

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/selfsubscribe
 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” (IN ASCII FORMAT) , and exit WS_FTP.
5. Using Telnet, go to “research.asu.edu”. Then do the followings.
6. Type your login name and password. Push the RETURN button. Push the button again. Then, you will see “research>”.
7. Go to the 'gauss' subdirectory by typing 'cd ~/gauss'.
8. Type the following while in the 'gauss' subdirectory:

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

9. Go back to the root directory ('cd ~').
10. 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.)
```

11. 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.)
12. 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.)
13. Type 'run mult.prg'. This runs the small test file. If you can see some output, it means that your setup is completed.
14. If you want to quit GAUSS, type “quit” in front of the gauss prompt (gauss).
15. 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 edit the content of the file. Remember that you should quit GAUSS to use the pico editor.
16. 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 Output

```
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 optimum;
#include optimum.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 ; @Controlling tolerance rate @
__opstmth = "bfgs, half" ; @algorithm @
__output = 0; @ Control output files @

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

@ b = the value of b minimizing f(b)@
@ retcode = 0 (normal convergence) @
@ retcode ≠ 0 (bad 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, $\text{polla}(b) = g_T(\theta)$ in GMM and $\text{grad} = G_T(\theta)$.

(5) Matrix Operations:

- $\text{rndns}(k,t,dd)$: $k = \text{rows}$; $t = \text{cols}$; $dd = \text{seed number}$
 - outcome: $k \times t$ matrix of iid $N(0,1)$ random numbers.
 - For uniform, use rndus .

- Let A and B be conformible matrices.
 - $A*B$ = product of A and B.
 - A' = transpose of A
 - $A'B$ = product of A' and B.
 - $A[:,1]$ = the first column of A.
 - $A[1,:]$ = the first row of A.
 - $A[1,2]$ = the (1,2)th element of A.
 - $A[1:5,:]$ = matrix of the 1st, 2nd, 3rd, 4th and 5th rows of A.
 - $\text{invpd}(A)$ = inverse of a positive definite matrix A.
 - $\text{diag}(A)$ = $n \times 1$ vector of diagonal elements of a $n \times n$ matrix A.
 - If $A = [a_{ij}]$, $\text{sqrt}(A) = [\sqrt{a_{ij}}]$; $A^2 = [a_{ij}^2]$.
 - $A|B$ = merging A and B vertically; $A \sim B$ = merging A and B horizontally.
 - $\text{sumc}(A)$ = $n \times 1$ vector of sums of individual columns for a $m \times n$ matrix A.

$$\text{EX: } A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}; \text{sumc}(A) = \begin{bmatrix} 9 \\ 12 \end{bmatrix};$$

- $\text{mean}(A)$ = $n \times 1$ vector of means of individual columns for a $m \times n$ matrix A.

$$\text{EX: } A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}; \text{meanc}(A) = \begin{bmatrix} 3 \\ 4 \end{bmatrix};$$

- $\text{mean}(A)$ = $n \times 1$ vector of standard errors of individual columns for a $m \times n$ matrix A.

- $A = m \times n$, $B = m \times n$, $C = m \times 1$, $d = \text{scalar}$
 - $A./B = [a_{ij}/b_{ij}]$ (element by element operation)
 - $A./C = [a_{ij}/b_j]$ (element by element operation)
 - $d - A = [d - a_{ij}]$; $d*A = [da_{ij}]$; $A/d = [a_{ij}/d]$.

[3] GAUSS PROGRAMS FOR GMM

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    ;

```

```

/* COMPUTING 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                                2/14/01
5:10 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 0.00083

```
=====
                                SECOND-STAGE GMM
=====
OPTMUM Version 3.1.4                                2/14/01
5:10 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 0.00083

=====

SECOND-STAGE GMM

=====

OPTMUM Version 3.1.4
5:10 pm

2/14/01

=====

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 0.00083

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

[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 observations 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 manufacturing 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 @
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. @

```

```

/*
** Here, rexp = years of full-time work experience at 1996.
** When your model contains time-dummy variables and has a variable (say, h)
** whose value increases by one every year automatically, it is not possible
** to estimate the coefficient of the variable by the within estimation
** method. In this case, you need to create a time-invariant variable whose
** value always equal to the value of h at the begining time period. If your
** model does not use time dummy variables, you can use h as an time-varying
** variable.
*/

rexp = {0,1,2,3,4,5,6} ;
rexp = ones(n,1) .* rexp ;
rexp = exp - rexp; @ time invariant variable of initial exp @

yy = lwage ; @ dependent var. @
xx = wks~south~smsa~ms~exp2~occ~ind~unionm~dyr ;
    @ time-varying indep. @

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

xxt = wks~south~smsa~ms~exp2~occ~ind~unionm ;

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

/*
** Exclude from xxt (i) variables whose values automatically
** increase by one every year; and (ii) their squares.
*/

xxtt = wks~south~smsa~ms~occ~ind~unionm ;
zz = ones(n*t,1)~fem~blk~edu~rexp ;

@ Clearing unnecessary variables @

clear id68, exp, exp2, 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 - pvmatl(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 observations 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] ;
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] ;
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 ;

/*
** 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

(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 and AM 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 observations and # of variables @
```

```
nobs = 4165 ;  
tim   = 7   ;  
nvar  = 23  ;
```

```
@ Read Data @
```

```
load dat[nobs,nvar] = 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 = rows(dat)/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 ;

/*
** 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

(4) MARYANN.PRG (SMLE for Tobit)

```
/*      TOBIT_1.PRG
**
**      PROGRAM FOR TOBIT REGRESSION OF PANEL DATA
**      USING SIMULATED MAXIMUM LIKELIHOOD ESTIMATION
**
**      BY MARY ANN HOFMANN
**
*/

/*      NOTE ABOUT DATA SET CONFIGURATION
**      THE DATA SET MUST BE ARRANGED SO THAT THERE IS ONE ROW
**      FOR EACH INDIVIDUAL, AND THE DEPENDENT AND INDEPENDENT
**      VARIABLES ARE ARRANGED IN COLUMNS BY YEAR.
**      THE DATA SET MUST INCLUDE A GROUP OF DUMMY VARIABLES,
**      WHERE THE VALUE IS ONE IF THE DEPENDENT VARIABLE IS NONZERO,
**      ZERO OTHERWISE.  FOR EXAMPLE:
**
**      OBS # DEP_T1 DEP_T2 X1_T1 X1_T2 X2_T1 X2_T2 DV_T1 DV_T2
**
**      1   100     101     5000  5025   3     3     1     1
**
**      2     0       0     4250  4300   4     5     0     0
**
**
**/

      new;

@ open output file and format it @
      output file = maryann.out reset;
      format /rd 12, 4;

@ define n and t @
      n = 103;      @ # of individuals @
      t = 4;        @ # of time periods @
```

```

@ cc is added to the mean function value before taking the log @
  cc = 0;

@ read data @
  load dat[n,17] = tobit.db;

@ define variables @
id    = dat[.,1];
ch87  = dat[.,2];
ch88  = dat[.,3]; @ logs of charitable contributions ($) @
ch89  = dat[.,4];
ch90  = dat[.,5];
pr87  = dat[.,6]; @after-tax contributions: log(1-tax rate) @
pr88  = dat[.,7];
pr89  = dat[.,8];
pr90  = dat[.,9];
inc87  = dat[.,10]; @ log of dispo. y ($1000) before contribution @
inc88  = dat[.,11];
inc89  = dat[.,12];
inc90  = dat[.,13];
dv87  = dat[.,14]; @ dummy variable = 1 if ch > 0 @
dv88  = dat[.,15];
dv89  = dat[.,16];
dv90  = dat[.,17];

@ define matrices @
  yy  = ch87~ch88~ch89~ch90; @ dependent variable @
  pr  = pr87~pr88~pr89~pr90; @ first X variable @
  inc = inc87~inc88~inc89~inc90; @ second X variable @
  dv  = dv87~dv88~dv89~dv90; @ dummy variables @

@ create matrix of random numbers for simulations @
  draws = 20; @ defines H (number of simulations) @
  seed1 = 1;
  rnd   = rndns(n,draws,seed1);

@ set start values (use best guess or results from Limdep) @
@ NOTE: B[5] cannot be zero @
  b0 = {1,-1,1,1,1};

```

```

@ define f-tilda function @
proc tilda(y,x1,x2,dum,rh,p);
local j, ff, ffj;
  j = 1;
  ff = 1;
  do while j <= t;
  ffj = (((1/abs(p[5]))*pdfn((y[1,j]-p[1]-p[2]*x1[1,j]-p[3]*x2[1,j]
    -p[4]*rh)/abs(p[5])))^dum[1,j])*
    ((cdfn((-p[1]-p[2]*x1[1,j]-p[3]*x2[1,j]-p[4]*rh)
    / abs(p[5])))^{(1-dum[1,j]}));
  ff = ff*ffj;
  j = j + 1;
  endo;
  retp(ff);
  endp;

```

```

@set up the Maximum Likelihood function @

```

```

library optmum;
#include optmum.ext;
OPTSET;

```

```

@ define likelihood function @

```

```

proc smle(b);
  local i, h, lnf, fst, fsth, mnfst, lnfst ;

```

```

@ set counter for individuals @

```

```

@ set up lnf to accumulate sum of ln f(.) over individuals @

```

```

  i=1;
  lnf = 0;
  do while i <= n;

```

```

@ set counter for draws @

```

```

@ set up fst to accumulate sum of simulations @

```

```

  h = 1;
  fst = 0;
  do while h <= draws;
  fsth = tilda(yy[i,.],pr[i,.],inc[i,.],dv[i,.],rnd[i,h], b);

```

```

        fst = fst + fsth;
        h   = h + 1;
    endo;

@ compute mean of fst and take its log @
    mnfst = fst/draws;
    lnfst = ln(mnfst+cc);

@ increase sum of ln f(.) @

    lnf = lnf + lnfst;
    i = i + 1;
    endo;

    retp(-lnf/n) ;
    endp;

__title = "TOBIT by SML";
_opgtol = 0.00001;
_opstmth = "BFGS HALF";

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

@ estimate standard errors @

@ create initial matrix used to compute covariance matrix @
sum = zeros(5,5);
i = 1;
do while i <= n;
    @ compute numerator (num) and denominator (den) for s @
    num = zeros(1,5);
    den = 0;
    h = 1;
    do while h <= draws;
        proc tildb(p);
            local til;
            til = tilda(yy[i,.],pr[i,.],inc[i,.],dv[i,.],rnd[i,h],p);
            retp(til);
        endp;
    endo;
endo;

```

```

        clo = tilddb(b);
        num = num + gradp(&tilddb,b);
        den = den + clo;
    h = h + 1;
    endo;

s = num/den;
ss = (s's);
sum = sum + ss;
i = i + 1;
endo;

@ compute the covariance matrix as the inverse of sum @
cov = invpd(sum);

@ obtain the standard errors and t-statistics @
se = sqrt(diag(cov));
tst = b./se;

@ print the results @
"";
"SMLE TOBIT RESULTS";
"";
" coefficient      std. error      t-statistic";
b~se~tst;
output off;

```

(OUTPUT)

```
=====  
iteration: 1  
algorithm: BFGS          step method: HALF  
function: 10.80337      step length: 0.00000      backsteps: 0  
-----
```

param.	param. value	relative grad.
1	1.0000	0.0409
2	-1.0000	0.0103
3	1.0000	0.1186
4	1.0000	0.3632
5	1.0000	0.6608

```
=====  
iteration: 2  
algorithm: BFGS          step method: HALF  
function: 9.27835       step length: 54.69406     backsteps: 1  
-----
```

param.	param. value	relative grad.
1	8.7451	0.0594
2	6.4838	0.0652
3	-0.0692	0.0439
4	7.4793	0.1755
5	1.1542	0.2583

```
=====  
:
```

```
=====  
iteration: 30  
algorithm: BFGS          step method: HALF  
function: 8.04522       step length: 1.12712     backsteps: 1  
-----
```

param.	param. value	relative grad.
1	2.3799	0.0000
2	-1.3729	0.0000
3	0.4751	0.0001
4	3.0677	0.0001
5	1.3676	0.0001

=====

TOBIT by SML

=====

OPTMUM Version 3.1.4
6:00 pm

2/14/01

=====

return code = 0
normal convergence

Value of objective function 8.045223

Parameters	Estimates	Gradient
P01	2.3865	0.0000
P02	-1.3740	-0.0000
P03	0.4731	0.0000
P04	3.0671	-0.0000
P05	1.3675	0.0000

Number of iterations 31
Minutes to convergence 3.73600

SMLE TOBIT RESULTS

coefficient	std. error	t-statistic
2.3865	0.6644	3.5918
-1.3740	1.3444	-1.0220
0.4731	0.2018	2.3448
3.0671	0.1119	27.4026
1.3675	0.0238	57.4916

(5) dynamic1.prg (Monte Carlo program for stationary dynamic panels)

```

/*****
** MONTE CARLO EXPERIMENTS FOR DYNAMIC PANEL DATA MODELS **
**                               WRITTEN BY S. C. AHN                               **
*****/

@ CLEAN MEMMORY @
  new ;

@ OPEN OUTPUT FILE @
  output file = dynamic1.out reset ;

@ FORMAT OUTPUT FILE @
  format /rd 8,3 ;

/*****
** Starting comments on the output file **
*****/

" MODEL SPECIFICATION: " ;
"   y_t = d*y_{t-1} + a + e_t ;" ;
"   y_0 = gamma*a + e_0      " ;
"   Stationary y: gamma = 1/(1-d)" ;

/*****
** Data Generating Process **
*****/

@ For the size of N @
  j1 = 2 ;
  let jj[4,1] = 50 100 300 500 ;
  do while j1 <= 2 ;
  jj1 = jj[j1,1] ;

@ For the size of T @
  j2 = 1 ;
  let jjj[3,1] = 4 7 11 ;
  do while j2 <= 1 ;
  jj2 = jjj[j2,1] ;

```

```

@ For the size of delta @
  j3 = 1          ;
  let jjjj[3,1] = 0.9 0.8 0.5 ;
  do while j3 <= 3 ;
  jj3 = jjjj[J3,1] ;

@ For siga @
  j4 = 3          ;
  let jjjjj[3,1] = 0 0.5 1 ;
  do while j4 <= 3 ;
  jj4 =jjjjj[J4,1] ;

@ Values of major parameters @
  ddd      = 1      ; @ SEED @
  iter     = 100    ; @ # OF ITERATIONS @
  tt       = jj2    ; @ TIME HORIZON @
  nn       = jj1    ; @ # OF CROSS-SECTIONAL UNITS @
  dd       = jj3    ; @ DELTA @
  se       = 1      ; @ SE(E) @
  sa       = jj4    ; @ SE(A) @
  rr       = 1/(1-dd) ; @ GAMMA @
  s0       = sqrt(se^2/(1-dd^2)) ; @ SE(Y0) @

  ::

```

(Output)

MODEL SPECIFICATION:

$$y_t = d*y_{t-1} + a + e_t ;$$

$$y_0 = \text{gamma}*a + e_0$$

$$\text{Stationary } y: \text{ gamma} = 1/(1-d)$$

=====

VALUES OF MAJOR PARAMETERS

SEED = 1.000
ITER = 100.000
T = 4.000
N = 100.000
DELTA = 0.900
SIGA = 1.000
SIGE = 1.000
GAMMA = 10.000
SIGO = 2.294

AB GMM RESULTS:

MEAN OF ESTIMATES : 0.360
MSE : 0.517
MEAN OF ASYM. VAR : 0.185
SIZE OF T-TEST : 0.340

AB0 GMM RESULTS:

MEAN OF ESTIMATES : 0.947
MSE : 0.011
MEAN OF ASYM. VAR : 0.003
SIZE OF T-TEST : 0.590

AS1 GMM RESULTS:

MEAN OF ESTIMATES : 0.964
MSE : 0.036
MEAN OF ASYM. VAR : 0.010
SIZE OF T-TEST : 0.560
SIZE OF AHN TEST : 0.530

=====

VALUES OF MAJOR PARAMETERS

SEED = 1.000
ITER = 100.000
T = 4.000
N = 100.000
DELTA = 0.800
SIGA = 1.000
SIGE = 1.000
GAMMA = 5.000
SIGO = 1.667

AB GMM RESULTS:

MEAN OF ESTIMATES : 0.555
MSE : 0.191
MEAN OF ASYM. VAR : 0.126
SIZE OF T-TEST : 0.210

AB0 GMM RESULTS:

MEAN OF ESTIMATES : 0.833
MSE : 0.016
MEAN OF ASYM. VAR : 0.007
SIZE OF T-TEST : 0.400

AS1 GMM RESULTS:

MEAN OF ESTIMATES : 0.892
MSE : 0.055
MEAN OF ASYM. VAR : 0.016
SIZE OF T-TEST : 0.500
SIZE OF AHN TEST : 0.430

=====

VALUES OF MAJOR PARAMETERS

SEED = 1.000
ITER = 100.000
T = 4.000
N = 100.000
DELTA = 0.500
SIGA = 1.000
SIGE = 1.000
GAMMA = 2.000
SIG0 = 1.155

AB GMM RESULTS:

MEAN OF ESTIMATES : 0.483
MSE : 0.036
MEAN OF ASYM. VAR : 0.026
SIZE OF T-TEST : 0.130

AB0 GMM RESULTS:

MEAN OF ESTIMATES : 0.522
MSE : 0.014
MEAN OF ASYM. VAR : 0.006
SIZE OF T-TEST : 0.240

AS1 GMM RESULTS:

MEAN OF ESTIMATES : 0.535
MSE : 0.035
MEAN OF ASYM. VAR : 0.011
SIZE OF T-TEST : 0.210
SIZE OF AHN TEST : 0.200

(5) dynamic2.prg (Monte Carlo program for nonstationary dynamic panels)

```

/*****
** MONTE CARLO EXPERIMENTS FOR DYNAMIC PANEL DATA MODELS **
**                WRITTEN BY S. C. AHN                **
*****/

@ CLEAN MEMMORY @
  new ;

@ OPEN OUTPUT FILE @
  output file = dynamic2.out reset ;

@ FORMAT OUTPUT FILE @
  format /rd 8,3 ;

/*****
** Starting comments on the output file **
*****/

" MODEL SPECIFICATION: " ;
"      y_t = d*y_{t-1} + a + e_t ;" ;
"      y_0 = gamma*a + e_0      " ;
"      Non-Stationary y:  gamma = 1/(1-d)+1 " ;

/*****
** Data Generating Process **
*****/

@ For the size of N @
  j1 = 2 ;
  let jj[4,1] = 50 100 300 500 ;
  do while j1 <= 2 ;
    jj1 = jj[j1,1] ;

@ For the size of T @
  j2 = 1 ;
  let jjj[3,1] = 4 7 11 ;
  do while j2 <= 1 ;
    jj2 = jjj[j2,1] ;

```

```

@ For the size of delta @
  j3 = 1          ;
  let jjjj[3,1] = 0.9 0.8 0.5 ;
  do while j3 <= 3 ;
  jj3 = jjjj[J3,1] ;

@ For siga @
  j4 = 3          ;
  let jjjjj[3,1] = 0 0.5 1 ;
  do while j4 <= 3 ;
  jj4 =jjjjj[J4,1] ;

@ Values of major parameters @
  ddd   = 1      ; @ SEED @
  iter  = 100    ; @ # OF ITERATIONS @
  tt    = jj2    ; @ TIME HORIZON @
  nn    = jj1    ; @ # OF CROSS-SECTIONAL UNITS @
  dd    = jj3    ; @ DELTA @
  se    = 1      ; @ SE(E) @
  sa    = jj4    ; @ SE(A) @
  rr    = 1/(1-dd)+1 ; @ GAMMA @
  s0    = sqrt(se^2/(1-dd^2)) ; @ SE(Y0) @

@ Printing out the parameter values in the output file @
  " " ;
  "===== " ;
  "VALUES OF MAJOR PARAMETERS" ;
  "SEED      = " ddd      ;
  "ITER      = " iter     ;
  "T         = " tt      ;
  "N         = " nn      ;
  "DELTA     = " dd      ;
  "SIGA      = " sa      ;
  "SIGE      = " se      ;
  "GAMMA     = " rr      ;
  "SIG0      = " s0      ;

```

(Output)

MODEL SPECIFICATION:

$$y_t = d*y_{t-1} + a + e_t ;$$

$$y_0 = \text{gamma}*a + e_0$$

$$\text{Non-Stationary } y: \text{ gamma} = 1/(1-d)+1$$

=====

VALUES OF MAJOR PARAMETERS

SEED	=	1.000
ITER	=	100.000
T	=	4.000
N	=	100.000
DELTA	=	0.900
SIGA	=	1.000
SIGE	=	1.000
GAMMA	=	11.000
SIG0	=	2.294

AB GMM RESULTS:

MEAN OF ESTIMATES	:	0.677
MSE	:	0.172
MEAN OF ASYM. VAR	:	0.110
SIZE OF T-TEST	:	0.200

AB0 GMM RESULTS:

MEAN OF ESTIMATES	:	1.002
MSE	:	0.015
MEAN OF ASYM. VAR	:	0.002
SIZE OF T-TEST	:	0.720

AS1 GMM RESULTS:

MEAN OF ESTIMATES	:	0.956
MSE	:	0.033
MEAN OF ASYM. VAR	:	0.004
SIZE OF T-TEST	:	0.530
SIZE OF AHN TEST	:	0.510

=====

VALUES OF MAJOR PARAMETERS

SEED = 1.000
ITER = 100.000
T = 4.000
N = 100.000
DELTA = 0.800
SIGA = 1.000
SIGE = 1.000
GAMMA = 6.000
SIG0 = 1.667

AB GMM RESULTS:

MEAN OF ESTIMATES : 0.776
MSE : 0.053
MEAN OF ASYM. VAR : 0.035
SIZE OF T-TEST : 0.170

AB0 GMM RESULTS:

MEAN OF ESTIMATES : 1.007
MSE : 0.049
MEAN OF ASYM. VAR : 0.005
SIZE OF T-TEST : 0.820

AS1 GMM RESULTS:

MEAN OF ESTIMATES : 0.874
MSE : 0.031
MEAN OF ASYM. VAR : 0.009
SIZE OF T-TEST : 0.400
SIZE OF AHN TEST : 0.380

=====

VALUES OF MAJOR PARAMETERS

SEED = 1.000
ITER = 100.000
T = 4.000
N = 100.000
DELTA = 0.500
SIGA = 1.000
SIGE = 1.000
GAMMA = 3.000
SIG0 = 1.155

AB GMM RESULTS:

MEAN OF ESTIMATES : 0.505
MSE : 0.013
MEAN OF ASYM. VAR : 0.008
SIZE OF T-TEST : 0.130

AB0 GMM RESULTS:

MEAN OF ESTIMATES : 0.895
MSE : 0.163
MEAN OF ASYM. VAR : 0.003
SIZE OF T-TEST : 1.000

AS1 GMM RESULTS:

MEAN OF ESTIMATES : 0.519
MSE : 0.013
MEAN OF ASYM. VAR : 0.006
SIZE OF T-TEST : 0.200
SIZE OF AHN TEST : 0.180