3.9 プログラミング 線形制約とテスト ver.0.1

　ここでは、推定がリニアの場合の「行列としての」制約のかけ方とＦ分布にもとづくテストについて説明します。トピックスとしてはよく使われるきわめて初級のことなのですが、行列の形で制約をかけるとなるとこれが少し厄介なことになります。（なお、この部分は4.7のGPE2パッケージの扱い方における線形制約のパラメータのつけ方と一部対応しています。）ここでは、さらから行列操作を行なってプログラムする方法を取り上げます。この章の行列として制約の形とＦ分布にもとづくテストの仕方をおさらいすることによって行列操作としてのGAUSSが、より身近に感じられるようになると思います。

Regression Fの意味とR2との関係

　OLSをすると統計計量分析でよく表示される”Ｆ”という統計量についての説明です。まず決定係数Ｒ2というのは、「ｙの全変動のうちどれだけモデルを説明できるか」で、すなわちy'yに対するy'yーe'eの割合で表せます。この場合の「モデル」とは、定数項を含むすべての係数を用いた推定式を意味します。ここで、定数項を除く、すべての係数を同時に０としてなくしてしまいましょう。その場合、定数項の推定はｙの平均値になります。これを制限したモデルとしましょう。このときのResidual Sum of Squaresはy'yそのものの全変動になります。もともとの制限がない場合のResidual Sum of Squaresはe'eです。したがって、<u>Kを定数項も含んだ係数の数とすると</u>、制約式の数はＫ－１本ですから、下のようなリニアな制約があるときのＦ値になるわけです（Ｒは制限、Ｕは無制限を表す）。

$$F=\frac{(RSS_R-RSS_U)/(K-1)}{RSS_U/(N-K)}$$

(N－K)は制限がないときの自由度です。それが、(Ｋ‐1)個の制限を加えることによって、Residual Sum of Squaresがその分だけ大きくなりますから、その増分のもともとのRSSに対する割合をＦ値として、下の帰無仮説を判断します。

　　H0: 定数項の除くすべての係数が０

　　H1: H0 is NOT true.

Ｆ(K-1,N-K)の適切なカットオフポイントよりも右のテイルにくればH0は棄却され、モデルが全く説明力がないということは棄却されます。反対に、Ｆ値が小さく、カットオフポイントよりも左にくれば、H0は棄却されなくて、このモデルは全く説明力はないということになります。上の基本式に、y'yとe'eを代入して、分子分母をy'yで割っていやると、

$$F=\frac{(y'y-e'e)/(K-1)}{e'e/(N-K)}=\frac{\dfrac{y'y-e'e}{y'y}/(K-1)}{\dfrac{e'e}{y'y}/(N-K)}=\frac{R2/(K-1)}{(1-R2)/(N-K)}$$

というふうなもともとのＲ2と自由度と制限式の本数だけの数になります。これがＦとＲ2

の関係式になります。すなわち、R2が大きくなれば（分子分母をR2で割って分析すれば）Ｆは大きくなり、R2が小さくなればＦも小さくなります。同じR2の値でも、Nが大きくなれば（データの数が多くなれば）Ｆは大きくなりまし、Kが大きくなれば（説明変数の数が大きくなれば）Ｆは小さくなります。そういうわけで、R2以外にもＦ値の値がOLS計算には通常表示されるわけです。したがって、Regression Fは説明変数の係数がすべて０であるときのリニアの係数制約を表していて、Ｆ値の式を簡単化するとR2から簡単に値が求まるわけです。

線形制約とＦ分布にもとづくテスト

いま、Ｄドライブにdatafile10.txtという30×３のデータファイルがあるとします(Judge[1988]Ch.12より)。１列目はL、２列目はK、最終３列目にQとします。モデルは、下のような形の２次形式までのトランスログ関数として、

$$\ln Q=\alpha+\beta_1 \ln L+\beta_2 \ln K+0.5\beta_3(\ln L)^2+\beta_4(\ln L)(\ln K)+0.5\beta_5(\ln K)^2$$

としてOLSをやってみます。OLSをやるのには、proc(n)で自作したlse(y,x)を使うことにします。その際Regression Fを求めるのにR2とこのFとの上述の関係式を使うことにします。（ただし、Multicollinearity問題は無視して、直接この関係式を推定することにします。）

プログラム

```
new; cls;
load data[30,3]=d:datafile10.txt;
l=data[.,1];k=data[.,2];q=data[.,3];
y=ln(q);x=ones(30,1)~ln(l)~ln(k)~(0.5*(ln(l)^2))~(ln(l).*ln(k))~(0.5*(ln(k)^2));

{b,se,t,shat,df,r2}=lse(y,x);
print "      Estimate          Standard Error             t-value";
print "--------------------------------------------------------------------";;
print b~se~t;
print " Std of est=" shat;
print /rz "                df=" df;
print /rz "                R2=" r2;
f= (r2/5) / ((1-r2)/df);
print "       F(5,24)=" f; print "       P(forF)=" cdffc(f,5,24);

proc(6)=lse(y,x);
    local b,uhat,n,k,df,s2hat,shat,varb,se,t,r2;
```

```
        b=inv(x'x)*x'y;
        uhat=y-x*b;
        n=rows(x);
        k=cols(x);
        df=n-k;
        s2hat=uhat'uhat/df;
        shat=sqrt(s2hat);
        varb=s2hat*inv(x'x);
        se=diag(sqrt(varb));
        t=b./se;
        r2=1-uhat'uhat/(y'(eye(n)-1/n*ones(n,1)*ones(n,1)')*y);
        retp(b,se,t,shat,df,r2);
    endp;
```

画面表示

```
    Estimate             Standard Error                  t-value
----------------------------------------------------------------
     0.33304                   0.15649                    2.1281
     0.73245                   0.13607                    5.3827
     0.90302                   0.17749                    5.0876
    -0.14123                  0.044451                   -3.1773
     0.32974                  0.085925                    3.8375
   -0.083760                  0.067089                   -1.2485
 Std of est=     0.28190
        df=           24
        R2=      0.96888
   F(5,24)=       149.45
   P(forF)= 2.8617e-017
```

上のプログラムでは、proc(n)の章のlseの計算とほぼ同じで、データを読み込んで加工する部分と、ｂとseとｔを水平方向にマージしてからラベルと横線とともに表示するように便宜上しているだけです。そのプログラムに、上の太字の部分のように、f= (r2/5) / ((1-r2)/df);の部分がR2を用いたF値の求め方を示しています。なおこの場合の帰無仮説は

H0: β1＝0かつβ2＝0かつβ3＝0かつβ4＝0かつβ5＝0

H1: H0 is NOT true.

で、この場合５本の制約式と24の自由度（すなわちX行列の列数）ですから、F(5,24)のＦ

分布にしたがうことになります。この場合のＦ値は149.45。またそれに対するＰ値はほぼ０ですから、帰無仮説H0は棄却されます。ここでＦ分布に関してＰ値を求めるにはcdffcという組込み関数を使います。第１要素にＦ値、第２、第３要素に自由度が入ります。

ポイント　cdffc(Ｆ値,DFa ,DFb)　　F(DFa,DFb)に従うＦ分布のそのＦ値の点のＰ値

なお、このRegression Fの値は、制限したRSS、無制限のRSSの両者の計算を直接代入することによっても求められることは冒頭に述べたとおりです。

　同様にして、このトランスログ関数の係数のうち２次に関する部分がすべて０であるとき、すなわち、β3＝0かつβ4＝0かつβ5＝0であるときのリニア制約を加えたときのテストをやってみましょう。これは、コブダグラス型とトランスログ型の判別をするテストでもあります。２つのOLSを実行してResidual Sum of Squaresを求めて、test statisticのＦ値を見る方法もありますが、ここでは、これを一般形で「行列の形で」解くことをやってみます。どんなリニア制約にも対応できるようになりますが少々厄介な計算になります。
この場合の制約は、

　　β3=0　かつ　β4＝0　かつ　β5＝0

ですから、Ｒβ＝ｑ方式で、<u>定数項も含んだ形で</u>、

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$3\times 6 \qquad 6\times 1 \quad 3\times 1$$

と書くことができます。この場合の係数パラメータb_rを求める方法は、この制約式のもとで、RSS(別名SSE)を最小化することで求まります。すなわち、

$$\text{MIN } (Y-Xb_r)'(Y-Xb_r)$$
$$\text{s.t.} \quad Rb_r-q=0$$

この場合のLagrangeanは、計算しやすいように後半を２倍して、

$$L=(Y-Xb_r)'(Y-Xb_r)-2\lambda'(Rb_r-q)$$

これを、b_rとλについて計算します。L_{br}はすべてK×１のタームであることに注意して

$$L_{br}=-2X'Y+2X'Xb_r-2R'\lambda=0$$

$L_\lambda = -2(Rb_r - q) = 0$

上の２つを解くと、最終的に、もともとの制限なしのOLSのβと制限ありのOLSのb_rの関係が次のようになります。

$$b_r = \beta + (X'X)^{-1}R'[R(X'X)^{-1}R']^{-1}(r - R\beta)$$

なお、b_rは定数項も含む係数パラメータの列ベクトル(K×１)。この制限がある場合のb_rによって、Residual Sum of Squaresを計算しなおして、test statisticである

$$F = \frac{(RSS_R - RSS_U)/(K-1)}{RSS_U/(N-K)}$$

を計算するわけです。

プログラム

```
new; cls;
load data[30,3]=d:datafile10.txt;
l=data[.,1];k=data[.,2];q=data[.,3];
y=ln(q);x=ones(30,1)~ln(l)~ln(k)~(0.5*(ln(l)^2))~(ln(l).*ln(k))~(0.5*(ln(k)^2));

b=inv(x'x)*x'y;
rssu=(y-x*b)'(y-x*b);
df=rows(x)-cols(x);

R={ 0 0 0 1 0 0,
    0 0 0 0 1 0,
    0 0 0 0 0 1};
q={0,
   0,
   0};
br=b+inv(x'x)*R'inv(R*inv(x'x)*R')*(q-R*b);
rssr=(y-x*br)'(y-x*br);

f= ((rssr-rssu)/3) / (rssu/df);
p=cdffc(f,rows(q),df);
print "F(3,24)=" f;
print "P       =" p;
```

画面表示

F(3,24)=　　　12.995
P　　　= 3.0389e-005

制限式３本、もともとの無制限のOLSの自由度が24ですから、test statisticはF(3,24)にしたがうＦ値になります。ここで、上の場合の帰無仮説は、

　H0：β3=0　かつ　β4＝0　かつ　β5＝0　（すなわちコブダグラス型）
　H1：H0 is NOT true.

でから、F(3,24)はおおよそ13、そのＰ値もほぼ０でＦ分布のかなり右のテイルにきますから、帰無仮説は棄却されます。すなわち、この場合、コブダグラス型であることは棄却されて、二者択一であれば、弱い意味でトランスログ型が選択されることになります。
　なお、アルゴリズムを一般化して、ｙとｙをＲとｑの行列とともに与えたときに、このＦ値とそれに対するＰ値を返すprocedureを作成すると次のようになります。

```
/*
** Linear Restriction to OLS based on F distribution
** (C) Copyright 1999-2002 Yosuke Amijima. All Rights Reserved.
**
**    PROC RQOLS
**            Restriction:   Rb=q
**    FORMAT
**            { f, p }= rqols(y,x,R,q);
**    INPUT
**            y- dependant variable, Nx1 vector
**            x- independent variables, NxK matrix
**            R- restriction matrix
**            q- restriction vector
**
**    OUTPUT
**            f - test statistic for restriction
**            p - p-value for test statistic f
*/
    proc(2)=rqols(y,x,R,q);
        local b,rssu,rssr,df,br,f,p;
        b=inv(x'x)*x'y;
        rssu=(y-x*b)'(y-x*b);
```

```
    df=rows(x)-cols(x);
    br=b+inv(x'x)*R'inv(R*inv(x'x)*R')*(q-R*b);
    rssr=(y-x*br)'(y-x*br);
    f= ((rssr-rssu)/rows(q)) / (rssu/df);
    p=cdffc(f,rows(q),df);
    retp(f,p);
endp;
```

上のprocedureでは、制限式の本数をrows(q)として、ｑベクトルの行数としています。これは行列Ｒの行数としても同じです。従属変数ｙ、定数項も含む独立変数ｘのほかに、制限をする行列Ｒと列ベクトルｑをインプットとして計算させ、リターンはＦ値とそれに対数Ｐ値の２個としています。なお、これがうまく動くかチェックするには、

プログラム

```
new; cls;
load data[30,3]=d:datafile10.txt;
l=data[.,1];k=data[.,2];q=data[.,3];
y=ln(q);x=ones(30,1)~ln(l)~ln(k)~(0.5*(ln(l)^2))~(ln(l).*ln(k))~(0.5*(ln(k)^2));

R={ 0 0 0 1 0 0,
    0 0 0 0 1 0,
    0 0 0 0 0 1};
q={0,
   0,
   0};
{f,p}=rqols(y,x,R,q);
print "F(3,24)=" f;
print "P       =" p;
```

の後にこのprocedureを置くと、procedureなしで行なった計算と同じ結果を表示します。つまり、proc(2)=rqols(y,x,R,q);からendp;までを、上のプログラムの直後に貼りつけてください。（または、rqols.gとしてsrcフォルダーに保存すれば組込み関数になります。）

　同様にして、今度は同じデータについて、すでにトランスログ関数が成り立つと仮定した上で、そのトランスログ関数が１次同次であるかのテストをしてみます。上の例と同じように、Ｒβ＝ｑの関係にあてはめて、Ｒ行列とｑベクトルを与えて、test statisticのＦ値とそれに対するＰ値を求めます。この場合の帰無仮説は、

H0：β1＋β2＝１かつβ3＋β4＝０かつβ4＋β5＝０　（すなわち１次同次）
H1：H0 is NOT true.

となり、

β1＋β2＝1
β3＋β4＝0
β4＋β5＝0

を満たす制限行列Ｒとベクトルｑは、Rβ＝ｑ方式で、定数項を含んだ形で、

$$\underset{3\times 6}{\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}} \underset{6\times 1}{\begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix}} = \underset{3\times 1}{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}$$　ですから

このＲとｑを作成したprocedureのrqols(y,x,R,q)にyおよびｘとともに代入して計算させます。プログラムは、Ｒとｑの指定以外は同じです。

プログラム

```
new; cls;
load data[30,3]=d:datafile10.txt;
l=data[.,1];k=data[.,2];q=data[.,3];
y=ln(q);x=ones(30,1)~ln(l)~ln(k)~(0.5*(ln(l)^2))~(ln(l).*ln(k))~(0.5*(ln(k)^2));

R={ 0 1 1 0 0 0,
     0 0 0 1 1 0,
     0 0 0 0 1 1};
q={1,
   0,
   0};
{f,p}=rqols(y,x,R,q);
```

```
    print "F(3,24)=" f;
    print "P        =" p;
```

画面表示

```
F(3,24)=        2.1255
P        =      0.12345
```

上のようなプログラムの後にrqolsのprocedure部分を置くと計算してくれて、この場合の結果は、F(3,24)が2.1255で、Ｆ分布のカットオフポイントよりも０側にくることになり、実際、Ｐ値も0.1を超えますから、５%に対しても、10%に対してもこの帰無仮説は棄却されないことになり、一次同次は成立すると言えます。

最後に注意を１つ。このＲとｑによるリニアの制限は、定数項も加えた、一般の行列計算の場合のやり方と同じ正式な方法です。（後の4.7で扱うGPE2上のパラメータ設定では、定数項があっても取り除いた形のRとｑを考えることになります。）こうした行列による設定、行列によるプログラミングがGAUSSのGAUSSたる由縁です。こうした行列による設定は、この後の節で扱う制約条件を考える諸ライブラリでも必要となってくる概念です。