3.13 プログラミング 系列相関とテスト(1)　　ver.0.2

Durbin-Watson Statistic

従属変数のラグを説明変数側に含まないモデルのAR(1)のテストする有名な統計量にDurbin-Watson statisticがあります。統計量から判断基準にするTableは別途用意する必要がありますが、統計量自体は極めて簡単にプログラムできます。すでにresidualsのベクトルｅが用意または計算されている場合には、DW統計量の計算方法は、

$$d = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$$

ですから、GAUSSの表示方式に直すと

```
dw=sumc((e[2:n,.]-e[1:n-1,.])^2)/sumc(e^2);
```

または、２乗和はvector'vectorで表せますから、

```
dw=(e[2:n,.]-e[1:n-1,.])'(e[2:n,.]-e[1:n-1,.]) / e'e;
```

とするだけで計算できます。または、新たにprocedureを書くのならば、

```
proc dwtest(y,x);
    local n,b,e,dw;
    n=rows(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    dw=sumc((e[2:n,.]-e[1:n-1,.])^2)/sumc(e^2);
    retp(dw);
endp;
```

のように、従属変数ベクトルｙと定数項ベクトルを含む独立変数行列Ｘがわかっているならば、通常のOLSの計算とそのresidualsを求める計算の後に、上のどちらか１行を入れてそのDWの値を返すようにします。ここではベクトルe_{t-1}を作るために、e[1:n-1],.]としているところがポイントです。このままでは、ｅベクトルの１行目からn-1行目までのサブベクトルを作成しただけで、何らラグをとることとは無関係なのですが、引かれる方のe_tにe[2:n,.]を用いることによって、両者ともに要素n-1個の長さの列ベクトルとなり、結果としてe_tとe_{t-1}の２つの列ベクトルは１期ずれることになります。ここで

N	e_t	e_{t-1}
1	1	
2	2	1
3	3	2
4	4	3

という具合にe_{t-1}に１から$n-1$までを用い、e_tに２からnまでを用いると、e_{t-1}は１つラグをもったと同じ結果となります。２乗和計算するには、行列計算としてe'eの形をこれまでどおり使うか、eのところの計算が長くなれば２つ同じことを書くのは面倒ですから、組込み関数sumcを使って、sumc(e^2)の計算をさせます。

Wallis' 4th order Durbin-Watson statistic

これに対して、４半期データを用いた場合のe_tと前年同期のe_{t-4}の関係見るのに、

$$e_t = \rho e_{t-4} + u$$

のときに、H0: $\rho = 0$ をテストするために、DWと同じような統計量

$$d = \frac{\sum_{t=5}^{n} (e_t - e_{t-4})^2}{\sum_{t=1}^{n} e_t^2}$$

を計算できます。GAUSSでは上のプログラムを若干変更するだけで、

```
dw4=sumc((e[5:n,.]-e[1:n-4,.])^2)/sumc(e^2);
```

とするだけで計算できます。または、新たにprocedureを書くのならば、

```
proc dw4test(y,x);
    local n,b,e,dw4;
    n=rows(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    dw4=sumc((e[5:n,.]-e[1:n-4,.])^2)/sumc(e^2);
    retp(dw4);
endp;
```

のようにします。この場合の上限duと下限dlについては、Wallis, K.F. (1972), "Testing for Fourth Order Autocorrelation in Quarterly Regression Equations," Econometrica, 40, 617-636.またはKmenta, J. (1997), Elements of Econometrics.等を参照して判断のこと。

Generalized Durbin-Watson statistic

上の２つの場合の中間の２期前、３期前などとのresidualsの相関については、ｊ期前との関係の場合の一般形を考えればよいことになります。

$$d = \frac{\sum_{t=j+1}^{n} (e_t - e_{t-j})^2}{\sum_{t=1}^{n} e_t^2}$$

となりますから、これをプログラムにするには、４期の場合の類推から一般形で書くと、

分子の引かれる前のｅをj+1からｎまでとし、後ろのｅを１からn-jとすればよいわけで、

```
gdw=sumc((e[j+1:n,.]-e[1:n-j,.])^2)/sumc(e^2);
```

の１行を付け加えることになります。または、procedureで表すのならば、その第３要素をラグｊとすると、

```
proc gdwtest(y,x,j);
    local n,b,e,gdw;
    n=rows(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    gdw=sumc((e[j+1:n,.]-e[1:n-j,.])^2)/sumc(e^2);
    retp(gdw);
endp;
```

として、呼び出しの部分で、新たにｊの部分に２とか３とかのラグの数を入れればよいことになります。GAUSS上のプログラムでは極めて簡単に変更ができます。Vinod, H.D. (1973), "Generalization of the Durbin-Watson Statistic for Higher Order Autoregressive Process," Communications in Statistics, 2, 115-144を参照のこと。なお、統計値と上限下限は独自にシュミレーションで求める必要があるりますが、DW値のtableと上のKmentaの4th orderのtableの２つの値から、その中間あたりを推測して判断することも十分可能です。これらの判断は、ピンポイントでその次数ラグのところだけをテストするものであって、後に述べる高次の場合のQおよびQ'テストとはまったくテストの対象が異なります。

Durbin's h

一方、従属変数のラグ変数が説明変数側に含まれている場合、通常のDurbin-Watson統計量とは異なるもので判断する必要があります。それがｈ統計量です。これは、データ数ｎが十分に大きな場合に用いられるテストです。下の計算は、ρをDWからの近似値ではなくて、

$$e_t = \rho e_{t-1} + \varepsilon$$

から定数項なしで推定するという計算方法を用いたものです。通常、「最小２乗原理」でのρの導出というのは、e_{t-1}をX、e_tをｙと見たてて、この式を行列形式で解いた、

$$\rho = \mathrm{inv}(e_{t-1}'e_{t-1})\ e_{t-1}'e_t$$

の計算のことを指します。ここでの統計量は、従属変数のラグ項の係数をβ_4とすれば、

$$h = \hat{\rho}\sqrt{\frac{n}{1 - nVar(\hat{\beta})}} \sim N(0,1)$$

となり、standard normalな分布にしたがう。系列相関のところで用いたdatafile12.txtをそこと同じモデルとして呼び出して、行列Ｘの４列目に従属変数のラグ項があって、５%の

有意水準でテストするとするならば、

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 hstat(y,x,4,0.05);

proc hstat(y,x,j,pp);
    local n,k,b,e,rho,s2hat,varb,h,p;
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    s2hat=e'e/(n-k);
    rho=inv(e[1:n-1,.]'e[1:n-1,.])*e[2:n,.]'e[1:n-1,.];
    varb=s2hat*inv(x'x);
    if (1-n*varb[j,j])<0;
        errorlog "Can't calculate h stat.";
        retp(".");
    endif;
    h=rho*sqrt(n/(1-n*varb[j,j]));
    p=2*cdfnc(abs(h));
    if p>=pp;
        print "h stat test (RESULT:Not reject H0)";
        print "No 1st order autocorrelation";
    else;
        if h>0;
            print "h Test (RESULT:Reject H0)";
            print "Positive 1st order autocorrelation exists";
        endif;
        if h<0;
            print "h Test (RESULT:Reject H0)";
```

```
                print "Negative 1st order autocorrelation exists";
            endif;
        endif;
        print "rho=" rho; print "    h=" h; print "    P=" p;
        retp(h);
    endp;
```

画面表示

```
h stat test (RESULT:Not reject H0)
No 1st order autocorrelation
rho=         0.22634165
  h=          1.4147812
  P=         0.15713265
```

この場合には、ρが正ではあるものの「H0:１階の自己相関がない」という帰無仮説が棄却されないので、１階のAutocorrelationはないと言えます。上のprocedureでは、通常通りのOLSでresidualsを求めて、ρの推定値の計算をします。この場合、invの中味も、それ以外のところも共にスケラーですから、invの中身を割る数として分母にもってきても同じ計算になります。また、転置の記号はかけ合わせの意味も含んでいるばかりでなく、転置をはさんだ項を先に計算する演算の優先順位もあるので、それぞれに括弧のしるしは必要ありません。なお、ｈ統計量のルートのなかの分母は常に０よりも大きい必要があるので、条件分岐で一応チェックをしています。このラグ項の係数のVarianceが1/nよりも大きければこのルートのなかの分母はマイナスの値になって、この統計量自体の計算は不可能です。もちこの場合に該当する場合には、文字列としてのドットのマークをリターンとして返すようにして、そこでprocedure内の計算は終わります。このretpはprocedureからの出口と考えてください。そうでなければ、そのあとにアルゴリズムの流れは続き、両側検定の場合のＰ値を計算して､それが設定した有意水準0.05よりも大きければ帰無仮説は棄却されなくて、それが0.05よりも小さければ棄却されるという条件分岐をしています。最終的に、リターンとしてではなくprocedureの内部で、ρとｈ統計量とＰ値を画面表示させて、ｈの値をリターンとして返すようにしています。なお、呼び出し部分でcallで呼び出し、そのリターンを表示しないでとめています。

m test

従属変数のラグ変数が説明変数側に含まれている場合において、データ数が十分にない場合や、ｈ統計量が計算不能の場合などの場合に、手軽にOLSを用いて状況を見る際に用いられるのが、このm testです。一度もともとのモデルをOLSしてresidualsを求めます。

この１期ラグをｋ＋１番目の説明変数と加え、もう一度OLSをして、そのresidualsのラグ項が有意であるかどうかをｔテストで調べるものである。すなわち、

$$Yt = \beta_1 + \beta_2 X_2 + \beta_3 X_3 + \ \ldots\ + \beta_k X_k + \beta_{k+1} e_{t-1}$$

を推定して、

H0: $\beta_{k+1} = 0$　１階の自己相関がない

H1: $\beta_{k+1} \neq 0$　１階の自己相関がある

（$\beta_{k+1} > 0$　正の１階の自己相関、$\beta_{k+1} < 0$　負の１階の自己相関）

の判断をｔ値によって行ないます。

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 mtest(y,x,0.05);

proc(0)=mtest(y,x,pp);
    local n,k,b,bz,e,ez,z,s2hat,varb,se,t,p;
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    y=y[2:n,.]; z=x[2:n,.]~e[1:n-1,.];
    bz=inv(z'z)*z'y;
    ez=y-z*bz;
    s2hat=ez'ez/(n-k-1);
    varb=s2hat*inv(z'z);
    se=diag(sqrt(varb));
    t=bz./se;
    p=2*cdftc(abs(t),n-k-1);
    if p[k+1]>=pp;
        print "m Test (RESULT:Not reject H0)";
        print "No 1st order autocorrelation";
    else;
        if bz[k+1]>0;
```

```
                print "m Test (RESULT:Reject H0)";
                print "Positive 1st order autocorrelation exists";
            endif;
            if bz[k+1]<0;
                print "m Test (RESULT:Reject H0)";
                print "Negative 1st order autocorrelation exists";
            endif;
        endif;
        print "    t=" t[k+1]; print "    P=" p[k+1];
        retp;
    endp;
```

画面表示

```
m Test (RESULT:Not reject H0)
No 1st order autocorrelation
   t=          1.8482894
   P=       0.076919982
```

上のように、当該箇所の係数は有意ではなくて、結果はｈ統計量のものと同じように、１階の自己相関がないと考えられます。プログラムでは同じように呼び出して、procedureの第３要素に、有意水準を小数として入れます。そして、通常どおりOLSを行なってｅを計算し、その１期前のラグ項を付け加えたものを説明変数とするOLSをもう一度行なって、通常通りその項のｔテストを行なって、その両側検定のP値を計算の上、与えられた0.05の値よりも大きければ有意でない、小さければ有意であるという判定をして、procedure内部でｔ値とそれに対するP値を表示して、リターンは０個で終了させています。

Box-Pierce Q Test

従属変数のラグが説明変数側にない場合の高次の系列相関

H0: $\rho_1 = \ldots = \rho_k$

をテストするものです。統計量は

$$Q = n\sum_{j=1}^{p}\left[\frac{\sum_{t=j+1}^{n} e_t e_{t-j}}{\sum_{t=1}^{n} e_t^2}\right]^2 \sim \chi^2(p)$$

をテストするもので、ｐ次までにARまたはMAがあるかどうかをテストするという累積和

的なものであります。一度有意になるとそのあとの次数は、通常の場合、有意になり続けます。今度はラグ項のないdatafile13.txtというデータがDドライブにあるものとします。それを、DW値のprocedureとともに、このQテストの値をラグ12期まで、有意水準５％でテストします。

プログラム

```
new; cls;
load data[40,3]=d:datafile13.txt;
Y=data[2:40,1]; L=data[2:40,2]; K=data[2:40,3];
y=ln(Y); x=ones(39,1)~ln(L)~ln(K);
print "DW=" dwtest(y,x);
call bpqtest(y,x,12,0.05);

proc dwtest(y,x);
        local n,b,e,dw;
   n=rows(x);
   b=inv(x'x)*x'y;
   e=y-x*b;
   dw= sumc((e[2:n,.]-e[1:n-1,.])^2) / sumc(e^2);
   retp(dw);
endp;

proc(0)=bpqtest(y,x,p,pp);
        local n,b,e,bpq,bpqj,p,j;
        n=rows(x);
        b=inv(x'x)*x'y;
        e=y-x*b;
        j=1; bpq=0;
        do while j<=p;
                print/rz j "th order";;
                bpqj=n*(sumc(e[j+1:n,.].*e[1:n-j,.])/sumc(e^2))^2;
                bpq=bpq+bpqj;
                if cdfchic(bpq,j)>=pp;
                        print " Box-Pierce Q = " bpq "(RESULT: Not reject H0)";
                else;
                        print " Box-Pierce Q = " bpq "(RESULT: Reject H0)";
```

```
                endif;
                j=j+1;
            endo;
        endp;
```

画面表示

DW= 0.85808023

1 th order Box-Pierce Q =	12.183290 (RESULT: Reject H0)
2 th order Box-Pierce Q =	12.267335 (RESULT: Reject H0)
3 th order Box-Pierce Q =	13.726258 (RESULT: Reject H0)
4 th order Box-Pierce Q =	13.843002 (RESULT: Reject H0)
5 th order Box-Pierce Q =	15.419349 (RESULT: Reject H0)
6 th order Box-Pierce Q =	15.507005 (RESULT: Reject H0)
7 th order Box-Pierce Q =	17.127147 (RESULT: Reject H0)
8 th order Box-Pierce Q =	21.245578 (RESULT: Reject H0)
9 th order Box-Pierce Q =	22.894072 (RESULT: Reject H0)
10 th order Box-Pierce Q =	23.869820 (RESULT: Reject H0)
11 th order Box-Pierce Q =	26.059211 (RESULT: Reject H0)
12 th order Box-Pierce Q =	28.494196 (RESULT: Reject H0)

上の結果は、DWの値とともに、Ｑ Testの値と５％有意水準での判断を12ラグまで求めたものです。すでに１期目のラグで有意になっているので、そのあとはずっと同じ結果になることは前述した通りです。現に、DW値も正の１次の系列相関を示しています。上のプログラムのprocedure部では、まず１期目のQ統計量を求めます。統計量のｎの値はΣの外でも内でも定数なので同じことになります。これをｊ＝１からｐまで順に足し合わせていくことになります。そのループの中で、おのおのについてχ2の片側検定のP値を求め、それを与えられた0.05と比べて、それよりも小さくなれば有意で系列相関がその時点までのどれかの時点にあるという結果が得られます。

Ljung-Box Q' Test

上のQテストと同様に、OLSのresidualsをもとに、

$$Q = n(n+2)\sum_{j=1}^{p}\left[\frac{\sum_{t=j+1}^{n} e_t e_{t-j}}{\sum_{t=1}^{n} e_t^2}\right]^2 /(n-j) \sim \chi^2(p)$$

の統計量を同じようにテストするものです。

プログラム

```
new; cls;
load data[40,3]=d:datafile13.txt;
Y=data[2:40,1]; L=data[2:40,2]; K=data[2:40,3];
y=ln(Y); x=ones(39,1)~ln(L)~ln(K);
call lbqtest(y,x,12,0.05);

proc (0) = lbqtest(y,x,p,pp);
    local n,b,e,lbq,lbqj,p,j;
    n=rows(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    j=1; lbq=0;
    do while j<=p;
        print/rz j "th order";;
        lbqj=n*(n+2)*(sumc(e[j+1:n,.].*e[1:n-j,.])/sumc(e^2))^2/(n-j);
        lbq=lbq+lbqj;
        if cdfchic(lbq,j)>=pp;
            print " Ljung-Box Q' = " lbq "(RESULT: Not reject H0)";
        else;
            print " Ljung-Box Q' = " lbq "(RESULT: Reject H0)";
        endif;
        j=j+1;
    endo;
endp;
```

画面表示

```
 1 th order Ljung-Box Q' =        13.145129 (RESULT: Reject H0)
 2 th order Ljung-Box Q' =        13.238260 (RESULT: Reject H0)
 3 th order Ljung-Box Q' =        14.899811 (RESULT: Reject H0)
 4 th order Ljung-Box Q' =        15.036568 (RESULT: Reject H0)
 5 th order Ljung-Box Q' =        16.937457 (RESULT: Reject H0)
 6 th order Ljung-Box Q' =        17.046363 (RESULT: Reject H0)
 7 th order Ljung-Box Q' =        19.122169 (RESULT: Reject H0)
 8 th order Ljung-Box Q' =        24.569127 (RESULT: Reject H0)
 9 th order Ljung-Box Q' =        26.822069 (RESULT: Reject H0)
```

10 th order Ljung-Box Q' =	28.201574 (RESULT: Reject H0)
11 th order Ljung-Box Q' =	31.407469 (RESULT: Reject H0)
12 th order Ljung-Box Q' =	35.105038 (RESULT: Reject H0)

上のように、結果は若干統計量の差はあるものの、基本的にQ統計量の結果と同様に、ラグ１期から系列相関が見られます。プログラム的には、まったく同じ手順で計算できます。それぞれのｊ期の増分のところの計算に、ｎにかえてn(n+2)を前につけて、後ろで(n-j)で割ればよいことになります。なおn(n+2)をΣの中に入れても定数ですから結果は同じです。

Breusch-Godfrey LM Test

一方、従属変数のラグを説明変数側に持つ場合の高次のテストには、Breisch-GodfreyのLMテストをすることになります。これは、OLSから求められたresidualsのｅをもとに、

$$e_t = \beta 0 + \beta 1 X1 + \ldots + \beta k Xk + \rho_1 e_{t-1} + \rho_2 e_t + \ldots + \rho_{t-p} e_{t-p}$$

をOLSで推定してそのR2をもとめて

$$nR2 \quad \sim \quad \chi^2(p)$$

をテストするものです。また、その変種としては、これを非制限モデルとして、

$$\rho_1 = \ldots = \rho_{t-p}$$

のとき式を制限モデルとして、F 分布にもとづくテストをするもので、

$$F = \frac{(RSSr - RSSu)/p}{RSSu/(n-k-p)} \approx \frac{(RSSr - RSSu)/p}{RSSu/n} = F'$$

のように、ｎが十分に大きいとき近似置換して、このF'をもとにｐF'=ｎR2から

$$pF' \quad \sim \quad \chi_2(p)$$

をテストするものです。プログラムでは、ラグ項がないデータですが、上のQテストと同じdatafile13.txtを使って比較をしています。呼び出し部分は以前と同様で、procedure内部では、前半ではnR2のケースのもの、そして後半ではその変種のpF'のケースのものを用いて高次の系列相関のテストをしています。

プログラム

```
new; cls;
load data[40,3]=d:datafile13.txt;
Y=data[2:40,1]; L=data[2:40,2]; K=data[2:40,3];
y=ln(Y); x=ones(39,1)~ln(L)~ln(K);
call bglmtest(y,x,4,0.05);
/*
** Breusch-Godfrey LM Test for Autocorrelation AR(p)
** (C) Copyright 2002 Yosuke Amijima. All Rights Reserved.
```

```
*/
    proc(0)=bglmtest(y,x,p,pp);
        local n,b,e,i,j,ematrix,e0,y1,x1,b1,n1,e1,r2,chi2,pv,pf,pv1;
        print "Breusch-Godfrey LM Test for AR(p)";
        n=rows(x);
        b=inv(x'x)*x'y;
        e=y-x*b;

        i=1;
        do while i<=p;
            print/rz "(DF) Up to" i "th order";
            ematrix=zeros(n-i,i);
            j=1;
            do while j<=i;
                ematrix[.,j]=e[i+1-j:n-j,.];
                j=j+1;
            endo;
            e0=e[i+1:n,.];
            y1=e0; x1=x[i+1:n,.]~ematrix;

            b1=inv(x1'x1)*x1'y1;
            n1=rows(x1);
            e1=y1-x1*b1;
            r2=1-e1'e1/(y1'(eye(n1)-1/n1*ones(n1,1)*ones(n1,1)')*y1);
            chi2=n*r2;
            pv=cdfchic(chi2,i);
            if pv>=pp;
                print "LM Test (RESULT:Not reject H0)";
            else;
                print "LM Test (RESULT:Reject H0)";
            endif;
                print "nR2       =" chi2; print "Prob>X2=" pv;
            pf=(e'e-e1'e1)/e'e*n;
            pv1=cdfchic(pf,i);
            if pv1>=pp;
                print "LM Test (RESULT:Not reject H0)";
```

```
            else;
                print "LM Test (RESULT:Reject H0)";
            endif;
            print "pF'      =" pf; print "Prob>X2=" pv1;
            print;
            i=i+1;
        endo;
        retp;
    endp;
```

画面表示

```
Breusch-Godfrey LM Test for AR(p)
(DF) Up to                    1 th order
LM Test (RESULT:Reject H0)
nR2      =          13.298582
Prob>X2=     0.00026560691
LM Test (RESULT:Reject H0)
pF'      =          13.396907
Prob>X2=     0.00025203956

(DF) Up to                    2 th order
LM Test (RESULT:Reject H0)
nR2      =          20.587587
Prob>X2=    3.3842490e-005
LM Test (RESULT:Reject H0)
pF'      =          21.473635
Prob>X2=    2.1729983e-005

(DF) Up to                    3 th order
LM Test (RESULT:Reject H0)
nR2      =          22.016528
Prob>X2=    6.4716620e-005
LM Test (RESULT:Reject H0)
pF'      =          23.524008
Prob>X2=    3.1399963e-005
```

```
(DF) Up to                    4 th order
LM Test (RESULT:Reject H0)
nR2      =          22.555236
Prob>X2=     0.00015534841
LM Test (RESULT:Reject H0)
pF'      =          24.110031
Prob>X2=    7.5919038e-005
```

上の前半はR2をもとにしたテスト、後半はｎが十分に大きい場合のmodified Fをもとにしたテストを行なって、ラグ１期からラグｐ期までの計算を繰り返しループで行なっています。プログラムの核になるのは、residualsのラグ１期からｐ期までの値を求めて下のように可変グリッドに格納するところです。

```
i=1;
do while i<=p;
    ematrix=zeros(n-i,i);
    j=1;
    do while j<=i;
        ematrix[.,j]=e[i+1-j:n-j,.];
        j=j+1;
    endo;
e0=e[i+1:n,.];
i=i+1;
endo;
```

水平方向にlagn(e,i)としてeのラグを１期からｐ期までとって、それらを水平方向にマージしてから、上からｐ列だけ切り取ることも可能ですが、それはラグの数があらかじめ決まっている場合にのみプログラムできます。ここでは、ｐ期のラグまでという可変なラグ数ｐの値に応じて変化するresidualsのグリッドを考えてみましょう。そのためには、まずラグｐ期までの項を作るとして、zeros(n-p,p)のグリッドを初期値として確保します。何もないところには代入していけないので、この作業はプログラムでは必須です。それを１期からｐ期までのラグについて可変グリッドにするためにｐをｉに書き換えて考えます。その上で、ｊ番目の列にｅのｊラグを代入します。この場合のｅはもとのi+1-jからn-jまでのものをとればよいことになります。この場合のresidualsのｅのラグはn-iとなります。これはzeros(n-i,i)として初期化したematrixの縦の行数に必ず等しくなります。これをｊの１列目から最高ｉ列目まで作って、ematrixの行列の要素と置き換えます。これを外側のｉループでラグｐ期までresidualsの可変グリッドを１つ１つ作っていきます。それに対して、従属変数となるラグなしのｅの上からｉ列も切り取ります。この他に、当然のことながら、行

列Xの説明変数も同様の処理が必要になります。プログラムでは、このematrixを説明変数行列Xに水平方向にマージして、これを従属変数ｅに対してOLSをします。そしてR2を計算して、ラグをとってカットする前のもともとのデータ全体の数ｎをかけ合わせたものを統計量として、χ2のP値を求めます。それがあらかじめ定めた0.05よりも小さければ、帰無仮説は棄却されて、そのｐ期までに少なくとも系列相関があると判断できます。後半は、このラグ付のOLSのRSSを求め、非制限モデルのRSSとし、このラグ可変グリッドのないOLSをさらに推定して、そのRSSを求め、それを制限モデルのRSSとして、F'をもとめさらにｐをかけて、同様なχ2にしたがうテストをしています。計算では、ｐが既にキャンセルアウトされた形をプログラム上では計算しています。（なお、上の結果は、従属変数のラグ１の項を最後まで考えないプログラムをしています。ラグ項を説明変数に含めて最初から計算すれば結果は若干異なってきます。）

Correlogram

ここでは、自己相関係数とそれから得られる偏自己相関係数を計算します。通常のパッケージソフトの大半には既に組み込まれていますが、これをプログラムするには一工夫必要です。また、パッケージごと微妙な計算の違いが見られます。ここでは、自己相関係数ACをラグ相互の自己共分散と分散関数の積のルートをとったもの、すなわち標準偏差をかけあわせたものとの比として計算します。すなわち、

$$\rho(j)=\frac{Cov(e_t,e_{t-j})}{\sqrt{Var(e_t)}\sqrt{Var(e_{t-j})}}=\frac{Cov(e_t,e_{t-j})}{Std(e_t)Std(e_{t-j})}$$

を自己相関係数ACとします。一方、偏自己相関係数を求めるのには、このρ(ｊ)行列

$$P_j=\begin{bmatrix}1 & \rho_1 & \rho_2 & . & \rho_{j-1}\\ \rho_1 & 1 & \rho_1 & . & \rho_{j-2}\\ \rho_2 & . & 1 & . & .\\ . & . & . & . & .\\ \rho_{j-1} & \rho_{j-2} & \rho_{j-3} & . & 1\end{bmatrix}$$

として、その最終ｊ列目をρ1, ρ2, … , ρjの列ベクトルで置き換えたものを

$$P_j^{*}=\begin{bmatrix}1 & \rho_1 & \rho_2 & . & \rho_1\\ \rho_1 & 1 & \rho_1 & . & \rho_2\\ \rho_2 & . & 1 & . & \rho_3.\\ . & . & . & . & .\\ \rho_{j-1} & \rho_{j-2} & \rho_{j-3} & . & \rho_j\end{bmatrix}$$

としたときに、

Pjj＝｜Pj*|/|Pj|

として求めます。逐次計算法と結果は同じになるはずです。（ただし、もともとのACの方が用いる行数の関係などで若干の違いが出ると計算結果はPACの方でもずれが生じます。）

プログラム

```
new; cls;
load data[40,3]=d:datafile13.txt;
Y=data[2:40,1]; L=data[2:40,2]; K=data[2:40,3];
y=ln(Y); x=ones(39,1)~ln(L)~ln(K);
call correlogram(y,x);
/*
** Correlogram(Autocorrelation and Partial Auctocorrelation) up to 12 lags
** (C) Copyright 2002 Yosuke Amijima. All Rights Reserved.
*/
proc(0)=correlogram(y,x);
    local n,k,b,e,i,j,ej,e0,cov,pmatrix,psmatrix,pcol,ac,pac,axis;
    n=rows(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    pcol=1|zeros(12,1);
    i=1;
    do while i<=12;
        j=1;
        do while j<=i;
            e0=e[i+1:n,.];
            ej=e[i+1-j:n-j,.];
            j=j+1;
        endo;
        cov=vcx(e0~ej)/stdc(e0)/stdc(ej);
        pcol[i+1,1]=cov[2,1];
    i=i+1;
    endo;
    ac=pcol[2:13];


pmatrix=pcol~lagn(pcol,1)~lagn(pcol,2)~lagn(pcol,3)~lagn(pcol,4)~lagn(pcol,5)~lagn(pco
```

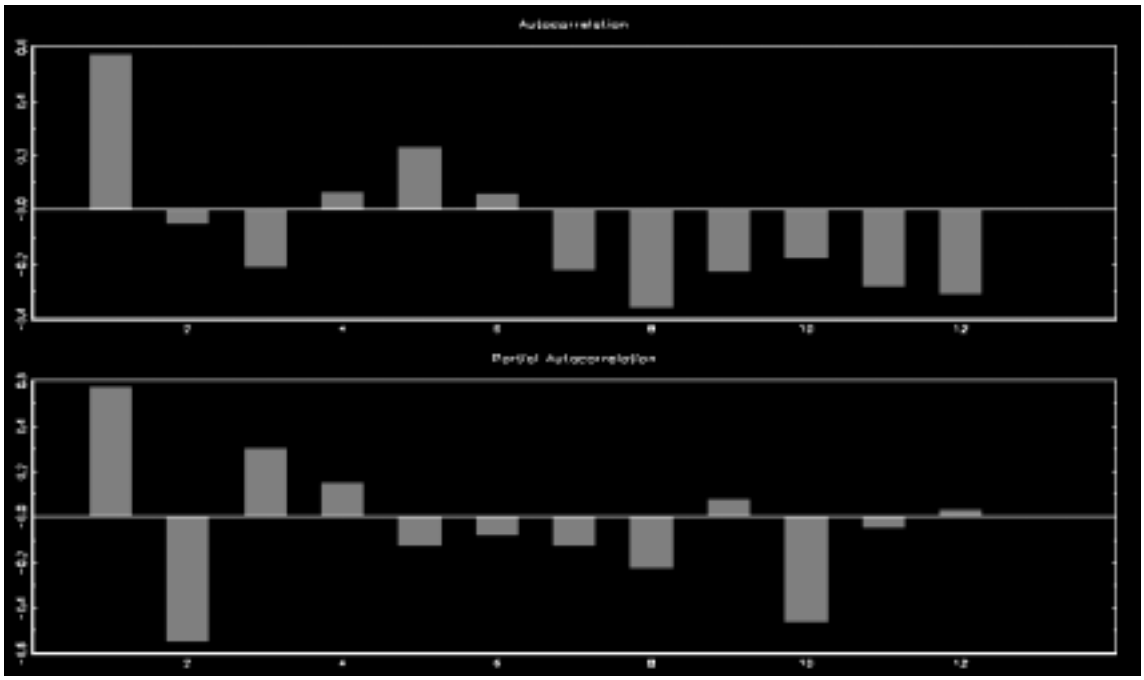
```
l,6)~lagn(pcol,7)~lagn(pcol,8)~lagn(pcol,9)~lagn(pcol,10)~lagn(pcol,11)~lagn(pcol,12)
;
    pmatrix=lowmat(pmatrix)+lowmat(pmatrix)'-eye(13);
    pmatrix=pmatrix[1:12,1:12];
    psmatrix=pmatrix;
    k=1; pac=zeros(12,1);
    do while k<=12;
        psmatrix[.,k]=pcol[2:13];
        pac[k,.]=det(psmatrix[1:k,1:k])/det(pmatrix[1:k,1:k]);
        psmatrix=pmatrix;
        k=k+1;
    endo;

    print "LAG                AC                PAC";;
    axis=seqa(1,1,12);
    print/lz axis~ac~pac;
    library pgraph;
    graphset;
    begwind; window(2,1,0);
    setwind(1);
    title("Autocorrelation");
    bar(axis,zeros(12,1)~ac);
    setwind(2);
    title("Partial Autocorrelation")
    bar(axis,zeros(12,1)~pac);
    endwind;
    retp;
endp;
```

プログラムでは、procdedureの前半では、e_tとe_{t-j}を作成します。それをもとに自己相関係数ACを上の式にもとづいて求めます。なお、vcxではこの場合、２×２のVariance-Covariance行列が計算されますから、その(2,1)または(1,2)の要素だけを計算に用います。この作業を、ｉループで、１から12まで求めます。あらかじめ最初を１として後の部分をすべて０とする列ベクトルpcolの２行目から順に代入していきます。これは上のP行列の１列目に当たります。この列ベクトルを用いて、行列Pの残りの部分を計算します。LMテストのように、さらにループを使って一般形で書いてもよいのですが、さらに数個のループ

を階層的に用いることになりますから、ここでは12ラグ固定で計算することにします。すなわち、P行列の１列目を基準にして、lagnで１列ずつずらせます。ラグ関数を使うと、下の部分は考慮しなくてもいいのですが、上の部分は欠落する部分にドットが入ります。その部分を切るなり、修正するなりしなくてはいけません。ドットの部分を０にしてしまう１つのてっとり早い方法に、lowmatがあります。下三角行列だけを得て、その他の要素に０を入れてくれます。こうしてできた行列と、これをさらに転置させて上三角行列にしたものとを足し合わせます。対角要素が１が２回足しあわされてすべて２となっているはずですから、eye(13)を引きます。さらに12×12の行列にすると求めるP行列となります。偏自己相関係数PACを求めるのには、最終ｋ列目の列をρ_1から始まる列と入れ替えます。それともともとのP行列とのdeterminantの比の計算、すなわち、クレイマーのルールでPACが求まります。この作業を１から12までｋループで繰り返しをやらせています。その後、ラグ数とACとPACを表示して、(2,1)の２分割のグラフで表示させています。分割グラフはI／Oの節のグラフ表示(3)のところでやりましたように、begwindで始まりendwindで終わります。分割はwindow(2,1,0)で２行１列になります。なお、第３要素の値は１でも０でもこの場合は重なりあいませんから任意です。上のグラフはsetwind(１)で指定されます。次のグラフはsetwind(2)と上のようにするか、またはnextwindと指定します。そして、retpには何も返さなくて、procedureのリターンの数のところも０になっています。この最後のretpはリターンが０の場合、別にあってもなくてもかまいません。これらの画面表示とグラフ表示は、リターンとは関係のない、procedure内部で行なわれるものです。

グラフ表示



画面表示

LAG	AC	PAC
1	0.56570204	0.56570204
2	-0.049131509	-0.54288312
3	-0.20623685	0.29975625
4	0.059664836	0.14693002
5	0.22462201	-0.12163165
6	0.058698634	-0.078372911
7	-0.21948295	-0.12020132
8	-0.35702212	-0.22028321
9	-0.22387284	0.071793599
10	-0.17547278	-0.46167669
11	-0.2772428	-0.044944533
12	-0.30358287	0.02739833

Newey-West Covariance Estimator

WhiteのHCSEと同じような考え方で、OLSによる係数βを変えないで保持したままでNewey-Westによる次のようなCovariance Matrixを計算して、係数のそれぞれのSEを求めるものです。

$$Var(\beta) = (X'X)^{-1}\left[\sum_{t=1}^{n} e_t^2 x_t x_t^{'} + \sum_{i=1}^{q}\sum_{t=i+1}^{n} w_i e_t e_{t-i}(x_t x_{t-i}^{'} + x_{t-i} x_i^{'})\right](X'X)^{-1}$$

ここで　$w_i = 1 - \dfrac{i}{q+1}$

上の式の大きな四角括弧の中の計算を下のプログラムでは、e.*xの転置行列を用いて繰り返し計算で後半部分を計算した上で、前半部分を足し合わせる形で、smatrixを計算しています。なお、さらに内側の丸括弧の中の２項は通常は等しいですから２倍して使います。

プログラム

```
new; cls;
load data[40,3]=d:datafile13.txt;
Y=data[2:40,1]; L=data[2:40,2]; K=data[2:40,3];
y=ln(Y); x=ones(39,1)~ln(L)~ln(K);

{b,se,t,p,varb,nwse,tnw,pnw,nwvarb,df,r2,r2bar}=neweywest(y,x,1);
print "OLS with SE";
print "          beta          se          beta/se          p";;
```

```
print b~se~t~p;
print "Original Covariance Estimator" varb;
print "OLS with Newey-West SE";
print "              beta                nwse          beta/nwse                p";;
print b~nwse~tnw~pnw;
print "Newey-West Covariance Estimator" nwvarb;
print "     R2=" r2; print "R2bar=" r2bar;

proc(12)=neweywest(y,x,nlag);
    local n,k,df,b,e,s2hat,varb,se,t,p,pnw,r2,r2bar,h,smatrix,i,w,nwvarb,nwse,tnw;
    n=rows(x); k=cols(x); df=n-k;
    b=inv(x'x)*x'y;
    e=y-x*b;
    s2hat=e'e/(n-k);
    varb=s2hat*inv(x'x);
    se=sqrt(diag(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);

    h=(e.*x)';
    smatrix=zeros(k,k);
    i=1;
    do while i<=nlag;
        w=1-i/(nlag+1);
        smatrix=smatrix+2*w*(h[.,i+1:n]*h[.,1:n-i]');
        i=i+1;
    endo;
    smatrix=h[.,1:n]*h[.,1:n]'+smatrix;
    nwvarb=inv(x'x)*smatrix*inv(x'x);

    nwse=sqrt(diag(nwvarb));
    tnw=b./nwse;
    pnw=2*cdftc(abs(tnw),df);
    retp(b,se,t,p,varb,nwse,tnw,pnw,nwvarb,df,r2,r2bar);
```

```
endp;
```

画面表示

```
OLS with SE
            beta                se           beta/se                 p
      -3.9377145       0.23699929       -16.614879   1.8332298e-018
       1.4507860       0.083228446        17.431372   3.9260341e-019
      0.38380813       0.048017824        7.9930345   1.7129662e-009
Original Covariance Estimator
      0.056168663      -0.018805540      0.0092666405
     -0.018805540      0.0069269742     -0.0038019678
     0.0092666405     -0.0038019678      0.0023057114
OLS with Newey-West SE
            beta               nwse          beta/nwse                 p
      -3.9377145        0.30408788       -12.949265   4.1003916e-015
       1.4507860        0.11276231        12.865877   4.9699434e-015
      0.38380813        0.064882436        5.9154395   9.0143451e-007
Newey-West Covariance Estimator
      0.092469438      -0.033431796       0.017864672
     -0.033291740       0.012715339     -0.0071628394
      0.017710153     -0.0071341890      0.0042097304
   R2=       0.99462716
R2bar=       0.99432866
```

上の計算のほとんどの部分は系列相関のところで、その修正で行なったWhiteのHCSEの計算と同じです。コアになるsmatrixの計算では、(h[.,i+1:n]*h[.,1:n-i]')の計算をｉが１からラグ数だけ繰り返すことによって、内側のΣが計算され、もともとｈはe.*xが転置されたものですから、そのクロスプロダクトをとることによって、結果として通常のe'eの計算のように外側のΣの計算を行なったことになります。（ただし、このプログラムではラグを１に設定してありますので、ループは１回しかまわっていません。ｎを入れるとｎ回ループが回ります。そしてループの外で、h[.,1:n]*h[.,1:n]'を足し合わせます。こちらも転置したもののクロスプロダクトですから、もともとの列のｔ＝１からｎまでの２乗和と同じことになります。これに両側からX'Xのinverseをかけてやるとと Newey-WestのVar(β)ができます。あとの計算は、通常のOLSと同じ手順です。R2関連の統計量は、この修正されたVar(β)には関係がありませんから、OLSのものと同じ結果になります。この場合の係数βは、言うまでもなく定義にしたがって、OLSの計算と同じものです。Var(β)が変更されますから、

各変数に対するSEもかわり、係数とSEの割合のｔ値に相当する部分もかわることになります。なお、これと似たルーチンは単位根のテストのPP Testのところでも使われます。