GAUSS CORNER

[1] 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.

[Example 1]

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

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

[Example 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 = AGMM@ ; @ Writing title for output @
__opgtol = 0.00001 ; @Controling tolerance rate @
__opstmth = Abfgs,half@; @algorithm @
__output = 0; @ Control output files @

{b, func, grad, retcode} = optprt(optmum(&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 young(b) ; @ b must be a vector @
local minsu, chulsoo ... ;
      :
retp(...) ;
endp ;

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

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

(5) Matrix Operations:

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

- Let A and B be conformable matrices.
  - A*B = product of A and B.
  - A= = transpose of A
  - A=B = produnct 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.
  - invpd(A) = inverse of a positive definite matrix A.
  - diag(A) = n×1 vector of diagonal elements of a n×n matrix A.
  - If A = [a_{ij}], sqrt(A) = [$\sqrt{a_{ij}}$]; A^2 = [a_{ij}^2].
  - A|B = merging A and B vertically; A~B = merging A and B horizontally.
  - sumc(A) = n×1 vector of sums of individual columns for a m×n matrix A.

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

  - meanc(A) = n≠1 vector of means of individual columns for a m×n matrix A.

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

  - stdc(A) = n×1 vector of standard errors of individual columns for a m×n matrix A.

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

[2] Program for OLS

Program: ols.prg (Can download from the website for ECN 525)

```
/*
**  OLS Program
*/

@ Data loading   @

load data[100,5] = exer.txt;

@   Define # of observations, # of regressors, X and Y @

tt   = rows(data); @ # of observations @
kk   = 5; @ # of regressors @
y    = data[.,1];
x2   = data[.,2];
x3   = data[.,3];
x4   = data[.,4];
x5   = data[.,5];
vny  = {"y"}; @ name of the dependent variable @

@ Dependent variable @

    yy = y ;
    vny = {"y"};

@ Regressors @

    xx = ones(tt,1)~x2~x3~x4~x5;
    vnx = {"cons", "x2", "x3", "x4", "x5" } ;

@     Do not change below  @
@     OLS using yy and xx  @

b  = invpd(xx'xx)*(xx'yy);
e  = yy - xx*b ;

s2 = (e'e)/(tt-kk);
v  = s2*invpd(xx'xx);

econ = b~sqrt(diag(v))~(b./sqrt(diag(v)));
econ = vnx~econ;

se = sqrt(diag(v));
```

```
sst   = yy'yy - tt*meanc(yy)^2;
sse   = e'e;

r2    = 1 - sse/sst;

@ Printing out OLS results @

output file = ols.out reset;

let mask[1,4] = 0 1 1 1;
let fmt[4,3] =
    "-*.*s" 8 8
    "*.*lf" 10 4
    "*.*lf" 10 4
    "*.*lf" 10 4;

format /rd 10,4 ;
"" ;
"OLS Regression Result" ;
"------------------------" ;
" dependent variable: " $vny ;
"" ;
" R-Squares           " r2 ;
"" ;
"variable     coeff.  std. err.   t-st " ;
yyprin = printfm(econ,mask,fmt);
"" ;

output off ;
```

Outcome in old.out

```
OLS Regression Result
------------------------
 dependent variable:          y

 R-Squares              0.8960

variable     coeff.      std. err.      t-st
cons       -0.0009    0.2225      -0.0042
x2           0.9617    0.2040       4.7145
x3           1.7759    0.2659       6.6790
x4           3.0003    0.1903      15.7669
x5           3.8823    0.2035      19.0790
```

[3] Program for Monte Carlo Experiments

Program: olsmonte.prg (Can download from the ECN 525 website)

```
/*
**  Monte Carlo Program
*/

@   Load Data @

load data[100,5] = exer.txt ;

@ Data generation under Strong Ideal Conditions @

/*
** The regressors are common to individual data sets.
** The errors are different across different data sets
** That is, regressors are nonstochastic but the errors are
*/

seed   = 1;
tt    = 100; @ # of observations @
kk   = 5;  @ # of betas @
iter  = 5000; @ # of sets of different data @
xx  = ones(tt,1)~data[.,2:5]; @ Regressors are fixed @
tb  = {0,1,2,3,4} ; @ y = x(2)*1 + x(3)*2 + x(4)*3 + x(5)*4 + e @

storb  = zeros(iter,1);
storse = zeros(iter,1);

i = 1; do while i <= iter;

@ Generating y @
yy  = xx*tb + 2*rndns(tt,1,seed);

@ OLS using yy and xx  @

b  = invpd(xx'xx)*(xx'yy);
e  = yy - xx*b ;

s2 = (e'e)/(tt-kk);
v  = s2*invpd(xx'xx);

se = sqrt(diag(v));

storb[i,1]  = b[2,1];
```

```
storse[i,1] = se[2,1];

i = i + 1; endo;

@ Reporting Monte Carlo results @

output file = olsmonte.out reset;

format /rd 12,3;

"Monte Carlo results";
"-----------";
"Mean of OLS b(2)                =" meanc(storb);
"s.e. of OLS b(2)                =" stdc(storb);
"mean of estimated s.e. of OLS b(2) =" meanc(storse) ;

library pgraph;
graphset;
{a1,a2,a3}=hist(storb,50);
output off ;
```

Outcome in olsmonte.out

```
Monte Carlo results
-----------
Mean of OLS b(2)              =       0.998
s.e. of OLS b(2)              =       0.184
mean of estimated s.e. of OLS b(2) = 0.185
```

[4] Programs for Basic Panel Data Models

Program name: pan_gls.prg

```
new  ;

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

      #include mgiv.col ;

@   Open an output file  @

      output file = pan_gls.out reset ;

@   Formatting output file  @

      format /rd 12,4 ;

@   Provide # of ovservations and # of variables @

    nobs = 336 ;
    nvar = 13   ;

@   Read Data   @

    load dat[nobs,nvar] = auto_1.txt ;
      @ 48 states (N = 48), 1982 - 1988 (T = 7)@


@  Define Variables @

    id       = dat[.,1]  ; @ ID for States @
    year     = dat[.,2]  ; @ year @
    spircons = dat[.,3]  ; @ Spirits consumption @
    unrate   = dat[.,4]  ; @ Unemployment rate @
    perinc   = dat[.,5]  ; @ Personal Income @
    emppop   = dat[.,6]  ; @ Employment/Population Ration @
    beertax  = dat[.,7]  ; @ Tax on Case of Beer @
    mlda     = dat[.,8]  ; @ Minimum Legal Drinking Age @
    vmiles   = dat[.,9]  ; @ Ave. mile per driver @
    jaild    = dat[.,10] ; @ Mandatory Jail Sentense = 1 @
   comserd  = dat[.,11] ; @ Mandotory Jail Sentence @
   allmort  = dat[.,12] ; @ # of Vehicle Fatalities @
   mrall    = dat[.,13] ; @ Vehicle Fatality Rate (VFR) @

@ Creating Time dummy variables @

      v    = {1982.5, 1983.5, 1984.5, 1985.5, 1986.5, 1987.5 };
    dyr  = dummy(year,v)};

@   Define N and T @

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

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

```
    yy    = mrall*10000  ; @ dependent var. @
    xx    = beertax~mlda~jaild~comserd~unrate~ln(perinc)~dyr[.,2:7]; @ time-varying indep. @
      zz    = ones(rows(yy),1); @ time-invariant indep. @

      vny   = {"VFR"};
      vnx   = {"beertax","mlda","jailed","comserd","unrate","lpinc",
                  "yr83", "yr84", "yr85", "yr86", "yr87", "yr88"};
    vnz   = {"cons"};

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

    xxt   = beertax~mlda~jaild~comserd~unrate~ln(perinc) ;

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

    clear dat ;

      let mask[1,4] = 0 1 1 1;
      let fmt[4,3] =
    "-*.*s" 8 8
    "*.*lf" 10 4
    "*.*lf" 10 4
    "*.*lf" 10 4;

@   Define k and g @

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

@ Creating AM and Mean Variables @

    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) ;

@   OLS estimation  @

      ow   = xx~zz;
      od   = invpd(ow'ow)*(ow'yy);
      oe   = yy - ow*od;
      os2  = (oe'oe)/(rows(ow)-cols(ow));
      ov   = os2*invpd(ow'ow);
      ose  = sqrt(diag(ov));
      orsq = 1-(oe'oe)/(yy'yy-rows(yy)*meanc(yy)^2);

"OLS Estimation Result" ;
"-----------------------" ;
```

```
" dependent variable: " $vny ;
"";
"variable     coeff.  std. err.   t-st " ;
yyprin = printfm((vnx|vnz)~od~ose~(od./ose),mask,fmt);
"" ;
"R-Square =" orsq;
"";

@   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)) ;
      wrsq = 1 - (we'we)/(yy'yy-rows(yy)*meanc(yy)^2);

"Within Estimation Result" ;
"------------------------" ;
" dependent variable: " $vny ;
"";
"variable     coeff.  std. err.   t-st " ;
yyprin = printfm(vnx~wb~ws~(wb./ws),mask,fmt);
"" ;
"R-Square =" wrsq;
"";

@   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)) ;
      gee  = qvyy -qvxx*gd[1:cols(xx)] ;
      grsq = 1 - gee'gee/(yy'yy-rows(yy)*meanc(yy)^2);

"GLS Estimation Result" ;
"------------------------" ;
" dependent variable: " $vny ;
"";
"variable     coeff.  std. err.   t-st " ;
yyprin = printfm((vnx|vnz)~gd~gs~(gd./gs),mask,fmt);
"" ;
"R-Square =" grsq;
```

```
"";

   " 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 ;

OUTPUT OFF
```

Output: pan_gls.out

```
OLS Estimation Result
------------------------
 dependent variable:          VFR

variable     coeff.  std. err.   t-st
beertax     0.1112    0.0624    1.7832
mlda       -0.0297    0.0317   -0.9367
jailed      0.1959    0.0723    2.7085
comserd     0.1460    0.0813    1.7951
unrate     -0.0227    0.0143   -1.5852
lpinc      -1.9018    0.2265   -8.3957
yr83       -0.0900    0.0959   -0.9389
yr84       -0.0648    0.0996   -0.6504
yr85       -0.0783    0.1006   -0.7782
yr86        0.0632    0.1022    0.6185
yr87        0.1032    0.1067    0.9671
yr88        0.1404    0.1107    1.2679
cons       20.7805    2.3157    8.9738

R-Square =      0.3482
```

Within Estimation Result

----

dependent variable: VFR

variable	coeff.	std. err.	t-st
beertax	-0.4768	0.1657	-2.8773
mlda	-0.0019	0.0178	-0.1053
jailed	0.0147	0.1201	0.1222
comserd	0.0345	0.1377	0.2503
unrate	-0.0629	0.0111	-5.6629
lpinc	1.7964	0.3625	4.9560
yr83	-0.0972	0.0322	-3.0232
yr84	-0.2812	0.0371	-7.5740
yr85	-0.3745	0.0389	-9.6220
yr86	-0.3376	0.0422	-8.0090
yr87	-0.4347	0.0481	-9.0369
yr88	-0.5213	0.0537	-9.7103

R-Square = 0.9390

GLS Estimation Result

----

dependent variable: VFR

variable	coeff.	std. err.	t-st
beertax	0.0052	0.1140	0.0454
mlda	0.0005	0.0175	0.0307
jailed	0.1429	0.0992	1.4404
comserd	-0.0704	0.1145	-0.6149
unrate	-0.0780	0.0105	-7.4267
lpinc	0.4189	0.3071	1.3641
yr83	-0.0918	0.0321	-2.8580
yr84	-0.2577	0.0368	-6.9957
yr85	-0.3237	0.0383	-8.4470
yr86	-0.2533	0.0409	-6.1996
yr87	-0.3193	0.0460	-6.9357
yr88	-0.3794	0.0510	-7.4451
cons	-1.1846	2.9655	-0.3995

R-Square = 0.9328

THETA = 0.1213
SIGE2 = 0.0241
SIGA2 = 0.2304
S.E.R.= 0.1551

Hausman Test, p-val, df =	76.4475	0.0000	6.0000
ALT1 Test, p-val, df =	66.3981	0.0000	6.0000

Program name: pan_gls2.prg

```
     new ;

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

     #include mgiv.col ;

@   Open output file @

     output file = pan_gls2.out reset ;

@   Format output file @

     format /rd 12,4 ;

@   Provide # of ovservations and # of variables @

   nobs = 336 ;
   nvar = 13   ;

@   Read Data   @

   load dat[nobs,nvar] = auto_1.txt ;
     @ 48 states (N = 48), 1982 - 1988 (T = 7)@

@  Define Variables @

   id       = dat[.,1]  ; @ ID for States @
   year     = dat[.,2]  ; @ year @
   spircons = dat[.,3]  ; @ Spirits consumption @
   unrate   = dat[.,4]  ; @ Unemployment rate @
   perinc   = dat[.,5]  ; @ Personal Income @
   emppop   = dat[.,6]  ; @ Employment/Population Ration @
   beertax  = dat[.,7]  ; @ Tax on Case of Beer @
   mlda     = dat[.,8]  ; @ Minimum Legal Drinking Age @
   vmiles   = dat[.,9]  ; @ Ave. mile per driver @
   jaild    = dat[.,10] ; @ Mandatory Jail Sentense = 1 @
   comserd  = dat[.,11] ; @ Mandotory Jail Sentence @
   allmort  = dat[.,12] ; @ # of Vehicle Fatalities @
   mrall    = dat[.,13] ; @ Vehicle Fatality Rate (VFR) @

@ Creating Time dummy variables @

     v    = {1982.5, 1983.5, 1984.5, 1985.5, 1986.5, 1987.5 };
   dyr  = dummy(year,v)};

@   Define N and T @

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

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

   yy    = mrall*10000  ; @ dependent var. @
   xx    = beertax~mlda~jaild~comserd~unrate~ln(perinc)~dyr[.,2:7]; @ time-varying indep. @
```

```
     zz    = ones(rows(yy),1); @ time-invariant indep. @

     vny   = {"VFR"};
     vnx   = {"beertax","mlda","jailed","comserd","unrate","lpinc",
                 "yr83", "yr84", "yr85", "yr86", "yr87", "yr88"};
   vnz   = {"cons"};

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

   clear dat   ;

@   Define k and g @

   k  = cols(xx)  ;
   g  = cols(zz)  ;

     let mask[1,4] = 0 1 1 1;
     let fmt[4,3] =
   "-*.*s" 8 8
   "*.*lf" 10 4
   "*.*lf" 10 4
   "*.*lf" 10 4;

@   Within with HET-AUTO adjustment @

   {wb,wcovh} = w_ha(xx,yy,n,t) ;
   wsh = sqrt(diag(wcovh)) ;

     "Within Estimation Results (HETERO/AUTO ADJUSTED)" ;
     "-----------------------" ;
     " dependent variable: " $vny ;
     "";
     "variable     coeff.  std. err.   t-st " ;
     yyprin = printfm(vnx~wb~wsh~(wb./wsh),mask,fmt);
     "" ;

@   Kiefer's Estimation @

   {kb,kcov,kcovh} = kiefer(xx,yy,n,t) ;
   ks  = sqrt(diag(kcov))  ;
   ksh = sqrt(diag(kcovh)) ;

     "Kiefer's Within Estimation Results" ;
     "-----------------------" ;
     " dependent variable: " $vny ;
     "";
     "variable     coeff.  std. err.   t-st " ;
     yyprin = printfm(vnx~kb~ks~(kb./ks),mask,fmt);
     "" ;

     "Kiefer's Within Estimation Results (HETERO ADJUSTED)" ;
     "-----------------------" ;
     " dependent variable: " $vny ;
     "";
     "variable     coeff.  std. err.   t-st " ;
```

```
      yyprin = printfm(vnx~kb~ksh~(kb./ksh),mask,fmt);
      "" ;

@   RE-GLS Estimation @

    {rb,rcov,rcovh} = regls(xx,yy,n,t) ;
    rs  = sqrt(diag(rcov))  ;
    rsh = sqrt(diag(rcovh)) ;

      "RE-GLS Estimation Results" ;
      "------------------------" ;
      " dependent variable: " $vny ;
      "";
      "variable     coeff.  std. err.   t-st " ;
      yyprin = printfm(vnx~rb~rs~(rb./rs),mask,fmt);
      "" ;

      "RE-GLS Estimation Results (CROSS-SECTION HETERO ADJUSTED)" ;
      "------------------------" ;
      " dependent variable: " $vny ;
      "";
      "variable     coeff.  std. err.   t-st " ;
      yyprin = printfm(vnx~rb~rsh~(rb./rsh),mask,fmt);
      "" ;

output off
```

Output file: pan_gls2.prg

```
Within Estimation Results (HETERO/AUTO ADJUSTED)
------------------------
 dependent variable:          VFR

variable     coeff.  std. err.   t-st
beertax    -0.4768    0.2949   -1.6167
mlda       -0.0019    0.0209   -0.0894
jailed      0.0147    0.0158    0.9292
comserd     0.0345    0.1285    0.2684
unrate     -0.0629    0.0127   -4.9657
lpinc       1.7964    0.6243    2.8775

Kiefer's Within Estimation Results
------------------------
 dependent variable:          VFR

variable     coeff.  std. err.   t-st
beertax    -0.1833    0.1837   -0.9982
mlda       -0.0058    0.0175   -0.3343
jailed     -0.0026    0.0882   -0.0295
comserd     0.0420    0.1110    0.3784
unrate     -0.0518    0.0112   -4.6255
lpinc       2.0991    0.3932    5.3387
```

Kiefer's Within Estimation Results (HETERO ADJUSTED)

----

dependent variable: VFR

variable	coeff.	std. err.	t-st
beertax	-0.1833	0.2254	-0.8135
mlda	-0.0058	0.0157	-0.3720
jailed	-0.0026	0.0138	-0.1885
comserd	0.0420	0.0924	0.4545
unrate	-0.0518	0.0107	-4.8367
lpinc	2.0991	0.5934	3.5373

RE-GLS Estimation Results

----

dependent variable: VFR

variable	coeff.	std. err.	t-st
beertax	0.4601	0.1210	3.8032
mlda	-0.0112	0.0233	-0.4819
jailed	0.1532	0.0957	1.5998
comserd	-0.0521	0.1147	-0.4538
unrate	-0.0062	0.0126	-0.4970
lpinc	0.2217	0.0521	4.2559

RE-GLS Estimation Results (CROSS-SECTION HETERO ADJUSTED)

----

dependent variable: VFR

variable	coeff.	std. err.	t-st
beertax	0.4601	0.1028	4.4774
mlda	-0.0112	0.0174	-0.6464
jailed	0.1532	0.1249	1.2261
comserd	-0.0521	0.1346	-0.3868
unrate	-0.0062	0.0123	-0.5084
lpinc	0.2217	0.0430	5.1512