3.11 プログラミング 不均一分散とテスト　　ver.0.2

ここでは、非常に基本的な不均一分散の諸テストを扱います。統計量とテストをGAUSSでどのようにしてプログラムするのかを理解するのに、初歩として絶好の対象になると思われます。まずは、Dドライブにdatafile12.txt(蓑谷[1997]『計量経済学』のデータ)があるものとします。その１列目をｙ、そして２列目をｘとして考えます。

$$y = \beta_1 + \beta_2 X_{-1} + \beta_3 X_{-2} + \beta_4 y_{-1} + \varepsilon$$

をモデルに推定をし、その残差residualsを分析します。なお、ここでは系列相関の問題は無視して、右辺のラグ項は通常の独立変数と同じように考えるものとします。プログラムでは、あらためて従属変数をｙ、定数項の１の列ベクトルも含めて独立変数をまとめてｘの行列とします。

Godlfeld-Quandt Test

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 gqtest(y,x,0.05);      /* The number 0.05 here means 5% significance level. */


proc(2)=gqtest(y,x,pp);
     local n,k,t,y1,x1,y2,x2,b1,e1,s2hat1,b2,e2,s2hat2,f,p;
    n=rows(x); k=cols(x); t=round(n/2);
    y1=y[1:t,.]; x1=x[1:t,.]; y2=y[t+1:n,.]; x2=x[t+1:n,.];
    b1=inv(x1'x1)*x1'y1; e1=y1-x1*b1;
    s2hat1=e1'e1/(t-k);
    b2=inv(x2'x2)*x2'y2; e2=y2-x2*b2;
    s2hat2=e2'e2/(n-t-k);
    f=s2hat2/s2hat1;
    p=cdffc(f,n-t-k,t-k);
    if p>=pp;
```

```
        print "Goldfeld=Quandt Test (RESULT:Not reject H0)";
    else;
        print "Goldfeld=Quandt Test (RESULT:Reject H0)";
    endif;
    print "F(" n-t-k "," t-k ")=" f; print "P=" p;
    retp(f,p);
endp;
```

画面表示

```
Goldfeld=Quandt Test (RESULT:Reject H0)
F(          10.000000 ,          11.000000 )=          3.8001217
P=        0.019232945
```

この場合、時系列データに対して、１期からｔ期までと、ｔ＋１期から最終ｎ期までの２つにデータを分割して、それぞれのResidual Sum of SquaresをRSS1とRSS2として、

$$GQ = \frac{\frac{RSS2}{(n-t)-k}}{\frac{RSS1}{t-k}} \sim F(n-t-k, t-k)$$

となる統計量、すなわち、それぞれのグループの分散の比をとります。ここでは、分子の方が大きくなるようにします。つまり、上の統計量は常に１よりも大きい値になるようにします。（もし、時系列データでRSS1の法が大きければ分母分子を逆転させます。また、非時系列データでは、変数について列ごとにソートをあらかじめ必要とします。）そこで

H0: Homescedasticity
H1: H0 is NOT true. (Heteroscedasticity)

という帰無仮説をテストします。上のプログラムでは、まずlagnというラグをとるオペレータ関数で変数を加工します。例えば、lagn(x,t)であれば、ｘのｔ期ラグ分したデータが作成されます。ただし、この場合、上からｔ期はドットなって欠落値になりますから、ここでは３期目からｎ期までのデータを用います。そして、qptest(y,x,0.05);でそのあとに置く自作のprocedureを呼び出して、ＧＱテストを行なっています。インプットには、ｙとｘのシリーズデータのほかに、基準とする有意水準の小数値ppを与えます。なお、procedureの内部で、アウトプットとは別に、結果を表示させているので、callでとめてリターンがあらためて出力されないようにしています。後半の部分のprocedure内部では、統計量ｆとそれに対するＰ値ｐがリターン値として一応（ほかのプログラムで将来利用できるように）設定しています。したがって、冒頭ではproc(2)としなくてはなりません。ここで、データを

２分割するために、t=round(n/2);を用いて、小数をラウンドしておおよそ等分にしています。ｎが偶数ではなくて、奇数の場合には、前半のデータが１つだけ多くなるようにここでは設定することになります。このｔ期を用いて、データを１期からｔ期までと、ｔ＋１期からｎ期までの２分割して、それぞれのグループのRSSを求めて、その非を自由度でバランスさせた計算をしているのです。第１グループにはb1,y1,e1などの変数を使い、第２グループには、b2,y2,e2などの変数を使って便宜上区別して内容的にはＯＬＳ計算と同じようにして、RSS1とRSS2を計算しています。ここで、Ｘの列数ｋは一定ですが、それぞれのグループのデータの個数は第１グループではｔ個、第２グループではｎ－ｔ個であることに注意してください。上にあげたＧＱ統計量を計算したものをｆとおいて、これを分子分母の自由度とともにp=cdffc(f,n-t-k,t-k)として、ＧＱ統計量ｆに対するＦ分布にもとづくpercentage pointを求めています。これをインプットで与えられた基準有意水準ppとIF文で比較して、以上であれば棄却されないくて、そうではなければ棄却されるように条件分岐させてそれぞれのメッセージをprint文で出しています。最後に、ｆ値とそれに対するＰ値を内部で画面表示させています。そして、リターンとしてｆとｐの２つを返してします。なお、繰り返しますが、これらのリターンはcall文でとめていますので使われてはいません。

このデータとモデルの場合、結果では、F(10,11)=3.8で、そのＰ値は約0.019ですからＦ分布のかなり右端のテイルにきていますから、５％の有意水準でH0は棄却され、５％の有意水準ではHeteroscedasticityがあると認められます。ただし、ここで注意すべきことは、このGoldfeld-Quandt Testは、不均一分散の要因が複数ではなくて、ただ１つであると仮定されている点にあります。しかしながら、データ数が少ない時に力を発揮し、極めて簡便であってとても有用です。

Jarque-Bera Normality Test

要因が１つではなくて、複数疑われる場合に、少なくとも１つの要因があるかどうかテストする前に、あらかじめ正規性のテストをした上で上のＧＱテストとは別のテストをします。正規性のテストは、OLS残差residualsを利用して、skewnessとkurtosisを計算してから、これらの値をもとにしてJarque-Bera Normality Testの

$$JB = \frac{n}{6}\left[Skew^2 + \frac{(Kurtosis - 3)^2}{4} \right] \sim \chi^2(2)$$

となる統計量(残差ベースの計算ではｎのところがｎ-ｋになるケースもある)をもとに、

H0: Residuals are normally distributed.

H1: H0 is NOT true.

というテストをします。

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 jbtest(y,x,0.05);



proc(4)=jbtest(y,x,pp);
    local n,k,b,e,skew,kurtosis,jb,p;
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    skew=meanc(e^3)/(meanc(e^2)^1.5);
    kurtosis=meanc(e^4)/(meanc(e^2)^2);
    print "Skewness of Residuals=" skew;
    print "Kurtosis of Residuals=" kurtosis;
    jb=n/6*(skew^2+(kurtosis-3)^2/4);
    p=cdfchic(jb,2);
    if p>=pp;
        print "Jarque-Bera Normality Test (RESULT:Not reject H0)";
    else;
        print "Jarque-Bera Normality Test (RESULT:Reject H0)";
    endif;
    print "X2( 2 )=" jb ; print "P=" p;
    retp(jb,p,skew,kurtosis);
endp;
```

画面表示

```
Skewness of Residuals=          0.47274221
Kurtosis of Residuals=          2.9377513
Jarque-Bera Normality Test (RESULT:Not reject H0)
```

X2(2)=　　1.0848606
P=　　0.58133371

同じデータを使って、通常のOLSと同じく残差を求めます。上の統計量をjbとおいて計算して、それに対するP値をp=cdfchic(jb,2);としてχ²(2)にしたがう分布から求めています。条件分岐では、このP値が、あらかじめインプットで与えられている有意水準pp以上であれば帰無仮説は棄却されなくて、そうでなければ棄却されるようにし、それぞれのメッセージを出しています。統計量jbとそれに対するP値をprint文でアウトプットとは別に内部で表示しています。最後に、アウトプットして、これらjbとpのほかに、skewとkurtosisの値を返しています。したがって、procedure冒頭はproc(4)になります。

結果は、この場合のχ²(2)の値は１前後と相対的に小さく、そのP値も0.5前後と0.05をはるかにうわまわりますから、帰無仮説は棄却されなくて、データ数が少ないという問題はあるものの正規性は確保されていると判断でき、次のテストに向かうことができます。実際、Kurtosisも３に近い数です。これとは反対に、仮に帰無仮説が棄却されるならば、残差residualsは正規分布にしたがわないと言えますが、その滝には、重要な説明変数の欠落によって、見せかけの非正規性があらわれていることも考えられるので注意が必要です。

Breusch-Pagan Test

複数の要因が不均一分散に影響を及ぼしている場合のテストで、もともとのy=Xβというモデルからσ²ₜを求めた上で、

$$\sigma^2_t = \alpha_1 + \alpha_2 Z_{2t} + \ldots + \alpha_k Z_{kt} \qquad t=1,2,\ldots,n$$

ここで、Ｚは任意、例えばＸ自身がよく使われる。

σ²ₜはe²ₜを自由度ｎの分散s²で割ってstandardizedしたもの。

としたときに、

H0: α2＝ … ＝αk＝0　　Homoscedasticity

H1: H0 is NOT true. （αのうち１つでも０でない）Heteroscedasticity

の帰無仮説のテストをします。この帰無仮説が棄却されれば、「少なくとも１つの要素」が不均一分散に影響を与えていると考えられます。一方、棄却されないのならば、均一分散であると言えます。これをテストする統計量は、残差に正規性が保たれている場合には、

$$BP = \frac{ESS}{2} \sim \chi^2(k-1)$$

を使います。正規性が保たれていない場合には、

$$nR^2 \sim \chi^2(k-1)$$

を使ってテストします。これはBP2 Test（またはGPE2ではKoenkar-Basset TestとしてBPテストをするとこの統計量もあわせて列挙されます）と呼ぶこともあります。

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 bptest(y,x,0.05);



proc(2)= bptest(y,x,pp);
    local n,k,b,e,s2hat,y1,z,b1,e1,r2,chi,p,ess;
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    s2hat=e'e/n;
    y1=e^2/s2hat;
    z=x;            /* You could change this to z=x.*x; or z=x.*x.*x; */
    b1=inv(z'z)*z'y1;
    e1=y1-z*b1;
    r2=1-e1'e1/(y1'(eye(n)-1/n*ones(n,1)*ones(n,1)')*y1);
    ess=(z*b1-meanc(y1))'(z*b1-meanc(y1)); chi=ess/2; /* chi=n*r2; */
    p=cdfchic(chi,k-1);
    if p>=pp;
        print "Breusch-Pagan Test (RESULT:Not reject H0)";
    else;
        print "Breusch-Pagan Test (RESULT:Reject H0)";
    endif;
    print/rz "X2(" k-1 ")=" chi ; print "P=" p;
    retp(chi,p);
endp;
```

画面表示

```
Breusch-Pagan Test (RESULT:Reject H0)
```

X2(3)= 7.9355046

P= 0.047363608

画面表示（正規性が保たれない場合のchi=n*r2;に対する結果）

Breusch=Pagan Test (RESULT:Reject H0)

X2(3)= 8.1904264

P= 0.042235818

プログラムでは、呼び出し方は前の２つのプログラムと同様です。次のProcedureの部分は一度OLSを計算して残差ｅを求めた後で、s2hat=e'e/n;(ここでn-kで割ってはいけません)およびy1=e^2/s2hat;として２回目のテスト統計量を求めるOLSのモデルの従属変数ｙ１を求めます。これに対して、独立変数側には、ＺとしてＸそのものを採用しています。なお、変数Ｘの２乗などを採用する場合もあります。このときのESSの２分の１を統計量にしたものをχ^2のP値と比較して、それ以上であれば棄却されなくて、そうでなければ棄却されるように条件分岐をさせています。内部で、そのχ^2の値と、それに対するP値を画面表示したあとで、リターンとしてそれら２つを返しています。なお、この場合の自由度は、帰無仮説からも明らかなように、定数項を除いたｋ－１となります。BP2のバージョンでは、単にnに統計量R^2の値をかけたものをchiとしてやって、同様な条件分岐と画面表示を行ないます。結果は、２つのバージョンともに、５％の有意水準で帰無仮説は棄却され、少なくとも１つの変数になんらかの影響を受けていると判断できます。なお、Jarque-Bera Test等で正規性が疑われる場合には、後者のバージョンの結果を採用する必要があります。正規性が失われれば失われるほど、後者の統計値は前者のBP統計量よりも下方に乖離していき、均一分散であると結論づけやすくなる傾向があります。反対に、完全な純粋な正規性が保たれるならば、両者の値は理論的には完全に一致することになります。なお、上のケースでは、Jarque-Bera Testですでに正規性が確認されているので、どちらのテスト結果を採用しても、結論がくつがえるほどの乖離は見られません。

Glesjer TestとHarvey-Godfrey Test

上のBreusch-Pagan Testでは、テストするOLS回帰モデルの従属変数に

$$\sigma^2_t = \alpha_1 + \alpha_2 Z_{2t} + \ \ldots \ + \alpha_k Z_{kt} \qquad t=1,2,\ldots,n \quad [\text{Breusch-Pagan}]$$

というようにσ^2_tが使われていました。いろいろなバージョンがありますが、この仲間に

$$\sigma_t = \alpha_1 + \alpha_2 Z_{2t} + \ \ldots \ + \alpha_k Z_{kt} \qquad t=1,2,\ldots,n \quad [\text{Glesjer}]$$

$$\ln \sigma^2_t = \alpha_1 + \alpha_2 Z_{2t} + \ \ldots \ + \alpha_k Z_{kt} \qquad t=1,2,\ldots,n \quad [\text{Harvey-Godfrey}]$$

というふうに従属変数にσtの２乗ではなくて、その平方根のσtを用いるGlesjer Test、それからログをとったｌｎσ^2_tを用いるHarvey-Godfrey Testなどがあります。さまざまなバージョンがあるのですが、BPテストと整合性を持たせて一般性を保つために、ここでは２番目のBPテストのバージョンのnR2を共通の統計量としてテストを帰無仮説をテストします。帰無仮説はBPテストとまったく同じです。ただし、対象とするHeteroscedasticityの型が異なることになります。

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 bptest(y,x,0.05);



proc(2)= bptest(y,x,pp);
    local n,k,b,e,s2hat,y1,z,b1,e1,r2,chi,p,ess;
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    s2hat=e'e/n;
    /*    y1=sqrt(e^2/s2hat); */    /* BP Test */
    /* y1=sqrt(e^2/s2hat);    */    /* Glesjer test */
    y1=ln(e^2/s2hat); /* Harvey-Godfrey Test */
    z=x;              /* You could change this to z=x.*x; or z=x.*x.*x; */
    b1=inv(z'z)*z'y1;
    e1=y1-z*b1;
    r2=1-e1'e1/(y1'(eye(n)-1/n*ones(n,1)*ones(n,1)')*y1);
    chi=n*r2;    /* ess=(z*b1-meanc(y1))'(z*b1-meanc(y1)); chi=ess/2; */
    p=cdfchic(chi,k-1);
    if p>=pp;
        print "Breusch=Pagan Test (RESULT:Not reject H0)";
    else;
```

```
        print "Breusch=Pagan Test (RESULT:Reject H0)";
    endif;
    print/rz "X2(" k-1 ")=" chi ; print "P=" p;
    retp(chi,p);
endp;
```

画面表示(BP2)

```
Breusch=Pagan Test (RESULT:Reject H0)
X2(                3 )=        8.1904264
P=       0.042235818
```

画面表示(Glesjer)

```
Breusch=Pagan Test (RESULT:Not reject H0)
X2(                3 )=         5.992959
P=        0.11195329
```

画面表示(Harvey-Godfrey)

```
Breusch=Pagan Test (RESULT:Not reject H0)
X2(                3 )=        2.5221416
P=       0.47130289
```

上のプログラムでは、BPテストのy=e^2/s2hat;のところをsqrtをつけたものがGlesjerで、lnをつけたものがHarvey-Godfreyとなります。結果のところのメッセージを直したければそれに対応したメッセージ表示のprint文の引用符の中味を変更してください。結果は、３つのケースを一同に列挙しています。BPテスト以外は、５％でも10％の有意水準でさえも帰無仮説を棄却することはできない結果になっています。すなわち、下の２つの型のHeteroscedasticityは観測されていないということになります。したがって、新奇をてらって、例えばGlesjer Testだけをとりあげて、Heteroscedasticityがないと結論づけることは、いささか乱暴な議論と言わざるをえません。なお、これらのテストには呼び方やテストの仕方にいろいろなバージョンがあることを、ここでは、付け加えておきます。

Godfrey-Koenker Test

このGKテストは、漸近性を用いたテストでｎが十分に大きくないと一般には有効ではありませんが、

$\sigma^2_t = \sigma^2 \exp[\theta E(Yt)]$

とするときに、

H0: $\theta = 0$　　Homoscedasticity

H1: H0 is NOT true.　　Heteroscedasticity

というテストをσ^2_tとE(Yt)の推定値にもとづく２つの値の相関係数から統計量

$$GK = n[\rho(\hat{e}_t^2, \hat{Y}_t)]^2 \sim \chi^2(1)$$　だたし、ρは２変数の相関係数

をもとにテストするものです。

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 gktest(y,x,0.05);



proc(2)=gktest(y,x,pp);
    local n,k,b,e,gk,p,corr;
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    corr=corrx((e^2)~(x*b));
    gk=n*corr[2,1]^2;
    p=cdfchic(gk,1);
    if p>=pp;
        print "Godfrey=Koenker Test (RESULT:Not reject H0)";
    else;
        print " Godfrey=Koenker Test (RESULT:Reject H0)";
    endif;
    print "X2( 1 )=" gk ; print "P=" p;
    retp(gk,p);
endp;
```

画面表示

```
Godfrey=Koenker Test (RESULT:Reject H0)
X2( 1 )=          6.4554203
P=        0.011061438
```

プログラムでは、OLS残差を求めるところまでこれまでのプログラムと同様で、残差の核要素の２乗とｙの推定値、すなわち、Ｘβの相関係数を求めている。そのために、

corr=corrx((e^2)~(x*b));

というふうに、両者を水平方向にマージしたｎ×２の行列からcorrxという相関係数を求める組込み関数を用いて計算させている。なお、なお、その結果の２×２の相関行列のうちの(2,1)の要素だけが必要で、そのcorr[2,1]を２乗したものにｎをかけあわせてGK統計量としている。自由度は、この場合常に１で、それに対するP値はcdfchicを用いて、自由度が１のときのP値pを計算させている。このｐが与えられた有意水準ppより以上であれば、帰無仮説は棄却されなくて、そうでなければ、棄却される。このgkとpを内部で画面表示させた後に、gkとpの２つをリターンとして返している。

結果は、データ数からいうと、このテストを使うのには問題が残りますが、５％の有意水準で帰無仮説は棄却され、Heteroscedasticityの存在が疑われる結果になっています。

White Test

こちらのWhite Testは正規性が保たれていない場合に試してみるべき一般的なテストです。OLS残差をもとに、これを２乗したものを従属変数として、それまでの独立変数のほかに交差項を含む２辞までのすべての変数をかけ合わせた変数を作り、これらをすべてまとめて独立変数とします。例えば、定数項とX1とX2の列ベクトルの３つが従来の独立変数Ｘの内容とするならば、

$$e_t^2 = \alpha_1 + \alpha_2 X_{1t} + \alpha_3 X_{2t} + \alpha_4 X_{1t}^2 + \alpha_5 X_{2t}^2 + \alpha_6 X_{1t} X_{2t} + \eta$$

を推定します。定数項を含んだもともとの独立変数の個数をｋとするならば、この場合の係数の個数は、k(k+1)/2となります。この場合、3(3+1)/2=6と残差のηとなっています。これをもとに、ＢＰテストのように定数項の係数以外について、

H0: $\alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 = 0$

一般形は、$\alpha_2 = \alpha_3 = \ldots = \alpha_{k(k+1)/2} = 0$

H1: H0 is NOT true.

という帰無仮説の統計量

$$nR2 \sim \chi^2(k(k+1)/2-1)$$

をもとにテストをします。自由度は、定数項を除く、作り出した独立変数の総数です。

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 whitetest(y,x,0.05);

proc(2)=whitetest(y,x,pp);
    local n,k,z,i,j,h,p,wh,b,y1,b1,e1,r2;
    n=rows(x); k=cols(x);
    z=zeros(n,k*(k+1)/2);
    h=1; i=1;
    do while i<=k;
        j=i;
        do while j<=k;
            z[.,h]=x[.,i].*x[.,j];
            j=j+1; h=h+1;
        endo;
        i=i+1;
    endo;

    b=inv(x'x)*x'y;
    y1=(y-x*b)^2;
    b1=inv(z'z)*z'y1;
    e1=y1-z*b1;
    r2=1-e1'e1/(y1'(eye(n)-1/n*ones(n,1)*ones(n,1)')*y1);
    wh=n*r2;
    p=cdfchic(wh,k*(k+1)/2-1);

    if p>=pp;
        print "White Test (RESULT:Not reject H0)";
    else;
        print "White Test (RESULT:Reject H0)";
    endif;
```

```
    print/rz "X2(" k*(k+1)/2-1 ")=" wh ; print "P=" p;
    retp(wh,p);
endp;
```

画面表示

```
White Test (RESULT:Not reject H0)
X2(                    9 )=          15.052771
P=       0.089496599
```

プログラムでは、呼び出し部とprocedureの後半部分はこれまでのものとほぼ同様です。問題は、もともとのOLSの独立変数の個数が決まっている場合には楽にプログラムできますが、これを定数項を含めてｋ個の独立変数の場合について、一般化するのに少々工夫が必要となります。すなわち、もともとのOLSに定数項も含めてｋ個の独立変数がある場合、

```
    z=zeros(n,k*(k+1)/2);
    h=1; i=1;
    do while i<=k;
        j=i;
        do while j<=k;
            z[.,h]=x[.,i].*x[.,j];
            j=j+1; h=h+1;
        endo;
        i=i+1;
    endo;
```

とすることによって、k(k+1)/2個分の新しい（既存のものも含む）独立変数を一般化して作り出すことができます。具体的には、まず、ｎ×k(k+1)/2のディメンションの独立変数行列を零行列Ｚとして確保しておきます。（この作業は、プログラムには必須です。マージしていく方法でなければ、行列の大きさを先に決めておいて、１か０か何かで埋めておく必要が便宜上あります。）そして、３変数を用いた２重ループで、何かかける何かの独立変数行列を作成します。ループの中では、もともとのＸ行列のｉ番目の列とｊ番目の列とを順にかけあわせて、Ｚ行列のｈ列目の０の入っているとことに上書きしています。すなわち、h=1から順に１つずつ増加するように、ループ最深部にh=h+1;をおいて、

外側ｉループ１回目

```
i=1, j=1        内側ｊループ１回目
i=1, j=2                    ２回目
i=1, j=3                    ３回目
```

i=1, j=4　　４回目

外側ｉループ２回目
i=2, j=2　　内側ｊループ１回目
i=2, j=3　　２回目
i=2, j=4　　３回目

外側ｉループ３回目
i=3, j=3　　内側ｊループ１回目
i=3, j=4　　２回目

外側ｉループ４回目
i=4, j=4　　内側ｊループ１回目

としてループを終了します。これによって、ｉとｊのそれぞれの１番目には１の列ベクトルがきていますから、それとなにか別のものをかけることによって、定数項ともともとの変数の列が作成されることになり、Ｚ行列に１列目からk(k+1)/2列目までにx[.,i].*x[.,j]の要素対要素の計算結果が入ることになります。トリックはここだけで、ほかの部分は、このＺを独立変数とすることによって、いままでのプログラムと同様に計算できます。R^2の計算を残差を利用して計算して、これにｎをかけあわせたものを統計量whとします。あとの条件分岐とχ2にしたがうＰ値計算、画面表示、リターンの作成はまったく同じです。自由度は、定数項を含まない係数の数の合計のk(k+1)/2-1になります。

結果は、少し奇異に感じるかもしれませんが、今までの結果と違ってHomoscedasticityの帰無仮説を５％の有意水準で棄却することはできません。ただし、10％では棄却されます。ＢＰテストの仲間のHeteroscedasticityの型が異なるものと同様に、帰無仮説を棄却されにくい状況になっています。従属変数のラグ項が独立変数に含まれる場合、経験的に、このような結果はしばしば遭遇するものです。このような場合には、原始的なＧＱテストやＢＰテストが有効であると言えます。Whiteテストが常に一般的な有効な手段とは言われていますが、必ずしもそうとは言えないわけです。

Park Test

なお、説明変数側の疑わしきファクターをピンポイントでテストするには、Parkテストがあります。このテストは、heteroscedasticityが疑われる単一ファクターをＺとして

$$\ln e^2 = \beta_0 + \beta_1 \ln Z + \eta$$

としてβ_1が有意に0と異なるかをtテストするものです。

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 parktest(y,x,0.05);



proc(0)=parktest(y,x,pp);
    local n,k,b,e,s2hat,y1,i,z,b1,e1,varb,se,t,p;
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    y1=ln(e^2); /* Try (e^2) instead. */
    i=2;
    do while i<=k;
        z=ones(n,1)~(ln(x[.,i]));
        b1=inv(z'z)*z'y1;
        e1=y1-z*b1;
        s2hat=e1'*e1/(n-2);
        varb=s2hat*inv(z'z);
        se=diag(sqrt(varb));
        t=b1./se;
        p=2*cdftc(abs(t),n-2);
        print/rz i "rd/th column of X";
        if p[2]>=pp;
            print "Park Test (RESULT:Not reject H0)";
        else;
            print "Park Test (RESULT:Reject H0)";
        endif;
        print " b:" b1[2]; print "se:" se[2]; print " t:" t[2]; print " P:" p[2];
        print;
```

```
        i=i+1;
    endo;
endp;
```

画面表示

```
                2 rd/th column of X
Park Test (RESULT:Not reject H0)
 b:         0.70824585
se:         0.59583364
 t:          1.1886637
 P:         0.24492207

                3 rd/th column of X
Park Test (RESULT:Not reject H0)
 b:         0.70150138
se:         0.56596921
 t:          1.2394692
 P:         0.22583769

                4 rd/th column of X
Park Test (RESULT:Not reject H0)
 b:         0.57018008
se:         0.44116896
 t:          1.2924302
 P:         0.20716017
```

残念ながら、ＢＰテストとその仲間の関係からもわかるように、このテストでは単一のファクターを特定することはできません。ただし、従属変数を$\ln e^2$ではなくてe^2にすれば、当然のことながらうまくいくはずです。

画面表示（従属変数e^2のケース）

```
                2 rd/th column of X
Park Test (RESULT:Not reject H0)
 b:          9.2529888
se:          4.7146193
 t:          1.9626163
```

P: 0.060071271

3 rd/th column of X

Park Test (RESULT:Not reject H0)

b: 8.7868936

se: 4.4895757

t: 1.9571768

P: 0.060739205

4 rd/th column of X

Park Test (RESULT:Reject H0)

b: 7.9206012

se: 3.4244464

t: 2.3129581

P: 0.028581879

データ的に、この例はあまりよくありませんが、最終項の自然対数からｅ^{2}に対しては、その係数は５％の水準で有意であると言えて、この項が疑われます。よく考えれば、この最終項は従属変数のラグ項であって、いくら系列相関がないからと前もって仮定しても、結局はこの項が疑われるわけです。したがって、ここで言いたいのは、データがよくないというのではなくて、Parkテストの標準形は上に上げた形のテストをするわけですが、ＢＰテストとその仲間の関係と同じように、従属変数の側を変更したバージョン、例えば、ログをはずしたケースを試みることも必要になる場合もあるということです。

WhiteのHCSE

不均一分散の問題を解消するためには、いろいろな方法がありますが、まずは、係数βをもともとのOLSの値と同じに固定して、不均一分散の型や程度がわからなくてもVar(β)の値を修正してｔ値を有効にする方法をプログラムします。これは、WhiteのHCSEという方法(Heteroscedasticity Consistent Standard Error)で、

$$\frac{1}{n}\sum \sigma_i^2 x_i x_i' \text{の} ConsistentEstimator \text{が} \frac{1}{n}\sum \hat{e}_i^2 x_i x_i'$$

であることを利用して、

$$Var(\beta) = (X'X)^{-1}(\sum_{t=1}^{n} \hat{e}_t^2 x_t x_t')(X'X)^{-1} = (X'X)^{-1} X' diag(\hat{e}_t^2) X (X'X)^{-1}$$

を計算するものです。なお、OLSのβはそのまま同じものを使います。上の式をもとに、SEを計算してｔ値を求めるならば、漸近的にｔ分布にもとづく真のテストができるように

なるというものです。

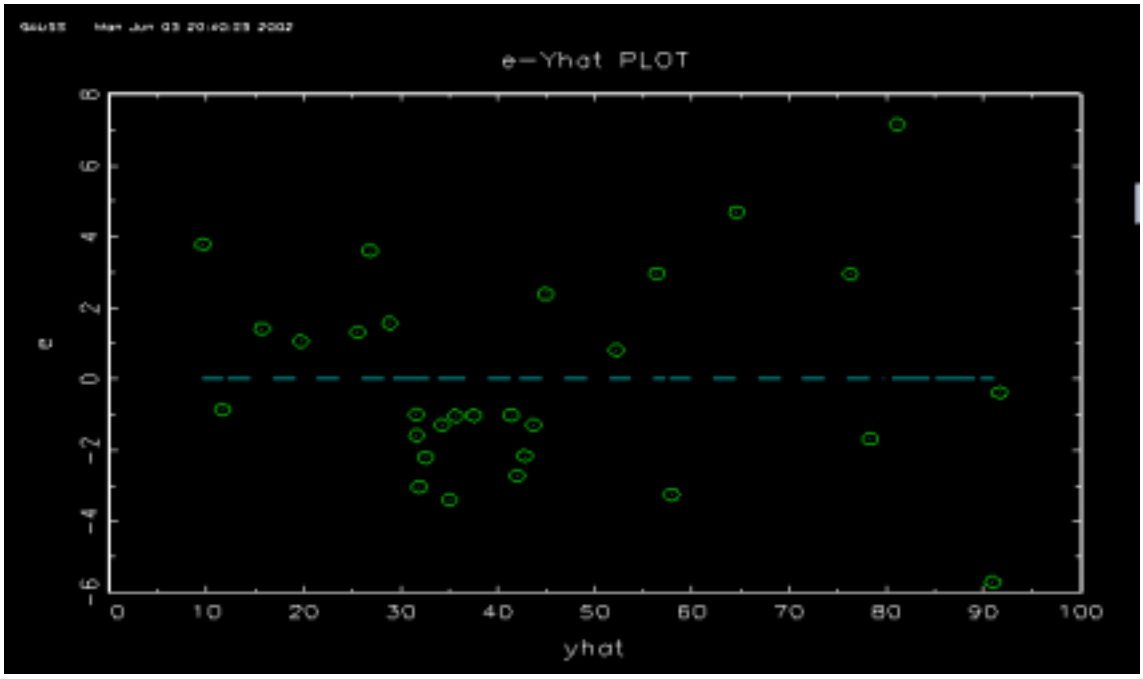
プログラム

```
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,.];
{b,se,t,hcse,tw,shat,df,r2,r2bar}=lsew(y,x);
print "OLS with SE";
print "                beta                    se                    t";;
print b~se~t;
print "      s=" shat; print "     r2=" r2; print "r2bar=" r2bar;
print;
print "OLS with White's SE";
print "                beta                hcse          beta/hcse";;
print b~hcse~tw;


proc(9)=lsew(y,x);
    local b,e,n,k,df,s2hat,shat,varb,se,t,sigma,varbw,hcse,tw,r2,r2bar,yhat;
    b=inv(x'x)*x'y;
    e=y-x*b;
    n=rows(x); k=cols(x);
    df=n-k;
    s2hat=e'e/df;
    shat=sqrt(s2hat);
    varb=s2hat*inv(x'x);
    se=sqrt(diag(varb));
    t=b./se;
    sigma=zeros(n,n);
    sigma=diagrv(sigma,(e.*e));
    varbw=inv(x'x)*(x'sigma*x)*inv(x'x);
    hcse=sqrt(diag(varbw));
    tw=b./hcse;
```

```
    r2=1-e'e/(y'(eye(n)-1/n*ones(n,1)*ones(n,1)')*y);
    r2bar=1-(1-r2)*(n-1)/(n-k);
        library pgraph;
        graphset;
        yhat=x*b;
        xlabel("yhat"); ylabel("e"); title("e-Yhat PLOT");
        _plctrl={-1,0};
        xy(yhat,e~zeros(n,1));
    retp(b,se,t,hcse,tw,shat,df,r2,r2bar);
endp;
```

グラフ表示



画面表示

OLS with SE

beta	se	t
-7.8922653	2.6077444	-3.0264720
0.69276561	0.13253255	5.2271354
-0.63412016	0.12615757	-5.0264139
0.71261457	0.094276392	7.5587807

s= 3.0081537

r2= 0.98442177

r2bar= 0.98255238

OLS with White's SE

beta	hcse	beta/hcse
-7.8922653	2.4049556	-3.2816678
0.69276561	0.11839425	5.8513449
-0.63412016	0.10931719	-5.8007360
0.71261457	0.10054830	7.0872858

プログラムでは、下のように、

```
sigma=zeros(n,n);
sigma=diagrv(sigma,(e.*e));
varbw=inv(x'x)*(x'sigma*x)*inv(x'x);
hcse=sqrt(diag(varbw));
tw=b./hcse;
```

まず、ｎ×ｎのΣ行列を零行列することで確保しておきます。そして、そこに残差の要素対要素で計算したe.*eを対角成分とするようにdiagrv(sigma,(e.*e))により、零行列に上書きして、あらためてsigmaと置きなおします。このoff diagonalがすべて０であるsigmaの行列をサンドイッチするように上の式のようにvar(β)を求め、varbwとします。その対角成分のルートをとったものが補正されたSE、すなわちHCSEとなります。これを従来のSEに代わり用いて、従来のOLSの係数βを割ったものが、この場合のWhiteのいう意味でのｔ値になります。なお、後半の一段下がった描画の部分では、まず、pgraphのライブラリを呼び出し、そのグローバル変数をgraphset;で初期化した後、計算されたyhatの列ベクトルの値をｘ軸方向にとり、残差ｅのベクトルをｙ軸方向にとり、xlabelとylabel、それにtitleを文字列として引用符に包んで設定します。なお、GAUSSには基準線などを自動的に描くというのはなくて、基準線も自分で設定します。そのために、ｎ×１の零ベクトルを残差ｅのあとにマージしてｙ軸方向に設定します。これにより、０のところで基準線が引けます。なお、_plctrl={-1,0};は、pgraph line controlの略で、ラインの属性を設定するグローバル変数の操作をしています。－１を入れると１つずつ点だけを表示します。－２なら１つとばしの２つごとの点になります。デフォルトは０が入っていて、線を引くものです。この場合、第１要素に対しては点だけで表示し、第２要素（すなわち基準線）には線を引くというグローバル変数設定をしています。

Weighted Least Squares vs. GLS

一方、Heteroscedasticityの型がわかっているときに、BLUEなestimatorβを根本から求めるには、WLSを用います。すなわち、定数項を含む独立変数および従属変数のすべての変数を「ある変数ベクトルで要素対要素でウエイトした」OLSを行なうものです。下の場

合、Heteroscedasticityの型があらかじめ、

$$\sigma^2_t = \sigma^2 X[.,4]$$

だとわかっていれば、

$$1/\sqrt{X[.,4]}$$

でウエイトするものです。

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,.];
weight=1/sqrt(x[.,4]);                          /* weight=1; for OLS */
{b,se,t,p,shat,df,r2,r2bar}=wls(y,x,weight);
print "b :" b; print "se:" se; print "t :" t; print "p :" p;
print "     r2=" r2'; print "r2bar=" r2bar; print "       s=" shat;


proc(8)=wls(y,x,weight);
    local b,e,n,k,df,s2hat,shat,varb,se,t,p,r2,r2bar;
    y=y.*weight; x=x.*weight;
    b=inv(x'x)*x'y;
    e=y-x*b;
    n=rows(x); k=cols(x);
    df=n-k;
    s2hat=e'e/df;
    shat=sqrt(s2hat);
    varb=s2hat*inv(x'x);
    se=diag(sqrt(varb));
    t=b./se;
    p=2*cdftc(abs(t),df);
    r2=1-e'e/(y'(eye(n)-1/n*ones(n,1)*ones(n,1)')*y);
    r2bar=1-(1-r2)*(n-1)/(n-k);
    retp(b,se,t,p,shat,df,r2,r2bar);
endp;
```

画面表示

```
b :
        -5.0968659
         0.55560212
        -0.51156468
         0.76793366
se:
          2.0231515
         0.12639717
         0.12007229
         0.10138137
t :
        -2.5192705
          4.3956847
        -4.2604724
          7.5747017
p :
       0.018522828
    0.00017845262
    0.00025319537
   6.2680474e-008
    r2=         0.93636219
r2bar=          0.92872566
     s=         0.45701134
```

単純ですが、上で注意すべきことは、定数項も含めてすべてウエイトしなければならなくて、事実上、定数項なしで計算することになります。上のプログラムでは、基本的には今まで組み立ててきたOLSのprocedureと同じですが、ｙとｘの列、行列を受け渡す際に、新たにウエイトを設定して、そのベクトル値をprocedure内部でｙと定数項も含んだｘの系列に要素対要素でかけ合わせたものです。もし、呼び出しの部分でweight=1;に設定すれば、通常のOLSが計算されます。なお、ｔ値に対するP値を計算するには、

```
p=2*cdftc(abs(t),df);
```

として、ｔ値の絶対値と自由度dfを与えてやって、ｔ分布にしたがうP値を求めます。なお、通常は方向性をもたない両側検定なので、２倍した値を計算しています。これまでのF分布やχ²分布にもとづくテストは片側テストがその大部分でしたから、それらのP値を求める

計算には２がかかっていなかったわけです。

これと同じ計算をΩのoff diagonalがすべて０（系列相関を考えない）のケースのGLSとして計算するのが、下のプログラムです。基本的には、ウエイトの２乗の逆数がΩの対角成分になります。

$$b_{gls}=(X'\Omega^{-1}X)^{-1}X\Omega^{-1}y$$

$$Var(\beta_{gls})=\sigma 2(X'\Omega X)^{-1}$$

となる計算をもとに考えます。

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,.];
omega=zeros(rows(x),rows(x)); omega=diagrv(omega,x[.,4]);
{b,se,t,p,df,r2,r2bar}=gls(y,x,omega);
print "b :" b; print "se:" se; print "t :" t; print "p :" p;
print "     R2=" r2'; print "R2bar=" r2bar;


proc(7)=gls(y,x,omega);
    local b,e,n,k,df,s2hat,varb,se,t,p,r2,r2bar;
    n=rows(x); k=cols(x);
    df=n-k;
    b=inv(x'inv(omega)*x)*x'inv(omega)*y;
    e=y-x*b;
    s2hat=e'inv(omega)*e/df;
    varb=s2hat*inv(x'inv(omega)*x);
    se=diag(sqrt(varb));
    t=b./se;
    p=2*cdftc(abs(t),df);
    r2=1-e'e/(y'(eye(n)-1/n*ones(n,1)*ones(n,1)')*y);
    r2bar=1-(1-r2)*(n-1)/(n-k);
    retp(b,se,t,p,df,r2,r2bar);
```

```
endp;

画面表示
b :
        -5.0968659
         0.55560212
        -0.51156468
         0.76793366
se:
          2.0231515
         0.12639717
         0.12007229
         0.10138137
t :
        -2.5192705
          4.3956847
        -4.2604724
          7.5747017
p :
       0.018522828
    0.00017845262
    0.00025319537
   6.2680474e-008
    R2=         0.98355885
R2bar=          0.98158591
```

*WeightedLeastSquares*の*weight*を$\frac{1}{\sqrt{w_t}}$とすれば

$$\Omega^{-1}=\begin{bmatrix} \frac{1}{(\sqrt{w_1})^2} & 0 & \cdots & \cdots & 0 \\ 0 & \frac{1}{(\sqrt{w_2})^2} & 0 & . & \vdots \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & . & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \frac{1}{(\sqrt{w_n})^2} \end{bmatrix}$$

ここで、$\Omega=\begin{bmatrix} w_1 & 0 & \cdots & \cdots & 0 \\ 0 & w_2 & 0 & . & \vdots \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & . & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & w_n \end{bmatrix}$

という関係になり、このΩがGLSで使われています。なお、ここでは系列相関については考えていないので、off diagonalはすべて０になっています。プログラムでは、呼び出し部分で、ｎ×ｎのΩ行列を指定します。最初にΩには零行列にしておいて、そこにここでは行列Ｘの４列目の列ベクトルを組込み関数diagrvでもって、流し込みます。そのほかの部分はOLS計算とほぼ同じで、上の２つの公式を用いてあてはめて計算をします。いったんVar(b_{gls})が求まれば、そなとのSE、ｔ値、ｐ値の計算は以前とまったく同じ要領です。

結果は、Weighted Least Squaresの結果と係数、SE、P値は同じになります。すなわちウエイトの２乗の列ベクトルを対角成分とするのが、Ωの逆行列になっています。これは、すべての変数にweightをかけるというのは、この場合のＸの４列目の変数のシリーズのルートをとった数ですべての変数で割っていますから、行列の２回かけ合わさっているところがあるGLSの計算では、すべての２変数のかけ合わせの部分にサンドイッチされる形でこのルートをとった形で割るという意味の行列の２乗分の行列がくるわけです。このΩの逆行列は、weightが２つかけ合わさっている行列と考えてもよいでしょう。

上の方法以外に、よく手軽に行なわれるものに対数変換をするもの、さらに意味づけは難しくなりますがBox-Cox変換をするものがあります。