b+1)(a+b)^2}$$

ポイント　rndbeta(**行数**,**列数**,a,b)　　**パラメーター**a,bのBeta**分布を生成する乱数関数**

最後に、１パラメーターの特殊なケースについてのガンマ分布を発生させる乱数関数について説明しておきます。

プログラム(Gamma分布)

```
new;  cls;
rndseed 10000;
x=rndgam(1000,1,3);
library pgraph;
graphset;
hist(x,10);
print meanc(x) vcx(x);
```

これは、一般的な形のガンマ分布のdensity functionである

$$P(X-x)=\frac{1}{\Gamma(\gamma)}\alpha^{\gamma}x^{\gamma-1}e^{-\alpha x}$$

$\gamma>0,\alpha>0,x>0$

のもとでαを

$\alpha=1$

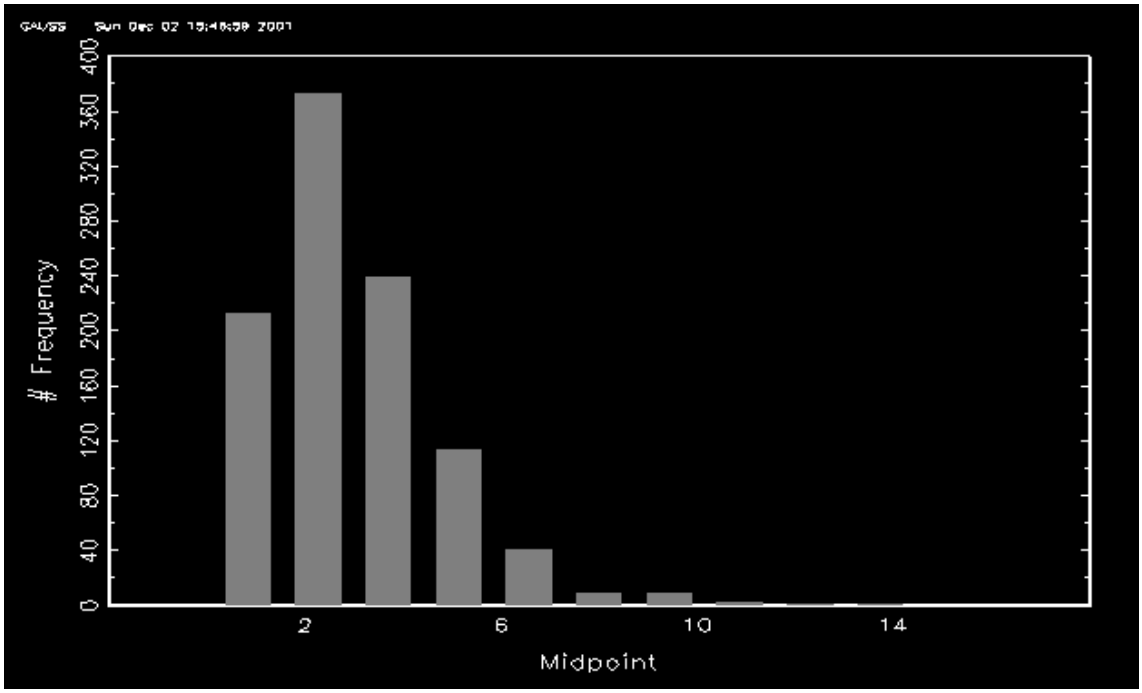
としたときの特殊なケースのガンマ分布を示しています。つまり、densityは

$$P(X=x)=\frac{1}{\Gamma(\gamma)}x^{\gamma-1}e^{-x}$$

ということになります。上のプログラムでは、第３要素γが３についてのケースです。平均分散は、E(x)＝γ，Var(x)＝γとなるはずです。

ポイント　rndgam(**行数**,**列数**,γ)　　**パラメーター**γ**のガンマ分布を生成する乱数関数**
（ただし、もう１つのパラメーターα**＝１）**

グラフ表示



画面表示

2.9476511　　3.0612730

そのほかの分布の乱数関数は、GAUSSのバージョンが上がるにつれ、徐々に加えられていますが、基本的には、自分でプログラムを書くことができます。カイ二乗χ(n)の分布を示す乱数は、standard normalのrndnを用いて、次のようにプログラムできます。まずは、一番簡単なdf＝１のケースについて示します。ちなみに、df＝ｎの場合のχ^{2}分布の平均と分散は、

E(x)＝ｎ，Var(x)＝２ｎ

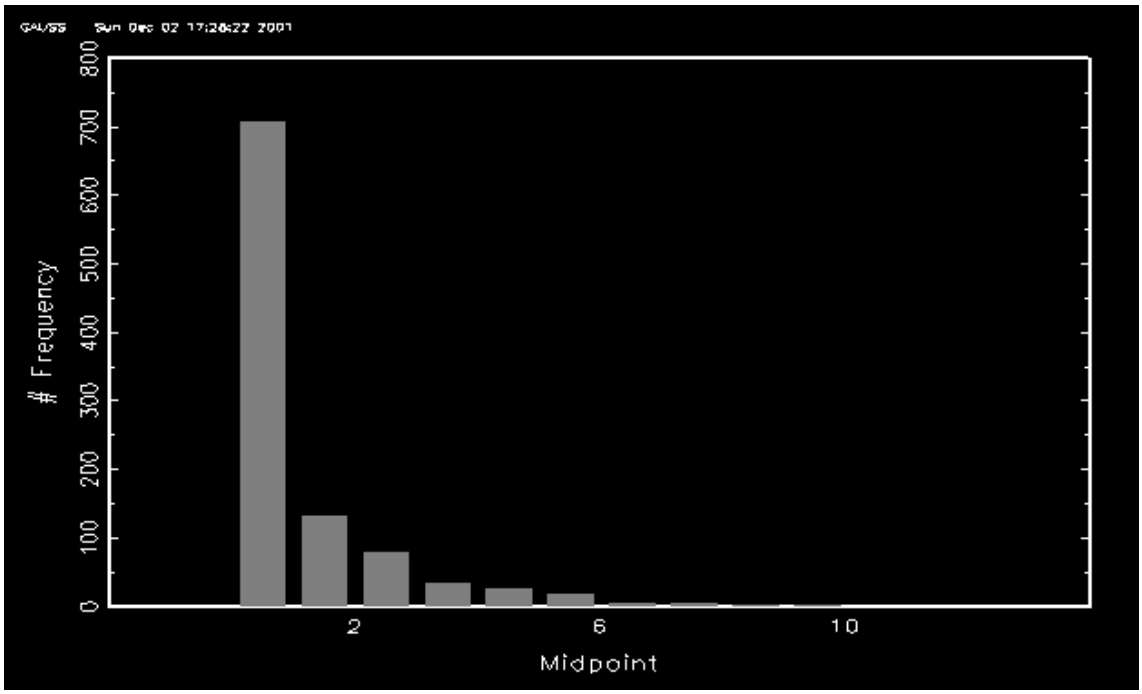
となるはずです。

プログラム

```
new;  cls;
rndseed 10000;
x=rndn(1000,1);
x=x.*x;
library pgraph;
graphset;
hist(x,10);
```

```
print meanc(x) vcx(x);
```

グラフ表示



画面表示

```
0.97723603          1.8572844
```

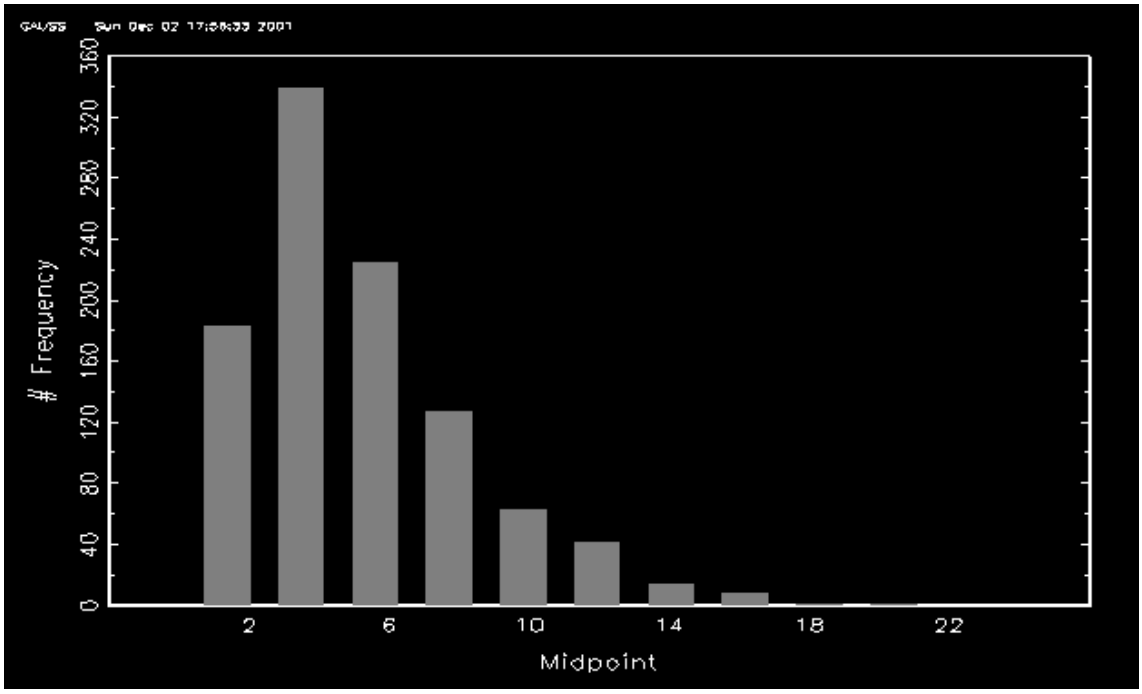
プログラムは、standard normal の分布にふるまうそれぞれの値を２乗したものを自由度１の χ^2 分布としています。ここでは、”. ＊”の演算子によって、要素対要素の積を作っています。平均、分散は若干のところずれていますが、rndn の行数を飛躍的に増やせば、E(x)＝n＝1，Var(x)＝２n＝２に一致するはずです。（なお、GAUSS の Light 版では、扱える行列のディメンションが限られているため、一列の場合、数万程度の行が限界です。GAUSS のフルバージョンでは、扱えるデータの大きさは、メモリの大きさにのみ依存します。また、GAUSS のバージョンによって、乱数の値は異なる場合があります。）自由度 df＝２以上の分布については、χ^2 分布の定義「N(0,1)にしたがう互いに独立な df 個の確率変数の２乗和の分布」を利用すると次のようなプログラムとなります。

プログラム

```
new;  cls;
rndseed 1000;
df=5;
x=rndn(1000,df);
```

```
x=sumc((x.*x)');
library pgraph;
graphset;
hist(x,10);
print meanc(x) vcx(x);
```

グラフ表示



画面結果

```
5.0930879        10.066111
```

ここでは、列の要素すべてを合計 sum する sumc という組込み関数を使っていますが rndn でできた行列変数を横方向に足さなければ行けないため、sumc の引数のところをダッシュのマークでさらに転置させて、実質行方向に合計しています。なお、sumc の答えである戻り値は、列ベクトルとして表示されるようになっていますから、そのまま分布として使えます。なお、light 版では、1000 行程度が複数列の場合、限界です。rndseed の値によって、より理想的な分布を探してください。なお、フルバージョンで、もっと大きな行数を考えると、平均分散は理想値に近づきます。

さて、すこし違った行列生成のトピックスとして、I 行列、ゼロ行列、すべての要素が１である行列などの作成があります。それぞれ、eye、zeros、ones という組込み関数が用意されていて、そのディメンションを引数として与えてやれば、手で打ち込むことなく、行列がすばやく作成されます。

プログラム

```
new;  cls;
    format /rz;
    print eye(3); print zeros(2,5); print ones(3,2);
```

画面結果

```
    1       0       0
    0       1       0
    0       0       1

    0       0       0       0       0
    0       0       0       0       0

    1       1
    1       1
    1       1
```

eye(n)はｎ×ｎのＩ行列を、zeros(m,n)はm行ｎ列のゼロ行列を、ones(m,n)はm行ｎ列の要素がすべて１の行列を、それぞれ生成します。ここでは、わかりやすいように、小数点以下の０をすべて省略するｚオプションをフォーマット文に書いています。それぞれのprint 文にオプションをつけても同様な画面表示が得られます。ただし、これは見かけの表示であって、以前やりましたように、format で途中計算の有効数字が変更になるわけではありませんので誤解のないようにしてください。Ｉ行列は常に正方行列ですので、行と列の数が等しいので、引数は１つしかありません。Format 文を使ったら、その後必ず 1 度GAUSS を終了させてください。format の内容が以前のまま残ることがあります。

zeros は主にプログラムを組む際の行列の初期値に使います。ones は、ones(n,1)としてｎ行のデータの regression をする際の切片を求める際に、要素がすべて１であるｎ行１列の行列をデータの独立変数行列にくっつける際に使います。例えば、次のようなｘという独立変数データの行列があるとします。それに、切片を含んだ regression をするために、１列すべて１をマージしなければなりません。その際には、以下のようにします。

プログラム

```
new;  cls;
x={3 4 5,
   2 3 4,
   4 2 5,
   3 2 4,
  2 3 3};
```

```
n=rows(x);
x=ones(n,1)~x;
print x;
```

画面結果

1.0000000	3.0000000	4.0000000	5.0000000
1.0000000	2.0000000	3.0000000	4.0000000
1.0000000	4.0000000	2.0000000	5.0000000
1.0000000	3.0000000	2.0000000	4.0000000
1.0000000	2.0000000	3.0000000	3.0000000

ポイント　eye(n)　　n×nの I 行列
ポイント　zeros(m,n)　　m×n のゼロ行列
ポイント　ones(m,n)　　m×n のすべての要素が１の行列

また、大変重要なことなのですが、ダミー変数を作成する際には、この切片の１の１列をつけないで、 I 行列を縦にデータの行数だけのばしたものを代わりにマージします。これは、組込み関数 reshape と eye の組合せで実現できます。

プログラム

```
new;  cls;
   d=reshape(eye(4),20,4);
   print /rz d;
```

画面結果

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0
0	1	0	0

0	0	1	0
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

この例は、ダミー変数、例えば４期ごとにダミー変数をデータにつけていく際のものです。eye(4)でまず４期分のダミー変数の１と０の組合せを作って、それを reshape の第 1 要素に入れます。そして、第 2 要素に、それをのばしてできる行列の行数、第３要素に、その列数（これは eye の引数に同じにしておくこと）を入れます。そうすると、reshape の戻り値はダミー変数を全期間分だけのばしたものになります。12 期ごとであれば、eye(12)を使って、reshape(eye(12),20 ,12)となります。行数は 12 で必ずしも割りきれる必要はなくて、余れば、そこまで行列のパターンが入って入りきれない分は切り捨てられます。Reshape の第 1 変数のところは、一般に行列変数が直接または間接に変数として入るのですが、必ずしもダミー変数の必要はなくて、一般的な行列のパターンを入れることができます。

これとは別に、GAUSS では正式なダミー変数を生成する dummy という組込み関数が備え付けられています。こちらは、循環するような時系列ではない、質的なダミー変数を作成するのに向いた機能をもっています。以前用いた datafile2.txt を使って実際にダミー変数を作ってみましょう。

プログラム

```
new;  cls;
load data[392,5]=d:datafile2.txt;
  v={3,4,5,6,7};
  d=dummy(data[.,2],v);
  x=d~data;
  print /rz x;
```

上のdummyの引数の中には、まず第１要素としてダミー変数を作る上で基準とする１列の縦ベクトルが入ります。ここでは、行列変数dataのうち２列目を基準としています。当然のことながら、data[.,2]は392行１列の１列のベクトルとなっています。そして、第２要素にはダミー変数を作る際の基準となる列の数値の区切りの数をいくつか指定しています。少しわかりづらいですが、ここでは、data[.,2]という列ベクトルのおのおのの数値を基準にして、３以下（３を含む）をまず第１番目のダミー変数にして１とおいて、３より大きく４以下を第２番目のダミー変数として１とおいて、４より大きく５以下を第３番目のダミー変数にして１とおいて、５より大きく６以下を第４番目のダミー変数にして１とおいて、６より大きく７以下を第５番目のダミー変数にして１とおいて、最後７より大きいものを第６番目のダミー変数にして１とおいて、その他すべてに０を入れたダミー変数の行列、

この場合、392行×６列の１と０からできた行列を作ります。ここでは、そのダミー変数の行列ｄともともとの行列dataを水平方向につなぎ合わせて行列ｘとしたうえで、小数点以下の０を省略する表示のフォーマットであるｚオプションで右そろえで（ｒ）で画面表示させています。v={3,4,5,6,7};はv=3|4|5|6|7;としても同じ意味になります。なお一括してd=dummy(data[.,2],{3,4,5,6,7});とは( )の中に{ }を書くことは文法違反で、そうは記述できませんが、d=dummy(data[.,2],3|4|5|6|7);と一括して書くことは可能です。もう少しわかりやすい単純な例で説明すると、例えばｘ１＝{3,3,4,8,5,7,6}という１列のベクトルがあるとします。これにv={3,4,5,6,7}のダミー変数の区切りで対応するならば、文法的には、

v={3,4,5,6,7};　d=dummy(x1,v);

と書きます。その結果は、図解すると次のようになります。

基準列	≦3	≦4	≦5	≦6	≦7	7＜
ｘ１	d[.,1]	d[.,2]	d[.,3]	d[.,4]	d[.,5]	d[.,6]
3	1	0	0	0	0	0
3	1	0	0	0	0	0
4	0	1	0	0	0	0
8	0	0	0	0	0	1
5	0	0	1	0	0	0
7	0	0	0	0	1	0
6	0	0	0	1	0	0

d[.,1]＝１　　ｘ１の要素≦３　の場合
d[.,2]＝１　３＜ｘ１の要素≦４　の場合
d[.,3]＝１　４＜ｘ１の要素≦５　の場合
d[.,4]＝１　５＜ｘ１の要素≦６　の場合
d[.,5]＝１　６＜ｘ１の要素≦７　の場合
d[.,6]＝１　７＜ｘ１の要素　　の場合

この場合、気をつけなければならないことは、区切りの数値の数、この場合５個に１を足したものがダミー変数の個数になることです。行列変数dataの２列目は３，４，５，６，７，８のいずれかの数字からなっていますから、v={3,4,5,6,7,8}としたいところですが、そうすると、８より大きい場合も考えられて、最後の列にはすべて０が入ってしまいます。フェンスポール問題をよく考えた上で、区切りの数は設定する必要があると言えます。またregressionをする場合には、切片の１の列ベクトルをなくしたうえで、これらのダミー変数の行列をマージしてやらないといけません。

ポイント　ダミー変数作成方法には２通りあって

１）　循環的な季節ダミーの場合

reshape(eye(**ｎ**),**行数**,**ｎ**)

２）　その他の場合

d=dummy(**基準縦１列ベクトル，区切りの値の入った縦一列ベクトル**)

そのほか、ＧＬＳなどでよく使う技法として、行列のdiagonalの要素だけを置き換えることがあります。その際には、

プログラム

```
new;  cls;
  x={1 2 3 4,
     5 6 7 8,
     9 10 11 12,
    13 14 15 16};
  v=ones(4,1);
  x1=diagrv(x,v);
  d=diag(x);
  print /rz x; print /rz x1; print /rz d;
```

画面結果

```
    1     2     3     4
    5     6     7     8
    9    10    11    12
   13    14    15    16

    1     2     3     4
    5     1     7     8
    9    10     1    12
   13    14    15     1

    1
    6
   11
   16
```

diagrv は diag の reverse のことをやるという意味で rv と略してつけられています。組込

み関数 diag は､その引数に入る正方行列の diagonal を列ベクトルとして取り出して表示するもので、diagrv はその反対に、列ベクトルと行列の diagonal とを置き換えるものです。diagrv の引数の第１要素には対象となる正方行列が、第２要素には置き換えられるべき列ベクトルが入ります。上の例では、ｘという行列の diagonal を ones(4,1)でできたｖという列行列に置き換えるものです。その後に、diag でもともとの行列ｘの diagonal を列ベクトルとして取り出して表示させています。見やすいように、/rz オプションによって、小数点以下の０を省略して表示させています。なお、diagrv のなかは x1=diagrv(x,ones(4,1));とは一括して書けますが、x1=diagrv(x,{1,1,1,1});とは（　）の中に{　}が入るという文法違反のため、そうは書けません。

ポイント　diag(正方行列)　　diagonal の要素を取り出して縦１列のベクトルにする
ポイント　diagrv(正方行列,縦ベクトル)　縦ベクトルの要素を正方行列の diagonal の要素と置き換える

GAUSS では、以前にも取り上げましたように、行列変数のなかにプログラム上から数値を手で入れるのには、ｘ＝{1,2,3,4,5};という形がとられました。let x={1,2,3,4,5};と let をつけて書くことも可能ですが、通常 let は省略されます。それは、行列の要素を表す{ }とカンマがついていてどこからどこまでが行かがはっきりしているからです。一方、行列のディメンションを指定した上で{　}なしに数値を順に代入していく場合には、let は必須となります。

プログラム

```
new;  cls;
let x[3,2]=1 2 3 4 5 6;
print x;
```

プログラム

```
new;  cls;
let x[3,2]=1,2,3,4,5,6;
print x;
```

画面表示

```
1.0000000      2.0000000
3.0000000      4.0000000
5.0000000      6.0000000
```

数値が入る行列変数に[行数,列数]でディメンション指定してやれば、カンマがついていようとなかろうと数値は行列変数のディメンションに順に代入されていきます。上の２つのプログラムは同一の動作をします。ただし、letはプログラム上での数値代入、loadは外部

ファイルの行列変数への代入で、両者は全く違いますので注意が必要です。

そのほか、入力を簡便にするため、GAUSSでは行列のlower triangularの部分に列ベクトルの値を展開する組込み関数xpndが備え付けられています。これはexpandの略で、引数に１列の縦ベクトルを代入して、そのそれぞれの値を行列のlower triangularの部分に展開して、対称な正方行列を生成するものです。この機能は、variance-covariance分析や、それを利用したdirected graph studyにおいて入力の手間を省いてくれる便利な機能です。その逆に、行列のlower triangularの部分を１列にベクトル化する組込み関数としてはvechというものがあります。これはvectorizeの発音からの略でvechとなっています。こちらは引数に正方行列が入り、その戻り値は縦１列ベクトルになります。下の例をみてください。

プログラム

```
new;  cls;
v={1,2,3,4,5,6,7,8,9,10};
x=xpnd(v);
print /rz x;
u=vech(x);
print /rz u;
```

画面表示

```
     1          2          4          7
     2          3          5          8
     4          5          6          9
     7          8          9         10


     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```

１から１０までの数列からなる縦１列のベクトルをxpndによって、行列の下半分の三角行列に展開して、その対称な正方行列ｘを生成して、画面表示しています。見やすいように/rz

オプションで小数点以下の０を省略して見せています。その行列ｘの下半分の三角行列をvechによってベクトル化して縦一列にしてｕとしてそれを同様に画面表示させています。

ポイント　xpnd(**縦１列ベクトル**)　　**下方三角行列に展開して対称な正方行列を作成**
ポイント　vech(**正方行列**)　　**下方三角行列を縦１列ベクトルにベクトル化**