GAUSS CORNER

[1] STARTUP

(1) OBTAINING RESEARCH ACCOUNT

● ASU has a site licence for Unix version of GAUSS. Your GAUSS programs will work for both Unix and Windows95 without any modification, as long as you write the programs using the lower-case letters.

● To access Unix-GAUSS, you need to subscribe the research cluster at ASU. Do one of the followings.

- By Internet

1. Go to www.asu.edu
2. Click the "Let's Go" button.
3. Type your ASURITE user ID and click on "Continue".
4. Type your PASSWORD and click on "Continue".
5. Go to "Services you are not subscribe to" and click on "Research Cluster". Then, click on "Subscribe to Selected Service".

It would take several hours before your account becomes activated.

(2) By Walk

Go to Computer Commons (First floor, Lobby)

(2) CONFIGURING YOUR AFS ACCOUNT

● Once you can access the research cluster, you need to modify the environment of your AFS account so that you can use GAUSS. Do the followings.

● The instruction below was prepared by Chris Gadarowski at Finance Department. Many thanks to him!!!

1. Go to my Web page and download three Gauss setup programs:

   gauss.cfg
   gauss.start
   mult.prg

2. Using WS_FTP, go to your AFS account.

3. Create in your account a subdirectory called “gauss”. Do not use capital letters.
4. Upload the above three files to the directory “gauss”, and exit WS_FTP.
3. Using Telnet, go to “research.asu.edu”. Then do the followings.
4. Type your login name and password. Push the RETURN button. Push the button again. Then, you will see “research>”.
5. Go to the 'gauss' subdirectory by typing 'cd ~/gauss'.
6. Type the following while in the 'gauss' subdirectory:

    chmod +x gauss.cfg
    chmod +x gauss.start

7. Go back to the root directory ('cd ~').
8. Add the following lines to your '.profile' file: e.g. type 'pico .profile' (do not forget the '.'). Then, go to the bottom of the file and enter the following:

    # set up references to gauss
    export GAUSS=/afs/asu.edu/dist/gauss
    export GAUSSHOME=/afs/asu.edu/dist/gauss
    export GAUSS_CFG=$HOME/gauss
    export LSHOST=research2.asu.edu
    alias -x rgauss="$GAUSS/gauss -v "
    (Don't forget the closing " in the last line.)

9. Logout of the account then log back onto the research cluster.
   (It is a common mistake to try to run gauss from some other cluster, e.g. stats or general.)
10. Go to the 'gauss' subdirectory. Type 'rgauss'. this should put you into interactive mode and you will see the gauss prompt (gauss). (If you get an error message about 'cannot execute', most likely you are not on the research cluster.)
11. Type 'run mult.prg'. This runs the small test file. If you can see some output, it means that your setup is completed.
12. If you want to quit GAUSS, type “quit” in front of the gauss prompt (gauss).
13. When you want to run other GAUSS programs, upload them (and data) to the gauss directory using WS_FTP. Then, do the same thing as in Step 10 using Telnet. If you want to modify the program, use the Pico editor. (For example, if you type ‘pico mult.prg” in the research prompt, Research>, you can see and eidt the content of the file. Remember that you should quit GAUSS to use the pico editor.
14. Good luck!!!

[2] SOME TIPS FOR GAUSS

(1) Procedure (Defining functions)

[Basic Structure]

```
proc f(a,b,c) ; @a,b,c (input; scalar, vector or matrix)@
local v,u,x ; @declare local variables @
    :
retp(v+u+x) ; @output @
endp ;
```

- All of the variables defined in proc are local.
- Global variables can be used.

[EX 1]

```
proc f(b) ;
retp(1+b+b^2) ;
endp ;

c = f(1); @ c = 3 @
d = f(2); @ d = 7 @
```

[EX 2] Multiple Ouput

```
proc(2) = f(b) ;
retp(1+b,1+2*b^2) ;
endp ;

{a,c} = f(1); @a = 2, c = 3.@
```

(2) Minimization

[Basic Structure]

```
library optmum;
#include optmum.ext;
optset;

proc f(b) ; @b = parameter vector @
     :     @ f(b) = function to be minimized @
retp(...) ;
endp     ;

b0 = {1,2,...,0}; @ initial value of b @

__title = “GMM” ; @ Writing title for output @
__opgtol = 0.00001 ; @Controling tolerance rate @
__opstmth = “bfgs,half”; @algorithm @
__output = 20; @ Control outpur files @

{b, func, grad, retcode} = optprt(optmum(&f,b0)) ;

@ b = the value of b minimizing f(b)@
@ retcode = 0 (normal convergence) @
@ retcode ≠ 0 (basd news)         @
```

(3) Gosub

```
     :
Gosub weight ;
     :
     :
end;
weight:
     :
```

```
return ;
```

- weight: a label for a subroutine.

(4) Computing gradient:

```
proc polla(b) ; @ b must be a vector @
local jorge,slava,... ;
    :
retp(...) ;
endp ;

grad = gradp(&polla,b) ;
```

- For example, polla(b) = $g_T(\theta)$ in GMM and grad = $G_T(\theta)$.

[3] GAUSS PROGRAMS FOR GMM

(1) GMM_1.PRG:

```
/*
** GMM FOR SINGLE EQUATION
** Using Newey-West and Andrews Covariance Matrix
*/

new ;

@ LOADING DATA @
    load dat[923,17] = mwemp.db ; @ Data on employed married women @

@ OPEN OUTPUT FILE @
    output file = gmm_1.out reset ;

@ DEFINE VARIABLES @
    lrate   = dat[.,1]  ;
    edu     = dat[.,2]  ;
    urb     = dat[.,3]  ;
    minor   = dat[.,4]  ;
    age     = dat[.,5]  ;
    tenure  = dat[.,6]  ;
    expp    = dat[.,7]  ;
    regs    = dat[.,8]  ;
    occw    = dat[.,9]  ;
    occb    = dat[.,10] ;
    indumg  = dat[.,11] ;
    indumn  = dat[.,12] ;
    unionn  = dat[.,13] ;
    unempr  = dat[.,14] ;
    lofinc  = dat[.,15] ;
    hwork   = dat[.,16] ;
    kids5   = dat[.,17] ;
    lhwork  = ln(hwork+1) ;

@ Define # of observations @
   n = rows(dat) ;
```

```
@  DEFINE EQUATION @
   proc eqq(b)  ;
   local equ    ;

      equ = lhwork - b[1]*ones(n,1) - b[2]*lrate ;

   retp(equ) ;
   endp      ;

@   DEFINE INSTRUMENTAL VARIABLES  @
    iv = ones(n,1)~age~expp~edu;

@   DEFINE INITIAL VALUE @
    bb = {1,1} ;

@   Choose np = 1 for Newey-West;
           np = 0 for Andrews        @
    np = 1 ;

@   Specify Bandwidth @
    bandw = 0 ; @ if bandw = 0, White-Correction @

@   iter = 0 for two-step GMM ; iter = 1 for iterative GMM @
    iter = 1 ;


/*
**  DO NOT CHANGE FROM HERE
*/

@   Procedure for Newey-West   @

proc nw(uv,l) ;
local j,omeo,n,w,ome  ;
    n    = rows(uv)  ;
    omeo = uv'uv/n   ;
    j = 1; do while j <= l  ;
        w    = 1 - j/(l+1)       ;
        ome  = uv[j+1:n,.]'uv[1:n-j,.]/n ;
        omeo = omeo + w*(ome+ome')       ;
```

```
    j = j + 1 ; endo ;
retp(omeo) ; endp    ;

@   Procedure for Andrews (1991) @

proc quadk(z) ;
local rr ;
  rr = ( 3/(6*pi*z/5)^2 )
        *( 5*sin(6*pi*z/5)/(6*pi*z) - cos(6*pi*z/5) ) ;
retp(rr) ; endp     ;

proc andrews(uv,l) ;
local j,omeo,n,w,ome  ;
    n    = rows(uv)  ;
    omeo = uv'uv/n   ;
    j = 1; do while j <= n-1 ;
        w    = j/(l+1)                    ;
        ome  = uv[j+1:n,.]'uv[1:n-j,.]/n  ;
        omeo = omeo + quadk(w)*(ome+ome') ;
    j = j + 1 ; endo ;
retp(omeo) ; endp       ;

library optmum; #include optmum.ext; optset ;

/* FIRST ROUND */

@   Creating weighting matrix @

    w = invpd(iv'iv) ;

proc  f(b) ;
local m    ;
    m = iv'eqq(b) ;
retp(  m'w*m  ) ; endp  ;

@ starting values @

b0 = bb  ;

__title = "FIRST-STAGE GMM ";
```

```
_opgtol = 1e-4;
_opstmth = "BFGS HALF";
__output = 0 ;

{b,func,grad,retcode} = optprt(optmum(&f,b0)) ;

initb = b ;

/* SECOND ROUND */

cre = 10 ; do while cre >= 0.1e-4 ;
initb = initb ;

@ CREATING WEIGHTING MATRICES @

    if np == 1 ; gosub weight1 ;
    else       ; gosub weight2 ;
    endif      ;

library optmum; #include optmum.ext; optset ;

                     /* Second Round GMM */

proc  f1(b) ;
local m    ;
    m = iv'eqq(b) ;
retp(  m'w*m ) ; endp  ;

@ starting values @

b0 = initb ;

__title = "SECOND-STAGE GMM ";
_opgtol = 1e-4;
_opstmth = "BFGS HALF";
__output = 0 ;

{b,func,grad,retcode} = optprt(optmum(&f1,b0)) ;

cre = iter*(initb-b)'(initb-b) ;
```

```
initb = b ;
endo      ;

/* CONPUTING STANDARD ERRORS */

format /rd 12,4 ;

proc mom(b)     ;
retp(iv'eqq(b)) ; endp            ;

gra = gradp(&mom,b) ;
cov = invpd(gra'w*gra) ;
se  = sqrt(diag(cov))  ;
tst = b./se  ;

"" ;
"GMM RESULTS" ;
"" ;
"      coeff.      std. err.      t-st " ;
b~se~tst     ;

/*  HANSEN TEST */

df = cols(iv) - rows(b) ;
"" ;
"J TEST, DF, P-Val =" func  df  cdfchic(func,df) ;

end ; @ This means no longer using subroutines  @

output off ;

weight1:
    w  = invpd( n*nw(iv.*eqq(initb),bandw) ) ;
    return ;
weight2:
    w  = invpd( n*andrews(iv.*eqq(initb),bandw) ) ;
    return ;
```

[OUTPUT]

```
==============================================================================
                                FIRST-STAGE GMM
==============================================================================
 OPTMUM Version 3.1.4                                  2/17/00   9:25 pm
==============================================================================
return code =    0
normal convergence

Value of objective function        8.235389

Parameters    Estimates    Gradient
----------------------------------------
P01              3.1998        0.0002
P02             -0.1747        0.0004

Number of iterations       14
Minutes to convergence     0.00183

==============================================================================
                                SECOND-STAGE GMM
==============================================================================
 OPTMUM Version 3.1.4                                  2/17/00   9:25 pm
==============================================================================
return code =    0
normal convergence

Value of objective function       26.570387

Parameters    Estimates    Gradient
----------------------------------------
P01              3.2146        0.0003
P02             -0.1738        0.0004

Number of iterations       14
Minutes to convergence     0.00267
```

```
================================================================================
                                SECOND-STAGE GMM
================================================================================
 OPTMUM Version 3.1.4                                    2/17/00   9:25 pm
================================================================================
return code =    0
normal convergence

Value of objective function       26.686095

Parameters    Estimates    Gradient
----------------------------------------
P01              3.2167       -0.0002
P02             -0.1744       -0.0003

Number of iterations       15
Minutes to convergence     0.00267

GMM RESULTS

      coeff.      std. err.      t-st

      3.2167       0.1445      22.2558
     -0.1744       0.0862      -2.0241

J TEST, DF, P-Val =     26.6861       2.0000       0.0000
```

(2) GMM_2.PRG:

```
/*
** GMM FOR TWO EQUATIONS
** Using Newey-West and Andrews Covariance Matrix
*/

new ;

@ LOADING DATA @
    load dat[923,17] = mwemp.db ; @ Data on employed married women @

@ OPEN OUTPUT FILE @
    output file = gmm_2.out reset ;

@ DEFINE VARIABLES @
    lrate   = dat[.,1]  ;
    edu     = dat[.,2]  ;
    urb     = dat[.,3]  ;
    minor   = dat[.,4]  ;
    age     = dat[.,5]  ;
    tenure  = dat[.,6]  ;
    expp    = dat[.,7]  ;
    regs    = dat[.,8]  ;
    occw    = dat[.,9]  ;
    occb    = dat[.,10] ;
    indumg  = dat[.,11] ;
    indumn  = dat[.,12] ;
    unionn  = dat[.,13] ;
    unempr  = dat[.,14] ;
    lofinc  = dat[.,15] ;
    hwork   = dat[.,16] ;
    kids5   = dat[.,17] ;
    lhwork  = ln(hwork+1) ;

@   Define # of observations in GMM  @
    n = rows(dat) ;

@  DEFINE EQUATIONS @
```

```
proc eqq1(b) ;
local equ1   ;

   equ1 = lrate-b[1]*lhwork-b[2]*ones(n,1)-b[3]*tenure-b[4]*edu ;

retp(equ1) ;
endp       ;

proc eqq2(b) ;
local equ2   ;

   equ2 = lhwork-b[5]*lrate-b[6]*ones(n,1)-b[7]*kids5-b[8]*lofinc ;

retp(equ2) ;
endp       ;

@   DEFINE INSTRUMENTAL VARIABLES FOR EACH EQUATION  @
    iv1 = ones(n,1)~tenure~kids5~edu~lofinc ;
    iv2 = ones(n,1)~tenure~kids5~edu~lofinc ;

@   DEFINE INITIAL VALUE @
    bb = { 0, 0, 0, 0, 0, 0, 0, 0 } ;

@   Choose np = 1 for Newey-West;
           np = 0 for Andrews        @
    np = 1 ;

@   Specify Bandwidth @
    bandw = 0 ; @ if bandw = 0, White-Correction @

@   iter = 0 for two-step GMM; iter = 1 for iterative GMM @
    iter = 0 ;

/*
**  DO NOT CHANGE FROM HERE
*/

          :
          :
```

[OUTPUT]

```
================================================================================
                                FIRST-STAGE GMM
================================================================================
 OPTMUM Version 3.1.4                                  2/17/00   9:55 pm
================================================================================
return code =    0
normal convergence

Value of objective function        4.305843

Parameters    Estimates    Gradient
-----------------------------------------
P01              1.1975        0.0000
P02             -3.1597        0.0000
P03              0.0201        0.0001
P04              0.1003        0.0000
P05             -0.1623        0.0000
P06              2.2177        0.0000
P07              0.0541        0.0000
P08              0.0959        0.0000

Number of iterations       60
Minutes to convergence     0.04767

================================================================================
                                SECOND-STAGE GMM
================================================================================
 OPTMUM Version 3.1.4                                      2/17/00
9:55 pm
================================================================================
return code =    0
normal convergence

Value of objective function        6.596957

Parameters    Estimates    Gradient
-----------------------------------------
P01              1.7093       -0.0000
```

```
P02             -4.6104       -0.0000
P03              0.0234       -0.0003
P04              0.0961       -0.0006
P05             -0.1284       -0.0001
P06              2.2242       -0.0000
P07             -0.0127        0.0004
P08              0.0915       -0.0002

Number of iterations       67
Minutes to convergence     0.05400

GMM RESULTS

      coeff.      std. err.      t-st

      1.7093       0.3980       4.2948
     -4.6104       1.2309      -3.7454
      0.0234       0.0040       5.8598
      0.0961       0.0121       7.9114
     -0.1284       0.0865      -1.4841
      2.2242       0.2714       8.1955
     -0.0127       0.0210      -0.6063
      0.0915       0.0327       2.7980

J TEST, DF, P-Val =      6.5970       2.0000       0.0369
```

(3) GMM_3.PRG:

```
/*
** GMM FOR THTREE EQUATIONS
** Using Newey-West or Andrews Covariance Matrix
*/

NEW ;
@ LOADING DATA @
    load dat[923,17] = mwemp.db ; @ Data on employed married women @

@ OPEN OUTPUT FILE @
    output file = gmm_3.out reset ;

@ DEFINE VARIABLES @
    lrate   = dat[.,1]  ;
    edu     = dat[.,2]  ;
    urb     = dat[.,3]  ;
    minor   = dat[.,4]  ;
    age     = dat[.,5]  ;
    tenure  = dat[.,6]  ;
    expp    = dat[.,7]  ;
    regs    = dat[.,8]  ;
    occw    = dat[.,9]  ;
    occb    = dat[.,10] ;
    indumg  = dat[.,11] ;
    indumn  = dat[.,12] ;
    unionn  = dat[.,13] ;
    unempr  = dat[.,14] ;
    lofinc  = dat[.,15] ;
    hwork   = dat[.,16] ;
    kids5   = dat[.,17] ;
    lhwork  = ln(hwork+1) ;

@   define # of observations in GMM  @
    n = rows(dat) ;

@  DEFINE EQUATIONS @
```

```
   proc eqq1(b) ;
   local equ1   ;
      equ1 = lrate-b[1]*lhwork-b[2]*ones(n,1)-b[3]*tenure ;
   retp(equ1) ;
   endp       ;

   proc eqq2(b) ;
   local equ2   ;
      equ2 = lhwork-b[4]*lrate-b[5]*ones(n,1)-b[6]*kids5 ;
   retp(equ2) ;
   endp       ;

   proc eqq3(b) ;
   local equ3   ;
      equ3 = lofinc-b[7]*lrate-b[8]*lhwork - b[9]*unempr  ;
   retp(equ3) ;
   endp       ;

@   DEFINE INSTRUMENTAL VARIABLES FOR EACH EQUATION  @
    iv1 = ones(n,1)~tenure~kids5~unempr ;
    iv2 = ones(n,1)~tenure~kids5~unempr ;
    iv3 = ones(n,1)~tenure~kids5~unempr ;

@   DEFINE INITIAL VALUE @
    bb = { 1, 1, 1, 1, 1, 1, 1, 1, 1 } ;

@   Choose np = 1 for Newey-West;
           np = 0 for Andrews        @
    np = 1 ;

@   Specify Bandwidth @
    bandw = 0 ; @ if bandw = 0, White-Correction @

@ iter = 0 for two-step GMM; iter = 1 for iterative GMM @

    iter = 0 ;
```

[OUTPUT]

```
================================================================================
                                FIRST-STAGE GMM
================================================================================
 OPTMUM Version 3.1.4                                  2/17/00  10:04 pm
================================================================================
return code =    0
normal convergence

Value of objective function       27.656967

Parameters    Estimates    Gradient
----------------------------------------
P01             -0.0700        0.0003
P02              1.7782        0.0001
P03              0.0200       -0.0002
P04              0.2509        0.0016
P05              2.4733        0.0010
P06              0.0582        0.0003
P07              0.7448        0.0005
P08              2.9141        0.0005
P09              1.7224        0.0000

Number of iterations       60
Minutes to convergence     0.05867

================================================================================
                                SECOND-STAGE GMM
================================================================================
 OPTMUM Version 3.1.4                                  2/17/00  10:04 pm
================================================================================
return code =    0
normal convergence

Value of objective function       10.717010

Parameters    Estimates    Gradient
----------------------------------------
P01             -0.2735        0.0005
```

```
P02              2.3681        0.0002
P03              0.0202        0.0004
P04              0.0472       -0.0006
P05              2.8390       -0.0004
P06             -0.0324       -0.0001
P07              0.4926       -0.0002
P08              3.1213       -0.0003
P09             -0.7053       -0.0000

Number of iterations       66
Minutes to convergence     0.06500

GMM RESULTS

      coeff.      std. err.      t-st

     -0.2735       0.3079      -0.8882
      2.3681       0.8931       2.6514
      0.0202       0.0033       6.1114
      0.0472       0.1605       0.2943
      2.8390       0.2697      10.5251
     -0.0324       0.0143      -2.2614
      0.4926       0.5269       0.9349
      3.1213       0.3052      10.2274
     -0.7053       0.9534      -0.7398

J TEST, DF, P-Val =     10.7170       3.0000       0.0134
```

(4) BOOT1.PRG

```
/*
** TESTING MEAN BY BOOTSTRAP
** PREPARED BY S.C. AHN
** FEB. 10, 2000
** DESIGNED TO EXAMINE
** FINITE-SAMPLE SIZE PROPERTIES OF MEAN TESTS
*/

@ CLEAN UP MEMORY @
    new ;

@ OPEN OUTPUT FILE @
    output file = boot1.out reset ;

@ DATA GENERATION @
    sd    = 4    ; @ seed number for random number generation @
    tt    = 10   ; @ number of observation @
    nboot = 1000 ; @ number of bootstrap samples @

@ THE VALUE OF POPULATION MEAN TO TEST @
    mu = 5   ; @ H_0: mu = 5 @

@ X IID FROM CHI(KK) @
    kk = 5                          ; @ degrees of freedom @
    xx = sumc(rndns(kk,tt,sd)^2)  ; @ tt*1 data vector @

@ COMPUTING SAMPLE MEAN AND SAMPLE VARIANCE @

    xb  = meanc(xx)   ; @ sample mean     @
    xs2 = stdc(xx)^2  ; @ sample variance @

@ ASYMPTOTIC CHI_SQUARE WALD TEST FOR H_0: MU = TRUE MU @

    awt  = tt*(xb-mu)^2/xs2  ; @ chi^2 Wald stat.  @
    pawt = cdfchic(awt,1)    ; @ p-value           @
```

```
/* BOOTSTRAP WITH PROB = 1/T FOR ALL OBSERVATIONS */

bootxb = zeros(nboot,1) ;
bootwt = zeros(nboot,1) ;

i = 1 ; do while i <= nboot ;

    cboot=ceil(tt*rndus(tt,1,sd)); @ choosing bootstrap data points @
    bsam = xx[cboot,.] ;

    bxb   = meanc(bsam)  ; @ boot-sample mean     @
    bxs2  = stdc(bsam)^2 ; @ boot-sample variance @

    bwt       = tt*(bxb-xb)^2/bxs2 ; @ boot-chi^2 Wald stat. @
    bootxb[i] = bxb                ; @ saving boot-means     @
    bootwt[i] = bwt                ; @ saving boot-stat      @

i = i + 1 ; endo ;

@ BIAS @

    bias = meanc(bootxb) - xb ;

@ BIAS-CORRECTED ESTIMATE @

    bcxb = xb - bias ;

@ BOOTSTRAP CRITICAL VALUES @

bootwt= sortc(bootwt,1) ;
tc    = nboot           ;
bootc =bootwt[ceil(tc*.99)]|bootwt[ceil(tc*.95)]|bootwt[ceil(tc*.9)];

/* REPORTING RESULTS */

format /rd 12,4 ;

"POINT ESTIMATES " ;
"" ;
"   SAMPLE MEAN  BIAS-COR. MEAN" ;
```

```
xb bcxb;
"";
"ASYMPTOTIC WALD TEST FOR H_0: MU = " mu ;
"";
"    STATISTICS      P-VAL";
awt pawt ;
"" ;
"BOOTSTRAP WALD TEST FOR H_0: MU =  " mu ;
"";
"    STATISTICS    C-VAL-1%     C-VAL-5%     C-VAL-10% ";
awt bootc' ;

output off ;
```

[OUTPUT]

```
POINT ESTIMATES

   SAMPLE MEAN  BIAS-COR. MEAN
      3.7168       3.7134

ASYMPTOTIC WALD TEST FOR H_0: MU =  5.0000

    STATISTICS      P-VAL
      6.9088       0.0086

BOOTSTRAP WALD TEST FOR H_0: MU =   5.0000

    STATISTICS    C-VAL-1%     C-VAL-5%     C-VAL-10%
      6.9088      16.0641       5.1244       3.5424
```

(5) BOOT2.PRG

```
/*
** TESTING MEAN BY BOOTSTRAP **
**    X IID FROM CHI^2(5)    **
**   MONTE CARLO EXERCISE    **
**    PREPARED BY S.C. AHN   **
**       FEB. 10, 2000       */

@ CLEAN UP MEMORY @
    new ;

@ OPEN OUTPUT FILE @
    output file = boot2.out reset ;

@ SET-UP: CAN CHANGE @
   sd   = 1    ; @ initial seed number for random number generation @
   tt   = 10   ; @ number of observation @
   nboot= 1000 ; @ number of bootstrap samples @
   iter  = 1000 ; @ number of simulations @
   mu    = 5    ; @ value of mu under H_0 @

@ INITIAL MATRICES: DO NOT CHANGE @
   abias = zeros(iter,1) ;
   bbias = zeros(iter,1) ;
   amse  = zeros(iter,1) ;
   bmse  = zeros(iter,1) ;
   arej  = zeros(iter,3) ;
   brej  = zeros(iter,3) ;

/* STARTING MONTE CARLO */

j = 1 ; do while j <= iter ;

@ X IID FROM CHI(KK): CAN CHANGE @
  sd = j + 100                 ; @ updating seed number @
  kk = 5                       ; @ degrees of freedom @
  xx = sumc(rndns(kk,tt,sd)^2)  ; @ tt*1 data vector @

/* DO NOT CHANGE FROM HERE */
```

```
@ COMPUTING SAMPLE MEAN AND SAMPLE VARIANCE @
    xb  = meanc(xx)   ; @ sample mean     @
    xs2 = stdc(xx)^2  ; @ sample variance @

    abias[j] = xb - kk    ;
    amse[j] = (xb - kk)^2 ;

@ ASYMPTOTIC CHI_SQUARE WALD TEST FOR H_0: MU = TRUE MU @
   awt  = tt*(xb-mu)^2/xs2        ; @ chi^2 Wald stat.  @

   if awt > 6.63490 ; test1 = 1 ; else ; test1 = 0 ; endif ;
   if awt > 3.84146 ; test5 = 1 ; else ; test5 = 0 ; endif ;
   if awt > 2.70554 ; test0 = 1 ; else ; test0 = 0 ; endif ;

   arej[j,.] = test1~test5~test0 ;

/* BOOTSTRAP WITH PROB = 1/T FOR ALL OBSERVATIONS */

   bootxb = zeros(nboot,1) ;
   bootwt = zeros(nboot,1) ;

i = 1 ; do while i <= nboot ;

   cboot = ceil( tt*rndus(tt,1,sd) ) ; @ choosing bootstrap data @
   bsam  = xx[cboot,.] ;
   bxb   = meanc(bsam)  ; @ boot-sample mean     @
   bxs2  = stdc(bsam)^2 ; @ boot-sample variance @

   bwt       = tt*(bxb-xb)^2/bxs2 ; @ boot-chi^2 Wald stat. @
   bootxb[i] = bxb                ; @ saving boot-means     @
   bootwt[i] = bwt                ; @ saving boot-stat      @

i = i + 1 ; endo ;

@ BIAS @
    bias = meanc(bootxb) - xb ;

@ BIAS-CORRECTED ESTIMATE @

   bcxb = xb - bias ;
```

```
   bbias[j] = (bcxb - kk)   ;
   bmse[j]  = (bcxb - kk)^2 ;

@ BOOTSTRAP CRITICAL VALUES @

bootwt= sortc(bootwt,1) ;
tc    = nboot           ;
bootc = bootwt[ceil(tc*.99)]|bootwt[ceil(tc*.95)]|bootwt[ceil(tc*.9)];

   if awt > bootc[1] ; test1 = 1 ; else ; test1 = 0 ; endif ;
   if awt > bootc[2] ; test5 = 1 ; else ; test5 = 0 ; endif ;
   if awt > bootc[3] ; test0 = 1 ; else ; test0 = 0 ; endif ;

   brej[j,.] = test1~test5~test0 ;

j = j + 1 ; endo ;

/* REPORTING RESULTS */

format /rd 12,4 ;

"BIASES IN POINT ESTIMATES " ;
"" ;
"   SAMPLE MEAN  BIAS-COR. MEAN" ;
meanc(abias) meanc(bbias) ;
"";
"REJECTION RATE OF ASYMPTOTIC WALD TEST FOR H_0: MU = " mu ;
"";
"      at 1%        at 5%        at 10%";
meanc(arej)' ;
"" ;
"BOOTSTRAP WALD TEST FOR H_0: MU =  " mu ;
"";
"      at 1%        at 5%        at 10% ";
meanc(brej)' ;

output off ;
```

[OUTPUT]

```
BIASES IN POINT ESTIMATES

   SAMPLE MEAN  BIAS-COR. MEAN
     -0.0313      -0.0310

REJECTION RATE OF ASYMPTOTIC WALD TEST FOR H_0: MU = 5.0000

      at 1%        at 5%        at 10%
      0.0380       0.0990       0.1600

BOOTSTRAP WALD TEST FOR H_0: MU = 5.0000

      at 1%        at 5%        at 10%
      0.0120       0.0460       0.1000
```

(6) BOOT4.PRG

```
/*
** TESTING EXP(MEAN) = EXP(MU) BY BOOTSTRAP
** X IID FROM CHI^2(3)
** MONTE CARLO EXERCISE TO EXAMINE SIZE PROPERTIES OF TESTS
** PREPARED BY S.C. AHN
** FEB. 10, 2000
*/

@ CLEAN UP MEMORY @
    new ;

@ OPEN OUTPUT FILE @
    output file = boot4.out reset ;

@ SET-UP @
  sd    = 1   ; @ initial seed number for random number generation @
  tt    = 20  ; @ number of observation @
  nboot = 1000   ; @ number of bootstrap samples @
  iter  = 1000   ; @ number of simulations @
  emu   = exp(3) ; @ value of exp(mu) under H_0 @

@ INITIAL MATRICES @
   abias = zeros(iter,1) ;
   bbias = zeros(iter,1) ;
   amse  = zeros(iter,1) ;
   bmse  = zeros(iter,1) ;
   arej  = zeros(iter,3) ;
   brej  = zeros(iter,3) ;

/* STARTING MONTE CARLO */

j = 1 ; do while j <= iter ;

@ X IID FROM CHI(KK) @
   sd = j + 100                  ; @ updating seed number @
   kk = 3                       ; @ degrees of freedom @
   xx = sumc(rndns(kk,tt,sd)^2) ; @ tt*1 data vector @
```

```
@ COMPUTING SAMPLE MEAN AND SAMPLE VARIANCE @
   xb   = meanc(xx)       ; @ sample mean     @
   xs2  = stdc(xx)^2      ; @ sample variance @
   exb  = exp(xb)         ; @ exp(sample mean) @
   exs2 = (exp(xb)^2)*xs2 ; @ asym. var[sqrt(tt)*(exp(xb)-exp(mu))] @
   abias[j] = exb - exp(kk)    ;
   amse[j] = (exb - exp(kk))^2 ;

@ ASYMPTOTIC CHI_SQUARE WALD TEST FOR H_0: MU = TRUE MU @
    awt  = tt*(exb-emu)^2/exs2       ; @ chi^2 Wald stat.  @

   if awt > 6.63490; test1 = 1 ; else ; test1 = 0 ; endif ;
   if awt > 3.84146; test5 = 1 ; else ; test5 = 0 ; endif ;
   if awt > 2.70554; test0 = 1 ; else ; test0 = 0 ; endif ;
   arej[j,.] = test1~test5~test0 ;

/* BOOTSTRAP WITH PROB = 1/T FOR ALL OBSERVATIONS */

bootxb = zeros(nboot,1) ;
bootwt = zeros(nboot,1) ;

i = 1 ; do while i <= nboot ;

cboot = ceil( tt*rndus(tt,1,sd) ) ; @ choosing bootstrap data points @
bsam  = xx[cboot,.] ;
bxb   = meanc(bsam)        ; @ boot-sample mean      @
bxs2  = stdc(bsam)^2       ; @ boot-sample variance  @
ebxb  = exp(bxb)           ; @ exp(boot-sample mean) @
ebxs2 = (exp(bxb)^2)*bxs2  ; @ asym. var[sqrt(tt)*(exp(xb)-exp(mu))] @
bwt       = tt*(ebxb-exb)^2/ebxs2 ; @ boot-chi^2 Wald stat.  @
bootxb[i] = ebxb                  ; @ saving exp(boot-means) @
bootwt[i] = bwt                   ; @ saving boot-stat       @

i = i + 1 ; endo ;

@ BIAS @
    bias = meanc(bootxb) - exb ;

@ BIAS-CORRECTED ESTIMATE @
   bcxb = exb - bias ;
```

```
   bbias[j] = (bcxb - exp(kk))   ;
   bmse[j]  = (bcxb - exp(kk))^2 ;

@ BOOTSTRAP CRITICAL VALUES @
bootwt = sortc(bootwt,1) ;
tc     = nboot           ;
bootc = bootwt[ceil(tc*.99)]|bootwt[ceil(tc*.95)]|bootwt[ceil(tc*.9)];

If awt > bootc[1] ; test1 = 1 ; else ; test1 = 0 ; endif ;
if awt > bootc[2] ; test5 = 1 ; else ; test5 = 0 ; endif ;
if awt > bootc[3] ; test0 = 1 ; else ; test0 = 0 ; endif ;

brej[j,.] = test1~test5~test0 ;

j = j + 1 ; endo ;

/* REPORTING RESULTS */
format /rd 16,4 ;

"BIASES IN POINT ESTIMATES OF EXP(MU)" ;
"" ;
"      EXP(SAMPLE M)  EXP(BIAS-COR. M)" ;
meanc(abias) meanc(bbias) ;
"";
"MSE IN POINT ESTIMATES OF EXP(MU)" ;
"" ;
"      EXP(SAMPLE M)  EXP(BIAS-COR. M)" ;
meanc(amse) meanc(bmse) ;
"" ;
"REJECTION RATE OF ASYMPTOTIC WALD TEST FOR H_0: EXP(MU) =     " emu ;
"";
"           at 1%           at 5%           at 10%";
meanc(arej)' ;
"" ;
"REJECTION RATE OF BOOTSTRAP WALD TEST FOR H_0: exp(MU) =      " emu ;
"";
"           at 1%           at 5%           at 10% ";
meanc(brej)' ;

output off ;
```

[OUTPUT]

```
BIASES IN POINT ESTIMATES OF EXP(MU)

      EXP(SAMPLE M)  EXP(BIAS-COR. M)
          3.0026          -1.7271

MSE IN POINT ESTIMATES OF EXP(MU)

      EXP(SAMPLE M)  EXP(BIAS-COR. M)
        237.6453         105.9099

REJECTION RATE OF ASYMPTOTIC WALD TEST FOR H_0: EXP(MU) = 20.0855

           at 1%            at 5%           at 10%
          0.0890           0.1270           0.1450

REJECTION RATE OF BOOTSTRAP WALD TEST FOR H_0: exp(MU) = 20.0855

           at 1%            at 5%           at 10%
          0.0260           0.0720           0.1150
```

(7) BOOT5.PRG

```
/*
** BOOTSTRAP TWO-STEP GMM FOR SINGLE EQUATION
** IID DATA
** PREPARED BY MIN AHN
** FEB. 14, 2000
*/

new ;

@ LOADING DATA @
    load data[923,17] = mwemp.db ;

@ OPEN OUTPUT FILE @
    output file = boot5.out reset ;

@ NUMBER OF BOOTSTRAP @
    nboot = 200 ;

/**** DO NOT CHANGE THE TWO LINES BELOW ****/

i=0        ;
dat = data ;
goto min   ;
min:

/****        CAN CHANGE BELOW           ****/

@ DEFINE VARIABLES @
    lrate   = dat[.,1]  ;
    edu     = dat[.,2]  ;
    urb     = dat[.,3]  ;
    minor   = dat[.,4]  ;
    age     = dat[.,5]  ;
    tenure  = dat[.,6]  ;
    expp    = dat[.,7]  ;
    regs    = dat[.,8]  ;
    occw    = dat[.,9]  ;
```

```
    occb    = dat[.,10] ;
    indumg  = dat[.,11] ;
    indumn  = dat[.,12] ;
    unionn  = dat[.,13] ;
    unempr  = dat[.,14] ;
    lofinc  = dat[.,15] ;
    hwork   = dat[.,16] ;
    kids5   = dat[.,17] ;
    lhwork  = ln(hwork+1) ;

@ DEFINE # OF OBSERVATION @
   n = rows(data) ;

@  DEFINE EQUATION @

   proc eqq(b)  ;
   local equ    ;

      equ = lhwork - b[1]*ones(n,1) - b[2]*lrate ;

   retp(equ) ;
   endp      ;

@   DEFINE INSTRUMENTAL VARIABLES  @

    iv = ones(n,1)~age~expp~edu;

@   DEFINE INITIAL VALUE @

    bb = {1,1} ;

        :
        :
```

[OUTPUT]

```
ASYMPTOTIC GMM RESULTS
      coeff.      std. err.   abs(t-st)
      3.2146       0.1445      22.2411
     -0.1738       0.0862       2.0164

HANSEN TEST
      J-STAT         DF         P-VAL
     26.6892       2.0000       0.0000

BOOTSTRAP GMM RESULTS
     abs(tst)       C-1%         C-5%         C-10%
     22.2411       2.4996       1.9197       1.5847
      2.0164       2.5724       1.8437       1.5216

BOOTSTRAP HANSEN TEST
      J-stat        C-1%         C-5%         C-10%
     26.6892       8.1842       5.4266       4.3021
```

(8) BOOT6.PRG

```
/*
** BOOTSTRAP TWO-STEP GMM FOR SINGLE EQUATION
** IID DATA
**
** MODEL SPECIFICATION:
**    Y = B[1] + B[2]*X + E with B[1] = B[2] = 1
**    X = Z1 + Z2 + E
**
** THIS MONTE CARLO EXPERIMENT IS DESIGNED TO EXAMINE
** FINITE-SAMPLE SIZE PROPERTIES OF ASYMPTOTIC AND BOOSTRAP GMM
** T TESTS AND HANSEN TESTS.
**
*/

new ;

@ CONTROL VARIABLES: CAN CHANGE @
   iter   = 100   ; @ number of iteration @
   nboot  = 200   ; @ number of bootstrapping @
   sd     = 1     ; @ initial seed number @
   n      = 30    ; @ data size @
   tb     = {1,1} ; @ true parameter vector @

@ SOME INITIAL MATRICES: DO NOT CHANGE @
   attrej = zeros(iter,3*rows(tb)) ;
   ajtrej = zeros(iter,3) ;
   bttrej = zeros(iter,3*rows(tb)) ;
   bjtrej = zeros(iter,3) ;


@ START MONTE CARLO: DO NOT CHANGE @
kk = 1 ; do while kk <= iter ;

@ CREATING DATA: CAN CHANGE @
   sd = sd + kk             ;
   z  = rndns(n,2,sd)       ;
   e  = rndns(n,1,sd)       ;
   x  = (z[.,1]+z[.,2]+e)   ;
```

```
   y  = tb[1]+ tb[2]*x + e  ;
   data = y~x~z             ;

/**** DO NOT CHANGE THE TWO LINES BELOW ****/

i=0        ;
dat = data ;
goto min   ;
min:

/****      CAN CHANGE FROM BELOW       ****/


@ DEFINE VARIABLES: CAN CHANGE @
    y   = dat[.,1]  ;
    x   = dat[.,2]  ;
    z1  = dat[.,3]  ;
    z2  = dat[.,4]  ;

@  DEFINE EQUATION: CAN CHANGE @
   proc eqq(b)  ;
   local equ    ;

      equ = y - b[1]*ones(n,1) - b[2]*x ;

   retp(equ) ;
   endp      ;

@   DEFINE INSTRUMENTAL VARIABLES: CAN CHANGE  @
    iv = ones(n,1)~z1~z2;

@   DEFINE INITIAL VALUE: CAN CHANGE  @
    bb = {1,1} ;
     :
     :
```

[OUTPUT]

```
REJECTION RATES OF ASYMPTOTIC GMM T-TESTS
       C-1%          C-5%          C-10%
      0.0200        0.1000        0.1500
      0.1000        0.1500        0.2000

REJECTION RATE OF ASYMPTOTIC HANSEN TESTS
       C-1%          C-5%          C-10%
      0.0000        0.0300        0.0800

REJECTION RATES OF BOOTSTRAP GMM T-TESTS
       C-1%          C-5%          C-10%
      0.0000        0.0500        0.1000
      0.0600        0.0800        0.1200

REJECTION RATE OF BOOTSTRAP HANSEN TESTS
       C-1%          C-5%          C-10%
      0.0000        0.0100        0.0700
```

[4] GAUSS PROGRAM FOR PANEL DATA ANALYSIS

(1) CR_GLS.PRG (For Within and GLS)

```
/*
**          PROGRAM FOR WITHIN and GLS ESTIMATION OF PANEL DATA
**
**                              WRITTEN BY
**                            SEUNG CHAN AHN
**                       DEPARTMENT OF ECONOMICS
**                         COLLEGE OF BUSINESS
**                           TEMPE, AZ 85287
**
*/

/*
**  Computing within and GLS Estimators
**  With Hausman Tests
*/

new  ;

@  You must locate MGIV.COL in the directory you execute this program
@

   #include mgiv.col ;

@  Open an output file  @

   output file = cr_gls.out reset ;

@  Formatting output file  @

   format /rd 12,4 ;

@   Provide # of ovservations and # of variables @

    nobs = 4165 ;
    nvar = 23   ;
```

```
@   Read Data   @

    load dat[nobs,nvar] = cr.db ;
    @ dat = delif(dat,dat[.,10] .== 1) ; @

@  Define Variables @

    id68  = dat[.,1]  ; @ ID for individuals @
    expp  = dat[.,2]  ; @ years of full-time work experience @
    expp2 = dat[.,3]  ; @ expp squared @
    wks   = dat[.,4]  ; @ weeks worked @
    occ   = dat[.,5]  ; @ OCC = 1 if blue-collar @
    ind   = dat[.,6]  ; @ IND = 1 if manufactoring industry @
    south = dat[.,7]  ; @ SOUTH = 1 if resident in the South @
    smsa  = dat[.,8]  ; @ SMSA = 1 if resident in SMSA @
    ms    = dat[.,9]  ; @ MS = 1 if married @
    fem   = dat[.,10] ; @ FEM = 1 if female @
    unionm= dat[.,11] ; @ UNIONM = 1 if union contract @
    edu   = dat[.,12] ; @ years of schooling @
    blk   = dat[.,13] ; @ BLK = 1 if black @
    wage  = dat[.,14] ; @ hourly wage rate: cents @
@    UNKNOWN  = dat[.,15] ;
    UNKNOWN2 = dat[.,16] ; @
    y76      = dat[.,17] ; @ Y76 = 1 if 1976 @
    y77      = dat[.,18] ;
    y78      = dat[.,19] ;
    y79      = dat[.,20] ;
    y80      = dat[.,21] ;
    y81      = dat[.,22] ;
    y82      = dat[.,23] ;
    lwage    = ln(wage)  ;
    dyr      = y77~y78~y79~y80~y81~y82 ;

@   Define N and T @

    t  = 7   ;
    n  = rows(dat)/t ;

@   Define Dep. Var., Time-varying Reg. and Time-invariant Reg. @
```

```
    rexp  = {0,1,2,3,4,5,6}    ;
    rexp  = ones(n,1) .*. rexp ;
    rexp  = expp - rexp        ;

    yy    = lwage  ;
    xx    = wks~south~smsa~ms~expp2~occ~ind~unionm~dyr ;

@ Exclude year dummy vars. from xx to make pvxx full column @
@ Use xxt for ALT1 test @

    xxt   = wks~south~smsa~ms~expp2~occ~ind~unionm ;

@ Exclude year dummy var. and expp^2 from xx to make amxx full column
@
@ Use xxtt for ALT3 test @

    xxtt  = wks~south~smsa~ms~occ~ind~unionm ;

    zz    = ones(n*t,1)~fem~blk~edu~rexp ;

@ Clearing unnecessary variables @

   clear   id68, expp, expp2, wks, occ, ind, south, smsa, ms, fem,
         unionm, edu, blk, wage, y76, y77, y78, y79, y80, y81, y82,
         lwage, dyr ;



/*
**  From Here, Do Not Change
*/

    clear dat ;

@   Define k and g @

    k  = cols(xx)  ;
    kt = cols(xxt) ;
    g  = cols(zz)  ;
```

```
@ Creating AM and Mean Variables @

    amxxtt= ammat(xxtt,n,t)  ;
    pvxxt = pvmat1(xxt,n,t)   ;
    pvxx  = pvmat1(xx,n,t)    ;
    pvyy  = pvmat1(yy,n,t)    ;

@   creating Deviation-From-Mean Variables  @

    qvxx = qvmat(xx,n,t) ;
    qvyy = qvmat(yy,n,t) ;

@   Within Estimation  @

    wb = invpd(qvxx'qvxx)*(qvxx'qvyy) ;

    we  = qvyy - qvxx*wb       ;
    ssq  = (we'we)/(n*t-n-k)   ;
    wc   = ssq*inv(qvxx'qvxx)  ;
    ws   = sqrt(diag(wc)) ;

    "Within Estimation" ;
    "     Estimate     Std. Err.     T-stat " ;

        wb~ws~(wb./ws)  ;
        "" ;

@   GLS estimation  @

    bx    = xx~zz     ;
    bd    = invpd(bx'bx)*(bx'yy)     ;
    bee   = pvyy - pvmat1(bx,n,t)*bd ;
    ssqbb = (bee'bee)/(n-k-g) ;
    theta = sqrt(ssq/ssqbb)   ;
    ssqaa = (ssqbb-ssq)/t ;

    yystar = yy - (1-theta)*pvyy ;
    xxstar = xx - (1-theta)*pvxx ;
    zzstar = theta*zz ;
```

```
    regstar = xxstar~zzstar ;

    gd  = invpd(regstar'regstar)*(regstar'yystar) ;
    gc  = ssq*invpd(regstar'regstar)              ;
    gs  = sqrt(diag(gc)) ;

    "GLS Estimation" ;
    "     Estimate     Std. Err.     T-stat " ;

        gd~gs~(gd./gs)  ;
        "" ;
    " THETA = " theta   ;
    " SIGE2 = " ssq     ;
    " SIGA2 = " ssqaa   ;
    " S.E.R.= " sqrt(ssq) ;
    "" ;

@ CREATING GLS RESIDUALS @

    gee  = yystar - regstar*gd  ;

@ H TEST FOR W VS. GLS @

    gb   = gd[1:k]        ;
    gbc  = gc[1:k,1:k]    ;
    ht   = (wb-gb)'pinv(wc-gbc)*(wb-gb)  ;
    df   = rank(wc-gbc)   ;

    "Hausman Test, p-val, df =" ht cdfchic(ht,df) df ;

@ J TEST FOR W VS. GLS USING PX ONLY @

    axx = qvxx~pvxxt~zz ;
    abb = invpd(axx'axx)*(axx'gee) ;
    ru2 = (axx*abb)'(axx*abb)/(gee'gee) ;
    alt1 = t*n*ru2     ;
    df   = cols(pvxxt) ;

    "ALT1    Test, p-val, df =" alt1 cdfchic(alt1,df) df ;
```

```
@ J TEST FOR W VS. GLS USING AM Variables @

    axx = qvxx~amxxtt~zz ;

    abb = invpd(axx'axx)*(axx'gee)      ;
    ru2 = (axx*abb)'(axx*abb)/(gee'gee) ;
    alt3 = t*n*ru2      ;
    df   = cols(amxxtt) ;

    "ALT3    Test, p-val, df =" alt3 cdfchic(alt3,df) df ;

output off
```

(Output)

Within Estimation

	Estimate	Std. Err.	T-stat
WKS	0.0007	0.0006	1.1362
SOUTH	0.0031	0.0342	0.0903
SMSA	-0.0419	0.0194	-2.1618
MS	-0.0286	0.0189	-1.5100
EXPP^2	-0.0004	0.0001	-7.3267
OCC	-0.0192	0.0137	-1.3938
IND	0.0208	0.0154	1.3478
UNIONM	0.0295	0.0149	1.9836
YR77	0.1037	0.0090	11.5199
YR78	0.2485	0.0096	25.7857
YR79	0.3628	0.0107	33.9489
YR80	0.4700	0.0121	38.9211
YR81	0.5646	0.0138	41.0263
YR82	0.6687	0.0157	42.5741

GLS Estimation

	Estimate	Std. Err.	T-stat
WKS	0.0009	0.0006	1.5168
SOUTH	-0.0591	0.0203	-2.9033
SMSA	0.0453	0.0153	2.9602
MS	-0.0155	0.0176	-0.8843
EXPP^2	-0.0004	0.0000	-9.1135
OCC	-0.0439	0.0127	-3.4567
IND	0.0284	0.0132	2.1557
UNIONM	0.0435	0.0130	3.3377
YR77	0.1045	0.0090	11.6694
YR78	0.2515	0.0095	26.5855
YR79	0.3676	0.0103	35.6385
YR80	0.4752	0.0115	41.4729
YR81	0.5732	0.0129	44.5341
YR82	0.6800	0.0145	46.8354
C	5.2487	0.0772	68.0177
FE	-0.4238	0.0394	-10.7433
BLK	-0.1513	0.0446	-3.3887
ED	0.0660	0.0045	14.7962
EXP	0.0278	0.0024	11.5994

```
 THETA =       0.2110
 SIGE2 =       0.0229
 SIGA2 =       0.0703
 S.E.R.=       0.1514

Hausman Test, p-val, df =     91.7087       0.0000       8.0000
ALT1    Test, p-val, df =     90.5206       0.0000       8.0000
ALT3    Test, p-val, df =    137.9597       0.0000      49.0000
```

(2) CR_IV.PRG (For HT, AM and BMS)

```
/*
**        PROGRAM FOR HT ESTIMATION OF PANEL DATA
**
**                      WRITTEN BY
**                    SEUNG CHAN AHN
**               DEPARTMENT OF ECONOMICS
**                 COLLEGE OF BUSINESS
**                   TEMPE, AZ 85287
**
*/

/*
**  Computing HT, AM and BMS Estimators
**  With Hausman Tests
*/

   new ;

@ Put MGIV.COL in the directory you execute this program! @

   #include mgiv.col ;

@ Open Output file @

   output file = cr_iv.out reset ;

@ Format output file @

   format /rd 12,4 ;


@   Provide # of ovservations and # of variables @

    nobs = 4165 ;
    nvar = 23   ;

@   Read Data   @
```

```
    load dat[nobs,nvar] = c:\data\cornwell\cr.db ;
@    dat = delif(dat,dat[.,10] .== 1) ; @

@  Define Variables @

    id68  = dat[.,1]  ;
    expp  = dat[.,2]  ;
    expp2 = dat[.,3]  ;
    wks   = dat[.,4]  ;
    occ   = dat[.,5]  ;
    ind   = dat[.,6]  ;
    south = dat[.,7]  ;
    smsa  = dat[.,8]  ;
    ms    = dat[.,9]  ;
    fem   = dat[.,10] ;
    unionm= dat[.,11] ;
    edu   = dat[.,12] ;
    blk   = dat[.,13] ;
    wage  = dat[.,14] ;
    UNKNOWN  = dat[.,15] ;
    UNKNOWN2 = dat[.,16] ;
    y76      = dat[.,17] ;
    y77      = dat[.,18] ;
    y78      = dat[.,19] ;
    y79      = dat[.,20] ;
    y80      = dat[.,21] ;
    y81      = dat[.,22] ;
    y82      = dat[.,23] ;
    lwage    = ln(wage)  ;
    lwks     = ln(wks)   ;
    dyr      = y77~y78~y79~y80~y81~y82 ;

@   Define N and T @

    n  = 4165/7 ;
@    n  = 528    ; @
    t  = 7      ;

@   Define Dep. Var., Time-varying Reg. and Time-invariant Reg. @
```

```
    @ Treating expp as time-invariant @
    rexp = {0,1,2,3,4,5,6}    ;
    rexp = ones(n,1) .*. rexp ;
    rexp = expp - rexp        ;

    yy  = lwage  ;
    xx  = wks~south~smsa~ms~(expp^2)~occ~ind~unionm~dyr ;
    zz  = ones(n*t,1)~fem~blk~edu~rexp ;
    xx1 = wks~south~smsa~ms   ;
    zz1 = ones(n*t,1)~fem~blk  ;

    @ xxt is used for BMS @
    xxt = occ~ind~unionm   ;

/*
**  From Here, Do Not Change
*/
        :
        :
```

(Output)

Within Estimation

	Estimate	Std. Err.	T-stat
WKS	0.0007	0.0006	1.1362
SOUTH	0.0031	0.0342	0.0903
SMSA	-0.0419	0.0194	-2.1618
MS	-0.0286	0.0189	-1.5100
EXPP^2	-0.0004	0.0001	-7.3267
OCC	-0.0192	0.0137	-1.3938
IND	0.0208	0.0154	1.3478
UNIONM	0.0295	0.0149	1.9836
YR77	0.1037	0.0090	11.5199
YR78	0.2485	0.0096	25.7857
YR79	0.3628	0.0107	33.9489
YR80	0.4700	0.0121	38.9211
YR81	0.5646	0.0138	41.0263
YR82	0.6687	0.0157	42.5741

Hausman-Taylor Estimation

	Estimate	Std. Err.	T-stat
WKS	0.0008	0.0006	1.3294
SOUTH	0.0118	0.0293	0.4047
SMSA	-0.0393	0.0193	-2.0372
MS	-0.0258	0.0189	-1.3671
EXPP^2	-0.0004	0.0001	-7.3517
OCC	-0.0192	0.0137	-1.3953
IND	0.0211	0.0154	1.3720
UNIONM	0.0267	0.0148	1.8020
YR77	0.1036	0.0090	11.5110
YR78	0.2486	0.0096	25.8039
YR79	0.3630	0.0107	33.9649
YR80	0.4702	0.0121	38.9392
YR81	0.5649	0.0138	41.0510
YR82	0.6691	0.0157	42.6031
C	3.0065	0.5018	5.9910
FEM	-0.4041	0.0793	-5.0924
BLK	0.0159	0.1068	0.1487
ED	0.2236	0.0404	5.5410
EXPP	0.0416	0.0108	3.8418

THETA = 0.1127
SIGE2 = 0.0229
SIGAA = 0.2545

HAUSMAN TEST FOR HT VS. WITHIN
STATISTIC, P-VAL, DF = 4.7414 0.0934 2.0000

HANSEN TEST FOR HT
STATISTIC, P-VAL, DF = 4.7526 0.0929 2.0000

AM Estimation

	Estimate	Std. Err.	T-stat
	0.0008	0.0006	1.2980
	-0.0074	0.0286	-0.2602
	-0.0249	0.0186	-1.3371
	-0.0266	0.0186	-1.4288
	-0.0004	0.0001	-7.3988
	-0.0204	0.0137	-1.4867
	0.0214	0.0154	1.3905
	0.0287	0.0148	1.9418
	0.1036	0.0090	11.5120
	0.2487	0.0096	25.8230
	0.3633	0.0107	34.0114
	0.4703	0.0121	38.9861
	0.5654	0.0137	41.1268
	0.6698	0.0157	42.6947
	3.9719	0.3778	10.5129
	-0.4052	0.0737	-5.4948
	-0.0657	0.0920	-0.7133
ED	0.1540	0.0273	5.6497
	0.0376	0.0076	4.9480

HAUSMAN TEST FOR AM VS. WITHIN

Statistic, p-val, df =	17.2340	0.0278	8.0000

HAUSMAN TEST FOR AM VS. HT

Statistic, p-val, df =	12.4926	0.0518	6.0000

HANSEN TEST FOR AM

STATISTIC, P-VAL, DF =	25.3809	0.4975	26.0000

EHS TEST FOR AM VS. HT

STATISTIC, P-VAL, DF =	20.6283	0.6605	24.0000

BMS Estimation

	Estimate	Std. Err.	T-stat
	0.0008	0.0006	1.3018
	-0.0121	0.0284	-0.4259
	-0.0213	0.0184	-1.1584
	-0.0273	0.0186	-1.4692
	-0.0004	0.0001	-7.3910
	-0.0218	0.0137	-1.5915
	0.0211	0.0153	1.3719
	0.0278	0.0148	1.8857
	0.1035	0.0090	11.5051
	0.2487	0.0096	25.8207
	0.3632	0.0107	34.0124
	0.4702	0.0121	38.9866
	0.5653	0.0137	41.1339
	0.6696	0.0157	42.7075
	4.2077	0.3340	12.5991
	-0.4007	0.0725	-5.5262
	-0.0915	0.0880	-1.0394
ED	0.1352	0.0224	6.0360
	0.0380	0.0068	5.6293

Hausman Test for BMS vs. Within			
Statistic, p-val, df =	19.2012	0.0138	8.0000
Hausman Test for BMS vs. AM			
Statistic, p-val, df =	4.4729	0.6130	6.0000
HANSEN TEST FOR BMS			
STATISTIC, P-VAL, DF =	33.9117	0.8639	44.0000
EHS TEST FOR BMS VS. AM			
STATISTIC, P-VAL, DF =	20.6283	0.9696	18.0000

(3) CR_MGIV.PRG (MGIV for Within, GLS, HT, AM and BMS)

```
/*
**          PROGRAM FOR MGIV ESTIMATION OF PANEL DATA
**
**                          WRITTEN BY
**                       SEUNG CHAN AHN
**                   DEPARTMENT OF ECONOMICS
**                     COLLEGE OF BUSINESS
**                       TEMPE, AZ 85287
**
*/

/*
**  COMPUTING MGIV and GMM FOR HT, AM AND BMS ESTIMATOR
**  COMPUTING GMM USING HT, AM AND BMS INSTRUMENTS
**  HAUSMAN TESTS
**  HANSEN TESTS
**  EHS TESTS
**
*/

   new ;

@  Locate MGIV.COL in the directory you execute this program @

   #include mgiv.col ;

@  Open output file @

   output file = cr_mgiv.out reset ;

@  Format output file @

   format /rd 12,4 ;

@   Provide # of ovservations and # of variables @

    nobs = 4165 ;
    tim  = 7    ;
```

```
    nvar = 23   ;

@   Read Data   @

    load dat[nobs,nvar] = c:\data\cornwell\cr.db ;
   @ dat = delif(dat,dat[.,10] .== 1) ; @

@  Define Variables @

    id68  = dat[.,1]  ;
    expp  = dat[.,2]  ;
    expp2 = dat[.,3]  ;
    wks   = dat[.,4]  ;
    occ   = dat[.,5]  ;
    ind   = dat[.,6]  ;
    south = dat[.,7]  ;
    smsa  = dat[.,8]  ;
    ms    = dat[.,9]  ;
    fem   = dat[.,10] ;
    unionm= dat[.,11] ;
    edu   = dat[.,12] ;
    blk   = dat[.,13] ;
    wage  = dat[.,14] ;
    UNKNOWN  = dat[.,15] ;
    UNKNOWN2 = dat[.,16] ;
    y76      = dat[.,17] ;
    y77      = dat[.,18] ;
    y78      = dat[.,19] ;
    y79      = dat[.,20] ;
    y80      = dat[.,21] ;
    y81      = dat[.,22] ;
    y82      = dat[.,23] ;
    lwage    = ln(wage)  ;
    lwks     = ln(wks)   ;
    dyr      = y77~y78~y79~y80~y81~y82 ;

@   Define N and T @

    n  = nobs/tim ;
   t  = tim      ;
```

```
@   Define Dep. Var., Time-varying Reg. and Time-invariant Reg. @

    @ Treating expp as time-invariant @
    rexp = {0,1,2,3,4,5,6}    ;
    rexp = ones(n,1) .*. rexp ;
    rexp = expp - rexp        ;

    yy  = lwage  ;
    xx  = wks~south~smsa~ms~(expp^2)~occ~ind~unionm~dyr ;
    zz  = ones(n*t,1)~fem~blk~edu~rexp ;
    xx1 = wks~south~smsa~ms   ;
    zz1 = ones(n*t,1)~fem~blk ;

    @ xxt is used for BMS @
    xxt = occ~unionm~ind ;

/*
**  From Here, Do Not Change
*/
```

(Output)

Kiefer's Within Estimation

	Estimate	Std. Err.	T-stat
WKS	0.0006	0.0005	1.1759
SOUTH	0.0119	0.0376	0.3153
SMSA	-0.0347	0.0201	-1.7231
MS	-0.0339	0.0196	-1.7276
EXPP^2	-0.0005	0.0001	-5.7818
OCC	-0.0158	0.0126	-1.2561
IND	0.0262	0.0146	1.7931
UNIONM	0.0101	0.0140	0.7226
YR77	0.1052	0.0060	17.6375
YR78	0.2525	0.0108	23.2896
YR79	0.3691	0.0128	28.7942
YR80	0.4782	0.0148	32.2853
YR81	0.5752	0.0181	31.7280
YR82	0.6815	0.0214	31.8219

Kiefer's Within Estimation (HETERO ADJUSTED)

	Estimate	Std. Err.	T-stat
WKS	0.0006	0.0007	0.8973
SOUTH	0.0119	0.0734	0.1617
SMSA	-0.0347	0.0285	-1.2168
MS	-0.0339	0.0216	-1.5648
EXPP^2	-0.0005	0.0001	-5.5759
OCC	-0.0158	0.0174	-0.9073
IND	0.0262	0.0188	1.3933
UNIONM	0.0101	0.0189	0.5354
YR77	0.1052	0.0063	16.6378
YR78	0.2525	0.0114	22.2325
YR79	0.3691	0.0134	27.5679
YR80	0.4782	0.0159	30.0244
YR81	0.5752	0.0205	28.0584
YR82	0.6815	0.0246	27.7005

Hausman-Taylor MGIV Estimation

	Estimate	Std. Err.	T-stat
WKS	0.0007	0.0005	1.3865
SOUTH	0.0186	0.0306	0.6083
SMSA	-0.0315	0.0203	-1.5504
MS	-0.0289	0.0197	-1.4658
EXPP^2	-0.0005	0.0001	-5.7398
OCC	-0.0163	0.0127	-1.2841
IND	0.0267	0.0147	1.8130
YR77	0.0063	0.0141	0.4448
	0.1051	0.0060	17.3800
	0.2527	0.0111	22.8415
	0.3693	0.0130	28.4359
	0.4784	0.0150	31.9356
	0.5755	0.0184	31.3390
	0.6819	0.0216	31.5173
C	3.0190	0.4739	6.3709
FEM	-0.4040	0.0719	-5.6209
BLK	0.0149	0.0975	0.1531
ED	0.2208	0.0379	5.8243
EXPP	0.0444	0.0101	4.3735

Hausman-Taylor MGIV Estimation (HETERO ADJUSTED)

	Estimate	Std. Err.	T-stat
WKS	0.0007	0.0007	1.0786
SOUTH	0.0186	0.0515	0.3618
SMSA	-0.0315	0.0283	-1.1107
MS	-0.0289	0.0216	-1.3364
EXPP^2	-0.0005	0.0001	-5.5938
OCC	-0.0163	0.0174	-0.9359
IND	0.0267	0.0189	1.4155
UNIONM	0.0063	0.0189	0.3318
YR77	0.1051	0.0063	16.6317
	0.2527	0.0113	22.3048
	0.3693	0.0133	27.6746
	0.4784	0.0159	30.1072
	0.5755	0.0204	28.1598
	0.6819	0.0246	27.7402
C	3.0190	0.6037	5.0006
FEM	-0.4040	0.0690	-5.8535
BLK	0.0149	0.1078	0.1385
ED	0.2208	0.0461	4.7921
EXPP	0.0444	0.0112	3.9636

Hausman Test for HT MGIV VS. Within MGIV

stat, p-val, df =	4.6282	0.0989	2.0000

Hausman-Taylor GMM Estimation

	Estimate	Std. Err.	T-stat
WKS	0.0008	0.0009	0.9135
SOUTH	0.0241	0.0445	0.5416
SMSA	-0.0332	0.0302	-1.1013
MS	-0.0218	0.0291	-0.7478
EXPP^2	-0.0004	0.0001	-5.1422
OCC	-0.0191	0.0177	-1.0789
IND	0.0218	0.0227	0.9607
UNIONM	0.0201	0.0249	0.8071
YR77	0.1040	0.0065	16.0646
	0.2491	0.0113	22.0548
	0.3647	0.0133	27.3864
	0.4732	0.0155	30.4754
	0.5689	0.0197	28.8235
	0.6742	0.0238	28.3113
C	2.9538	0.5630	5.2462
FEM	-0.3990	0.0614	-6.4996
BLK	-0.0094	0.0992	-0.0951
ED	0.2232	0.0435	5.1283
EXPP	0.0451	0.0106	4.2569

Hansen Test for HT

stat, p-val, df =	4.3420	0.1141	2.0000

Amemiya-MaCurdy MGIV Estimation

	Estimate	Std. Err.	T-stat
WKS	0.0006	0.0005	1.2587
SOUTH	-0.0269	0.0219	-1.2306
SMSA	0.0204	0.0180	1.1347
MS	-0.0202	0.0189	-1.0699
EXPP^2	-0.0005	0.0001	-6.0422
OCC	-0.0202	0.0128	-1.5793
IND	0.0289	0.0149	1.9418
UNIONM	0.0092	0.0141	0.6514
YR77	0.1059	0.0061	17.2932
	0.2550	0.0112	22.7436
	0.3730	0.0132	28.3007
	0.4825	0.0152	31.8216
	0.5816	0.0186	31.2796
	0.6897	0.0219	31.4721
C	4.2978	0.2219	19.3715
FEM	-0.3906	0.0392	-9.9681
BLK	-0.0574	0.0477	-1.2032
ED	0.1301	0.0158	8.2119
EXPP	0.0370	0.0049	7.5178

Amemiya-MaCurdy MGIV Estimation (HETERO ADJUSTED)

	Estimate	Std. Err.	T-stat
WKS	0.0006	0.0007	0.9659
SOUTH	-0.0269	0.0327	-0.8220
SMSA	0.0204	0.0237	0.8628
MS	-0.0202	0.0220	-0.9161
EXPP^2	-0.0005	0.0001	-5.9049
OCC	-0.0202	0.0174	-1.1606
IND	0.0289	0.0185	1.5583
UNIONM	0.0092	0.0187	0.4898
YR77	0.1059	0.0063	16.6892
	0.2550	0.0114	22.3107
	0.3730	0.0135	27.5836
	0.4825	0.0161	29.9487
	0.5816	0.0206	28.2932
	0.6897	0.0248	27.8335
C	4.2978	0.2891	14.8649
FEM	-0.3906	0.0433	-9.0182
BLK	-0.0574	0.0587	-0.9792
ED	0.1301	0.0202	6.4579
EXPP	0.0370	0.0057	6.4654

Hausman Test for AM MGIV VS. Within MGIV

stat, p-val, df =	42.1182	0.0000	8.0000

```
Amemiya-MaCurdy GMM Estimation
     Estimate     Std. Err.     T-stat
      0.0012       0.0009       1.4357
     -0.0483       0.0263      -1.8385
      0.1150       0.0246       4.6767
      0.0115       0.0303       0.3813
     -0.0005       0.0001      -5.8212
     -0.0173       0.0166      -1.0410
      0.0206       0.0213       0.9677
      0.0286       0.0215       1.3282
      0.1040       0.0063      16.4659
      0.2426       0.0111      21.8568
      0.3629       0.0131      27.5978
      0.4688       0.0152      30.8802
      0.5711       0.0189      30.1783
      0.6757       0.0228      29.5733
      4.7568       0.2185      21.7731
     -0.3946       0.0392     -10.0543
     -0.1448       0.0431      -3.3623
ED    0.0897       0.0129       6.9436
      0.0340       0.0049       6.9472

Hansen Test for AM
     stat, p-val, df =     49.8631       0.0033      26.0000

EHS Test
     stat, p-val, df =     45.5211       0.0051      24.0000
```

BMS MGIV Estimation

	Estimate	Std. Err.	T-stat
	0.0007	0.0005	1.2862
	-0.0349	0.0216	-1.6130
	0.0304	0.0173	1.7524
	-0.0227	0.0187	-1.2100
	-0.0005	0.0001	-6.0435
	-0.0228	0.0127	-1.7940
	0.0248	0.0147	1.6911
	0.0062	0.0139	0.4444
	0.1057	0.0061	17.2328
	0.2548	0.0112	22.7414
	0.3727	0.0132	28.2969
	0.4820	0.0151	31.8636
	0.5811	0.0186	31.3275
	0.6893	0.0218	31.5921
	4.5265	0.1904	23.7686
	-0.3892	0.0392	-9.9339
	-0.0868	0.0455	-1.9060
ED	0.1112	0.0127	8.7811
	0.0380	0.0046	8.2351

BMS MGIV Estimation (HETERO ADJUSTED)

	Estimate	Std. Err.	T-stat
	0.0007	0.0007	0.9910
	-0.0349	0.0314	-1.1111
	0.0304	0.0226	1.3445
	-0.0227	0.0217	-1.0450
	-0.0005	0.0001	-5.9130
	-0.0228	0.0173	-1.3195
	0.0248	0.0189	1.3172
	0.0062	0.0184	0.3371
	0.1057	0.0063	16.6839
	0.2548	0.0114	22.3386
	0.3727	0.0135	27.6382
	0.4820	0.0161	29.9963
	0.5811	0.0205	28.3558
	0.6893	0.0247	27.8966
	4.5265	0.2325	19.4653
	-0.3892	0.0408	-9.5417
	-0.0868	0.0518	-1.6749
ED	0.1112	0.0152	7.3241
	0.0380	0.0051	7.4690

Hausman Test for BMS MGIV VS. Within MGIV

stat, p-val, df =	45.1177	0.0000	8.0000

BMS GMM Estimation

	Estimate	Std. Err.	T-stat
	0.0011	0.0009	1.2782
	-0.0593	0.0255	-2.3226
	0.1156	0.0229	5.0429
	0.0050	0.0296	0.1694
	-0.0005	0.0001	-6.5611
	-0.0300	0.0167	-1.8003
	0.0160	0.0202	0.7916
	0.0146	0.0199	0.7338
	0.1053	0.0062	17.0083
	0.2437	0.0109	22.2590
	0.3664	0.0128	28.5366
	0.4741	0.0148	31.9359
	0.5784	0.0185	31.2877
	0.6858	0.0222	30.8518
	4.8261	0.1918	25.1608
	-0.4050	0.0378	-10.7025
	-0.1403	0.0424	-3.3070
ED	0.0867	0.0110	7.8478
	0.0350	0.0045	7.7548

Hansen Test for BMS

stat, p-val, df =	62.4753	0.0347	44.0000

EHS Test for BMS

stat, p-val, df =	12.6123	0.8141	18.0000

[1] GAUSS PROGRAM FOR GARCH

(PROGRAM)

```
new;

/*
**             PROGRAM FOR GARCH(p,q)
**
**                  WRITTEN BY
**                SEUNG CHAN AHN
**            DEPARTMENT OF ECONOMICS
**              COLLEGE OF BUSINESS
**                TEMPE, AZ 85287
**
*/

@   OPEN OUTPUT FILE  @

/* ---------------------------------------- */

    output file = ahngarch.out reset ;

/* ---------------------------------------- */

@   DATA LOADING AND TRANSFORMATION  @

/* ------------------------------------------------------ */

    load dat[301,1] = exdmdo.db;
    y = 100*ln(dat[2:rows(dat)]./p[1:rows(dat)-1]) ;

/* ------------------------------------------------------ */

@   DEPENDENT VARIABLE AND EXOGENOUS REGRESSORS @
@   Z: VARIABLES WHICH MAY APPEAR IN THE GARCH TERM @

/* ------------------------------------------------------ */
```

```
    yy = y ;
    xx = ones(rows(y),1)   ;
    zz = ones(rows(y),1)~y ;

/* ------------------------------------------------------ */

@   CONDITIONAL MEAN SPECIFICATION  @

proc res(mb)  ;
local ms      ;

/* ------------------------------------------------------ */

    ms = yy - xx*mb ;

/* ------------------------------------------------------ */

retp(ms)  ;
endp      ;

@ DEFINE K, P AND Q FOR GARCH @

/* ------------------ */

    k = cols(xx) ;
    p = 1 ;
    q = 1 ;

/* ------------------ */

@ ARCH-M @
@ IP1 = 0 IF NO ARCH-M ; IP1 = 1 IF ARCH-M @
@ IP2 = 0 IF ARCH-M WITH DEV; IP2 = 1 IF WITH VAR @

/* ------------------------------- */

    ip1 = 0 ;
    ip2 = 0 ;

/* ------------------------------- */
```

```
@ SPECIFY GARCH TERM  @

proc hterm(a,h,u2,tar,z) ;
local hc,pp,qq ;

    pp = p+1 ; qq = q+1 ;

/* ------------------------------------------------------------ */

   hc = a[1] + a[2]*u2[qq-1] + a[3]*h[pp-1] ;

   @ TARCH(0,1)  hc = a[1] + a[2]*u2[qq-1]
                      + a[3]*tar[qq-1]*u2[qq-1] ; @

   @ IGARCH(1,1) hc = a[1] + a[2]*u2[qq-1]
                      + (1-a[2])*h[pp-1] ;  @

   @ GARCH(1,1)  hc = a[1] + a[2]*u2[qq-1] + a[3]*h[pp-1]
                      + a[4]*h[pp-1]*zz[1] ;      @

/* ------------------------------------------------------------ */

retp(hc) ;
endp    ;

@ INITIAL VALUES FOR PARAMETER IN YOUR MODEL @

/* ------------------------------------------------------------ */

   inib1 = invpd(xx'xx)*(xx'yy) ; @ for mean @
   inib2 = {0.1, 0.2, 0.7 }     ; @ for conditional variances  @
   inib3 = {0.1}                ; @ for ARCH-M term (optimal) @
   inib = inib1|inib2 ;
   @ if ARCH-M, use inib = inib1|inib2|inib3 ; @

/* ------------------------------------------------------------ */

@   CONTROL ALGORITHM  @

/* ------------------------------------------------------------ */
```

```
    algo = 2 ; @ 2=BFGS; 3=DFP; 4=NEWTON; 5=BHHH @
    lser = 2 ; @1=one; 2=STEPBT; 4=BRENT; 5=BHHHSTEP @
    gtol = 1e-3 ;

/* ------------------------------------------------------- */

@   GIVE TITLE FOR YOUR OUTPUT  @

/* ------------------------------ */

    qqt = "GARCH(1,1)" ;

/* ------------------------------ */

@   Max. Lag for LM tests  @

/* ------------------------------ */

    maxlag = 10 ;

/* ------------------------------ */



                    /*************************
                     *  Do not change below   *
                     *************************/

@   GARCH Specification   @
@   The GARCH-M term is the last entry of vb   @

    if p == 0   ; wp = 1 ; else ; wp = p ; endif ;

proc (3) = garch(mb,vb,p,q) ;
local uncm,n,u,uu,hh,i,t,th,j,c,du,dh,tarch ;

    uncm  = meanc(res(mb)^2) ;
    n     = rows(yy)         ;

    u          = sqrt(uncm)*ones(n+q,1) ;
```

```
    u[q+1:q+n] = res(mb)                ;
    hh         = uncm*ones(n+p,1)       ;

    i  = 1 ; do while i <= n ;

        dh = hh[i:i+wp-1] ;
        du = u[i:i+q-1]   ;
        tarch = dummydn(du,0,2) ;
        hh[p+i] = hterm(vb,dh,du^2,tarch,zz[i,.]) ;

        th = hh[p+i]      ;
        c  = vb[rows(vb)] ;
        u[q+i] = u[q+i] - ip1*(ip2*c*th + (1-ip2)*c*sqrt(th)) ;

    i = i + 1 ; endo ;

    hh = hh[p+1:rows(hh)]  ;
    uu = u[q+1:rows(u)]^2  ;

retp(hh,uu,u[q+1:rows(u)]) ;
endp        ;

library maxlik;
#include maxlik.ext ;
maxset ;

@   Define log-likelihood function   @

cc = rows(inib2) ;
proc func(b,yy)  ;
local hh,uu,u,logl ;

  b[k+1:k+cc] = abs(b[k+1:k+cc]) ;

  {hh,uu,u} = garch(b[1:k],b[k+1:k+cc],p,q)    ;
  logl      = -0.5*(ln(hh) + uu./hh)-0.9189385 ;

retp(logl) ;
endp       ;
```

```
    b0 = inib ;

@ Calculate the maximum likelihood estimates @

_max_CovPar = 3        ;
_max_algorithm = algo  ;
_max_LineSearch = lser ;
_max_GradTol = gtol    ;
__output = 10          ;

{b,f1,f2,f3,retcode }= maxlik(yy,0,&func,b0) ;

b[k+1:k+cc] = abs(b[k+1:k+cc]) ;
n  = rows(yy) ;
rc = (n^2)*(_max_finalHess)*(_max_XprodCov)*(_max_finalHess) ;
rc = invpd(rc)      ;
uc = _max_XprodCov  ;

ww = rows(b) ;
{hh,uu,u} = garch(b[1:k],b[k+1:ww],p,q) ;

@  Computing QMLE using approximated Hessian @

proc dmu(bb)  ;
local mm      ;

    mm = res(bb[1:k]) ;

retp ( mm ) ;
endp ;

proc dhh(bb)      ;
local mm1,mm2,mm3 ;

  {mm1,mm2,mm3} = garch(bb[1:k],bb[k+1:k+cc],p,q)  ;

retp (mm1) ;
endp       ;

dm = gradp(&dmu,b)./sqrt(hh)  ;
```

```
dh = gradp(&dhh,b)./hh        ;
apphess = dm'dm + 0.5*(dh'dh) ;
rdd = (apphess)*(_max_XprodCov)*(apphess) ;
rdd =invpd(rdd)                      ;


/*
**  Output
*/

format /rd 8,3;
print " ";
print " ";
print "S. C. AHN ---   " qqt "";
print " " ;
print "=======================================";
print "log-likelihood =" n*f1 ;
print "" ;
print "=======================================";

st1=sqrt(diag(uc))  ;
st2=sqrt(diag(rc))  ;
st3=sqrt(diag(rdd)) ;

print "Results with Usual Covariance Matrix" ;
print "" ;
print "For mean Specification" ;
print " Estimate std. err. t-stat" ;
print b[1:k]~st1[1:k]~(b[1:k]./st1[1:k]) ;

print "";

print "For GARCH Specification" ;
print " Estimate std. err. t-stat" ;
print b[k+1:ww]~st1[k+1:ww]~(b[k+1:ww]./st1[k+1:ww]) ;

print "";

print "=======================================";
print "Results with Robust Covariance Matrix" ;
```

```
print "" ;
print "For mean Specification" ;
print " Estimate std. err. t-stat" ;
print b[1:k]~st2[1:k]~(b[1:k]./st2[1:k]) ;

print "";

print "For GARCH Specification" ;
print " Estimate std. err. t-stat" ;
print b[k+1:ww]~st2[k+1:ww]~(b[k+1:ww]./st2[k+1:ww]) ;

print "";

print "=======================================";
print "Results with Robust Covariance Matrix by Approx. Hessian" ;
print "" ;
print "For mean Specification" ;
print " Estimate std. err. t-stat" ;
print b[1:k]~st3[1:k]~(b[1:k]./st3[1:k]) ;

print "";

print "For GARCH Specification" ;
print " Estimate std. err. t-stat" ;
print b[k+1:ww]~st3[k+1:ww]~(b[k+1:ww]./st3[k+1:ww]) ;

print "";


/*
**  LM Test for ARCH effects
*/

proc lmt(r) ;
local y,x,r2,b,i,t,f,lm ;

    t = rows(uu) - r ;
    y = uu[r+1:r+t]  ;
    x = zeros(rows(y),r) ;
```

```
    i = 1 ; do while i <= r ;

        x[.,i] = uu[i:i+t-1] ;

    i = i + 1 ; endo ;

    x  = ones(t,1)~x      ;
    b  = invpd(x'x)*(x'y) ;
    f  = x*b              ;
    lm = t*(f'f)/(y'y)    ;

retp(lm) ;
endp     ;

print "=======================================";
print "LM TESTS";
print "" ;

    i = 1 ; do while i <= maxlag ;


    "LM for lag=" i ", P =" lmt(i) cdfchic(lmt(i),i) ;
    "" ;

    i = i + 1 ; endo ;

/*
**  Tests Based on Standardized Residuals
*/

uh  = u./sqrt(hh);
u2h = uu./hh     ;
m1  = meanc(uh)  ;
m2  = meanc(u2h) ;
m3  = meanc(uh.*u2h)  ;
m4  = meanc(u2h.*u2h) ;
b3  = m3/(m2^1.5) ;
b4  = m4/(m2*m2)  ;
bera=n*( (b3^2)/6 + ((b4-3)^2)/24 ) ;
print " ";
```

```
print "====================================" ;
print " Skewness =" b3;
print " Kurtosis =" b4;
print " Bera-Jarque Normality (df=2), p =" bera cdfchic(bera,2) ;
"";

print "====================================" ;
ncor=20;
cr=ones(ncor,1);
q=0.0;
uh1=uh-m1;
il=1;
do while il<=ncor;
     uh1=0.0|uh1[1:n-1];
     cr[il,1]=meanc((uh-m1).*uh1)/meanc((uh-m1).*(uh-m1));
     q=q+cr[il,1]*cr[il,1]/(n-il);
     il=il+1;
endo;
q=q*(n+2)*n;
print "  Q(20) for the levels (df=20), p =" q cdfchic(q,20) ;
print "  Lag 1,...,20:";
print cr[1:5]'    ;
print cr[6:10]'   ;
print cr[11:15]'  ;
print cr[16:20]'  ;
print "";

q=0.0;
uh1=u2h-m2;
il=1;
do while il<=ncor;
     uh1=0.0|uh1[1:n-1];
     cr[il,1]=meanc((u2h-m2).*uh1)/meanc((u2h-m2).*(u2h-m2));
     q=q+cr[il,1]*cr[il,1]/(n-il);
     il=il+1;
endo;
q=q*(n+2)*n;
print "  Q(20) for the squares (df=20), p =" q cdfchic(q,20) ;
print "  Lag 1,...,20:";
print cr[1:5]'   ;
```

```
print cr[6:10]'  ;
print cr[11:15]' ;
print cr[16:20]' ;
print "";
print "";

output off ;
```

(2) OUTPUT

```
================================================================
 iteration:  10
 algorithm: BFGS         step method: STEPBT
 function:  1.32970      step length:  0.39314      backsteps:  1
----------------------------------------------------------------
   param.      param. value     relative grad.
       1             0.1021            0.0014
       2             0.0436            0.0081
       3             0.0538            0.0032
       4             0.8945            0.0061


S. C. AHN ---   GARCH(1,1)

========================================
log-likelihood =-398.910

========================================
Results with Usual Covariance Matrix

For mean Specification
 Estimate std. err. t-stat
   0.101    0.055    1.845

For GARCH Specification
 Estimate std. err. t-stat

   0.044    0.040    1.092
   0.054    0.028    1.904
   0.894    0.069   13.032

========================================
Results with Robust Covariance Matrix

For mean Specification
 Estimate std. err. t-stat
   0.101    0.049    2.046
```

```
For GARCH Specification
 Estimate std. err. t-stat

   0.044    0.032    1.380
   0.054    0.027    2.025
   0.894    0.042   21.374

========================================
Results with Robust Covariance Matrix by Approx. Hessian

For mean Specification
 Estimate std. err. t-stat
   0.101    0.051    1.972

For GARCH Specification
 Estimate std. err. t-stat

   0.044    0.040    1.079
   0.054    0.038    1.441
   0.894    0.060   14.869

========================================
LM TESTS

LM for lag=   1.000 , P =  62.353    0.000
LM for lag=   2.000 , P =  62.175    0.000
LM for lag=   3.000 , P =  62.113    0.000
LM for lag=   4.000 , P =  62.628    0.000
LM for lag=   5.000 , P =  67.104    0.000
LM for lag=   6.000 , P =  67.474    0.000
LM for lag=   7.000 , P =  68.039    0.000
LM for lag=   8.000 , P =  68.572    0.000
LM for lag=   9.000 , P =  70.715    0.000
LM for lag=  10.000 , P =  70.739    0.000

====================================
 Skewness =  -0.015
 Kurtosis =   4.386
 Bera-Jarque Normality (df=2), p =  24.036    0.000
```

```
====================================
  Q(20) for the levels (df=20), p =  27.633    0.118
  Lag 1,...,20:
  -0.079    0.033   -0.033    0.106   -0.036
  -0.062   -0.080   -0.034    0.073   -0.086
   0.134   -0.003   -0.057   -0.016    0.075
   0.031    0.016    0.064    0.016    0.103

  Q(20) for the squares (df=20), p =  18.838    0.532
  Lag 1,...,20:
   0.016   -0.078   -0.001    0.041    0.051
  -0.052   -0.028    0.010    0.065    0.004
  -0.050    0.013   -0.060   -0.011    0.149
  -0.053   -0.010   -0.029    0.030    0.091
```