3.12 プログラミング モデル選択とテスト　　ver.0.1

Nestedのモデル選択のテストは、線形制約のテストのところで扱った同じように、F分布にしたがうテストを行なって、もともとのすべてを包含するモデルが受け入れられるのか、それとも係数のいくつかを同時に0と置いた時の制限モデルを受け入れるのかを判断しました。その場合、一方が棄却されれば、もう一方は必ず受容されるというものででした。 それに対して、ここでは、Non-Nestedのテストを中心に、両者棄却または両者受容ということをも想定したリニアモデルのテストを行ないます。

MacKinnon-White-Davidson Test

前回の不均一分散への対処法として、両辺のログをとるモデルを使うというのも考えられます。しかし、ここでは分散均一という視点からではなく、モデル選択の視点から、もとのリニアのモデルとログログのモデルの２つをNon-NestedのMWD Testをしてみます。全章と同様に、Dドライブにdatafile12.txtがあるものとします。なお、手順は、第１回目のregression

$Y=\beta_1+\beta_2 X_2+\beta_3 X_3+ \ \cdots \ +\beta_k x_k$

$lnY=\gamma_1+\gamma_2 ln X_2+\gamma_3 ln X_3+ \ \cdots \ +\gamma_k lnx_k$

から、推定値YhatとlnYhatを求める。

Z1=ln(Yhat)－lnyhat および

Z2=exp(lnYhat)－Yhat

を作成し、これらを説明変数に加えたregression

$Y=\beta_1+\beta_2 X_2+\beta_3 X_3+ \ \cdots \ +\beta_k x_k+\beta_{k+1} Z1$

$lnY=\gamma_1+\gamma_2 ln X_2+\gamma_3 ln X_3+ \ \cdots \ +\gamma_k lnx_k+\beta_{k+1} Z2$

をもう一度行ない、加えた変数に対する係数が有意であるかないかをｔテストする。

プログラム

```
new; cls;
load data[31,2]=d:datafile12.txt;
y=data[.,1]; x=data[.,2];
x_1=lagn(x,1); x_2=lagn(x,2); y_1=lagn(y,1);
x=ones(rows(data),1)~x_1~x_2~y_1;
y=y[3:31,.]; x=x[3:31,.];
call mwdtest(y,x,0.05);

proc(0)=mwdtest(y,x,pp);
    local n,k,b,yhat,lny,lnx,bln,lnyhat,z1,z2,x1,x2,b1,b2,e1,e2,
```

```
s2hat1,s2hat2,varb1,varb2,se1,se2,t1,t2,p1,p2;
n=rows(x); k=cols(x);
b=inv(x'x)*x'y;
yhat=x*b;
if x[.,1]/=ones(n,1);
    errorlog "ERROR: 1st column of X should be constant ones.";
    retp;
endif;
lny=ln(y); lnx=ones(n,1)~ln(x[.,2:k]);
bln=inv(lnx'lnx)*lnx'lny;
lnyhat=lnx*bln;
z1=ln(yhat)-lnyhat; z2=exp(lnyhat)-yhat;

x1=x~z1;
b1=inv(x1'x1)*x1'y;
e1=y-x1*b1;
s2hat1=e1'e1/(n-k-1);
varb1=s2hat1*inv(x1'x1);
se1=diag(sqrt(varb1));
t1=b1./se1;
p1=2*cdftc(abs(t1),n-k-1);
print "Linear Model";
if p1[k+1]>=pp;
    print "MWD Test (RESULT:Not reject H0)";
else;
    print "MWD Test (RESULT:Reject H0)";
endif;
print "t for Z1:" t1[k+1]; print "           P:" p1[k+1];
print;
x2=lnx~z2;
b2=inv(x2'x2)*x2'lny;
e2=lny-x2*b2;
s2hat2=e2'e2/(n-k-1);
varb2=s2hat2*inv(x2'x2);
se2=diag(sqrt(varb2));
t2=b2./se2;
```

```
        p2=2*cdftc(abs(t2),n-k-1);
        print "Log-Log Model";
        if p2[k+1]>=pp;
            print "MWD Test (RESULT:Not reject H0)";
        else;
            print "MWD Test (RESULT:Reject H0)";
        endif;
        print "t for Z2:" t2[k+1]; print "          P:" p2[k+1];
        retp;
    endp;
```

画面表示

```
Linear Model
MWD Test (RESULT:Not reject H0)
t for Z1:         -1.9154421
        P:        0.067425415

Log-Log Model
MWD Test (RESULT:Not reject H0)
t for Z2:          1.2685958
        P:         0.21675239
```

上のように、両者ともに、加えられたＺ変数はそれぞれ棄却されないので、どちらのモデルも選ばれる結果となります。このテストで、もし両者が棄却されれば、どちらのモデルも説明能力がないと言えます。反対に、この場合のように、両者が棄却されないのであれば、データが２つのモデルを区別するに足りるほど量的質的に豊富ではないと言えます。このように、Non-Nestedのテストは従来のＦテストなど（係数のリニア制約のケースなど）によるNestedの２者選択のモデル選択ではなくて、両者棄却または両者受容をも想定したテストです。この場合、そもそものデータ数をもっと増やす、または、このデータが何かのサブサンプルであればそのサンプリングの方法を変更することが必要になるでしょう。

プログラムでは、その呼び出し部分は全章とまったく同じです。そのあと、両辺の変数を自然対数変換する際に（計量経済学では通常ログ変換とは、たとえLOGと書いていても自然対数変換のことをさすので注意のこと）、ログをとった側の独立変数側の行列には（自然対数の１ではなくて）１の列を加えなければなりませんから、errorlogの分岐であらかじめもとのＸ行列の１行目に１の列がきているのかをチェックしています。もし１の列が来ていなければ、errorlogを表示してretp;でprocedureから何も返さずに出ます。そのあと、

通常通り２つのモデルの推定を行ない、それぞれの従属変数の推定値を求めます。そして

```
z1=ln(yhat)-lnyhat; z2=exp(lnyhat)-yhat;
```

というふうに、それら推定値yhatとlnyhatにもとづいて、Ｚ１とＺ２の変数を作ります。このＺ１をリニアのモデルに、Ｚ２をログログモデルに、新たな独立変数として追加してそれぞれのモデルをもう一度推定しなおしています。この場合の独立変数行列の列数はｋではなくてｋ＋１になります。したがって、自由度はｎ－ｋではなくｎ－ｋ－１となります。そして最終的に、定数項を含んでｋ＋１番目の変数のｔ値を求めて、その絶対値を

```
p=2*cdftc(abs(t),n-k-1);
```

に自由度とともに代入して、２をかけたものが最終的にＰ値になります。これが当初設定した５％（すなわち0.05）以上であれば、加えられた変数の係数を０とする帰無仮説は棄却させなくて、このモデルは受容されます。反対に、それよりも小さければ、帰無仮説を棄却して、このモデルは棄却されます。これをそれぞれのモデルに対して行なっています。

ちなみに、AICとSchwarz Criterionを求めたいのならば、ｎとｋとRSSの３つがわかれば簡単に求められますから（ｋは定数項も含んだ数)、

```
rss=(y-x*b)'(y-x*b);
aic=ln(rss/n)+2*k/n;     print "AIC=" aic;
sc=ln(rss/n)+k*ln(n)/n;     print " SC=" sc;
```

のようにしてやって、３つの変数を追加でローカル宣言してやって、プログラムの適切な場所におけば、リニアの場合の２つの情報量基準を計算表示できます。より小さい値をもたらすモデルを選択します。ただし、上の場合には、もう１つのモデルはログログモデルですから、この場合のRSSはリニアモデルより小さくなり、しかも上の式での２つの基準の値はマイナスになることが多いですからこの方法では比較することはできません。

<u>Ramsey RESET Test</u>

このテストは、真のモデルにさらに含まれるべき説明変数が既知ではない場合に、その代理変数として、もともとのモデルの従属変数の推定値の２乗項、３乗項、４乗項をとってやって、それらの係数が同時に０になる場合をH0、そうでない場合もH1として、線形制約と同じようなＦ分布にもとづくテストを行なうものです。

$Y=\beta_1+\beta_2 X_2+\beta_3 X_3+ \ldots +\beta_k x_k+\beta_{k+1}Yhat^2+\beta_{k+2}Yhat^3+\beta_{k+3}Yhat^4$

$H0: \beta_{k+1}=\beta_{k+2}=\beta_{k+3}=0$

H1: H0 is NOT true.

プログラム

```
new; cls;
load data[31,2]=d:datafile12.txt;
y=data[.,1]; x=data[.,2];
x_1=lagn(x,1); x_2=lagn(x,2); y_1=lagn(y,1);
```

```
x=ones(rows(data),1)~x_1~x_2~y_1;
y=y[3:31,.]; x=x[3:31,.];
call rreset(y,x,0.05);

proc(0)=rreset(y,x,pp);
    local n,k,b,rssr,yhat,yhat2,yhat3,yhat4,z,bz,rssu,f,p;
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    rssr=(y-x*b)'(y-x*b);
    yhat=x*b; yhat2=yhat^2; yhat3=yhat^3; yhat4=yhat^4;
    z=x~yhat2~yhat3~yhat4;
    bz=inv(z'z)*z'y;
    rssu=(y-z*bz)'(y-z*bz);
    f=((rssr-rssu)/3) / (rssu/(n-k-3));
    p=cdffc(f,3,n-k-3);
    if p>=pp;
        print "Ramsey RESET Test (RESULT:Not reject H0)";
    else;
        print "Ramsey RESET TEST (RESULT:Reject H0)";
    endif;
    print "F=" f ; print "P=" p;
    retp;
endp;
```

画面表示

```
Ramsey RESET TEST (RESULT:Reject H0)
F=          4.3234760
P=       0.015351173
```

プログラム的には、いままでのものとほとんど同じです。ただし、自由度がｎ－ｋ－３であることに注意してください。線形制約の場合と同様に、モデルのRSSをRSSuとし、２乗３乗４乗項の係数を同時に０とするときの制限モデルのRSSをRSSrとすると、

$$F = \frac{(RSSr - RSSu)/3}{RSSu(n-k-3)} \sim F(3, n-k-3)$$

にしたがう分布によりテストを行なっています。ここでは、結果は５％の有意水準でH0が棄却されますから、このリニアモデルにはさらに何か説明変数を加えられるべきです。

J Test

従属変数ｙが共通であって、説明変数が異なる（重なりがあってもよいがデータの行数は同じである必要がある）場合に、どちらを選択するか、あるいは両者選択・両者棄却するのかをテストする代表的なテストがこのＪテストである。一度回帰を行なった後、

$Yx=\beta_1+\beta_2 x_2+\beta_3 x_3+ \ldots +\beta_k x_k+\beta_{k+1}YHATz$

$Yz=\gamma_1+\gamma_2 z_2+\gamma_3 z_3+ \ldots +\gamma_k z_k+\gamma_{k+1}YHATx$

というふうにYHATzとYHATxをそれぞれ加えたモデルを推定して、定数項を含めてｋ＋1番目の係数が０であるかｔテストするものです。

プログラム

```
new; cls;
load data[50,5]=d:datafile11.txt;
y=data[.,1]; x=data[.,2:3]; z=data[.,4:5];
x=ones(rows(data),1)~x; z=ones(rows(data),1)~z;
call jtest(y,x,z,0.05);

proc(0)=jtest(y,x,z,pp);
    local n,bx,bz,yhatx,yhatz,k1,k2,b1,b2,e1,e2,
          s2hat1,s2hat2,varb1,varb2,se1,se2,t1,t2,p1,p2;
    if rows(x)/=rows(z);
        errorlog "ERROR: Each row does not match.";
        retp;
    endif;
    n=rows(x);
    bx=inv(x'x)*x'y; yhatx=x*bx;
    bz=inv(z'z)*z'y; yhatz=z*bz;
    x=x~yhatz; z=z~yhatx;
    k1=cols(x); k2=cols(z);
    /* Equation 1 */
    b1=inv(x'x)*x'y;
    e1=y-x*b1;
    s2hat1=e1'e1/(n-k1);
    varb1=s2hat1*inv(x'x);
    se1=diag(sqrt(varb1));
    t1=b1./se1;
    p1=2*cdftc(abs(t1),n-k1);
```

```
        print "Equation 1:";
        if p1[k1]>=pp;
        print "J Test (RESULT:Not reject H0)";
            else;
        print "J Test (RESULT:Reject H0)";
            endif;
        print "t=" t1[k1]; print "P=" p1[k1];
        /* Equation2 */
        b2=inv(z'z)*z'y;
        e2=y-z*b2;
        s2hat2=e2'e2/(n-k2);
        varb2=s2hat2*inv(z'z);
        se2=diag(sqrt(varb2));
        t2=b2./se2;
        p2=2*cdftc(abs(t2),n-k2);
        print; print "Equation 2:";
        if p2[k2]>=pp;
            print "J Test (RESULT:Not reject H0)";
        else;
            print "J Test (RESULT:Reject H0)";
        endif;
        print "t=" t2[k2]; print "P=" p2[k2];
        retp;
    endp;
```

画面表示

```
Equation 1:
J Test (RESULT:Reject H0)
t=          11.832589
P=   1.4826926e-015

Equation 2:
J Test (RESULT:Reject H0)
t=          16.270143
P=   1.0482163e-020
```

上の結果は、Equation 1とEquation 2ともに付け加えた説明変数の係数が０であることがかなりの有意性をもって棄却されます。なぜなら、ここで使われているデータは構造変化のテストで用いたもので、すべての説明変数を使うのではなくて、その前半の２列目と３列目を定数項つきで第１モデルとし、その後半の４列目と５列目を定数項つきでモデル２としているからです。両モデルともに棄却されるということは明らかです。プログラム的には、冒頭のMWD Testを簡単にしたようなアルゴリズムになって、もう一方の１回目のregressionから得られたYHATを加えた従属変数の最終列の係数をｔテストしています。

JA Test

このテストは、Ｊテストの変形バージョンで、Ｊテストにおいて付け加えるべきYHATの部分の計算を、従来までの従属変数Ｙそのものではなくて、alternativeのモデルの従属変数の推定値を用います。いわば、都合３回のregressionをそれぞれ行なっていると考えてください。原理的には、Ｊテストと同じです。

プログラム

```
new; cls;
load data[50,5]=d:datafile11.txt;
y=data[.,1]; x=data[.,2:3]; z=data[.,4:5];
x=ones(rows(data),1)~x; z=ones(rows(data),1)~z;
call jatest(y,x,z,0.05);

proc(0)=jatest(y,x,z,pp);
    local n,bx,bz,yhatx,yhatz,k1,k2,b1,b2,e1,e2,
          s2hat1,s2hat2,varb1,varb2,se1,se2,t1,t2,p1,p2,b,bb;
    if rows(x)/=rows(z);
        errorlog "ERROR: Each row does not match.";
        retp;
    endif;
    n=rows(x);
    b=inv(x'x)*x'y; bb=inv(z'z)*z'y;
    bx=inv(x'x)*x'(z*bb); yhatx=x*bx;
    bz=inv(z'z)*z'(x*b); yhatz=z*bz;
    x=x~yhatz; z=z~yhatx;
    k1=cols(x); k2=cols(z);
    /* Equation 1 */
    b1=inv(x'x)*x'y;
```

```
        e1=y-x*b1;
        s2hat1=e1'e1/(n-k1);
        varb1=s2hat1*inv(x'x);
        se1=diag(sqrt(varb1));
        t1=b1./se1;
        p1=2*cdftc(abs(t1),n-k1);
        print "Equation 1:";
        if p1[k1]>=pp;
            print "JA Test (RESULT:Not reject H0)";
        else;
            print "JA Test (RESULT:Reject H0)";
        endif;
        print "t=" t1[k1]; print "P=" p1[k1];
        /* Equation2 */
        b2=inv(z'z)*z'y;
        e2=y-z*b2;
        s2hat2=e2'e2/(n-k2);
        varb2=s2hat2*inv(z'z);
        se2=diag(sqrt(varb2));
        t2=b2./se2;
        p2=2*cdftc(abs(t2),n-k2);
        print; print "Equation 2:";
        if p2[k2]>=pp;
            print "JA Test (RESULT:Not reject H0)";
        else;
            print "JA Test (RESULT:Reject H0)";
        endif;
        print "t=" t2[k2]; print "P=" p2[k2];
        retp;
    endp;
```

画面表示

```
Equation 1:
JA Test (RESULT:Reject H0)
t=          10.277742
P=   1.6907622e-013
```

Equation 2:
JA Test (RESULT:Reject H0)
t= 9.3096991
P= 3.7634906e-012

上の結果も、Ｊテストと同様に、両者棄却となっています。プログラム的には、太字の部分の計算にＹそのものではなくて、alternativeの推定値が用いられているところ以外は、この前のＪテストとまったく同じです。

COX Test

今度のテストは漸近性を用いて、サンプルサイズｎに依存しない統計量をテストするものです。テスト統計量は$s^2x=(y-x\beta_x)'(y-x\beta_x)/n$　および　$s^2z=(y-z\beta_z)'(y-z\beta_z)/n$　をもとに

$M_x=I_n-x*inv(x'x)*x'$　と　$M_z=I_n-z*inv(z'z)*z'$

した上で、$s^2xz= s^2x+\beta_x'x'M_zx\beta_x/n$　を計算して、Cox統計量

$$q=\frac{\frac{n}{2}\ln\frac{s_z^2}{s_{xz}^2}}{\left[\frac{s_x^2}{(s_{xz}^2)^2}\beta_x'X'M_zM_xM_zX\beta_x\right]^{1/2}}\sim N(0,1)$$

を計算して、正規分布にしたがうテストをするものです。途中、XとZをひっくり返して同じ統計量を計算させ同様のテストを行なっています。

プログラム

```
/*
** Cox Test for Model Selection between Two Linear Models.
** (C) Copyright 1999 Yosuke Amijima. All Rights Reserved.
**
**    PROC COXTEST
**
**    FORMAT
**          call coxtest(y, X, Z, pp)
**    INPUT
**          y - Nx1 vector, dependent variable.
**          X - NxK matrix, independent variables including constant term.
**          Z - NxK matrix, independent variables including constant term.
```

```
**          pp – percentage point for the test significance
**
**  MODEL
**          MODEL 1: y=Xb_x+ ε_1          MODEL 2: y=Zb_z+ ε_2
**                      ( Both X and Z contain constant term vector if any. )
*/
    new; cls;
    load data[50,5]=d:datafile11.txt;
    y=data[.,1]; x=data[.,2:3]; z=data[.,4:5];
    x=ones(rows(data),1)~x; z=ones(rows(data),1)~z;
    call coxtest(y,x,z,0.05);


    proc(0)=coxtest(y,x,z,pp);
        local n,bx,bz,Mx,Mz,s2x,s2z,s2xz,q,p,temp;
        n=rows(y);
        /* Equation 1 */
        bx=inv(x'x)*x'y; Mx=eye(n)-x*inv(x'x)*x'; s2x=(y-x*bx)'(y-x*bx)/n;
        bz=inv(z'z)*z'y; Mz=eye(n)-z*inv(z'z)*z'; s2z=(y-z*bz)'(y-z*bz)/n;
        s2xz=s2x+bx'x'Mz*x*bx/n;
        q=(n/2)*ln(s2z/s2xz)/sqrt((s2x/s2xz^2)*bx'x'Mz*Mx*Mz*x*bx);
        p=2*cdfnc(abs(q));
        print "Equation 1:";
        if p>=pp;
            print "COX Test (RESULT:NOT reject H0)";
        else;
            print "COX Test (RESULT:Reject H0)";
        endif;
        print "q=" q; print;
        /* Equation 2 */
        temp=x; x=z; z=temp;
        bx=inv(x'x)*x'y; Mx=eye(n)-x*inv(x'x)*x'; s2x=(y-x*bx)'(y-x*bx)/n;
        bz=inv(z'z)*z'y; Mz=eye(n)-z*inv(z'z)*z'; s2z=(y-z*bz)'(y-z*bz)/n;
        s2xz=s2x+bx'x'Mz*x*bx/n;
        q=(n/2)*ln(s2z/s2xz)/sqrt((s2x/s2xz^2)*bx'x'Mz*Mx*Mz*x*bx);
        p=2*cdfnc(abs(q));
```

```
        print "Equation 2:";
        if p>=pp;
            print "COX Test (RESULT:NOT reject H0)";
        else;
            print "COX Test (RESULT:Reject H0)";
        endif;
        print "q=" q;
    endp;
```

画面表示

```
Equation 1:
COX Test (RESULT:Reject H0)
q=        -22.624222

Equation 2:
COX Test (RESULT:Reject H0)
q=        -42.155456
```

両者ともに、かなりの有意性をもって棄却されていることは、ほかのテストと同じ結果となっています。判断分岐には、p=2*cdfnc(abs(q))として、Cox統計量の絶対値をN(0,1)の分布におけるPercentage PointのComplimentを計算してくれる組込み関数cdfncで計算させています。分布はN(0,1)ですから、自由度はありません。なお、このテストはデータ数の少ない場合には問題があるのは言うまでもありません。この場合には、上であげたほかのテストの方が信頼性があると思われます。